Laboratoire d'Analyse et Modélisation de Systèmes pour d'Aide à la Décision
UMR 7243

CAHIER DU LAMSADE

358

Juillet 2014

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

I. Diarrassouba, V. Gabrel, L. Gouveia, A. R. Mahjoub, P. Pesneau

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

I. Diarrassouba[1], V. Gabrel[2], L. Gouveia[3], A. R. Mahjoub[2], P. Pesneau[4]

1. Laboratoire LMAH, Université du Havre, 25 Rue Philippe Lebon, 76600, Le Havre, France
ibrahima.diarrassouba@univ-lehavre.fr
2. Laboratoire LAMSADE, Université Paris Dauphine, Place du Maréchal De Lattre de Tassigny, 75775, Paris Cedex 16, France
{gabrel,mahjoub}@lamsade.dauphine.fr
3. Departamento de Estatística e Investigação Operacional - Centro de Investigação Operacional, Faculdade de Ciências, Universidade de Lisboa, Portugal
legouveia@fc.ul.pt
4. University of Bordeaux, INRIA Bordeaux - Sud-Ouest, IMB UMR 5251, France,
pierre.pesneau@math.u-bordeaux1.fr

Abstract. In this paper, 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 pairs $\{s, t\} \in D$.
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 compararison of these formulations in terms of LP-bound. Finally, these formulations are tested using CPLEX and compared in a computational study for the case $k = 3$.

Keywords. Survivable network, edge-disjoint paths, flow, hop-constrained path, integer programming formulation.

1 Introduction

Let $G = (V, E)$ be an undirected graph and $D \subseteq V \times V$, a set of pairs of nodes, called *demands*. 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 $k \geq 1$, 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, networks that are still functional after the failure of certain links. As pointed out in [34] (see also [32]), 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 telecommunications network. In fact, the alternative paths could be too long to guarantee an effective routing. In data networks, such as Internet, the elongation of the route

of the information could cause a strong 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 guaranties exactly the needed quality of the alternative routes. Moreover, in a telecommunication networks, usually several commodities have to be routed in the network between pairs of terminals. In order to guaranty 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. In particular, the associated polytope has recieved a special attention. In [31], Huygens et al. study the kHNDP for $|D| = 1$, $k = 2$ 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 [13], Dahl et al. study the kHNDP when $|D| = 1$, $L = 2$ 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 [10], Dahl considers the kHNDP for $|D| = 1$, $k = 1$ and $L = 3$. He gives a complete description of the dominant of the associated polytope Dahl and Gouveia [11] consider the directed hop-constrained path problem. They describe valid inequalities and characterize the associated polytope when $L \leq 3$. Huygens and Mahjoub [30] study the kHNDP when $|D| = 1$, $k = 2$ 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 presented 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 in the case 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 [20], [21] and [25]. Here, the hop-constraints limit to a positive integer H the number of links between the root and any terminal in the network. Dahl [9] study the problem where $H = 2$ from a polyhedral point of view and give a complete description of the associated polytope when the graph is a wheel. Finally, Huygens et al. [29] 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 order to limit the length of the routing, requires that each link of the network belongs to a ring (cycle) of bounded length. In [19], Fortz et al. consider the 2-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 [18], 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 of 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 [26] and Grötschel et al. [27, 28] 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 [15] study that problem and give a complete description of the associated polytope when the graph is series-parallel. In [16], 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 [14], extend the results of Fonlupt and Mahjoub [17] to the case $k \geq 3$ and introduce some graph reduction operations. Kerivin et al. [33] 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 [3] devise a Branch-and-Cut algorithm using the reduction operations of [14] and give some computational results for $k = 3, 4, 5$. A complete survey on the k-edge connected subgraph problem can be found in [32].

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 $k \geq 1$. One approach (called "separated" approach) uses a layered graph for each hop-constrained subproblem and the other (called "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.

We denote by $G = (V, E)$ an undirected graph with node set V and edge set E. 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. 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 are in W and destination are 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.

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.

