Integer Programming Formulations for the k-Edge-Connected 3-Hop-Constrained Network Design Problem

I. Diarrassouba
Laboratoire LMAH, Université du Havre, 25 Rue Philippe Lebon, 76600 Le Havre, France

V. Gabrel and A. R. Mahjoub
Laboratoire LAMSADE, Université Paris-Dauphine, Place du Maréchal De Lattre de Tassigny, 75775 Paris Cedex 16, France

L. Gouveia
Departamento de Estatística e Investigação Operacional, Centro de Investigação Operacional, Faculdade de Ciênçias, Universidade de Lisboa, Portugal

P. Pesneau
University of Bordeaux, INRIA Bordeaux - Sud-Ouest, IMB UMR 5251, France

In this article, we study the k-edge-connected L-hop-constrained network design problem. Given a weighted graph $G = (V, E)$, a set D of pairs of nodes, two integers $L \geq 2$ and $k \geq 2$, the problem consists in finding a minimum weight subgraph of G containing at least k edge-disjoint paths of length at most L between every pair $\{s, t\} \in D$. We consider the problem in the case where $L = 2$, 3 and $|D| \geq 2$. We first discuss integer programming formulations introduced in the literature. Then, we introduce new integer programming formulations for the problem that are based on the transformation of the initial undirected graph into directed layered graphs. We present a theoretical comparison of these formulations in terms of LP-bound. Finally, these formulations are tested using CPLEX and compared in a computational study for $k = 3$, 4, 5. **NETWORKS, Vol. 67(2), 148–169 2016**

Keywords: survivable network; edge-disjoint paths; flow; hop-constrained path; integer programming formulation; k-edge-connected; aggregated formulations; separated formulations

Received December 2012; accepted May 2015
Correspondence to: A. R. Mahjoub; e-mail: mahjoub@lamsade.dauphine.fr
Contract grant sponsor: National Funding from FCT—Fundaç ã o para a Ciência e a Tecnologia, under the project: PEst-OE/MAT/UI0152
DOI 10.1002/net.21667
Published online 26 December 2015 in Wiley Online Library (wileyonlinelibrary.com).

1. INTRODUCTION

Let $G = (V, E)$ be an undirected graph with node set V and edge set E and $D \subseteq V \times V$ a set of pairs of nodes, called *demands* with $|D| = d$. If a pair $\{s, t\}$ is a demand in D, we call s and t *demand nodes* or *terminal nodes*. Let $L \geq 2$ be a fixed integer. If s and t are two nodes of V, an L-st-path in G is a path between s and t of length at most L, where the length is the number of edges (also called *hops*).

Given a weight function $c : E \to \mathbb{R}$, which associates the weight $c(e)$ to each edge $e \in E$ and an integer $k \geq 2$, the *k-edge-connected L-hop-constrained network design problem* (kHNDP for short) consists in finding a minimum cost subgraph of G having at least k edge-disjoint L-st-paths between each demand $\{s, t\} \in D$.

The kHNDP has applications in the design of survivable telecommunication networks where bounded-length paths are required. Survivable networks must satisfy some connectivity requirements; that is, the networks should still be functional after the failure of certain links. As pointed out in [33] (see also [31]), the topology that seems to be very efficient (and needed in practice) is the uniform topology, that is to say that corresponding to networks that survive after the failure of $k - 1$ or fewer edges, for some $k \geq 2$. However, this requirement is often insufficient regarding the reliability of a telecommunication network. In fact, the alternative paths could be too long to guarantee an effective routing. In data networks, such as the Internet, the elongation of the route of the information could cause a major loss in the transfer speed. For other networks, the signal itself could be degraded by a longer routing. In such cases, the L-path requirement guarantees exactly the needed quality of the alternative routes. Moreover, in a telecommunication network, usually several commodities have to be routed in the network between pairs of terminals. To ensure an effective routing, there must exist a sufficient number of hop-constrained paths between each pair of terminals.

The kHNDP has been extensively investigated when there is only one demand in the network ($|D| = 1$). In particular, the associated polytope has received special attention. In [30], Huygens et al. study the kHNDP for $k = 2$ and $L = 2, 3$. They give an integer programming formulation for the problem and show that the linear programming relaxation of this formulation completely describes the associated polytope. From this, they obtain a minimal linear description of that polytope. They also show that this formulation is no longer valid when $L \geq 4$. In [11], Dahl et al. study the kHNDP when $L = 2$ and $k \geq 2$. They give a complete description of the associated polytope in this case and show that it can be solved in polynomial time using linear programming. In [9], Dahl considers the kHNDP for $k = 1$ and $L = 3$. He gives a complete description of the dominant of the associated polytope. Dahl and Gouveia [10] consider the directed hop-constrained path problem. They describe valid inequalities and characterize the associated polytope when $L \leq 3$. Huygens and Mahjoub [29] study the kHNDP when $k = 2$ and $L \geq 4$. They also study the variant of the problem where k node-disjoint paths of length at most L are required between two terminals. They give an integer programming formulation for these two problems when $L = 4$.

In [7], Coullard et al. investigate the structure of the polyhedron associated with the st-walks of length L of a graph, where a walk is a path that may go through the same node more than once. They present an extended formulation of the problem, and, using projection, they give a linear description of the associated polyhedron. They also discuss classes of facets of that polyhedron.

The kHNDP has also been studied when $|D| \geq 2$. In [12], Dahl and Johannessen consider the case where $k = 1$ and $L = 2$. They introduce valid inequalities and develop a branch-and-cut algorithm. The problem of finding a minimum cost spanning tree with hop-constraints is also considered in [21, 22, 24]. Here, the hop-constraints limit to a positive integer H the number of links between the root and any terminal in the network. Dahl [8] studies the problem when $H = 2$ from a polyhedral point of view, and gives a complete description of the associated polytope when the graph is a wheel. Finally, Huygens et al. [28] consider the problem of finding a minimum cost subgraph with at least two edge-disjoint L-hop-constrained paths between each given pair of terminal nodes. They give an integer programming formulation of that problem for $L = 2, 3$ and present several classes of valid inequalities. They also devise separation routines. Using these, they propose a branch-and-cut algorithm and discuss some computational results.

Besides hop-constraints, another reliability condition, which is used in to limit the length of the routing, requires that each link of the network belongs to a ring (cycle) of bounded length. In [18], Fortz et al. consider the two-node connected subgraph problem with bounded rings. This problem consists in finding a minimum cost 2-node connected subgraph (V, F) such that each edge of F belongs to a cycle of length at most L. They describe several classes of facet-defining inequalities for the associated polytope and devise a branch-and-cut algorithm for the problem. In [19], Fortz et al. study the edge version of that problem. They give an integer programming formulation for the problem in the space of the natural design variables and describe different classes of valid inequalities. They study the separation problem for these inequalities and discuss branch-and-cut algorithms.

The related k-edge-connected subgraph problem and its associated polytope have also been the subject of extensive research in the past years. Grötschel and Monma [25] and Grötschel et al. [26, 27] study the k-edge-connected subgraph problem within the framework of a general survivable model. They discuss polyhedral aspects and devise cutting plane algorithms. Didi Biha and Mahjoub [14] study that problem and give a complete description of the associated polytope when the graph is series-parallel. In [15], Didi Biha and Mahjoub study the Steiner version of that problem and characterize the polytope when k is even. Chopra in [6] studies the dominant of that problem and introduces a class of valid inequalities for its polyhedron. Barahona and Mahjoub [2] characterize the polytope for the class of Halin graphs. In [17], Fonlupt and Mahjoub study the fractional extreme points of the linear programming relaxation of the 2-edge-connected subgraph polytope. They introduce an ordering on these extreme points and characterize the minimal extreme points with respect to that ordering. As a consequence, they obtain a characterization of the graphs for which the linear programming relaxation of that problem is integral. Didi Biha and Mahjoub [16] extend the results of Fonlupt and Mahjoub [17] to the case $k \geq 3$ and introduce some graph reduction operations. Kerivin et al. [32] study that problem in the more general case where each node of the graph has a specific connectivity requirement. They present different classes of facets of the associated polytope when the connectivity requirement of each node is at most 2 and devise a branch-and-cut algorithm for the problem in this case. In [3], Bendali et al. study the k-edge-connected subgraph problem for the case $k \geq 3$. They introduce several classes of valid inequalities and discuss the separation algorithm for these inequalities. They devise a branch-and-cut algorithm using the reduction operations of [16] and give some computational results for $k = 3$, 4, 5. A complete survey on the k-edge-connected subgraph problem can be found in [31].

In this work, we introduce four new integer programming formulations for the kHNDP when $L = 2, 3$ and $k \geq 2$. The paper is organized as follows. In Section 2, we present integer programming formulations introduced in the literature and which are defined on the original graph. Then, in Sections 3 and 4, we propose two new approaches for the problem that are based on directed layered graphs, when $L = 2$, 3 and $k \geq 2$. One approach (called the "separated" approach) uses a layered graph for each hop-constrained subproblem and the other (called the "aggregated" approach) uses a single layered graph for the whole problem. These new approaches yield new integer programming formulations for the problem when $L = 2, 3$. In Section 5, we compare the different formulations in terms of linear programming relaxation. Finally, in the last section, we test these formulations using CPLEX and present some computational results for $k = 3, 4, 5$ and $L = 2, 3$.

The rest of this section is devoted to more definitions and notation. An edge $e \in E$ with endnodes u and v is denoted by uv. Given two node subsets W and W', we denote by $[W, W']$ the set of edges having one endnode in W and the other in W'. If $W = \{u\}$, we then write $[u, W']$ for $[\{u\}, W']$. We also denote by $\overline{W}$ the node set $V \setminus W$. The set of edges having only one node in W is called a *cut* and denoted by $\delta(W)$. We will write $\delta(u)$ for $\delta(\{u\})$. Given two nodes $s, t \in V$, a cut $\delta(W)$ such that $s \in W$ and $t \in \overline{W}$ is called an *st-cut*.

We will also denote by $H = (U, A)$ a directed graph where U is the set of nodes and A is the set of arcs. An arc a with origin u and destination v will be denoted by (u, v). Given two node subsets W and W' of U, we will denote by $[W, W']$ the set of arcs whose origin is in W and whose destination is in W'. As before, we will write $[u, W']$ for $[\{u\}, W']$ and $\overline{W}$ will denote the node set $U \setminus W$. The set of arcs having their origin in W and their destination in $\overline{W}$ is called a *cut* or *dicut* in H and is denoted by $\delta^+(W)$. We will also write $\delta^+(u)$ for $\delta^+(\{u\})$ with $u \in U$. If s and t are two nodes of H such that $s \in W$ and $t \in \overline{W}$, then $\delta^+(W)$ will be called an *st-cut* or *st-dicut* in H. If W and W' are two node subsets of H, then $[W, W']^+$ will denote the set of arcs of H whose origins are in W and destinations are in W'. As for undirected graphs, we will write $[u, W']^+$ for $[\{u\}, W']^+$.

Given an undirected graph $G = (V, E)$ (resp. a directed graph $H = (U, A)$) and an edge subset $F \subseteq E$ (resp. an arc subset $B \subseteq A$), we let $x^F \in \mathbb{R}^E$ (resp. $y^B \in \mathbb{R}^A$) be the *incidence vector* of F (resp. B), that is, the $0-1$ vector such that $x^F(e) = 1$ if $e \in F$ (resp. $y^B(a) = 1$ if $a \in B$) and 0 otherwise. Given F a subset of E (resp. A) and a vector $x \in \mathbb{R}^E$ (resp. $y \in \mathbb{R}^A$), $x(F)$ (resp. $y(F)$) will represent the term $\sum_{e \in F} x(e)$ (resp. $\sum_{e \in F} y(e)$).

2. ORIGINAL GRAPH-BASED FORMULATIONS

In this section, we present three integer programming formulations for the kHNDP. The first one is the so-called natural formulation which uses only the design variables. The two other formulations use paths and flows variables in the original space.

2.1. Natural Formulation

Let $G = (V, E)$ be an undirected graph, $D \subseteq V \times V$ be a demand set, and two integers $k \geq 2$ and $L \in \{2, 3\}$. If an edge subset $F \subseteq E$ induces a solution of the kHNDP, that is, a subgraph (V, F) containing k-edge-disjoint L-st-paths for every $\{s, t\} \in D$, then its incidence vector x satisfies the following inequalities.

$$x(\delta(W)) \geq k \text{ for all } st\text{-cuts } \delta(W),\ W \subset V, \{s, t\} \in D, \tag{2.1}$$

$$x(e) \geq 0 \qquad \text{for all } e \in E, \tag{2.2}$$

$$x(e) \leq 1 \qquad \text{for all } e \in E. \tag{2.3}$$

In [9], Dahl considers the problem of finding a minimum cost path between two given terminal nodes s and t of length at most L. He studies the polyhedron (the L-path polyhedron) associated with that problem and introduces a class of inequalities as follows.

Let $\{s, t\} \in D$ and let a partition $(V_0, V_1, \ldots, V_{L+1})$ of V be such that $s \in V_0$ and $t \in V_{L+1}$, and $V_i \neq \emptyset$ for all $i = 1, \ldots, L$. Let T be the set of edges $e = uv$, where $u \in V_i$, $v \in V_j$, and $|i - j| > 1$. Then the inequality

$$x(T) \geq 1$$

is valid for the L-path polyhedron.

Using the same partition, this inequality can be generalized in a straightforward way to the
kHNDP polytope as

$$x(T) \geq k. \tag{2.4}$$

The set T is called an L-st-path-cut, and a constraint of type (2.4) is called an L-st-path-cut inequality. See Figure 1 for an example of an L-st-path-cut inequality with $L = 3$ and, $V_0 = \{s\}$ and $V_{L+1} = \{t\}$.

Note that T intersects every L-st-path in at least one edge and each st-cut $\delta(W)$ intersects every st-path.

Huygens et al. [28] show that the kHNDP can be formulated as an integer program using the design variables when $L = 2, 3$.

Theorem 2.1 ([28]). *Let $G = (V, E)$ be a graph, $k \geq 2$ and $L \in \{2, 3\}$. Then the kHNDP is equivalent to the following integer program*

$$\min \left\{cx;\ \textit{subject to } (2.1)\text{– } (2.4),\ x \in \mathbb{Z}^E\right\}. \tag{2.5}$$

Formulation (2.5) is called the natural formulation and is denoted by $k\text{HNDP}_{Nat}$. In [28], Huygens et al. study the polytope associated with this formulation and introduce some facet-defining inequalities for the problem. They also develop a branch-and-cut algorithm for the kHNDP when $k = 2$ and $L = 2, 3$.

In [4], Bendali et al. study the kHNDP when $k \geq 2$, $L = 2$, 3 and $|D| = 1$. They study the polyhedral structure of that formulation and give necessary and sufficient conditions under which the L-st-path-cut inequalities (2.4) define facets. In particular, they show that an L-st-path-cut inequality induced by a partition $(V_0, \ldots, V_{L+1})$, with $s \in V_0$ and $t \in V_{L+1}$, is facet-defining only if $|V_0| = |V_{L+1}| = 1$. One can easily see that this condition also holds even when $|D| \geq 2$. Thus, we have the following theorem.

Theorem 2.2. *Let $\{s, t\} \in D$ and let $\pi = (V_0, \ldots, V_{L+1})$ be a partition of V with $s \in V_0$ and $t \in V_{L+1}$. The L-st-path-cut*

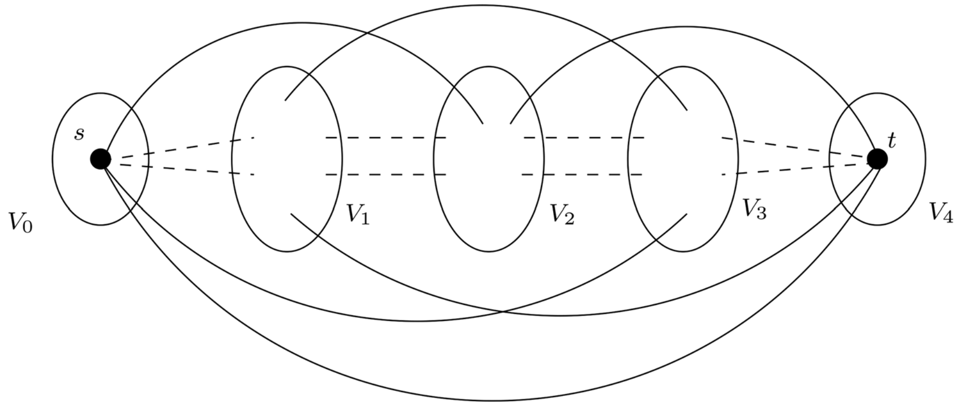

FIG. 1. Support graph of a L-st-path-cut with $L=3$, $V_0=\{s\}$, $V_{L+1}=\{t\}$ and T formed by the solid edges.

inequality (2.4) induced by π defines a facet of the kHNDP polytope only if $|V_0|=|V_{L+1}|=1$.

Theorem 2.2 points out the fact that an L-st-path-cut inequality induced by a partition $(V_0,\dots,V_{L+1})$ such that $|V_0|\geq 2$ or $|V_{L+1}|\geq 2$, is redundant with respect to those L-st-path-cut inequalities induced by partitions $(V_0',V_1',\dots,V_L',V_{L+1}')$ with $V_0'=\{s\}$ and $V_{L+1}'=\{t\}$. Therefore, in the remainder of the paper, the only L-st-path-cuts that we will consider are those induced by partitions of the form $(\{s\},V_1,\dots,V_L,\{t\})$.

2.2. Undirected Path-Based Formulation

A solution of the kHNDP can be modelled by a collection of L-st-paths of G, for all $\{s,t\}\in D$.

For all $\{s,t\}\in D$, let $\mathcal{P}_{st}$ be the set of L-st-paths of G, and let $\mu^{st}(P)$ be the $0-1$ variable which equals 1 if the path $P\in\mathcal{P}_{st}$ is chosen and 0 otherwise.