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 $L \in \{2, 3\}$. Given an edge subset $F \subseteq E$ which induces a solution of the kHNDP, that is the subgraph (V, F) contains k-edge-disjoint L-st-paths for every $\{s, t\} \in D$, then, clearly, its incidence vector x^F satisfies the following inequalities.

$$x(\delta(W)) \geq k \text{ for all } st\text{-cut}, \{s, t\} \in D, \tag{2.1}$$
$$x(e) \geq 0 \qquad \text{for all } e \in E, \tag{2.2}$$
$$x(e) \leq 1 \qquad \text{for all } e \in E. \tag{2.3}$$

In [10], Dahl considers the problem of finding a minimum cost path between two given terminal nodes s and t of length at most L. He studied the polyhedron (the L-path polyhedron) associated with that problem and introduced a class of inequalities as follows.
Let $\{s, t\} \in D$ and a partition $(V_0, V_1, ..., V_{L+1})$ of V such that $s \in V_0$ and $t \in V_{L+1}$, and $V_i \neq \emptyset$ for all $i = 1, ..., L$. 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 a L-st-path-cut inequality with $L = 3$ and, $V_0 = \{s\}$ and $V_{L+1} = \{t\}$.

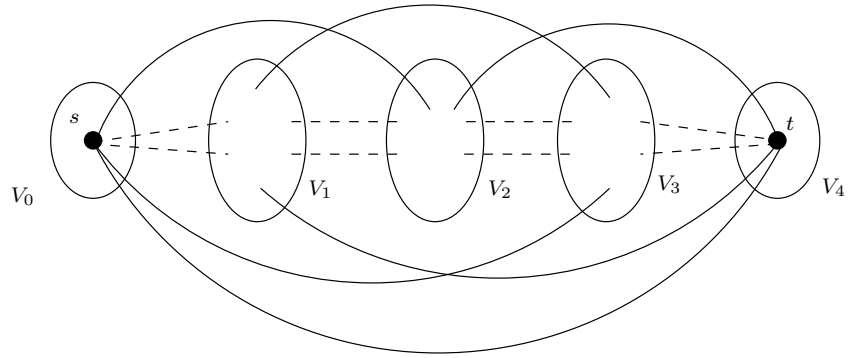


FIGURE 1 – Support graph of a L-st-path-cut with $L = 3$, $V_0 = \{s\}$, $V_{L+1} = \{t\}$ and T formed by the solid edges

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. [29] showed that the kHNDP can be formulated as an integer program using the design variables when $L = 2, 3$.

Theorem 2.1 [29] *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*

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

Formulation (2.5) is called *natural formulation* and is denoted by $k\text{HNDP}_{Nat}$. In [29], Huygens et al. studied the polytope associated with this formulation and introduced some facet-defining inequalities for the problem. They also develop a Branch-and-Cut algorithms for the kHNDP when $k = 2$ and $L = 2, 3$.

In [4], Bendali et al. studied the kHNDP in the case where $k \geq 2$, $L = 2, 3$ and $|D| = 1$. They studied the polyhedral structure of that formulation and gave necessary and sufficient conditions under which the L-st-path-cut inequalities (2.4) define facets. In particular, they showed that an L-st-path-cut inequality induced by a partition $(V_0, ..., V_{L+1})$, 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 $\pi = (V_0, ..., V_{L+1})$ be a partition of V with $s \in V_0$ and $t \in V_{L+1}$. The L-st-path-cut inequality (2.4) induced by π defines a facet of the kHNDP 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 $(V_0, ..., V_{L+1})$ 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 $(V'_0, V'_1, ..., V'_L, V'_{L+1})$ with $V_0 = \{s\}$ and $V_{L+1} = \{t\}$. Therefore, in the remain of the paper, the only L-st-path-cut that we will consider are those induced by partitions of the form $(\{s\}, V_1, ..., V_L, \{t\})$.

2.2 Undirected path-based formulation

In [35], Orlowski and Wessaly studied the survivable network design problem with hop constraint and where the paths must satisify some addiditional constraints. These constraints are related

to the protection scheme used to make the network survivable. Every solution of the problem can be seen as a collection of feasible hop-constrained st-paths in the graph G, for every demand $\{s,t\} \in D$. Here feasible means that these paths are consistent with respect to the protection scheme.

In [35], the authors considered as protection scheme the case where a given amount of flow must be routed between the terminals of every demand, using hop-constrained paths. They presented an integer programming formulation for this problem which uses path variables.

This formulation can be easily extended to the kHNDP. In fact, a solution of the kHNDP can be modelled by a collection of L-st-paths of G, for all $\{s,t\} \in D$. Using the remark above, one can model the kHNDP using path variables in the graph G as follows.

Let $\mathcal{P}_{st}$ be the set of L-st-paths of G. Given an edge subset $F \subseteq E$, we let $\mu_F^{st}(P)$, $\{s,t\} \in D$, be the $0-1$-variable which equals to 1 if the path $P \in \mathcal{P}_{st}$ is in the graph induced by F and 0 otherwise.
If F induces a solution of the kHNDP, then the following inequalities are satisified by its incidence vector x^F and $\mu_F^{st}(P)$, $P \in \mathcal{P}_{st}$, $\{s,t\} \in D$.

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

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

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

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

where $\mu^{st} \in \mathbb{R}^{\mathcal{P}_{st}}$ and $x \in \mathbb{R}^E$.

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

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

Formulation (3.12) is called *Undirected Path Formulation* and denoted by $k\text{HNDP}_{Path}^U$. Note that, in many combinatorial optimization problems, path-based formulations imply an exponential number of variables, since the number of paths in a graph is, in general, exponential. This leads 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 $k\text{HNDP}_{Path}^U$ is polynomial and its linear relaxation can be solved by enumerating all the paths 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 [11] 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 when $k \geq 2$ and $|D| \geq 2$. The formulation is described as follows.

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). Let $F \subseteq E$ be a subgraph of G Given a demand $\{s, t\} \in D$, we let $f^{st} \in \mathbb{R}^A$ be 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.

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

$$\sum_{(u,v) \in A, u \neq s} f^{st}(u, v) \leq \sum_{(v,t) \in A} f^{st}(v, t), \text{ for all } v \in V \setminus \{t\}, \tag{2.12}$$

$$f^{st}(u, s) = 0, \text{ for every arc } (s, u) \in A, \ u \neq s, \text{for all } \{s, t\} \in D, \tag{2.13}$$

$$f^{st}(t, v) = 0, \text{ for every arc } (v, t) \in A, \ v \neq t, \text{for all } \{s, t\} \in D, \tag{2.14}$$

$$f^{st}(u, v) + f^{st}(v, u) \leq x(uv), \text{ for all } uv \in E, \tag{2.15}$$

$$\left. \begin{array}{l} f^{st}(u, v) \\ f^{st}(v, u) \end{array} \right\} \geq 0, \text{ for all } \{s, t\} \in D, \ uv \in E, \tag{2.16}$$

$$x(uv) \leq 1, \text{ for all } uv \in E. \tag{2.17}$$

Note that Constraints (2.13) and (2.14) remove the flow variables for every arc entering node s and leaving node t, for all $\{s, t\} \in D$. 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 (2.15) 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 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.

Inequalities (2.12) indicate, for a node $v \neq t$, if no flow goes from v to t, then no st-path going through v is used. This implies the existence of L-st-paths in the solution.
It can be easily seen that when $L = 2$ and $L = 3$, inequalities (2.11)-(2.17), together with integrality constraints on variables x and f^{st} completely describes the solution set of the kHNDP. Thus, we have the following compact formulation for the problem.

Theorem 2.4 *The kHNDP for $L = 2, 3$ is equivalent to the following integer program*

$$\begin{array}{c} min\{cx; \text{ subject to } (2.11) - (2.17), \ x \in \mathbb{Z}_+^E, f^{st} \in \mathbb{Z}_+^A, \\ \text{for all } \{s, t\} \in D\}. \end{array} \tag{2.18}$$

This formulation will be called *Undirected Flow Formulation* and denoted by $k\text{HNDP}^U_{Flow}$.

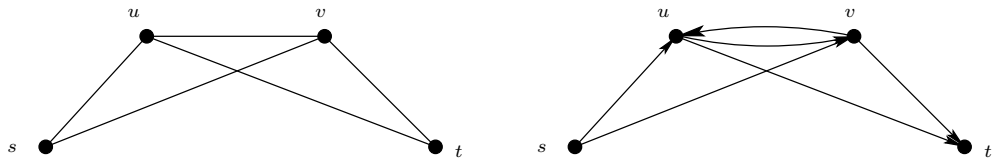


FIGURE 2 – Illustration of the linking inequalities for an edge uv with $u, v \neq s, t$.