If an edge subset $F\subseteq E$ induces a solution of the kHNDP, then the following inequalities are satisfied by its incidence vector x^F and $(\mu^{st}(P),\ P\in\mathcal{P}_{st},$ for all $\{s,t\}\in D)$.

$$\sum_{P\in\mathcal{P}_{st}}\mu^{st}(P)\geq k,\ \text{for all } \{s,t\}\in D, \tag{2.6}$$

$$\sum_{P\in\mathcal{P}_{st},\,e\in P}\mu^{st}(P)\leq x(e),\ \text{for all } e\in E,\ \{s,t\}\in D, \tag{2.7}$$

$$x(e)\leq 1,\ \text{for all } e\in E, \tag{2.8}$$

$$\mu^{st}(P)\geq 0,\ \text{for all } P\in\mathcal{P}_{st},\{s,t\}\in D. \tag{2.9}$$

Inequalities (2.6) state that a solution of the problem contains at least k L-st-paths of G, for all $\{s,t\}\in D$, while inequalities (2.7) and (2.8) ensure that these L-st-paths are edge-disjoint.

We have the following theorem which follows from the above remark.

Theorem 2.3. *The kHNDP for L = 2, 3 is equivalent to the following integer program*

$$\textit{min}\{cx;\ \textit{subject to}\ (2.6)-(2.9),\ x\in\mathbb{Z}_+^E,\mu^{st}\in\mathbb{Z}_+^{\mathcal{P}_{st}},\ \textit{for all}\ \{s,t\}\in D\}. \tag{2.10}$$

Formulation (2.10) is called the undirected path formulation and is denoted by $k\text{HNDP}^U_{\text{Path}}$. Note that, in many combinatorial optimization problems, path-based formulations imply an exponential number of variables, as the number of paths in a graph is, in general, exponential. This requires one to use appropriate methods like column generation to solve the linear relaxation of the problem. However, for the kHNDP, the number of L-st-paths is bounded by $|V|^{L-1}$, for each $\{s,t\}\in D$. Hence, the number of variables of $k\text{HNDP}^U_{\text{Path}}$ is polynomial (with $L=2$, 3) and its linear relaxation can be solved by enumerating all the L-st-paths in a single linear program.

2.3. Undirected Flow-Based Formulation

In this section, we introduce a flow-based model for the problem using flow variables in the graph G. A similar formulation has been proposed by Dahl and Gouveia [10] for the kHNDP with $k=1$ and $|D|=1$, and, to the best of our knowledge, this is the first time that such a formulation is given for $k\geq 2$ and $|D|\geq 2$.

Let $G'=(V,A)$ be the directed graph obtained from G by replacing each edge $uv\in E$ by two arcs (u,v) and (v,u). Given a demand $\{s,t\}\in D$, $f^{st}\in\mathbb{R}^A$ is a flow vector on G' between s and t of value k. Thus, for all $\{s,t\}\in D$, f^{st} satisfies the following constraints.

$$\sum_{a\in\delta^+(u)}f^{st}(a)-\sum_{a\in\delta^-(u)}f^{st}(a)=\left\{\begin{array}{c}k\text{ if } u=s,\\ 0\text{ if } u\in V\setminus\{s,t\},\\ -k\text{ if } u=t,\end{array}\right\},\quad \text{for all } u\in V, \tag{2.11}$$

$$f^{st}(u,s)=0,\ \text{for all } (s,u)\in A,\ u\neq s, \tag{2.12}$$

$$f^{st}(t,v)=0,\ \text{for all } (v,t)\in A,\ v\neq t, \tag{2.13}$$

$$f^{st}(u,v)+f^{st}(v,u)\leq x(uv),\ \text{for all } uv\in E, \tag{2.14}$$

$$\left.\begin{array}{r}f^{st}(u,v)\\ f^{st}(v,u)\end{array}\right\}\geq 0,\ \text{for all } uv\in E, \tag{2.15}$$

$$x(uv)\leq 1,\ \text{for all } uv\in E. \tag{2.16}$$

Note that constraints (2.12) and (2.13) remove the flow variables for every arc entering node s and leaving node t, for all

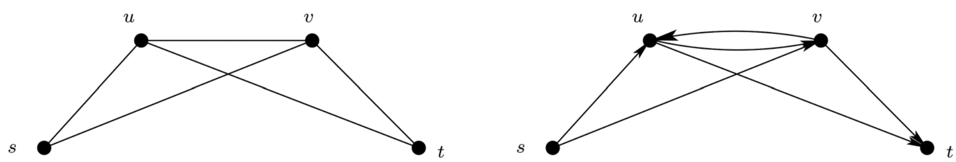

FIG. 2. Illustration of the linking inequalities for an edge uv with $u, v \neq s, t$.

$\{s,t\} \in D$. In fact, it is not hard to see that these arcs will never be used in an optimal solution of the problem. Thus the corresponding flow variables are set to 0. Also inequalities (2.14) are the *linking inequalities* which state that if an edge is not taken in the solution, then the two corresponding arcs have a flow equal to 0. They also indicate that for a given demand $\{s, t\}$ and an edge uv with $u, v \neq s, t$, if edge uv is taken in the solution, then only one of the arcs (u, v) and (v, u) can be used by the flow. This comes from the fact that in an optimal solution, the edge uv may be used in an st-path either from u to v (this is arc (u, v)) or from v to u (this is arc (v, u)). In fact, if both arcs (u, v) and (v, u) are used in the solution, then the solution in the original graph contains two 3-st-paths of the form (s, u, v, t) and (s, v, u, t) which share the edge uv. However, by removing the edge uv, these two st-paths are replaced by the two st-paths (s, u, t) and (s, v, t), of length 2, with lower cost. An illustration is given in Figure 2.

To represent the hop-constraint, we have to introduce additional inequalities for all demands $\{s,t\} \in D$:

- when $L=2$, it is sufficient to prevent in a solution the selection of an "internal" arc $(u, v) \in A$, that is, an arc incident neither to s nor to t, as follows:

$$f^{st}(u,v) = 0 \text{ for all } (u,v) \in A \text{ such that } u \neq s, v \neq t. \tag{2.17}$$

- when $L=3$, we have to add the following inequalities to the system:

$$\sum_{(u,v)\in A, v\neq t} f^{st}(u,v) \leq f^{st}(s,u), \text{ for all } (s,u) \in A, \tag{2.18}$$

$$\sum_{(u,v)\in A, v\neq t} f^{st}(u,v) \leq 0, \text{ for all } u \in V \setminus \{s,t\} \text{ such that } (s,u) \notin A. \tag{2.19}$$

Inequalities (2.18) indicate that if arc (s, u) is not in the solution, then any "internal" arc (u, v) cannot be in the solution. Inequalities (2.19), in a similar way, express the fact that any "internal" arc (u, v) cannot be in the solution when (s, u) does not belong to A. Both inequalities are necessary and sufficient to eliminate all paths of length greater than 3. Observe that, when $L=3$, constraints similar to (2.18) for arcs (v, t) are not necessary as they can be obtained from (2.18) and the flow conservation equations (2.11).

It can be easily seen that the solution set of the kHNDP is completely described by inequalities (2.11)–(2.16), together with integrality constraints, equations (2.17) for $L=2$ and inequalities (2.18)–(2.19) for $L=3$.

Theorem 2.4. *The kHNDP is equivalent to*

$$\left.\begin{array}{ll} \min\{cx;\ \text{subject to } (2.11)\text{–}(2.16), (2.17), & \\ x \in \mathbb{Z}_+^E, f^{st} \in \mathbb{Z}_+^A, \text{ for all } \{s,t\} \in D\} & \text{if } L = 2, \text{ and to} \\ \min\{cx;\ \text{subject to } (2.11)\text{–}(2.16), (2.18)-(2.19), & \\ x \in \mathbb{Z}_+^E, f^{st} \in \mathbb{Z}_+^A, \text{ for all } \{s,t\} \in D\}, & \text{if } L = 3. \end{array}\right\} \tag{2.20}$$

This formulation will be called the undirected flow formulation and is denoted by $k\text{HNDP}^U_{\text{Flow}}$.

3. DEMAND DECOMPOSITION BASED FORMULATIONS

In this section, we will introduce three integer programming formulations for the kHNDP for $L=2$, 3 where we use a directed layered graph to model each hop-constrained subproblem. These formulations will be called *separated formulations*.

In these layered digraphs, any dipath has at most 3 hops. The idea of replacing hop-constrained path subproblems in the original graph by unconstrained path subproblems in an adequate graph has been first suggested in Gouveia [22] and subsequently used in other related works (e.g., [5] and [23]) for more general hop-constrained network design problems. However, the directed graphs proposed here are different from the ones suggested in [22] when $L=3$.

3.1. Graph Transformation

Given $G = (V, E)$ and $\{s,t\} \in D$, let $\widetilde{G}_{st} = (\widetilde{V}_{st}, \widetilde{A}_{st})$ be the layered digraph obtained from G as follows:

- $\widetilde{V}_{st} = N_{st} \cup N'_{st} \cup \{s,t\}$ with $N_{st} = V \setminus \{s,t\}$ and N'_{st} is a copy of N_{st} (each node $u \in N_{st}$ corresponds to a node u' of N'_{st}),
- $\widetilde{A}_{st}$ is composed of four kinds of arcs:
  - for all $su \in E$, $(s,u) \in \widetilde{A}_{st}$,
  - for all $vt \in E$, $(v',t) \in \widetilde{A}_{st}$,
  - for all $u \in N_{st}$, we introduce $\min\{|[s,u]|, |[u,t]|, k\}$ arcs of the form $(u,u') \in \widetilde{A}_{st}$,
  - if $L=3$, for all $uv \in E \setminus \{st\}$ with $u, v \in N_{st}$, $\{(u,v'), (v,u')\} \in \widetilde{A}_{st}$ (see Fig. 3 for an illustration with $L=3$).

For an edge $e = uv \in E$, we denote by $\widetilde{A}_{st}(e)$ the set of arcs of $\widetilde{G}_{st}$ corresponding to the edge e:

- when $u=s$ (resp. $v=t$), $\widetilde{A}_{st}(e)$ contains (s, v) (resp. (u', t)),
- when $u \neq s$ and $v \neq t$, if $L=3$, $\widetilde{A}_{st}(e) = \{(u,v'), (v,u')\}$ and, if $L=2$, $\widetilde{A}_{st}(e)$ is empty.

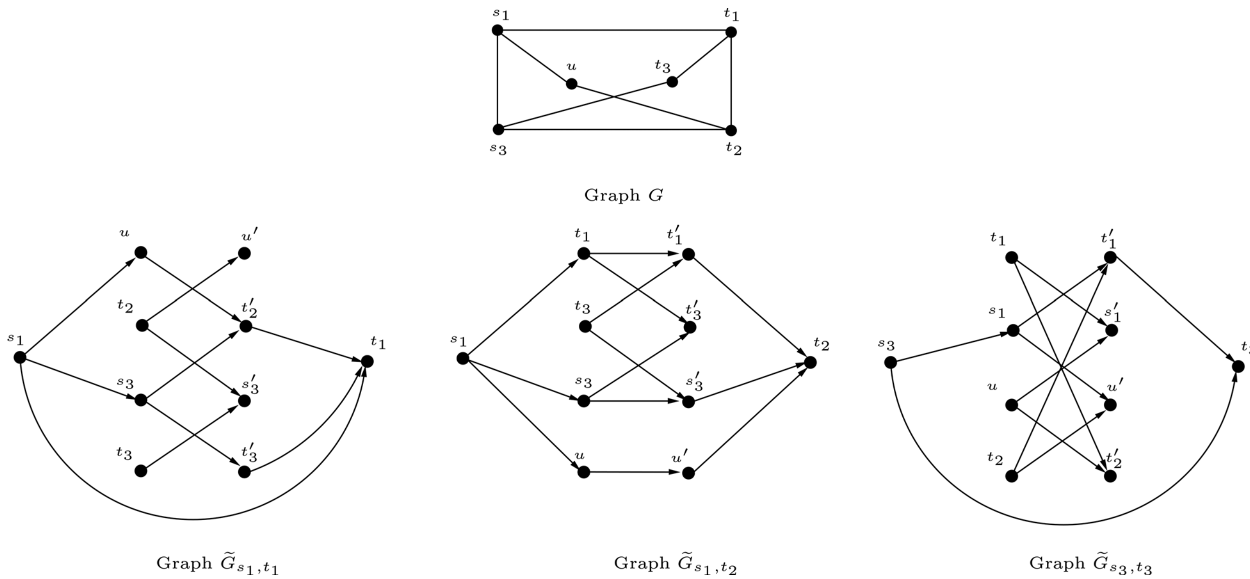

FIG. 3. Construction of graphs $\widetilde{G}_{st}$ with $D = \{\{s_1, t_1\}, \{s_1, t_2\}, \{s_3, t_3\}\}$ for $L = 3$ and $k = 1$.

Note that $\widetilde{G}_{st}$ may have nodes different from $u \in N_{st} \cup N'_{st}$ with indegree or outdegree equal to zero. These nodes can be removed from $\widetilde{G}_{st}$ after its construction.

$\widetilde{G}_{st}$ contains four layers: $\{s\}, N_{st}, N'_{st}, \{t\}$ and no circuit. Also, any st-dipath in $\widetilde{G}_{st}$ is of length no more than 3:

- the length is equal to 1 if the st-dipath is composed of the single arc (s, t),
- the length is equal to 3 for both st-dipaths of the form (s, u, u', t) corresponding to path (s, u, t) of length 2 in G, and st-dipaths of the form (s, u, v', t) corresponding to path (s, u, v, t), with $u \neq v$, of length 3 in G.

Moreover, notice that in $\widetilde{G}_{st}$ there exist exactly $\min\{|[s, u]|, |[u, t]|, k\}$ arcs between two vertices (u, u'), for every $u \in V \setminus \{s, t\}$. If G is simple, that is, does not contain parallel edges, then $\min\{|[s, u]|, |[u, t]|, k\} \leq 1$, for every $u \in V \setminus \{s, t\}$, and $\min\{|[s, u]|, |[u, t]|, k\} \leq k$ for general graphs. If G is simple and complete, then $\min\{|[s, u]|, |[u, t]|, k\} = 1$, for every $u \in V \setminus \{s, t\}$. The reason for adding such a number of arcs between two vertices (u, u') in $\widetilde{G}_{st}$ is that, as mentioned before, an st-dipath containing arc (u, u') in $\widetilde{G}_{st}$ corresponds to a 2-st-path (s, u, t) going through vertex u in G. Moreover, the number of edge-disjoint 2-st-paths of G going through node u is less than or equal to $\min\{|[s, u]|, |[u, t]|\}$ (notice that G may contain parallel edges). As at most k edge-disjoint 2-st-paths are chosen in a solution, we may then take at most $\min\{|[s, u]|, |[u, t]|, k\}$ edge-disjoint 2-st-paths of G. Thus, $\widetilde{G}_{st}$ must contain at least $\min\{|[s, u]|, |[u, t]|, k\}$ arcs of the form (u, u'), for every $u \in V \setminus \{s, t\}$. Obviously, it suffices to have exactly $\min\{|[s, u]|, |[u, t]|, k\}$ arcs of the form (u, u') in $\widetilde{G}_{st}$, for every $u \in V \setminus \{s, t\}$.

Therefore, each L-st-path in G corresponds to an st-dipath in $\widetilde{G}_{st}$ and conversely. We have the following lemma, given without proof. For the proof, the reader can refer to [13].

Lemma 3.1. *Let $L \in \{2, 3\}$ and $\{s, t\} \in D$.*

(i). If two L-st-paths of G are edge-disjoint, then the corresponding two st-dipaths in $\widetilde{G}_{st}$ are arc-disjoint.
(ii). If two st-dipaths of $\widetilde{G}_{st}$ are arc-disjoint, then the corresponding two L-st-paths in G are edge-disjoint.

Note that each graph $\widetilde{G}_{st}$ contains $|\widetilde{V}_{st}| = 2|V| - 2$ $(= |N_{st} \cup N'_{st} \cup \{s, t\}|)$ nodes and $|\widetilde{A}_{st}| \leq |\delta(s)| + |\delta(t)| - |[s, t]| + k(|V| - 2)$ arcs if L = 2 and $|\widetilde{A}_{st}| \leq 2|E| - |\delta(s)| - |\delta(t)| + |[s, t]| + k(|V| - 2)$ arcs if L = 3, for all $\{s, t\} \in D$.

As a consequence of Lemma 3.1, for $L = 2$, 3 and every demand $\{s, t\} \in D$, a set of k edge-disjoint L-st-paths of G corresponds to a set of k arc-disjoint st-dipaths of $\widetilde{G}_{st}$, and k arc-disjoint st-dipaths of $\widetilde{G}_{st}$ correspond to k edge-disjoint L-st-paths of G. Therefore, we have the following corollary.

Corollary 3.1. *Let H be a subgraph of G and let $\widetilde{H}_{st}$, $\{s, t\} \in D$, be the subgraph of $\widetilde{G}_{st}$ obtained by considering all the arcs of $\widetilde{G}_{st}$ corresponding to an edge of H, plus the arcs of the form (u, u'), $u \in V \setminus \{s, t\}$. Then H induces a solution of the kHNDP if $\widetilde{H}_{st}$ contains k arc-disjoint st-dipaths, for every $\{s, t\} \in D$. Conversely, given a set of subgraphs $\widetilde{H}_{st}$ of $\widetilde{G}_{st}$, $\{s, t\} \in D$, if H is the subgraph of G obtained by considering all the edges of G associated with at least one arc in a subgraph $\widetilde{H}_{st}$, then H induces a solution of the kHNDP only if $\widetilde{H}_{st}$ contains k arc-disjoint st-dipaths, for every $\{s, t\} \in D$.*

Corollary 3.1 suggests at once the following flow-based formulation.