3 Demand decomposition based formulations

Let $G = (V, E)$ be an undirected graph, $L \in \{2, 3\}$, $k \geq 2$, two integers, and D a set of demands. In this section, we introduce three integer programming formulations for the kHNDP where we use a directed layered graph to model each hop-constrained subproblem. These formulation will be called *separated formulations*.
As we shall show, in these directed graphs, every path 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 [21] and subsequently used in other related papers (eg. [5] and [23]) for more general hop-constrained newtork design problems. However, the directed graphs proposed here are different from the ones suggested in [21] when $L = 3$.

3.1 Graph transformation

Let $\{s, t\} \in D$ and $\widetilde{G}_{st} = (\widetilde{V}_{st}, \widetilde{A}_{st})$ be the directed graph obtained from G using the following procedure.
Let $N_{st} = V \setminus \{s, t\}$, N'_{st} be a copy of N_{st} and $\widetilde{V}_{st} = N_{st} \cup N'_{st} \cup \{s, t\}$. The copy in N'_{st} of a node $u \in N_{st}$ will be denoted by u'. To each edge $e = st \in E$, we associate an arc (s, t) in $\widetilde{G}_{st}$ with capacity 1. With each edge $su \in E$ (resp. $vt \in E$), we associate in $\widetilde{G}_{st}$ the arc (s, u), $u \in N_{st}$ (resp. (v', t), $v' \in N'_{st}$) with capacity 1. With each node $u \in V \setminus \{s, t\}$, we associate in $\widetilde{G}_{st}$ k arc (u, u') with an infinit capacity. Finally, if $L = 3$ we associate with each edge $uv \in E \setminus \{s, t\}$, two arcs (u, v') and (v, u'), with $u, v \in N_{st}$ and $u', v' \in N'_{st}$ with capacity 1 (see Figure 3 for an illustration with $L = 3$).

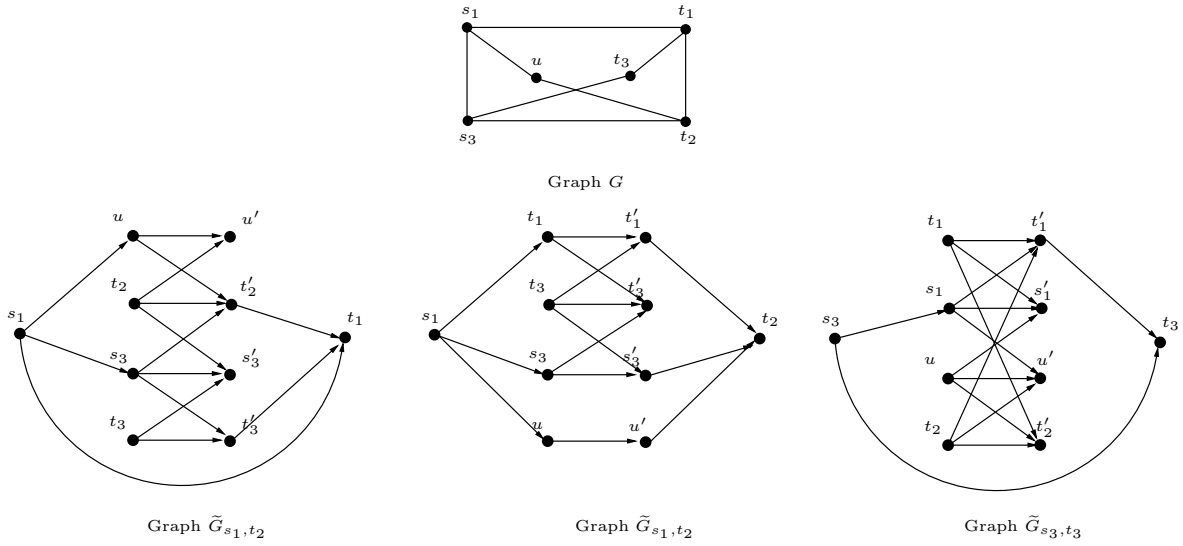


FIGURE 3 – Construction of graphs $\widetilde{G}_{st}$ with $D = \{\{s_1, t_1\}, \{s_1, t_2\}, \{s_3, t_3\}\}$ for $L = 3$