3.2. *Separated Flow Formulation*

Given a demand $\{s, t\}$, we let $f^{st} \in \mathbb{R}^{\widetilde{A}_{st}}$ be a flow vector on $\widetilde{G}_{st}$ of value k between s and t.

Then f^{st} satisfies the *flow conservation constraints* (3.1), given by

$$\sum_{a\in\delta^+(u)} f_a^{st} - \sum_{a\in\delta^-(u)} f_a^{st} = \left\{\begin{matrix} k & \text{if } u = s, \\ 0 & \text{if } u \in \widetilde{V}_{st}\setminus\{s,t\}, \\ -k & \text{if } u = t, \end{matrix}\right\}, \quad \text{for all } u \in \widetilde{V}_{st}, \{s,t\} \in D \tag{3.1}$$

and

$$f_a^{st} \le x(e), \text{ for all } a \in \widetilde{A}_{st}(e), e \in E, \{s,t\} \in D. \tag{3.2}$$

$$f_a^{st} \le 1, \text{ for all } a = (u,u'),\ u \in V\setminus\{s,t\},\ \{s,t\}\in D. \tag{3.3}$$

$$f_a^{st} \ge 0, \text{ for all } a \in \widetilde{A}_{st},\ \{s,t\} \in D. \tag{3.4}$$

$$x(e) \le 1, \text{ for all } e \in E. \tag{3.5}$$

Inequalities (3.2) are also called *linking inequalities*. They indicate that if an edge $e \in E$ is not in the solution, then the flow on every arc corresponding to e is 0. Inequalities (3.4)–(3.5) are the *trivial inequalities*.

Thus, we have the following theorem.

Theorem 3.1. *The kHNDP for L = 2, 3 is equivalent to the following integer program*

$$min\left\{cx;\ \textit{subject to}\ (3.1)\text{– }(3.5),\ x \in \mathbb{Z}_+^E, f^{st} \in \mathbb{Z}_+^{\widetilde{A}_{st}},\ \textit{for all}\ \{s,t\} \in D\right\}. \tag{3.6}$$

Formulation (3.6) will be called the separated flow formulation and will be denoted by $k\text{HNDP}^{\text{Sep}}_{\text{Flow}}$.

3.3. Separated Path Formulation

As the well-known work by Rardin and Choe [34], it is known that flows can also be modeled by paths. Every solution of the problem can, hence, be represented by directed st-paths in graphs $\widetilde{G}_{st}$, for all $\{s,t\} \in D$.

For each demand $\{s,t\} \in D$, let $\widetilde{\mathcal{P}}_{st}$ be the set of st-dipaths in $\widetilde{G}_{st}$ and, for each $P \in \widetilde{\mathcal{P}}_{st}$, let $\mu^{st}(P)$ be a binary variable whose value is 1 if P is used in a solution and 0 if not.

If an edge subset $F \subseteq E$ induces a solution of the kHNDP, then x^F and $(\mu^{st}(P),\ P \in \widetilde{\mathcal{P}}_{st}, \{s,t\} \in D)$ satisfy the following inequalities.

$$\sum_{P\in\widetilde{\mathcal{P}}_{st}} \mu^{st}(P) \ge k, \text{ for all } \{s,t\} \in D, \tag{3.7}$$

$$\sum_{P\in\widetilde{\mathcal{P}}_{st}, a\in P} \mu^{st}(P) \le x(e), \text{ for all } a \in \widetilde{A}_{st}(e),\ e \in E, \{s,t\} \in D, \tag{3.8}$$

$$\sum_{P\in\widetilde{P}_{st}, a\in P} \mu^{st}(P) \le 1, \text{ for all } a = (u,u'),\ u \in V\setminus\{s,t\},\ \{s,t\}\in D, \tag{3.9}$$

$$x(e) \le 1, \text{ for all edge } e \in E, \tag{3.10}$$

$$\mu^{st}(P) \ge 0, \text{ for all } P \in \widetilde{\mathcal{P}}_{st}, \{s,t\} \in D. \tag{3.11}$$

Inequalities (3.7) express the fact that the solution must contain at least k st-dipaths. Inequalities (3.8) and (3.9) indicate that these st-dipaths are arc-disjoint.

The following theorem gives an integer programming formulation for the kHNDP using the path-based model described above.

Theorem 3.2. *The kHNDP for L = 2, 3 is equivalent to the following integer program*

$$min\left\{cx;\ \textit{subject to}\ (3.7) - (3.11),\ x \in \mathbb{Z}_+^E, \mu^{st} \in \mathbb{Z}_+^{\widetilde{\mathcal{P}}_{st}},\ \textit{for all}\ \{s,t\} \in D\right\}. \tag{3.12}$$

Formulation (3.12) is called the separated path formulation and is denoted by $k\text{HNDP}^{\text{Sep}}_{\text{Path}}$. Note that for each demand $\{s,t\} \in D$, the number of st-paths in the graph $\widetilde{G}_{st}$ is bounded by $|V|^{L-1}$, which is polynomial for L = 2, 3. Thus, this formulation contains a polynomial number of variables while the number of nontrivial inequalities is at most

$$d + \sum_{\{s,t\}\in D} (|\delta(s)| + |\delta(t)| - |[s,t]|) + dk(|V| - 2) \quad \text{if } L = 2,$$

$$d + 2d|E| - \sum_{\{s,t\}\in D} (|\delta(s)| + |\delta(t)| - |[s,t]|) + dk(|V| - 2) \quad \text{if } L = 3,$$

which is polynomial.

Hence, as for the undirected path-based formulation $k\text{HNDP}^{U}_{\text{Path}}$, its linear relaxation can be solved in polynomial time using linear programming and by enumerating all the variables and constraints of the problem.

Still using Rardin and Choe [34], it can be shown that the separated flow and path formulations (3.6) and (3.12) provide the same LP-bound.

Also, one can easily observe that Formulation (3.12) is equivalent to Formulation (2.10), as L-st-paths in the original graph G correspond to st-dipaths in the directed graphs $\widetilde{G}_{st}$, for all $\{s,t\} \in D$, and vice versa. Thus, these formulations also produce the same LP-bound.

Proposition 3.1. *Formulation (3.12) and Formulation (2.10) are equivalent and produce the same LP-bound.*

3.4. Separated Cut Formulation

The previous two models include constraints guaranteeing that for each demand $\{s,t\} \in D$, there exists a flow of value

k under the arc capacities given by x. By the Max flow-Min cut theorem, such a flow exists if and only if the capacity of any st-dicut, in each graph $\widetilde{G}_{st}$, is at least k. This observation leads at once to the following formulation which provides the same LP bound as the previous separated flow and path formulations.

Let $H \subseteq E$ be an edge subset which induces a solution of the kHNDP in G and let $\widetilde{H}_{st}$ be the arc subset of $\widetilde{G}_{st}$, $\{s,t\} \in D$, corresponding to H. Then, the incidence vector x^H of H and the vectors $y_{st}^{\widetilde{H}_{st}}$, $\{s,t\} \in D$, satisfy the following inequalities.

$$y_{st}(\delta^+(\widetilde{W})) \geq k, \text{ for all } st-\text{dicut } \delta^+(\widetilde{W}) \text{ of } \widetilde{G}_{st}, \quad \text{for all } \{s,t\} \in D, \tag{3.13}$$

$$y_{st}(a) \leq x(e), \text{ for all } a \in \widetilde{A}(e),\ e \in E,\ \{s,t\} \in D, \tag{3.14}$$

$$y_{st}(a) \leq 1, \text{ for all } a = (u,u'), \quad \text{for all } u \in V \setminus \{s,t\},\ \{s,t\} \in D, \tag{3.15}$$

$$y_{st}(a) \geq 0, \text{ for all } a \in \widetilde{A}_{st},\ \{s,t\} \in D, \tag{3.16}$$

$$x(e) \leq 1, \text{ for all } e \in E. \tag{3.17}$$

Inequalities (3.13) will be called *directed st-cut inequalities* or *st-dicut inequalities* and inequalities (3.14) *linking inequalities*. Inequalities (3.14) indicate that an arc $a \in \widetilde{A}_{st}$ corresponding to an edge e is not in $\widetilde{H}_{st}$ if e is not taken in H. Inequalities (3.15)–(3.17) are the *trivial inequalities*.

We have the following result which is given without proof as it easily follows from the above discussion.

Theorem 3.3. *The kHNDP for $L=2, 3$ is equivalent to the following integer program*

$$\min \Big\{cx;\ \text{subject to } (3.13)-(3.17),\ x \in \mathbb{Z}_+^E,\ y_{st} \in \mathbb{Z}_+^{\widetilde{A}_{st}}, \quad \text{for all } \{s,t\} \in D\Big\}. \tag{3.18}$$

This formulation is called the separated cut formulation and is denoted by $k\text{HNDP}^{\text{Sep}}_{\text{Cut}}$. It contains a polynomial number of variables. Indeed, for $L=2$, the number of variables is

$$|E| + \sum_{\{s,t\}\in D} |\widetilde{A}_{st}| \leq |E| + \sum_{\{s,t\}\in D} (|\delta(s)| + |\delta(t)| - |[s,t]|) + dk(|V|-2),$$

and for $L=3$, it is

$$|E| + \sum_{\{s,t\}\in D} |\widetilde{A}_{st}| \leq |E| + 2d|E| - \sum_{\{s,t\}\in D} (|\delta(s)| + |\delta(t)| - |[s,t]|) + dk(|V|-2)$$

(recall that $d = |D|$).

However, the number of constraints is exponential as the number of directed st-cuts is exponential in the size of $\widetilde{G}_{st}$, for all $\{s,t\} \in D$. As we will see in Section 6, its linear programming relaxation can be solved in polynomial time using a cutting plane algorithm.

In the next section, we introduce a further formulation for the kHNDP also based on directed graphs. However, unlike the separated formulations, this formulation is supported by only one directed graph.

4. AGGREGATED FORMULATION FOR THE *k*HNDP

We denote by S_D and T_D respectively the sets of source and destination nodes of D. In the case where two demands $\{s_1,t_1\}$ and $\{s_2,t_2\}$ are such that $s_1 = t_2 = s$, we keep a copy of s in both S_D and T_D.

In this section, we will introduce a new formulation for the kHNDP which is supported by a single directed graph $\widetilde{G} = (\widetilde{V}, \widetilde{A})$ obtained from G as follows:

- $\widetilde{V} = S_D \cup N' \cup N'' \cup T_D$ with N' and N'' two copies of V; we denote by u' and u'' the nodes of N' and N'' corresponding to a node $u \in V$.
- $\widetilde{A}$ contains 6 kinds of arcs:
  1. for each demand $\{s,t\} \in D$
     - if $st \in E$, we add in $\widetilde{A}$ the arc (s,t'),
     - if $su \in E$ with $u \in V \setminus \{s,t\}$, we add an arc (s,u'),
     - if $vt \in E$ with $v \in V \setminus \{s,t\}$, we add an arc (v'',t),
  2. for each node $u \in V$, we add $\max_{\{s,t\}\in D}\{\min\{|[s,u]|, |[u,t]|, k\}\}$ arcs of the form (u',u''),
  3. for each $t \in T_D$, we add $\min\{k, \max\{|[s,t]|,\ s \in S_D$ with $\{s,t\} \in D\}\}$ arcs of the form (t',t),
  4. if $L=3$, for each edge $e = uv \in E$, we add two arcs (u',v'') and (v',u'').

Figures 4 and 5 show examples for $L=2$ and $L=3$ with $k=1$.

Notice that the digraph $\widetilde{G}$ may have nodes $u \in N \cup N'$ with indegree or outdegree equal to zero. These nodes can be removed from $\widetilde{G}$ after its construction. Also, note that when G is simple (that is with no parallel edges), $|[u',u'']^+| = |[t',t]^+| \leq 1$, for every $u \in V$ and every $t \in T_D$, and $|[u',u'']^+| = |[t',t]^+| = 1$ if G is complete.

$\widetilde{G}$ contains $|\widetilde{V}| = 2|V| + |S_D| + |T_D|$ nodes and $|\widetilde{A}| \leq k|V| + \sum_{s\in S} |\delta(s)| + \sum_{t\in T} |\delta(t)|$ arcs if $L=2$ and $|\widetilde{A}| \leq 2|E| + k|V| + \sum_{s\in S} |\delta(s)| + \sum_{t\in T} |\delta(t)|$ arcs if $L=3$.

If $\widetilde{G} = (\widetilde{V}, \widetilde{A})$ is the digraph associated with G, then for an edge $e \in E$, we denote by $\widetilde{A}(e)$ the set of arcs of $\widetilde{G}$ corresponding to e.

Observe that $\widetilde{G}$ is acyclic. Also note that for a given demand $\{s,t\} \in D$, every st-dipath in $\widetilde{G}$ contains at most 3 arcs. An L-st-path $P = (s,u,v,t)$ of G, where u and v may be the same, corresponds to an st-dipath $\widetilde{P} = (s,u',v'',t)$ in $\widetilde{G}$. Conversely, every st-dipath $\widetilde{P} = (s,u',v'',t)$ of $\widetilde{G}$, where u' and v'' may correspond to the same node of V, corresponds to an L-st-path $P = (s,u,v,t)$, where u and v may be the same. Moreover $\widetilde{G}$ does not contain any arc of the form (s,s') and (t'',t), for every $s \in S_D$ and $t \in T_D$. If a node $t \in T_D$ appears in exactly one demand $\{s,\ t\}$, then $[s'',t] = \emptyset$. In the

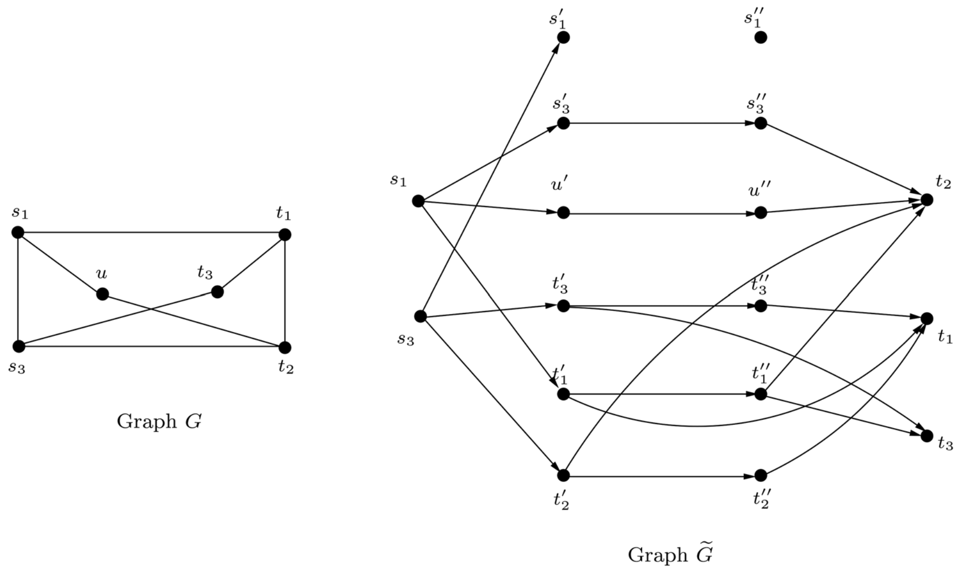

FIG. 4. Construction of graph $\widetilde{G}$ with $D = \{\{s_1, t_1\}, \{s_1, t_2\}, \{s_3, t_3\}\}$, $L = 2$ and $k = 1$.

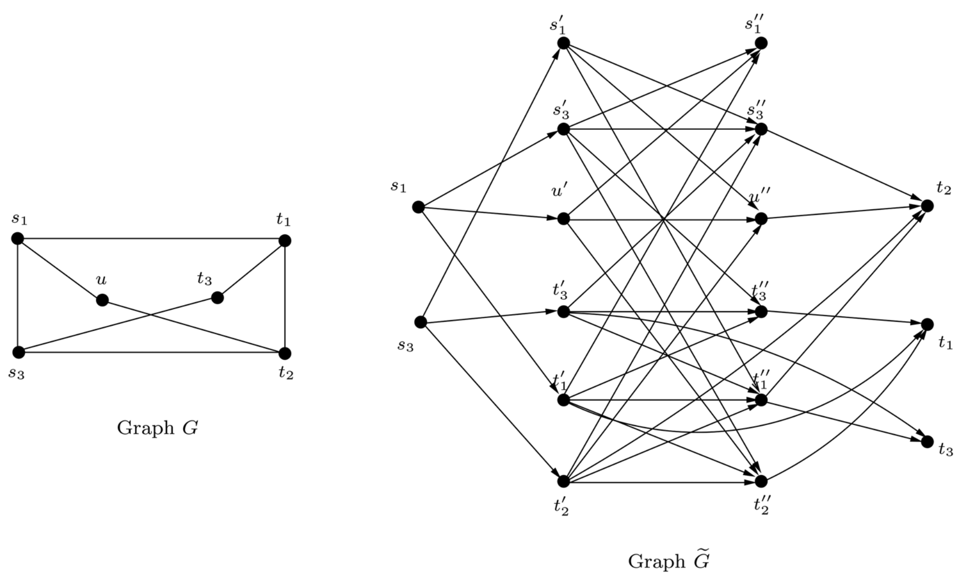

FIG. 5. Construction of graph $\widetilde{G}$ with $D = \{\{s_1, t_1\}, \{s_1, t_2\}, \{s_3, t_3\}\}$, $L = 3$ and $k = 1$.

remainder of this section we will suppose w.l.o.g. that each node of T_D is involved, as destination, in only one demand. In fact, in general, if a node $t \in T_D$ is involved, as destination, in more than one demand, say $\{s_1, t\}, \ldots, \{s_p, t\}$, with $p \geq 2$, then one may replace in T_D t by p nodes $t_1, \ldots, t_p$ and in D each demand $\{s_i, t\}$ by $\{s_i, t_i\}$, $i = 1, \ldots, p$.

We have the following result, given without proof. For a complete proof, the reader can refer to [13].

Lemma 4.1. *Let $L \in \{2, 3\}$. If each node $t \in T_D$ appears in exactly one demand, then for every $\{s, t\} \in D$,*

(i). if two L-st-paths of G are edge-disjoint, then the corresponding st-dipaths of $\widetilde{G}$ are arc-disjoint.
(ii). if two st-dipaths of $\widetilde{G}$ are arc-disjoint, then the corresponding st-paths in G contain two edge-disjoint L-st-paths.

As a consequence of Lemma 4.1, the graph G contains k edge-disjoint L-st-paths for a demand $\{s, t\}$ if and only if $\widetilde{G}$ contains at least k arc-disjoint st-dipaths. Thus, each solution of the kHNDP in G corresponds to a solution of the so-called Survivable Directed Network Design Problem (kSNDP for short) in $\widetilde{G}$ with demand set D. This latter problem consists in finding in a directed graph, and for a given set of demands, a subgraph containing k arc-disjoint st-dipaths for all demands $\{s, t\}$. Hence, we have the following corollary.

Corollary 4.1. *Let H be a subgraph of G and let $\widetilde{H}$ be the subgraph of $\widetilde{G}$ obtained by considering all the arcs of $\widetilde{G}$ corresponding to the edges of H together with the arcs of the form (u', u''), $u \in V$, and (t', t), for every $t \in T_D$. Then H induces a solution of the kHNDP if $\widetilde{H}$ is a solution of the Survivable Directed Network Design Problem (kSNDP). Conversely, if*

$\widetilde{H}$ is a subgraph of $\widetilde{G}$ and H is the subgraph of G obtained by considering all the edges which correspond to at least one arc of $\widetilde{H}$, then H induces a solution of the kHNDP only if $\widetilde{H}$ is a solution of the kSNDP.

By Menger's Theorem, $\widetilde{G}$ contains k arc-disjoint st-dipaths if and only if every st-dicut of $\widetilde{G}$ contains at least k arcs. If $F \subseteq E$ is a set of edges of G that induces a solution of the kHNDP, then $x \in \mathbb{R}^E$ and $y \in \mathbb{R}_+^{\widetilde{A}}$ satisfy the following inequalities

$$y(\delta^+(\widetilde{W})) \geq k, \text{ for all } st-\text{dicut } \delta^+(\widetilde{W}), \{s,t\} \in D, \quad (4.1)$$

$$y(a) \leq x(e), \text{ for all } a \in \widetilde{A}(e), e \in E, \quad (4.2)$$

$$y(a) \geq 0, \text{ for all } a \in \widetilde{A}, \quad (4.3)$$

$$x(e) \leq 1, \text{ for all } e \in E. \quad (4.4)$$

We have the following theorem, which easily follows from Corollary 4.1.

Theorem 4.1. *The kHNDP for $L = 2, 3$ is equivalent to the following integer program*

$$min \left\{cx;\ subject\ to\ (4.1)\text{–}\ (4.4),\ x \in \mathbb{Z}_+^E, y \in \mathbb{Z}_+^{\widetilde{A}}\right\}. \quad (4.5)$$

Formulation (4.5) will be called the aggregated formulation and is denoted by $k\text{HNDP}_{Ag}$. Inequalities (4.1) will be called directed st-cut inequalities or st-dicut inequalities and (4.2) will be called linking inequalities. The latter inequalities indicate that an arc a, corresponding to an edge e, is not chosen in the solution of kSNDP if e is not chosen in the solution of kHNDP.

This formulation contains $|E| + |\widetilde{A}| \leq |E| + k|V| + \sum_{s \in S_D} |\delta(s)| + \sum_{t \in T_D} |\delta(t)|$ variables if $L = 2$ and $|E| + |\widetilde{A}| \leq 3|E| + k|V| + \sum_{s \in S_D} |\delta(s)| + \sum_{t \in T_D} |\delta(t)|$ variables if $L = 3$. The number of constraints is exponential as the st-dicuts are exponential in number. But, as it will turn out, the separation problem for inequalities (4.1) can be solved in polynomial time and, hence, the linear programming relaxation of (4.5) is also.

Notice that the graph transformations introduced in this section do not work for $L \geq 4$. In fact, first the natural formulation is no longer valid for $L \geq 4$. As shown by Huygens and Mahjoub [29], further valid inequalities are required when $L = 4$ to have a natural formulation. Also, for $L \geq 5$, such a formulation is still unknown. Conversely, these graph transformations are valid only for $L = 2, 3$ and cannot be extended to the case $L \geq 4$. Indeed, as it has been seen above, each L-st-path in the original graph is transformed into a dipath in the transformed graph. The kHNDP then reduces to a Max flow-Min cut problem when restricted to a single demand, which is no longer valid when $L \geq 4$. Also, the reduction to a Max flow-Min cut problem does not hold for other graph transformations like that proposed by [5].

In the next section, we present a comparative study of the different formulations presented in the last section. In particular, we will show that the linear programming relaxation of these formulations are as strong as the linear programming relaxation of the natural formulation.

5. COMPARISON STUDY BETWEEN THE DIFFERENT FORMULATIONS

In this section, we compare the different formulations we have introduced before. We first focus on the number of variables of the different extended formulations. As noticed before, the undirected flow and separated flow formulations as well as the undirected and separated path formulations have a polynomial number of variables and constraints. However, the aggregated graph contains fewer arcs than occur in the union of the separated graphs. Thus, the aggregated formulation contains fewer variables than the separated and undirected formulations.

Now, we compare the formulations in terms of LP-bound. In fact, we will show that the extended formulations (undirected flow and path, separated and aggregated formulations) produce the same LP-bound, and this LP-bound is the same as that of the natural formulation.

First, we compare the separated formulations. In fact, as mentioned in Section 3.2, by the Max flow-Min cut Theorem, the separated flow and cut formulations produce the same LP-bound. Also, by Rardin and Choe [34], the separated flow and path formulations are equivalent, and, hence, give the same LP-bound. Moreover, as explained in Section 3.3, Formulations (3.12) and (2.10) are equivalent and, hence, give the same LP-bound. Therefore, we have the following theorem.

Theorem 5.1. *The linear programming relaxations of Formulations (3.6, 3.12, 3.18), and (2.10) have the same optimal value.*

Now we compare the undirected flow and path formulations. In fact, using the same argument as in [10] *(see the proof of Proposition* 2.2) *we can easily show that the undirected flow formulation can be obtained by projecting the undirected path formulation. For a detailed proof, the reader can also refer to* [13].

Theorem 5.2. *The undirected flow formulation can be obtained by projection of the undirected path formulation.*

Theorem 5.2 implies that the linear programming relaxation of Formulations (2.20) and (2.10) have the same optimal value, and, hence, produce the same LP-bound for the kHNDP when $L = 2, 3$.

Now we turn our attention to the natural formulation (2.5). We are going to show that the linear programming relaxation of the natural formulation has the same value as that of the separated cut formulation. To this end, we first introduce a procedure which associates with every st-cut and L-st-path-cut of G an st-dicut of $\widetilde{G}_{st}$, for every demand $\{s,t\} \in D$. This procedure, called *Procedure A*, produces, from an edge set $C \subseteq E$ and a demand $\{s,t\} \in D$, an arc subset $\widetilde{C}$ of $\widetilde{G}_{st}$ obtained as follows.

Procedure A.

(i). For an edge $st \in C$, add the arc (s, t) in $\widetilde{C}$;
(ii). for an edge $su \in C$, add the arc (s, u) in $\widetilde{C}$, $u \in N_{st}$;
(iii). for an edge $vt \in C$, add the arc (v', t) in $\widetilde{C}$, $v' \in N'_{st}$;
(iv). for an edge $uv \in C$, $u \neq v$, $u, v \in V \setminus \{s, t\}$,
  (iv.a). if $su \in C$ or $vt \in C$, then add (v, u') in $\widetilde{C}$, with $v \in N_{st}$ and $u' \in N'_{st}$;
  (iv.b). if $su \notin C$ and $vt \notin C$, then add the arc (u, v') in $\widetilde{C}$.

Note that each arc of $\widetilde{C}$ corresponds to a unique edge of C and vice versa.

Also, observe that the arc set $\widetilde{C}$ does not contain any arc of the form (u, u') with $u \in N_{st}$ and $u' \in N'_{st}$. Also note that $\widetilde{C}$ does not contain at the same time two arcs (u, v') and (v, u'), for an edge $uv \in E$ with $u, v \in V \setminus \{s, t\}$.

Conversely, an arc subset $\widetilde{C}$ of $\widetilde{A}_{st}$ can be obtained from an edge set $C \subseteq E$, using Procedure A, if $\widetilde{C}$ does not contain simultaneously two arcs (u, v') and (v, u'), $u, v \in N_{st}$, $u', v' \in N'_{st}$, and does not contain any arc of the form (u, u') with $u \in N_{st}$, $u' \in N'_{st}$.

Before going further, we give the following two lemmas whose proof can be found in [4].

Lemma 5.1. *Let $L = 2, 3$, $\{s, t\} \in D$ and let $C \subseteq E$ be an edge set of G which is an st-cut or an L-st-path-cut induced by a partition $(V_0, \ldots, V_{L+1})$ such that $|V_0| = |V_{L+1}| = 1$. Then, the arc set obtained from C by Procedure A is an st-dicut of $\widetilde{G}_{st}$.*

Proof. See the proof of Lemma 4.1 in [4]. ∎

Lemma 5.2. *Let $\bar{x}$ be a solution of the linear programming relaxation of kHNDP$_{Nat}$ and, for all $\{s, t\} \in D$, let $\bar{y}_{st} \in \mathbb{R}^{\widetilde{A}_{st}}$ be the vector obtained from $\bar{x}$ by*

$$\bar{y}_{st}(a) = \begin{cases} \bar{x}(su) & \text{if } a \text{ is of the form } (s,u),\ u \in N_{st}, \\ \bar{x}(vt) & \text{if } a \text{ is of the form } (v',t),\ v' \in N'_{st}, \\ \bar{x}(uv) & \text{if } a \text{ is of the form } (u,v') or (v',u), \\ & u,v \in N_{st},\ u',v' \in N'_{st},\ u \neq v,\ u' \neq v', \\ \bar{x}(st) & \text{if } a \text{ is of the form } (s,t), \\ 1 & \text{if } a \text{ is of the form } (u,u'),\ u \in N_{st},\ u' \in N'_{st}. \end{cases}$$

Given a demand $\{s, t\} \in D$, let $\widetilde{C}$ be an st-dicut of $\widetilde{G}_{st}$ such that $\widetilde{C}$ does not contain an arc of the form (u, u'), $u \in V \setminus \{s, t\}$. Then, there exists an st-cut or an L-st-path-cut $C \subseteq E$ in G such that $\bar{x}(C) \leq \bar{y}_{st}(\widetilde{C})$.

Proof. See the proof of Lemma 4.2 in [4]. ∎

We also give the following lemma which will be useful in the remainder of this section.

Lemma 5.3. *Let $\bar{x}$ be a solution of the linear programming relaxation of kHNDP$_{Nat}$ and, for all $\{s, t\} \in D$, let $\bar{y}_{st}$ be the vector of $\mathbb{R}^{\widetilde{A}_{st}}$ described in Lemma 5.2 and associated with $\bar{x}$, for all $\{s, t\} \in D$. Then, for every $\{s, t\} \in D$, every st-dicut $\widetilde{C}$ of $\widetilde{G}_{st}$ which contains an arc of the form (u, u') has a weight $\bar{y}_{st}(\widetilde{C}) \geq k$ if every st-dicut $\widetilde{C}'$ of $\widetilde{G}_{st}$ which does not contain an arc of the form (u, u') is such that $\bar{y}_{st}(\widetilde{C}') \geq k$.*

Proof. Consider a demand $\{s, t\} \in D$ and an st-dicut $\widetilde{C} = \delta^+_{\widetilde{G}_{st}}(\widetilde{W})$ of $\widetilde{G}_{st}$ which contains at least one arc of the form (u, u'), for some $u \in V \setminus \{s, t\}$. Notice, that if $\widetilde{C}$ contains one arc of the form (u, u'), then $[u, u']^+ \subseteq \widetilde{C}$. Moreover, by the construction of $\widetilde{G}_{st}$, $|[u, u']^+| = \min\{|[s, u]|, |[u, t]|, k\}$.

Now to prove the lemma, we show that either $|[u, u']^+ \cap \widetilde{C}| = k$ or there exists an st-dicut of $\widetilde{G}_{st}$, not containing an arc of the form (u, u') and whose weight, with respect to $\bar{y}_{st}$, is lower than that of $\widetilde{C}$.

Thus, suppose first that $|[u, u']^+ \cap \widetilde{C}| = k$, for some $u \in V \setminus \{s, t\}$ Then, as $\bar{y}_{st}(a) = 1$, for all $a \in [u, u']^+$, $u \in V \setminus \{s, t\}$, we have $\bar{y}_{st}(\widetilde{C}) \geq \bar{y}_{st}([u, u']^+) = |[u, u']^+| = k$, and the result holds.

Now suppose $|[u, u']^+| < k$, that is, $|[u, u']^+| = \min\{|[s, u]|, |[u, t]|, k\} = \min\{|[s, u]|, |[u, t]|\}$, for all $u \in V \setminus \{s, t\}$ with $[u, u']^+ \subseteq \widetilde{C}$. We iteratively build an arc set whose weight is lower than that of $\widetilde{C}$ and which does not contain an arc of the form (u, u'), for every $u \in V \setminus \{s, t\}$. For this, we let $u \in V \setminus \{s, t\}$ be a node such that $[u, u']^+ \subseteq \widetilde{C}$ and build a node set $\widetilde{W}'$ in the following way.

If $\min\{|[s, u]|, |[u, t]|\} = |[s, u]|$, that is $|[u, u']^+| = |[s, u]| = |[s, u]^+|$, then let $\widetilde{W}' = \widetilde{W} \setminus \{u\}$. In this case, we have

$$\bar{y}_{st}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W}')) = \bar{y}_{st}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W})) - \bar{y}_{st}([u, u']^+) - \bar{y}_{st}([u, N'_{st} \setminus \{u'\}]^+ \cap \widetilde{C}) + \bar{y}_{st}([s, u]^+).$$

As by assumption, $|[u, u']^+| = |[s, u]| = |[s, u]^+|$ and as $\bar{y}_{st}(a) \leq 1$, for all $a \in \widetilde{A}_{st}$, we have $\bar{y}_{st}([s, u]^+) \leq |[s, u]^+| = |[u, u']^+| = \bar{y}_{st}([u, u']^+)$. Thus,

$$\bar{y}_{st}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W}')) \leq \bar{y}_{st}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W})).$$

Now if $\min\{|[s, u]|, |[u, t]|\} = |[u, t]|$, that is $|[u, u']^+| = |[u, t]| = |[u, t]^+|$, then let $\widetilde{W}' = \widetilde{W} \cup \{u'\}$. In this case, we have

$$\bar{y}_{st}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W}')) = \bar{y}_{st}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W})) - \bar{y}_{st}([u, u']^+) - \bar{y}_{st}([N_{st} \setminus \{u\}, u']^+ \cap \widetilde{C}) + \bar{y}_{st}([u, t]).$$

As by assumption, $|[u, u']^+| = |[u, t]| = |[u, t]^+|$ and as $\bar{y}_{st}(a) \leq 1$, for all $a \in \widetilde{A}_{st}$, we have $\bar{y}_{st}([u, t]^+) \leq |[u, t]^+| = |[u, u']^+| = \bar{y}_{st}([u, u']^+)$. Thus,

$$\bar{y}_{st}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W}')) \leq \bar{y}_{st}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W})).$$

Therefore, the node set $\widetilde{W}'$ obtained by the above procedure is such that $\bar{y}_{st}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W}')) \leq \bar{y}_{st}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W}))$. Moreover, $[u, u']^+ \cap \delta^+_{\widetilde{G}_{st}}(\widetilde{W}') = \emptyset$. By repeating this procedure, one obtains a node set, say $\widetilde{W}'' \subset \widetilde{V}_{st}$, such that $\delta^+_{\widetilde{G}_{st}}(\widetilde{W}'')$ does not contain an arc of the form (u, u'), for every $u \in V \setminus \{s, t\}$. Notice that the node set $\widetilde{W}''$ is obtained in at most $|V| - 1$ iterations. Moreover,

$$\overline{y}_{st}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W}'')) \leq \overline{y}_{st}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W})). \tag{5.1}$$

Finally, if the weight of every st-dicut not containing an arc of the form (u, u') is greater than or equal to k, then $\overline{y}_{st}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W}'')) \geq k$. This, together with inequality (5.1), yield $\overline{y}_{st}(\delta^+_{\widetilde{G}_{st}}(\widetilde{W})) \geq k$, which ends the proof of the lemma. ∎

Now we give the following theorem which shows that the linear programming relaxation of the natural and separated cut formulations have the same values.

Theorem 5.3. *Let $\overline{Z}_{Nat}$ and $\overline{Z}^{Sep}_{Cut}$ denote respectively the optimal values of the linear programming relaxation of the natural and separated cut formulations. Then, $\overline{Z}_{Nat} = \overline{Z}^{Sep}_{Cut}$.*

Proof. First, we show that $\overline{Z}_{\text{Nat}} \leq \overline{Z}^{\text{Sep}}_{\text{Cut}}$. To do this, we consider an optimal solution $\overline{\Gamma} = (\overline{x}, \overline{y}_{s_1t_1}, \ldots, \overline{y}_{s_dt_d})$ of the linear programming relaxation of $k\text{HNDP}^{\text{Sep}}_{\text{Cut}}$. We are going to show that $\overline{x}$ also induces a solution of $k\text{HNDP}_{Nat}$. For this, let $\{s, t\} \in D$ and $C \subseteq E$ be an st-cut or an L-st-path-cut induced by a partition $(V_0, \ldots, V_{L+1})$, with $|V_0| = |V_{L+1}| = 1$, and let $\widetilde{C} \subseteq \widetilde{A}_{st}$ be the arc set of $\widetilde{G}_{st}$ obtained from C and $\{s, t\}$ by the application of Procedure A. By Lemma 5.1, $\widetilde{C}$ is an st-dicut of $\widetilde{G}_{st}$. As each arc of $\widetilde{C}$ corresponds to a unique edge of C and vice versa, and as $\overline{y}_{st}(a) \leq \overline{x}(e)$, for all $a \in \widetilde{A}_{st}(e)$, $e \in E$, we have $\overline{x}(C) \geq \overline{y}(\widetilde{C})$. As $\overline{\Gamma}$ is a solution of the $k\text{HNDP}^{\text{Sep}}_{\text{Cut}}$ and, hence, $\overline{y}_{st}$ satisfies the st-dicut inequalities, we get $\overline{x}(C) \geq k$. This implies that $\overline{x}$ induces a solution of the linear programming relaxation of $k\text{HNDP}_{Nat}$ yielding $\overline{Z}_{\text{Nat}} \leq \overline{Z}^{\text{Sep}}_{\text{Cut}}$.

Now we show that $\overline{Z}_{\text{Nat}} \geq \overline{Z}^{\text{Sep}}_{\text{Cut}}$. For this, we consider an optimal solution $\overline{x}$ of the linear programming relaxation of $k\text{HNDP}_{Nat}$. Let $\overline{y}_{st}$ be the vector of $\mathbb{R}^{\widetilde{A}_{st}}$ described in Lemma 5.2 associated with $\overline{x}$, for all $\{s, t\} \in D$. We will show in the following that $\overline{\Gamma} = (\overline{x}, \overline{y}_{s_1t_1}, \ldots, \overline{y}_{s_dt_d})$ induces a solution of $k\text{HNDP}^{\text{Sep}}_{\text{Cut}}$. To do this, we consider an st-dicut $\widetilde{C}$ of $\widetilde{G}_{st}$, for a given demand $\{s, t\} \in D$. First, if $\widetilde{C}$ does not contain any arc of the form (u, u'), then by Lemma 5.2, one can obtain an st-cut or an L-st-path-cut C of G with $\overline{x}(C) \leq \overline{y}_{st}(\widetilde{C})$. Clearly, as $\overline{x}$ is solution of $k\text{HNDP}_{Nat}$, we have $k \leq \overline{x}(C)$ and, hence, $\overline{y}_{st}(\widetilde{C}) \geq k$. Thus, every st-dicut $\widetilde{C}$ not containing arcs of the form (u, u') satisfies $\overline{y}_{st}(\widetilde{C}) \geq k$. Therefore, by Lemma 5.3, every st-dicut $\widetilde{C}$ containing an arc of the form (u, u') is such that $\overline{y}_{st}(\widetilde{C}) \geq k$.

Thus, every st-dicut $\widetilde{C}$ of $\widetilde{G}_{st}$ satisfies $\overline{y}_{st}(\widetilde{C}) \geq k$, for every $\{s, t\} \in D$. Also, it is not hard to see that $\overline{\Gamma}$ satisfies inequalities (3.14) and (3.15), and, hence, induces a solution of the linear programming relaxation of $k\text{HNDP}^{\text{Sep}}_{\text{Cut}}$ with cost $\overline{Z}_{\text{Nat}}$. Therefore, we get $\overline{Z}_{\text{Nat}} \geq \overline{Z}^{\text{Sep}}_{\text{Cut}}$, which ends the proof of the theorem. ∎

Next, we compare the linear programming relaxation of the aggregated formulation (4.5) and the natural formulation (3.18). But first, we introduce a procedure, called Procedure B, which transforms an edge set $C \subseteq E$ to an arc set $\widetilde{C}$ of $\widetilde{G}$. Let $C \subseteq E$ and $\{s, t\} \in D$, and let $\widetilde{C}$ be the arc set of $\widetilde{G}$ obtained using the following procedure.

Procedure B.

(i). For an edge $st \in C$, add the arc (s, t') in $\widetilde{C}$;
(ii). for an edge $su \in C$, add the arc (s, u') in $\widetilde{C}$, $u' \in N'$;
(iii). for an edge $vt \in C$, add the arc (v'', t) in $\widetilde{C}$, $v'' \in N''$;
(iv). for an edge $uv \in C$, $u \neq v$, $u, v \in V \setminus \{s, t\}$,
  (iv.a). if $su \in C$ or $vt \in C$, then add (v', u'') in $\widetilde{C}$, with $v' \in N'$ and $u'' \in N''$;
  (iv.b). if $su \notin C$ and $vt \notin C$, then add the arc (u', v'') in $\widetilde{C}$.

Observe that $\widetilde{C}$ does not contain any arc of the form (u', u'') with $u' \in N'$ and $u'' \in N''$, or of the form (t', t) for $t \in T_D$. Also note that $\widetilde{C}$ does not contain at the same time two arcs corresponding to the same edge of G.

Conversely, an arc subset $\widetilde{C}$ of $\widetilde{A}$ can be obtained by Procedure B from an edge set $C \subseteq E$ if $\widetilde{C}$ does not contain simultaneously two arcs corresponding to the same edge of G, and any arc of the form (u', u'') with $u' \in N'$, $u'' \in N''$ or (t', t), $t \in T_D$.

We have the following two lemmas.

Lemma 5.4. *Let $(\overline{x}, \overline{y})$ be a solution of the linear programming relaxation of Formulation (4.5). Let $C \subseteq E$ be an edge set of G which is an st-cut or a L-st-path-cut induced by a partition $(V_0, \ldots, V_{L+1})$ such that $|V_0| = |V_{L+1}| = 1$, with $L \in \{2, 3\}$. Then the arc set obtained from C and {s, t} by Procedure B is an st-dicut of $\widetilde{G}$.*

Proof. See the proof of Lemma 4.1 in [4]. ∎

Lemma 5.5. *Let $\overline{x}$ be a solution of the linear programming relaxation of kHNDP$_{Nat}$ and let $\overline{y} \in \mathbb{R}^{\widetilde{A}}$ be the vector obtained from $\overline{x}$ as*

$$\overline{y}(a) = \begin{cases} \overline{x}(e) & \text{if } a \in \widetilde{A}(e), \text{ for all } e \in E, \\ 1 & \text{if } a \text{ is of the form } (u', u''),\ u' \in N',\ u'' \in N'', \\ & \text{or of the form } (t', t),\ t \in T_D. \end{cases}$$

Given a demand $\{s, t\} \in D$, let $\widetilde{C}$ be an st-dicut of $\widetilde{G}$ such that $\widetilde{C}$ does not contain any arc of the form (u', u''), $u \in V$, or of the form (t', t), $t \in T_D$. Then, there exists an st-cut or an L-st-path-cut $C \subseteq E$ in G such that $\overline{x}(C) \leq \overline{y}(\widetilde{C})$.

Proof. The proof is similar to that of Lemma 4.2 in [4]. ∎

Also, we compare the aggregated formulation with the natural formulation, in terms of linear programming relaxation. We show that their linear programming relaxations also have the same value.

Theorem 5.4. *Let $\overline{Z}_{Nat}$ and $\overline{Z}_{Ag}$ denote respectively the optimal values of the linear programming relaxations of the natural and aggregated formulations. Then $\overline{Z}_{Nat} = \overline{Z}_{Ag}$.*

Proof. We will show first that $\overline{Z}_{\text{Ag}} \geq \overline{Z}_{\text{Nat}}$. For this, we will consider an optimal solution $(\overline{x}, \overline{y})$ of the linear programming relaxation of the kHNDP$_{Ag}$ and show that $\overline{x}$ induces a solution of the linear programming relaxation of the kHNDP$_{Nat}$. Let $\{s,t\} \in D$, and let $C \subseteq E$ be an st-cut or an L-st-path-cut of G induced by a partition $(V_0, \ldots, V_{L+1})$, with $|V_0| = |V_{L+1}| = 1$. Also let $\widetilde{C}$ be the arc set obtained from C and $\{s,t\}$ by application of Procedure B. By Lemma 5.4, the arc set $\widetilde{C}$ induces an st-dicut of $\widetilde{G}$. As each arc of $\widetilde{C}$ corresponds to a unique edge of C and vice versa, and as $\overline{y}(a) \leq \overline{x}(e)$, for all $a \in \widetilde{A}(e)$, $e \in E$, we have $\overline{x}(C) \geq \overline{y}(\widetilde{C})$. Moreover, $\overline{y}$ satisfies all the st-dicut inequalities. Thus, we have $\overline{x}(C) \geq \overline{y}(\widetilde{C}) \geq k$, and $\overline{x}$ induces a solution of the linear programming relaxation of kHNDP$_{Nat}$, yielding $\overline{Z}_{\text{Ag}} \geq \overline{Z}_{\text{Nat}}$.

Now, we are going to show that $\overline{Z}_{\text{Nat}} \geq \overline{Z}_{\text{Ag}}$. To this end, we will show that an optimal solution $\overline{x}$ of the linear programming relaxation of kHNDP$_{Nat}$ induces a solution of the linear programming relaxation of the kHNDP$_{Ag}$ with cost $\overline{Z}_{\text{Nat}}$, implying that $\overline{Z}_{\text{Nat}} \geq \overline{Z}_{\text{Ag}}$. Let $\overline{y}$ be the vector of $\mathbb{R}^{\widetilde{A}}$ obtained from $\overline{x}$ as described in Lemma 5.5, and let $\overline{\Gamma} = (\overline{x}, \overline{y})$. We claim that $\overline{y}$ satisfies all the st-dicut inequalities (4.1). To prove this, we consider an st-dicut $\widetilde{C} = \delta^+(\widetilde{W})$ of $\widetilde{G}$ and distinguish four cases. W.l.o.g., we will suppose that $\widetilde{W} \cap S_D = \{s\}$ and $(\widetilde{V} \setminus \widetilde{W}) \cap T_D = \{t\}$. Otherwise, one can easily observe that the st-dicut inequality induced by $\widetilde{C}$ is redundant with respect to that induced by the node set $(\widetilde{W} \setminus (S_D \setminus \{s\})) \cup (T_D \setminus \{t\})$. ∎

Case 1. If $\widetilde{C}$ does not contain any arc of the form (u', u''), $u \in V$, or of the form (t', t), $t \in T_D$, and does not contain simultaneously two arcs corresponding to the same edge, then $\widetilde{C}$ can be obtained by application of Procedure B for an edge set $C \subseteq E$. From Lemma 5.5, the edge set C is either an st-cut or an L-st-path-cut, and $\overline{x}(C) \leq \overline{y}(C)$. As $\overline{x}$ is a solution of kHNDP$_{Nat}$ and, hence, $\overline{x}(C) \geq k$, we also have $\overline{y}(\widetilde{C}) \geq k$.

Case 2. If $\widetilde{C}$ does not contain any arc of the form (u', u''), $u \in V$, or of the form (t', t), $t \in T_D$, but contains two arcs corresponding to the same edge, then, as $\widetilde{C}$ is an st-dicut, these two arcs are either (s, u') and (s', u'') or (v', t'') and (v'', t), for some edge su or vt, with $u, v \in V \setminus \{s,t\}$. If $\widetilde{C}$ contains two arcs (s, u') and (s', u''), then $\{s, s'\} \subseteq \widetilde{W}$, and $\widetilde{W}' = \widetilde{W} \setminus \{s'\}$ induces an st-dicut. As by construction of $\widetilde{G}$, $[s, s']^+ = \emptyset$, we have $\delta^+(\widetilde{W}') = \widetilde{C} \setminus \{(s', u'')\}$. If $\widetilde{C}$ contains two arcs (v', t'') and (v'', t), then $\{v', v''\} \subseteq \widetilde{W}$ and $t'' \notin \widetilde{W}$. As before, the node set $\widetilde{W}'' = \widetilde{W} \cup \{t''\}$ induces an st-dicut, and as $[t'', t]^+ = \emptyset$, we have $\delta^+(\widetilde{W}'') = \widetilde{C} \setminus \{(v', t'')\}$. By repeating this procedure for every pair of arcs of $\widetilde{C}$ corresponding to the same edge, we obtain a minimal arc set $\widetilde{C}' \subset \widetilde{C}$, which does not contain any arc of the form (u', u''), $u \in V$ or of the form (t', t), $t \in T_D$, and which does not contain two arcs corresponding to the same edge of G. Thus, from Case 1, we have $\overline{y}(\widetilde{C}') \geq k$. As $\widetilde{C}' \subset \widetilde{C}$, we have $\overline{y}(\widetilde{C}) \geq \overline{y}(\widetilde{C}')$ and, hence, get $\overline{y}(\widetilde{C}) \geq k$.

Case 3. Now suppose that $\widetilde{C}$ contains an arc of the form (u', u''), for some $u \in V$, but no arc of the form (t', t), for every $t \in T_D$. If $|[u', u'']^+| = k$, then, as $\overline{y}(a) = 1$, for all $a \in [u', u'']^+$, we have $\overline{y}(\widetilde{C}) \geq k$.

If $|[u', u'']^+| = \max_{\{s,t\} \in D} \{\min\{|[s,u]|, |[u,t]|, k\}\} < k$, then

$$\min\{|[s,u]|, |[u,t]|, k\} = \min\{|[s,u]|, |[u,t]|\} \leq |[u', u'']^+|,$$
$$\text{for all } \{s,t\} \in D.$$

Now if $\min\{|[s_0, u]|, |[u, t_0]|\} = |[s_0, u]|$, for some $s_0 \in \widetilde{W} \cap S_D$ and $t_0 \notin \widetilde{W}$, then by considering $\widetilde{W}' = \widetilde{W} \setminus (\{u'\} \cup (\widetilde{W} \cap S_D \setminus \{s_0\}))$, we have

$$\begin{aligned}\overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W}')) = \overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W})) + \overline{y}([s_0, u']^+) - \overline{y}([u', u'']^+) \\ - \overline{y}(\delta^+_{\widetilde{G}}(u') \cap (\widetilde{C} \setminus [u', u'']^+)) \\ - \overline{y}([\widetilde{W} \cap (S_D \setminus \{s_0\}), \widetilde{V} \setminus \widetilde{W}]^+).\end{aligned}$$

As $|[s_0, u]| = \min\{|[s_0, u]|, |[u, t_0]|\} \leq |[u', u'']^+|$, $0 \leq \overline{y}(a) \leq 1$, for all $a \in \widetilde{A}$, and $\overline{y}(a) = 1$, for all $a = (u', u'')$, we have $\overline{y}([s_0, u']^+) \leq |[s_0, u']^+| \leq |[u', u'']^+| = \overline{y}(|[u', u'']^+|)$. Therefore,

$$\overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W}')) \leq \overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W})).$$

Moreover, $\delta^+_{\widetilde{G}}(\widetilde{W}')$ does not contain any arc of the form (t', t).

If $\min\{|[s_0, u]|, |[u, t_0]|\} = |[u, t_0]|$, then one can consider node set $\widetilde{W}' = \widetilde{W} \cup \{u''\} \cup ((\widetilde{V} \setminus \widetilde{W}) \cap (T_D \setminus \{t_0\})$, and using similar arguments as before, one obtains

$$\overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W}')) \leq \overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W})).$$

Notice that here also, $\delta^+_{\widetilde{G}}(\widetilde{W}')$ does not contain any arc of the form (t', t).

By repeating this procedure, one gets a node set $\widetilde{W}''$ whose induced arc set $\delta^+_{\widetilde{G}}(\widetilde{W}'')$ does not contain any arc of the form (u', u'') and of the form (t', t), and such that

$$\overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W})) \geq \overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W}'')).$$

Thus, from Cases 1 and 2, we have

$$\overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W})) \geq \overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W}'')) \geq k.$$

Case 4. Finally suppose that $\widetilde{C} = \delta^+_{\widetilde{G}}(\widetilde{W})$ contains an arc of the form (t', t). If $|[t', t]^+| = k$, then we have $\overline{y}(\widetilde{C}) \geq k$. If $|[t', t]^+| = \max_{s \in S_D, \{s,t\} \in D} \{|[s,t]|\} = |[s_0, t]| < k$, for some $s_0 \in S_D$, then by considering node set $\widetilde{W}' = \widetilde{W} \setminus (\{t'\} \cup (\widetilde{W} \cap (S_D \setminus \{s_0\})))$ and using similar arguments as in Case 3, we get

$$\overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W})) \geq \overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W}')).$$

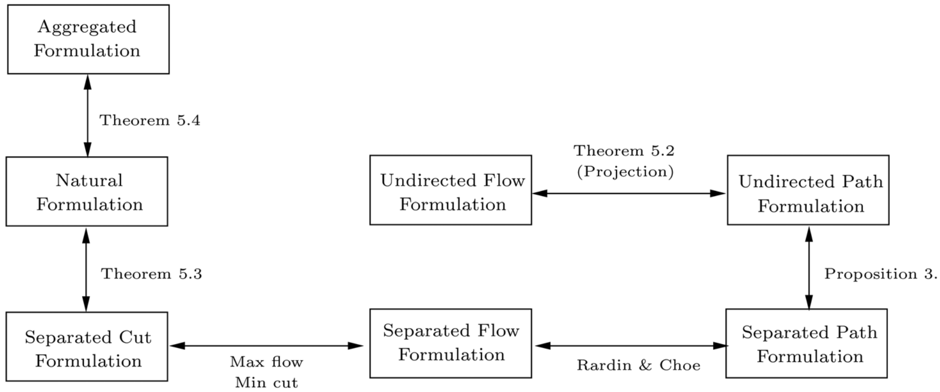

FIG. 6. Comparison of LP-bounds of all the formulations.

By repeating this procedure one obtains a node set, say $\widetilde{W}'' \subset \widetilde{V}$, such that $\delta^+_{\widetilde{G}}(\widetilde{W}'')$ does not contain any arc of the form (t', t), and such that

$$\overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W})) \geq \overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W}'')),$$

Thus, by Cases 1, 2, and 3, we get

$$\overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W})) \geq \overline{y}(\delta^+_{\widetilde{G}}(\widetilde{W}'')) \geq k.$$

Therefore, $\overline{y}$ satisfies every st-dicut inequality (4.1) and, as $\overline{y}(a) = \overline{x}(e)$, if $a \in \widetilde{A}(e)$, for all $e \in E$, and $\overline{y}(a) = 1$, otherwise, $(\overline{x}, \overline{y})$ is solution of the linear programming relaxation of kHNDP$_{Ag}$, whose value is $\overline{Z}_{\text{Nat}}$. Thus, $\overline{Z}_{\text{Nat}} \geq \overline{Z}_{\text{Ag}}$, which ends the proof of the theorem.

One may notice that Theorems 5.3 and 5.4 point out the fact that the separated and aggregated formulations produce the same LP-bound as the natural formulation. Also, by Theorems 5.1 and 5.2, the undirected path and flow formulations produce the same LP-bound as the separated formulation.

As a consequence, the undirected formulations, the separated and aggregated formulations produce the same LP-bound as the natural formulation, and all the formulations produce the same LP-bound. These results are summarized in Corollary 5.1 and Figure 6.

Corollary 5.1. *Formulations (2.5), (2.10), (2.20), (3.6), (3.12), (3.18), and (4.5), produce the same LP-bound for the kHNDP.*

6. COMPUTATIONAL RESULTS

In this section, we present a computational study of the different formulations introduced in the paper. The main objective is to check their efficiency for solving the problem for large scale instances, and compare them to each other from a computational point of view.

We solve each formulation using CPLEX 12.5 and Concert Technology, implemented in C++, on a DELL Workstation T3500 with an Intel Xeon Quad-Core 2.26GHz and 3Gb of RAM. For our computational experiments, we use instances from TSPLIB [35] (euclidean complete graphs) with randomly generated demand sets. We consider single-source multi-destination instances and multisource multidestination instances. The number of nodes of the graphs varies from 21 to 52. The number of demands, in turn, varies from 15 to 50, for the rooted case and from 10 to 26 for the arbitrary case.

The following tables give computational results for the separated flow and path formulations, the aggregated formulation and natural formulation. We do not give the results for the separated cut formulation. We will discuss this formulation later.

Note that the linear programming relaxations of the separated flow and path formulations are solved using a linear program, as they contain a polynomial number of variables and constraints. For the aggregated and natural formulations, we use a cutting plane algorithm to solve their linear programming relaxation, as they both contain an exponential number of constraints (st-cut (2.1) and L-st-path-cut inequalities (2.4) for the natural formulation, and st-dicut inequalities (4.1) for the aggregated inequalities). The separation problem associated with these constraints reduces to a maximum flow problem and can be solved in polynomial time (see for example [3] and [19]).

The results given in the following tables are obtained for $k=3$ and $L=2$, 3.

Each instance is described by the number of nodes and the type of the demand set, indicated by "r" for rooted demands and "a" for arbitrary demands. The other entries of the various tables are:

$\|V\|$	: number of nodes,
$\|D\|$	: number of demands,
COpt	: weight of the best upper bound obtained,
Gap	: the relative error between the best upper bound (the optimal solution if the problem has been solved to optimality) and the lower bound obtained at the root node (LPRoot) of the branch-and-cut tree, that is, Gap = (COpt − LPRoot)/COpt,
NSub	: number of subproblems in the branch-and-cut tree,
TT	: total CPU time in h:min:s

In all the tables, the instances indicated with "*" are instances for which the algorithm has reached the maximum

TABLE 1. Results for separated flow formulation with $k = 3$

$\lvert V \rvert$	$\lvert D \rvert$	COpt	Gap	NSub	TT
(a) Results for $L = 2$					
r 21	15	7,138	3.36	98	00:00:01
r 21	17	7,790	5.15	178	00:00:01
r 21	20	8,762	6.66	1,993	00:00:02
a 21	10	8,313	0.00	1	00:00:01
a 21	11	8,677	0.00	1	00:00:01
r 30	15	12,512	0.00	1	00:00:01
r 30	20	14,215	1.19	84	00:00:01
r 30	25	15,610	2.83	365	00:00:02
a 30	10	12,124	2.31	115	00:00:01
a 30	15	15,868	1.43	139	00:00:01
r 48	20	21,586	2.07	94	00:00:02
r 48	30	29,326	8.12	15,260	00:00:22
r 48	40	37,458	14.42	84,70,79	00:13:09
a 48	15	32,097	0.54	34	00:00:01
a 48	20	44,400	0.34	269	00:00:02
a 48	24	52,619	0.14	77	00:00:02
r 52	20	14,093	1.57	71	00:00:01
r 52	30	17,643	4.66	3,736	00:00:06
r 52	40	21,041	8.08	75,187	00:01:21
r 52	50	24,619	8.89	2,11,817	00:08:54
a 52	20	18,480	0.88	374	00:00:01
a 52	26	24,125	0.51	360	00:00:01
(b) Results for $L = 3$					
r 21	15	54,72	6.70	954	00:00:26
r 21	17	58,64	7.97	24,72	00:00:50
r 21	20	64,66	8.95	53,972	00:22:39
a 21	10	66,75	8.43	20,292	00:03:45
a 21	11	6,770	6.54	1,612	00:00:39
r 30	15	10,109	5.68	2,023	00:01:19
r 30	20	11,182	6.89	32,625	00:30:13
*r 30	25	12,482	9.72	2,18,700	05:00:00
a 30	10	10,254	5.08	3,482	00:00:44
***a 30*	*15*	*1,33,04*	*7.28*	*10,25,10*	*01:21:12*
r 48	20	16,684	9.10	62,793	04:52:49
*r 48	30	21,415	15.21	27,834	05:00:00
*r 48	40	27,546	19.54	2,823	05:00:00
***a 48*	*15*	*25,171*	*17.54*	*19,357*	*02:56:47*
*** a 48*	*20*	*34,569*	*22.52*	*9,341*	*04:59:60*
*a 48	24	41,347	23.99	4,690	05:00:00
r 52	20	11,154	7.88	36,720	01:53:47
***r 52*	*30*	*13,792*	*11.40*	*6,662*	*03:01:02*
***r 52*	*40*	*16,797*	*16.42*	*4,812*	*04:50:15*
*r 52	50	19,592	18.94	2,463	05:00:00
***a 52*	*20*	*16,049*	*9.47*	*9,273*	*03:25:50*
*a 52	26	21,095	14.98	3,905	05:00:00

TABLE 2. Results for separated path formulation with $k = 3$

$\lvert V \rvert$	$\lvert D \rvert$	COpt	Gap	NSub	TT
(a) Results for $L = 2$					
r 21	15	7,138	0.00	1	00:00:01
r 21	17	7,790	0.00	1	00:00:01
r 21	20	8,762	7.38	1322	00:00:02
a 21	10	8,313	0.00	1	00:00:01
a 21	11	8,677	0.00	1	00:00:01
r 30	15	12,512	1.74	129	00:00:01
r 30	20	14,215	1.68	247	00:00:02
r 30	25	15,610	2.78	145	00:00:03
a 30	10	12,124	3.33	103	00:00:01
a 30	15	15,868	1.12	54	00:00:01
r 48	20	21,586	2.59	153	00:00:03
r 48	30	29,326	8.10	10829	00:00:20
r 48	40	37,458	12.86	1203056	00:19:24
a 48	15	32,097	0.09	3	00:00:01
a 48	20	44,400	0.24	196	00:00:02
a 48	24	52,619	0.15	61	00:00:02
r 52	20	14,093	1.20	22	00:00:01
r 52	30	17,643	5.82	3,722	00:00:13
r 52	40	21,041	8.47	91,633	00:02:05
r 52	50	24619	9.97	56,27,02	00:18:48
a 52	20	18,480	0.85	333	00:00:01
a 52	26	24,125	0.32	65	00:00:01
(b) Results for $L = 3$					
r 21	15	5,472	7.74	4,755	00:00:36
r 21	17	5,864	7.75	6,033	00:01:45
r 21	20	6,466	9.32	21,323	00:06:37
a 21	10	6,675	8.34	30,167	00:04:19
a 21	11	6,770	6.62	8,917	00:02:16
r 30	15	10,109	5.94	3,892	00:02:09
r 30	20	11,182	7.60	12,189	00:09:17
***r 30*	*25*	*12,488*	*10.13*	*76,177*	*01:50:47*
a 30	10	10,254	5.47	10,404	00:01:28
***a 30*	*15*	*13,518*	*8.82*	*55,720*	*00:30:35*
***r 48*	*20*	*16,856*	*10.27*	*19,132*	*01:32:57*
*r 48	30	21,162	14.12	24,142	05:00:00
***r 48*	*40*	*27,893*	*20.50*	*5,404*	*03:35:16*
***a 48*	*15*	*25,515*	*18.65*	*15,251*	*01:39:31*
***a 48*	*20*	*34,457*	*22.27*	*13,824*	*04:35:36*
*a 48	24	41,122	23.57	5,010	05:00:00
r 52	20	11,154	8.43	25,113	01:23:41
*r 52	30	13,689	10.83	26,583	05:00:00
*r 52	40	16,291	13.67	9,629	05:00:00
*r 52	50	19,292	17.78	4,991	05:00:00
***a 52*	*20*	*15,837*	*8.31*	*9,023*	*02:33:18*
*a 52	26	20,405	12.31	5,219	05:00:00

CPU time, 5 h, while instances with "**" are those for which the algorithm runs out of resources (lack of memory). For all these instances, we give, in italics, the best results obtained at the end of the execution of the algorithm.

Note that, according to Corollary 5.1, the linear relaxations of all the formulations tested here have the same value. However, the computation of the LP-value obtained at the root node of the Branch-and-Cut tree also uses CPLEX general-purpose cutting planes. Thus, this LP-value may be different from one formulation to another and, consequently, the Gap achieved by the Branch-and-Cut algorithm for all the formulations may be different from one another.

Table 1(a) and 1(b) give the results for the separated flow formulation for $L = 2$ and $L = 3$, respectively, and $k = 3$.

By observing Table 1(a), we notice that for $L = 2$, the problem is solved to optimality by the separated flow formulation for all the instances. The CPU time is less than 1 min in almost all cases, and the gap between the LP-root node and the optimal solution is low (less than 5% for 16 instances over 22, and between 5% and 10% for 5 instances). For $L = 3$ (Table 2(b)), the separated flow formulation solves to optimality only 10 instances. For the remaining instances, upper bounds are obtained by CPLEX, with a relative gap of at most 23.99%.

TABLE 3. Results for aggregated formulation with $k = 3$

$\|V\|$	$\|D\|$	NCut	COpt	Gap	NSub	TT
(a) Results for $L = 2$						
r 21	15	1,042	7,138	9.51	78	00:00:03
r 21	17	1,144	7,790	9.35	77	00:00:04
r 21	20	1,569	8,762	11.61	514	00:00:19
a 21	10	208	8,313	3.55	89	00:00:03
a 21	11	254	8,677	3.53	115	00:00:03
r 30	15	1,734	1,2512	5.56	134	00:00:10
r 30	20	2,596	1,4215	6.84	276	00:00:27
r 30	25	3,034	15,610	8.58	576	00:00:60
a 30	10	336	12,124	5.21	270	00:00:14
a 30	15	501	15,868	3.69	1,200	00:01:16
r 48	20	4,881	21,586	8.17	234	00:01:39
*r 48	30	44,027	29,329	15.23	20,961	05:00:00
***r 48*	*40*	*59,913*	*37,821*	*17.52*	*10,012*	*03:54:46*
a 48	15	1,134	32,097	3.09	987	00:04:33
*a 48	20	2,838	44,466	57.89	5,3910	05:00:00
*a 48	24	3,409	52,828	4.39	39,101	05:00:00
r 52	20	6,111	14,093	6.21	166	00:01:52
r 52	30	18,798	17,643	10.72	5,607	00:56:01
***r 52*	*40*	*50,569*	*21,160*	*13.40*	*12,513*	*04:09:06*
*r 52	50	45,205	24,739	13.76	10,801	05:00:00
a 52	20	937	18,480	3.44	2,013	00:12:23
a 52	26	2,064	24,125	4.12	16,539	02:36:53
(b) Results for $L = 3$						
r 21	15	9,187	5,472	8.34	890	00:03:21
r 21	17	15,790	5,864	8.25	1,622	00:09:15
r 21	20	48,679	6,466	9.54	11,806	02:10:33
*a 21	10	66,978	6,708	9.05	7,754	05:00:00
a 21	11	46,649	6,770	6.80	2,620	01:56:15
r 30	15	18,767	10,109	58.95	904	00:19:06
r 30	20	36,292	11,182	7.98	8,501	02:38:41
***r 30*	*25*	*71,352*	*12,569*	*11.69*	*7,739*	*04:37:07*
*a 30	10	44,501	10,287	6.03	6,495	05:00:00
*a 30	15	57,828	14,116	12.92	3,566	05:00:00
*r 48	20	60,270	17,154	12.45	2,742	05:00:00
*r 48	30	72,513	23,075	66.08	1,943	05:00:00
*r 48	40	68,534	29,332	24.56	2,007	05:00:00
*a 48	15	35,396	–	–	313	05:00:00
*a 48	20	32,177	–	–	87	05:00:00
*a 48	24	26,715	–	–	31	05:00:00
*r 52	20	49,816	11,207	9.45	2,441	05:00:00
*r 52	30	71,167	14,272	15.06	1,375	05:00:00
*r 52	40	68,627	16,963	17.45	1,528	05:00:00
*r 52	50	6,1504	20,604	23.51	2,040	05:00:00
*a 52	20	3,4627	–	–	251	05:00:00
*a 52	26	28,028	–	–	73	05:00:00

TABLE 4. Results for natural formulation with $k = 3$

$\|V\|$	$\|D\|$	NCut	COpt	Gap	NSub	TT
(a) Results for $L = 2$						
r 21	15	900	7,138	9.51	150	00:00:02
r 21	17	1,056	7,790	9.35	118	00:00:02
r 21	20	1,282	8,762	11.61	819	00:00:09
a 21	10	245	8,313	3.55	236	00:00:02
a 21	11	248	8,677	3.53	157	00:00:02
r 30	15	1,654	1,2512	5.56	61	00:00:04
r 30	20	4,408	14,215	43.37	763	00:00:29
r 30	25	2,826	15,610	8.58	631	00:00:20
a 30	10	287	12,124	5.21	359	00:00:04
a 30	15	482	15,868	3.69	1,207	00:00:17
r 48	20	4,447	21,586	8.17	209	00:00:44
r 48	30	4,8160	29,326	15.22	19,516	04:05:33
***r 48*	*40*	*61,444*	*37,798*	*58.44*	*10,678*	*02:13:37*
a 48	15	1,187	32,097	3.09	1,602	00:02:02
*a 48	20	2,832	44,527	4.43	15,5033	05:00:00
*a 48	24	3,276	52844	49.53	12,14,86	05:00:00
r 52	20	6,096	14,093	6.21	229	00:01:06
r 52	30	26,416	17,643	10.72	4,056	00:30:42
***r 52*	*40*	*58,080*	*21,079*	*13.06*	*11,018*	*02:23:08*
***r 52*	*50*	*51060*	*24877*	*14.24*	*12,868*	*02:37:38*
a 52	20	1,112	18,480	3.44	3,783	00:04:52
a 52	26	2,031	24,125	4.12	16,444	00:34:43
(b) Results for $L = 3$						
r 21	15	8,782	5,472	8.34	946	00:01:45
r 21	17	12,072	5,864	8.25	1,525	00:03:47
r 21	20	44,595	6,466	9.54	13,837	01:22:47
*a 21	10	70,023	6,675	8.60	8,416	05:00:00
a 21	11	58,300	6770	6.80	3,496	03:41:11
r 30	15	14,483	10,109	6.87	791	00:06:58
r 30	20	43,112	11,182	29.65	6,527	01:54:40
***r 30*	*25*	*73,513*	*12,585*	*11.80*	*7,433*	*02:58:31*
a 30	10	44,462	10,254	5.73	7,627	03:50:46
*a 30	15	67,886	13,730	10.47	4,271	05:00:00
*r 48	20	73,929	17,417	45.89	2,872	05:00:00
*r 48	30	86,009	22,310	18.69	2,331	05:00:00
* r 48	40	92,333	28,720	22.95	2,260	05:00:00
*a 48	15	42,475	–	–	409	05:00:00
*a 48	20	35,387	–	–	127	05:00:00
*a 48	24	29,533	–	–	53	05:00:00
*r 52	20	58,213	11,254	9.83	3,244	05:00:00
*r 52	30	90,777	14,205	14.66	2,182	05:00:00
*r 52	40	94,667	17,202	18.60	2,493	05:00:00
*r 52	50	95,905	19,153	17.71	3,691	05:00:00
*a 52	20	44,549	–	–	341	05:00:00
*a 52	26	33,201	–	–	145	05:00:00

The problem seems to be easier for this formulation when $L = 2$ than when $L = 3$. The same observation applies to the other formulations. More instances are solved to optimality within 5 h when $L = 2$ than when $L = 3$. This observation confirms the idea that the kHNDP is easier when $L = 2$ than when $L = 3$.

Subsequently, we focus on the comparison of each formulation to the others in terms of CPU time and in terms of best solution. The following tables give the results for the separated path, the aggregated and the natural formulation for $L = 2$, 3 and $k = 3$.

We start the comparison by considering $L = 2$. The comparison between Tables 1(a), 2(a), 3(a), and 4(a) shows first that, for $L = 2$, the separated flow and path formulations produce quite similar results, and that they achieve better results than the aggregated and natural formulations. The formulations having results in the first two tables are able to solve to optimality 100% of the instances, while the aggregated and natural formulations solve 77.27% and 72.72% of the instances to optimality, respectively. Also, for the instances solved to optimality, the total CPU time for the separated flow and path formulations is better than that for the aggregated

TABLE 5. Comparison between best upper bounds for $L=3$ and $k=3$

$\|V\|$	$\|D\|$	GSPath	GAgg	GNat
r 30	25	6	87	103
a 30	15	214	812	426
r 48	20	172	470	733
r 48	30	2727	1660	895
r 48	40	347	∞	1174
a 48	15	344	∞	∞
a 48	20	−112	∞	∞
a 48	24	−225	∞	∞
r 52	30	−103	480	413
r 52	40	−506	166	405
r 52	50	−300	1012	−439
a 52	20	−212	∞	∞
a 52	26	−690	∞	∞

and natural formulations. This can be explained by the fact that these latter formulations contain an exponential number of constraints and their linear programming relaxation is solved using the cutting plane method. Thus, the difference of CPU time mainly is the time spent by the algorithm for the separation of the cut constraints (4.1) and (2.1), and (2.4).

Now, we turn our attention to the case where $L=3$. As mentioned before, the problem becomes harder in this case. We compare the different formulations in terms of upper bound. For this, we choose the results 1(b) of the separated flow formulation as reference. The following table gives, for some instances and for each formulation, the difference between the upper bound achieved for a given formulation and the one achieved by the separated flow formulation, that is, $G_i = \text{COpt}_i - \text{COpt}_{SFlow}$, where i is SPath, Agg, or Nat, standing, respectively, for separated path formulation, aggregated and natural formulation. Notice that the results given below measure the ability of CPLEX to efficiently solve each formulation.

The instances reported in the table are those for which at least one formulation does not give the optimal solution.

A negative value in Table 5 for a given formulation indicates that the formulation gives a better bound than that obtained by the separated flow formulation while a positive value indicates a greater bound. From this table, we can see that the separated path formulation produces, for many cases, a better bound than the separated flow formulation. Also, this formulation produces, in most cases, better bounds than the natural and aggregated formulations.

The comparison between the natural and aggregated formulations in Table 5 shows inconclusive results. For 3 instances of 7, the aggregated formulation outperforms the natural formulation, while it is the contrary for the 4 others.

We conclude this series of experiments by making a comment on the separated cut formulation. This formulation produces poor results in terms of CPU time and in terms of upper bound. For several instances, the algorithm is not able to solve the linear programming relaxation of the root node of the branch-and-cut tree after 5 h of CPU time, and this, even for $L=2$. And for all of these instances, the algorithm does not produce an upper bound. This is explained by the long time spent by the algorithm in the separation of the cut constraints (3.13).

TABLE 6. Results for separated flow formulation with $k=4$

$\|V\|$	$\|D\|$	COpt	Gap	NSub	TT
(a) Results for $L=2$					
r 21	15	8,931	0.00	1	00:00:01
r 21	17	9,818	3.15	35	00:00:01
r 21	20	10,953	3.63	323	00:00:02
a 21	10	10,915	0.00	1	00:00:01
a 21	11	11,664	0.30	48	00:00:01
r 30	15	16,235	0.22	6	00:00:01
r 30	20	18,011	0.00	1	00:00:01
r 30	25	19,830	1.81	1,034	00:00:02
a 30	10	16,097	0.00	1	00:00:01
a 30	15	21,333	0.00	1	00:00:01
r 48	20	27,476	0.85	74	00:00:01
r 48	30	37,118	7.29	7,949	00:00:15
r 48	40	47,305	10.73	17,3605	00:03:10
a 48	15	42,503	0.11	14	00:00:01
a 48	20	57,508	0.00	1	00:00:01
a 48	24	68,370	0.00	1	00:00:01
r 52	20	17,887	0.00	1	00:00:01
r 52	30	22,545	3.82	2,409	00:00:04
r 52	40	26,633	6.63	20,834	00:00:34
r 52	50	31,457	7.62	27,5803	00:07:05
a 52	20	24,586	0.17	16	00:00:01
a 52	26	32,175	0.13	8	00:00:01
(b) Results for $L=3$					
r 21	15	72,73	2.85	668	00:00:18
r 21	17	78,24	4.22	1,996	00:00:35
r 21	20	8,556	4.77	48,970	00:17:11
a 21	10	8,929	5.96	89,237	00:13:38
a 21	11	9,232	6.05	10,03,99	00:22:04
r 30	15	13,963	3.83	7,797	00:05:57
r 30	20	15,041	3.54	16,574	00:15:02
r 30	25	16,268	4.91	16,55,47	04:10:20
a 30	10	14,058	2.51	11,415	00:01:55
***a 30*	*15*	*18,138*	*6.05*	*80,473*	*01:05:29*
*r 48	20	22,063	5.58	79,488	05:00:00
*r 48	30	27,910	11.02	28,363	05:00:00
*r 48	40	35128	14.19	8578	05:00:00
*a 48	15	32,588	14.04	45,352	05:00:00
***a 48*	*20*	*45,234*	*20.76*	*9,324*	*04:37:53*
*a 48	24	55,307	24.08	4,927	05:00:00
*r 52	20	14,979	5.43	10,33,72	05:00:00
*r 52	30	18,172	7.56	23,061	05:00:00
* r 52	40	21,265	10.09	7,077	05:00:00
*r 52	50	24,679	11.60	2,698	05:00:00
*a 52	20	21,206	6.45	16,947	05:00:00
*a 52	26	28,350	13.48	4,061	05:00:00

Our next series of experiments concerns the kHNDP when $k=4$ and $k=5$. For each case, we have solved the problem when $L=2$ and $L=3$. All the instances previously tested for $k=3$ have been solved for $k=4$ and $k=5$. The results are given only for large size instances, that is, with graphs having 48 or 52 nodes. They are presented in Tables 6–9 for $k=4$ and Tables 10–13 for $k=5$.

The first observation is that, for $k=4$ (Tables 6–9), almost all the instances are solved to optimality when $L=2$ while

TABLE 7. Results for separated path formulation with $k = 4$

$\|V\|$	$\|D\|$	COpt	Gap	NSub	TT
(a) Results for $L = 2$					
r 21	15	8,931	0.00	1	00:00:01
r 21	17	9,818	0.00	1	00:00:01
r 21	20	10,953	4.17	403	00:00:01
a 21	10	10,915	0.00	1	00:00:01
a 21	11	11,664	0.00	1	00:00:01
r 30	15	16,235	0.09	5	00:00:01
r 30	20	18,011	0.19	3	00:00:01
r 30	25	19,830	1.25	386	00:00:02
a 30	10	16,097	0.00	1	00:00:01
a 30	15	21,333	0.00	1	00:00:01
r 48	20	27,476	0.87	73	00:00:01
r 48	30	37,118	7.66	6,722	00:00:16
r 48	40	47,305	10.65	17,1625	00:02:50
a 48	15	42,503	0.00	1	00:00:01
a 48	20	57,508	100.00	4	00:00:01
a 48	24	68,370	100.00	2	00:00:01
r 52	20	17,887	0.00	1	00:00:01
r 52	30	22545	4.60	3,059	00:00:04
r 52	40	26,633	5.92	38,675	00:00:51
r 52	50	31,457	7.40	40,37,77	00:09:03
a 52	20	24,586	0.14	13	00:00:01
a 52	26	32,175	100.00	4	00:00:01
(b) Results for $L = 3$					
r 21	15	7,273	3.50	4,961	00:00:27
r 21	17	7,824	4.29	6,267	00:01:31
r 21	20	8,556	5.19	24,448	00:06:20
a 21	10	8,929	6.20	73,604	00:10:36
a 21	11	9,232	6.45	14,2836	00:27:48
r 30	15	13963	4.30	10,861	00:04:51
r 30	20	15041	3.82	15,990	00:08:47
r 30	25	16,268	4.91	11,99,54	02:25:47
a 30	10	14,058	2.49	5,221	00:01:08
***a 30*	*15*	*18,066*	*5.73*	*97,960*	*01:00:08*
*r 48	20	22,044	5.68	79,943	05:00:00
*r 48	30	27,744	10.59	36,252	05:00:00
*r 48	40	34,732	13.17	13,286	05:00:00
***a 48*	*15*	*33,819*	*16.92*	*15,462*	*01:45:17*
***a 48*	*20*	*44,080*	*18.68*	*13,841*	*04:50:50*
*a 48	24	52,719	20.35	5,226	05:00:00
***r 52*	*20*	*14,979*	*5.47*	*31,689*	*01:44:28*
*r 52	30	18,139	7.57	22,907	05:00:00
*r 52	40	20,961	8.59	12,208	05:00:00
*r 52	50	24,576	11.16	4,416	05:00:00
***a 52*	*20*	*21,347*	*7.03*	*10,554*	*03:18:19*
*a 52	26	27,920	12.19	4,504	05:00:00

TABLE 8. Results for aggregated formulation with $k = 4$

$\|V\|$	$\|D\|$	NCut	COpt	Gap	NSub	TT
(a) Results for $L = 2$						
r 21	15	576	8,931	3.55	31	00:00:02
r 21	17	568	9,818	4.09	10	00:00:02
r 21	20	719	10,953	5.72	71	00:00:04
a 21	10	92	10,915	15.04	7	00:00:01
a 21	11	239	1,1664	2.13	133	00:00:03
r 30	15	1,053	16,235	0.66	12	00:00:03
r 30	20	1,417	18,011	1.21	50	00:00:07
r 30	25	1,788	19,830	3.34	195	00:00:19
a 30	10	124	16,097	0.40	7	00:00:02
a 30	15	238	21,333	1.18	38	00:00:05
r 48	20	3,284	27,476	2.97	111	00:00:51
r 48	30	21,913	37,118	10.43	47,87	00:51:19
***r 48*	*40*	*46,101*	*47,397*	*12.13*	*1,4389*	*04:02:21*
a 48	15	647	42,503	0.47	44	00:00:27
a 48	20	915	57,508	0.26	42	00:00:40
a 48	24	936	68,370	0.24	21	00:00:33
r 52	20	4,778	17,887	0.73	47	00:01:07
r 52	30	10,891	22,545	6.67	1,196	00:11:52
r 52	40	34,527	26,633	8.13	14,150	04:09:54
*r 52	50	28,687	31,553	9.83	17,024	05:00:00
a 52	20	468	24,586	0.44	69	00:00:50
a 52	26	739	32,175	0.87	73	00:01:16
(b) Results for $L = 3$						
r 21	15	3,383	7,273	3.93	439	00:00:50
r 21	17	7,638	7,824	4.54	1,459	00:03:57
r 21	20	28,466	8,556	38.85	16,885	01:35:12
***a 21*	*10*	*66,386*	*9,128*	*8.30*	*9,682*	*04:33:56*
***a 21*	*11*	*66,713*	*9,297*	*7.19*	*9,820*	*04:09:36*
r 30	15	18,218	13,963	46.66	1,664	00:23:07
r 30	20	45,563	15,041	4.20	9,195	03:30:04
***r 30*	*25*	*60,852*	*16,302*	*5.49*	*9,494*	*04:10:04*
a 30	10	26,949	14,058	2.81	3,730	01:43:48
*a 30	15	52,978	18,079	5.82	3,151	05:00:00
*r 48	20	55,157	22,680	53.26	2,945	05:00:00
*r 48	30	66,314	28,522	13.13	2,216	05:00:00
*r 48	40	66,868	34,560	12.85	2,142	05:00:00
*a 48	15	35,986	–	–	463	05:00:00
*a 48	20	31,374	–	–	95	05:00:00
*a 48	24	28,236	–	–	39	05:00:00
*r 52	20	56,345	15,216	7.16	2,715	05:00:00
*r 52	30	58,776	18,890	11.48	2,623	05:00:00
*r 52	40	67,966	21,254	10.21	1,551	05:00:00
*r 52	50	57,261	24,402	10.73	2,465	05:00:00
*a 52	20	38,017	–	–	283	05:00:00
*a 52	26	28,081	–	–	93	05:00:00

most of them are not solved to optimality within the CPU time limit when $L = 3$. The observation is the same for $k = 5$ (Tables 10–13). This leads to the conclusion that the kHNDP appears to be easier when $L = 2$ than when $L = 3$ also in the case where $k > 3$.

Also, we note that for $k = 4$ and $k = 5$, the separated flow and path formulations outperform the aggregated and natural formulations. For example, for $k = 4$ and $L = 2$, the CPU time is less than one minute for the separated flow and path formulations and for almost all the instances. However, the CPU time is much higher for the aggregated and natural formulations. A similar observation applies for $L = 3$ and $k = 4$ and for $k = 5$ and $L = 2, 3$. Considering these two latter formulations, we remark that, as for $k = 3$, neither of them outperforms the other.

Another observation for $k = 4$, 5 and all the formulations is that the gap achieved between the best upper bound and the LP-root node in most of the cases, is better than that obtained for $k = 3$ and for the same instances. Moreover, one can notice that some instances have been solved to optimality for $k = 4$, 5 and not for $k = 3$. For example, instance r30 − 25 (that is, $|V| = 30$, rooted set of demands and $|D| = 25$) has been solved to optimality by the separated flow formulation for $k = 4$, 5 and $L = 3$, while it has not

TABLE 9. Results for natural formulation with $k = 4$

$\lvert V\rvert$	$\lvert D\rvert$	NCut	COpt	Gap	NSub	TT
(a) Results for $L = 2$						
r 21	15	426	8,931	3.55	26	00:00:01
r 21	17	462	9,818	4.09	16	00:00:01
r 21	20	708	1,0953	5.72	142	00:00:02
a 21	10	83	10,915	0.31	4	00:00:01
a 21	11	280	11,664	2.13	198	00:00:02
r 30	15	956	16,235	0.66	26	00:00:02
r 30	20	1,105	18,011	1.21	54	00:00:03
r 30	25	2,687	19,830	32.07	578	00:00:17
a 30	10	70	16,097	0.40	6	00:00:01
a 30	15	221	21,333	1.18	60	00:00:02
r 48	20	2,896	27,476	2.97	174	00:00:27
r 48	30	19,618	37,118	10.43	3,920	00:19:55
**r 48	40	42,894	47,348	12.04	15,572	02:05:02
a 48	15	536	42,503	0.47	39	00:00:06
a 48	20	827	57,508	43.36	42	00:00:10
a 48	24	961	68,370	0.24	52	00:00:18
r 52	20	3,892	17,887	0.73	19	00:00:22
r 52	30	9,777	22,545	6.67	896	00:03:45
**r 52	40	35,089	26,698	42.80	21,312	02:44:36
**r 52	50	49,525	31,546	9.81	14,482	02:44:12
a 52	20	476	24,586	0.44	82	00:00:11
a 52	26	685	32,175	0.99	86	00:00:15
(b) Results for $L = 3$						
r 21	15	3,886	7,273	3.93	581	00:00:28
r 21	17	8,175	7,824	4.54	1,622	00:02:29
r 21	20	37,918	8,556	5.32	19,320	01:23:06
**a 21	10	58,118	8,987	6.86	11,841	03:38:50
*a 21	11	74,430	9,509	9.25	10,822	05:00:00
r 30	15	37,818	13,963	4.31	4,587	01:10:27
r 30	20	31,405	15,041	4.20	4,419	00:51:58
**r 30	25	68,478	16,309	5.53	8,960	02:56:51
a 30	10	19,901	14,058	2.81	3,586	00:35:19
*a 30	15	63,324	18,594	8.43	5,667	05:00:00
*r 48	20	73,416	22,462	7.94	3,842	05:00:00
**r 48	30	85,854	28,785	13.92	3,679	04:59:09
*r 48	40	94,810	35,016	13.98	2,635	05:00:00
*a 48	15	44,783	–	–	653	05:00:00
*a 48	20	33,680	–	–	145	05:00:00
*a 48	24	29,990	–	–	49	05:00:00
*r 52	20	72,672	15,343	31.36	3,508	05:00:00
*r 52	30	76,392	18,570	9.96	3,452	05:00:00
*r 52	40	93,699	21,431	10.95	2,923	05:00:00
**r 52	50	85,126	24,748	11.98	3,787	03:39:22
*a 52	20	37,328	–	–	333	05:00:00
*a 52	26	32,565	–	–	135	05:00:00

TABLE 10. Results for separated flow formulation with $k = 5$

$\lvert V\rvert$	$\lvert D\rvert$	COpt	Gap	NSub	TT
(a) Results for $L = 2$					
r 21	15	10,805	0.00	1	00:00:01
r 21	17	12,030	0.00	1	00:00:01
r 21	20	13,288	0.96	72	00:00:01
a 21	10	14,345	0.39	17	00:00:01
a 21	11	14,690	0.00	1	00:00:01
r 30	15	20,926	0.00	1	00:00:01
r 30	20	22,880	0.00	1	00:00:01
r 30	25	24,609	0.28	94	00:00:01
a 30	10	21,324	0.00	1	00:00:01
a 30	15	27,513	0.44	50	00:00:01
r 48	20	34,287	0.31	16	00:00:01
r 48	30	44,411	3.04	1,479	00:00:04
r 48	40	56,200	5.97	16,536	00:00:35
a 48	15	54,554	0.09	16	00:00:01
a 48	20	74,317	0.16	556	00:00:02
a 48	24	88,147	0.12	209	00:00:02
r 52	20	22,869	0.27	29	00:00:01
r 52	30	27,611	2.08	506	00:00:02
r 52	40	31,942	2.38	5,073	00:00:19
r 52	50	37,649	3.44	11,167	00:00:43
a 52	20	31,854	0.31	294	00:00:01
a 52	26	42,145	0.38	745	00:00:02
(b) Results for $L = 3$					
r 21	15	9,505	2.19	1,951	00:00:27
r 21	17	10,048	1.71	775	00:00:20
r 21	20	10,982	2.84	21,006	00:09:22
a 21	10	11,313	4.40	11,37,86	00:16:29
a 21	11	11,510	3.42	57,306	00:12:18
r 30	15	18,278	2.96	27,538	00:14:37
r 30	20	19,939	3.07	12,09,66	01:40:03
r 30	25	21,112	3.02	63,120	02:09:15
a 30	10	18,491	1.82	11,994	00:01:22
**a 30	15	23,011	4.07	53,273	00:36:27
r 48	20	27,818	3.30	76,367	03:31:20
*r 48	30	34,353	6.52	32,924	05:00:00
*r 48	40	42,815	9.88	10,038	05:00:00
**a 48	15	41,616	14.58	13,061	01:47:21
**a 48	20	54,001	16.51	10,308	04:14:42
* a 48	24	66,251	20.42	4,620	05:00:00
**r 52	20	19,482	5.59	12,038	01:05:45
*r 52	30	22,833	5.71	24,710	05:00:00
*r 52	40	26,094	6.12	8,072	05:00:00
*r 52	50	30,596	8.00	2,893	05:00:00
**a 52	20	27,307	6.78	8,160	03:13:54
*a 52	26	35,133	9.98	4,681	05:00:00

been solved to optimality for $k = 3$ within the CPU time limit.

Now comparing the formulations for $k = 4$ and $k = 5$, it appears that the problem is easier for $k = 5$ as both the gap and CPU time are, in general, better for $k = 5$ than for $k = 4$. For example, instance r48 − 40 could not be solved to optimality by the natural formulation for $k = 3$, 4 and $L = 2$, whereas it has been solved in less than 3 hours for $k = 5$. The same observation applies for other instances and other formulations.

From all these observation, we believe that the kHNDP becomes easier as k increases for $L = 2$, 3. This has already been mentioned by Bendali et al. [3] for the k-edge-connected subgraph problem (kECSP) (which corresponds to the kHNDP with $L = |V|$ and $D = V \times V$). The computational study they conducted for this latter problem showed that the problem becomes easier as k increases.

Also, Gabow et al. [20] considered the minimum size kECSP and the LP relaxation associated with its natural formulation in both directed and undirected graphs. They showed that the ratio of the total weight of fractional edges over all the edges in a minimum size solution is bounded, in undirected graphs, by $1 + \frac{3}{k}$, for k odd, and by $1 + \frac{2}{k}$, for k even, and, in directed graphs, by $1 + \frac{2}{k}$. Clearly, this ratio

TABLE 11. Results for separated path formulation with $k = 5$

$\|V\|$	$\|D\|$	COpt	Gap	NSub	TT
(a) Results for $L = 2$					
r 21	15	10,805	0.00	1	00:00:01
r 21	17	12,030	0.00	1	00:00:01
r 21	20	13,288	0.61	63	00:00:01
a 21	10	14,345	0.32	18	00:00:01
a 21	11	14,690	0.00	1	00:00:01
r 30	15	20,926	0.00	1	00:00:01
r 30	20	22,880	0.00	1	00:00:01
r 30	25	24,609	0.31	43	00:00:01
a 30	10	21,324	0.00	1	00:00:01
a 30	15	27,513	100.00	2	00:00:01
r 48	20	34,287	0.23	8	00:00:01
r 48	30	44,411	3.07	1,671	00:00:05
r 48	40	56,200	5.37	22,309	00:00:50
a 48	15	54,554	0.14	25	00:00:01
a 48	20	74,317	0.25	938	00:00:02
a 48	24	88,147	0.15	155	00:00:02
r 52	20	22,869	0.38	17	00:00:01
r 52	30	27,611	2.23	372	00:00:02
r 52	40	31,942	2.92	6,614	00:00:24
r 52	50	37,649	3.27	10,074	00:00:40
a 52	20	31,854	0.34	398	00:00:01
a 52	26	42,145	0.33	920	00:00:03
(b) Results for L = 3					
r 21	15	9,505	2.33	4,948	00:00:59
r 21	17	10,048	1.85	3,435	00:00:53
r 21	20	10,982	2.88	15,114	00:04:14
a 21	10	11,313	4.51	98,096	00:11:59
a 21	11	11,510	3.51	43,873	00:08:22
r 30	15	18,278	3.09	10,681	00:04:02
r 30	20	19,939	3.15	68,118	00:41:33
r 30	25	21,112	2.94	66,468	01:53:35
a 30	10	18,491	2.11	11,995	00:01:23
**a 30	15	22,929	3.72	68,444	00:50:43
r 48	20	27,818	3.40	30,902	01:59:50
*r 48	30	34,516	6.98	38,927	05:00:00
*r 48	40	42,210	8.58	14,034	05:00:00
**a 48	15	40,473	11.96	15,175	01:41:33
**a 48	20	54,303	16.98	11,170	03:57:55
*a 48	24	64,744	18.57	5,499	05:00:00
*r 52	20	19,236	4.34	64,073	05:00:00
*r 52	30	22,841	5.82	38,089	05:00:00
*r 52	40	26,095	6.15	13,250	05:00:00
*r 52	50	30,276	6.99	4,562	05:00:00
*a 52	20	27,486	7.26	22,709	05:00:00
*a 52	26	35,317	10.48	4,864	05:00:00

TABLE 12. Results for aggregated formulation with $k = 5$

$\|V\|$	$\|D\|$	NCut	COpt	Gap	NSub	TT
(a) Results for $L = 2$						
r 21	15	339	10,805	0.00	1	00:00:01
r 21	17	456	12,030	1.65	8	00:00:01
r 21	20	450	13,288	2.23	18	00:00:02
a 21	10	266	14,345	3.08	498	00:00:10
a 21	11	196	14,690	1.22	40	00:00:02
r 30	15	836	20,926	0.00	1	00:00:02
r 30	20	1,143	22,880	0.20	7	00:00:04
r 30	25	1,259	24,609	0.84	18	00:00:06
a 30	10	242	21,324	2.17	224	00:00:11
a 30	15	434	27,513	1.87	774	00:00:52
r 48	20	2,669	34,287	0.96	26	00:00:30
r 48	30	7,108	44,411	5.89	670	00:05:02
r 48	40	15,840	56,200	7.16	4,513	00:46:31
a 48	15	1,079	54,554	41.80	1,933	00:07:40
*a 48	20	2,480	74,434	2.27	56,096	05:00:00
*a 48	24	3,088	88,398	2.33	40,654	05:00:00
r 52	20	4,023	22,869	0.45	56	00:00:57
r 52	30	7,061	27,611	3.74	304	00:03:56
r 52	40	11,102	31,942	4.06	1,260	00:16:43
r 52	50	16,367	37,649	5.42	5,282	01:33:54
a 52	20	1,169	31,854	2.00	14,586	01:25:37
*a 52	26	2,163	42,166	2.41	35,514	05:00:00
(b) Results for L = 3						
r 21	15	4,779	9505	2.68	1,003	00:02:01
r 21	17	2,588	10048	2.15	316	00:00:43
r 21	20	7,617	10982	3.12	4,783	00:10:09
**a 21	10	49,393	11,548	6.56	13,863	03:04:51
**a 21	11	51,753	11,776	5.90	12,832	03:12:27
r 30	15	40,608	18,278	3.39	3,701	01:40:08
**r 30	20	67,356	19,939	3.40	9,943	04:31:03
**r 30	25	61,629	21,119	3.14	9,685	04:02:44
a 30	10	18,735	18,491	2.30	7,231	01:29:05
*a 30	15	53,657	23,506	6.22	5,167	05:00:00
*r 48	20	51,656	27,819	4.09	3,576	05:00:00
*r 48	30	63,706	35,363	9.48	2,548	05:00:00
*r 48	40	65,905	42,890	10.14	2,568	05:00:00
*a 48	15	38,014	–	–	611	05:00:00
*a 48	20	30,907	–	–	145	05:00:00
*a 48	24	27,600	–	–	39	05:00:00
*r 52	20	60,832	19848	7.51	2,623	05:00:00
*r 52	30	58,346	23,357	8.02	2,915	05:00:00
*r 52	40	61,595	26,263	6.94	1,921	05:00:00
*r 52	50	52,968	30,486	7.76	2,591	05:00:00
*a 52	20	33,367	–	–	493	05:00:00
*a 52	26	29,925	–	–	93	05:00:00

decreases as k increases, implying that an LP-rounding-based heuristic for the minimum size kECSP would become more efficient in finding near minimal solution when k increases. This argument also suggests that the probability of finding a fractional solution, in a branch-and-cut framework, decreases when k increases (implying a higher probability of finding integer feasible solutions). This could also explain the fact that the kHNDP is easier to solve when k increases. In fact, by the graph transformations, as it has been seen, the kHNDP reduces to the kECSP in directed graphs.

7. CONCLUDING REMARKS

In this article, we have studied the k-edge-connected hop-constrained network design problem when $k \geq 3$ and $L = 2$, 3. We have presented four integer programming formulations based on the transformation of the initial graph into directed layered graphs. We have also compared the linear programming relaxation of these formulations and shown that all of them give the same LP-bound.

We have also compared these formulations in a computational study for $k = 3, 4, 5$, which shows that, as expected, the

TABLE 13. Results for natural formulation with k = 5

$\|V\|$	$\|D\|$	NCut	COpt	Gap	NSub	TT
(a) Results for L = 2						
r 21	15	376	10,805	16.27	11	00:00:01
r 21	17	373	12,030	1.65	12	00:00:01
r 21	20	376	13,288	2.23	17	00:00:01
a 21	10	305	14,345	3.08	527	00:00:03
a 21	11	324	14,690	1.22	126	00:00:02
r 30	15	680	20,926	0.00	1	00:00:01
r 30	20	868	22,880	0.20	4	00:00:01
r 30	25	1,050	24,609	0.84	20	00:00:02
a 30	10	273	21,324	2.17	213	00:00:04
a 30	15	505	27,513	1.87	855	00:00:15
r 48	20	1,920	34,287	0.96	31	00:00:10
r 48	30	6,152	44,411	5.89	810	00:02:04
r 48	40	24,912	56,200	45.48	29,096	02:57:39
a 48	15	1,096	54,554	1.37	1,640	00:02:10
* a 48	20	2,849	74,420	38.61	15,89,59	05:00:00
* a 48	24	2,737	88,176	2.08	13,40,58	05:00:00
r 52	20	3,586	22,869	0.45	43	00:00:23
r 52	30	6,214	27,611	3.74	252	00:01:08
r 52	40	8,697	31,942	4.06	1,055	00:04:19
r 52	50	15,074	37,649	5.42	6,115	00:29:50
a 52	20	1,280	31,854	2.00	13,490	00:19:43
*a 52	26	2,577	42,162	2.40	14,64,19	05:00:00
(b) Results for L = 3						
r 21	15	5,488	9,505	11.12	1,575	00:01:24
r 21	17	3,152	10,048	2.15	489	00:00:24
r 21	20	7,179	10,982	3.12	4,776	00:04:41
**a 21	10	48,459	11,572	6.76	14,318	02:02:36
**a 21	11	56,306	11,605	4.52	16,604	03:41:42
r 30	15	41,281	18,278	3.39	4,433	01:22:52
**r 30	20	71,202	19,982	3.61	9,622	02:34:53
**r 30	25	62,966	21,296	3.96	9,788	02:19:40
a 30	10	16,702	18,491	2.30	6,750	00:35:13
*a 30	15	63,972	23,282	5.31	6,984	05:00:00
*r 48	20	65,829	27,860	4.23	4,196	05:00:00
**r 48	30	82,400	35,193	9.04	3,301	04:34:24
**r 48	40	87,271	43,227	10.84	3,688	04:15:30
*a 48	15	45,494	–	–	879	05:00:00
*a 48	20	35,193	–	–	155	05:00:00
*a 48	24	32,428	–	–	81	05:00:00
*r 52	20	76,674	19,358	5.17	3,127	05:00:00
**r 52	30	75,920	23,222	7.48	3,342	03:52:52
**r 52	40	78,823	26,419	7.49	3,351	04:22:07
**r 52	50	77,399	30,838	8.80	4,830	03:35:56
*a 52	20	38,898	–	–	521	05:00:00
*a 52	26	32,260	–	–	151	05:00:00

resolution of the problem is significantly easier when $L = 2$. It also shows that the flow-based and path-based formulations produce better results than the other formulations when $L = 2$. For $L = 3$, the path-based formulation outperforms the other formulations in terms of obtaining upper bounds and the aggregated formulation produces, in some cases, good results. The results also show that the separated cut formulation achieves poor results for both $L = 2$ and $L = 3$ and is, apparently, unusable from a practical point of view. Finally, the computational study shows that the problem seems to be easier when k increases, corroborating a similar observation previously made for the k-edge-connected subgraph problem.

The experiments conducted in this article show that the aggregated formulation is less effective in solving the kHNDP than the separated flow and path formulations. This result is quite surprising as, in general, aggregated formulations for network design problems (like the kECSP) outperform flow-based ones, especially for large size instances. In our case, this unusual result can be explained by the fact that CPLEX 12.5 uses several and effective tools for solving the separated flow and path formulations while not for the aggregated formulation. In fact, for problems formulated with an exponential number of constraints and when these contraints are handled using cutting plane algorithms (like st-cut constraints (4.1)) CPLEX does not use some of the improvement tools it uses for flow-based formulations. However, we believe that the aggregated formulation outperforms the separated flow and path formulations if we consider very large scale instances. For this, we would probably need to use other tools like additional valid inequalities to strengthen the aggregated formulation.

Also, this work indicates that some improvement may be needed in the resolution of the problem, especially for $L = 3$ (gaps relatively high) and for all the formulations. Hence, it would be interesting to use other techniques to solve the problem, like Benders decomposition-based algorithm (as in [5]), or improve the branch-and-cut algorithms using further valid inequalities in the cutting plane phase.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their comments that permitted us to considerably improve the presentation of the paper.

REFERENCES

[1] IBM, IBM ILOG CPLEX Optimization Studio 12.5 Documentation. Available at: http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.5.1/maps/ic-homepage.html.

[2] F. Barahona and A.R. Mahjoub, On two-connected subgraph polytopes, Discrete Math 147 (1995), 19–34.

[3] F. Bendali, I. Diarrassouba, M. Didi Biha, A.R. Mahjoub, and J. Mailfert, A Branch-and-cut algorithm for the k-edge connected subgraph problem, Networks 55 (2010), 13–32.

[4] 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.

[5] Q. Botton, B. Fortz, L. Gouveia, and M. Poss, Benders decomposition for the hop-constrained survivable network design problem, INFORMS J Comput 25 (2013), 13–26.

[6] S. Chopra, The k-edge connected spanning subgraph polyhedron, SIAM J Discrete Math 7 (1994), 245–259.

[7] C.R. Coullard, A.B. Gamble, and J. Lui, "The k-walk polyhedron", Advances in optimization and approximation, nonconvex optimization application 1, D-Z Du and J. Sun (Editors), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994, pp. 9–29.

[8] G. Dahl, The 2-hop spanning tree problem, Oper Res Lett 23 (1999), 21–26.

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

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

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

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

[13] I. Diarrassouba, V. Gabrel, L. Gouveia, P. Pesneau, and A.R. Mahjoub, Integer programming formulations for the k-edge-connected 3-hop-constrained network design problem, Technical report N^o 358, LAMSADE, Université Paris Dauphine, France, 2014.

[14] M. Didi Biha and A.R. Mahjoub, k-edge connected polyhedra on series-parallel graphs, Oper Res Lett 19 (1996), 71–78.

[15] M. Didi Biha and A.R. Mahjoub, Steiner k-edge connected subgraph polyhedra, J Comb Optim 4 (2000), 131–134.

[16] M. Didi Biha and A.R. Mahjoub, The k-edge connected subgraph problem I: Polytopes and critical extreme points, Linear Algebra Appl 381 (2004), 117–139.

[17] J. Fonlupt and A.R. Mahjoub, Critical extreme points of the 2-edge connected spanning subgraph polytope, Math Program 105 (2006), 289–310.

[18] B. Fortz, M. Labbe, and F. Maffioli, Solving the two-connected network with bounded meshes problem, Oper Res Lett 48 (2000), 866–877.

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

[20] H.N. Gabow, M.X. Goemans, E. Tardos, and D. Williamson, Approximating the smallest k-edge connected spanning subgraph by LP-rounding, Networks 53 (2009), 345–357.

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

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

[23] L. Gouveia, L. Patricio, and P.F. Sousa, "Compact models for hop-constrained node survivable network design: An application to MPLS," Telecommunications network planning: Innovations and pricing, network design and management, G. Anandaligam and S. Raghavan (Editors), Springer, 2005, pp. 167–180.

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

[25] M. Grötschel and C.L. Monma, Integer polyhedra arising from certain network design problems with connectivity constraints, SIAM J Discrete Math 3 (1990), 502–523.

[26] M. Grötschel, C.L. Monma, and M. Stoer, "Polyhedral approches to network survivability," Reliability of computer and communication networks 5, F. Roberts, F. Hwang, and C.L. Monma (Editors), Discrete Mathematics and Computer Science, AMS/ACM, 1991, pp. 121–141.

[27] M. Grötschel, C.L. Monma, and M. Stoer, Polyhedral and computational investigations arising for designing communication networks with high survivability requirements, Oper Res 43 (1995), 1012–1024.

[28] D. Huygens, M. Labbe, 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.

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

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

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

[32] H. Kerivin, A.R. Mahjoub, and C. Nocq, "(1,2)-survivable networks: Facets and branch-and-Cut," The sharpest cut, M. Grötschel (Editor), MPS-SIAM Series in Optimization, 2004, pp. 121–152.

[33] C.-W. Ko and C.L. Monma, Heuristics for designing highly survivable communication networks, Technical report, Bellcore, Morristown, NJ, 1989.

[34] R. Rardin and U. Choe, Tighter relaxations of fixed charge network flow problems, Technical report J-79-18, Georgia Institute of Technology, Atlanta, Georgia, 1979.

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