Note that each graph $\widetilde{G}_{st}$ contains $|\widetilde{V}_{st}| = 2|V| - 2$ $(= |N_{st} \cup N'_{st} \cup \{s,t\}|)$ nodes and $|\widetilde{A}_{st}| = |\delta(s)| + |\delta(t)| - |[s,t]| + |V| - 2$ arcs if $L = 2$ and $|\widetilde{A}_{st}| = 2|E| - |\delta(s)| - |\delta(t)| + |[s,t]| + |V| - 2$ arcs if $L = 3$, for all $\{s,t\} \in D$.
Given a demand $\{s,t\}$, the associated graph $\widetilde{G}_{st} = (\widetilde{V}_{st}, \widetilde{A}_{st})$, and for an edge $e \in E$, we denote by $\widetilde{A}_{st}(e)$ the set of arcs of $\widetilde{G}_{st}$ corresponding to the edge e.
It is not hard to see that $\widetilde{G}_{st}$ does not contain any circuit. Also, observe that any st-dipath in $\widetilde{G}_{st}$ is of length no more than 3. Moreover each L-st-path in G corresponds to an st-dipath in $\widetilde{G}_{st}$ and conversely. In fact, if $L \in \{2,3\}$, every 3-st-path (s,u,v,t), with $u \neq v$, $u,v \in V \setminus \{s,t\}$, corresponds to an st-dipath in $\widetilde{G}_{st}$ of the form (s,u,v',t) with $u \in N_{st}$ and $v' \in N'_{st}$. Every 2-st-path (s,u,t), $u \in V \setminus \{s,t\}$, corresponds to an st-dipath in $\widetilde{G}_{st}$ of the form (s,u,u',t).
We have the following lemma.

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 st-dipaths in $\widetilde{G}_{st}$ are arc-disjoint.

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

Proof. We will suppose, w.l.o.g., that $L = 3$. The proof is similar for $L = 2$.
i) Let P_1 and P_2 be two edge-disjoint 3-st-paths of G. Let $\widetilde{P}_1$ and $\widetilde{P}_2$ be the two st-dipaths of $\widetilde{G}_{st}$ corresponding to P_1 and P_2, respectively. We will show that $\widetilde{P}_1$ and $\widetilde{P}_2$ are arc-disjoint. Let us assume that this is not the case. Then $\widetilde{P}_1$ and $\widetilde{P}_2$ intersect on an arc a of the form either (s,t), (s,u), (v',t), (u,v') or (u,u'), with $u \in N_{st}$ and $v' \in N'_{st}$. If a is of the form (s,t), (s,u), (v',t) or (u,v'), then it corresponds to an edge e of G of the form either st, su, vt or uv. This implies that P_1 and P_2 contain both the edge e, a contradiction. Thus, $\widetilde{P}_1$ and $\widetilde{P}_2$ intersect on an arc of the form (u,u'), with $u \in N_{st}$. As the st-dipaths of $\widetilde{G}_{st}$ contain at most 3 arcs, $\widetilde{P}_1$ and $\widetilde{P}_2$ are of the form (s,u,u',t). But this implies that P_1 and P_2 contain simulataneously the edges su and ut, a contradiction.
ii) Now consider two arc-disjoint st-dipaths $\widetilde{P}_1$ and $\widetilde{P}_2$ of $\widetilde{G}_{st}$ and let P_1 and P_2 be the corresponding 3-st-paths of G. Suppose that $P_1 \cap P_2 \neq \emptyset$. If P_1 and P_2 intersect on an edge $e = st$, then $\widetilde{P}_1$ and $\widetilde{P}_2$ also contain the arc (s,t), a contradiction. If P_1 and P_2 intersect on an edge of the form su, $u \in V \setminus \{s,t\}$ (resp. vt, $v \in V \setminus \{s,t\}$), then, as before, both $\widetilde{P}_1$ and $\widetilde{P}_2$ contain the arc (s,u) (resp. (v',t)), yielding a contradiction. Now if P_1 and P_2 intersect on an edge of the form uv, $u,v \in V \setminus \{s,t\}$, then $\widetilde{P}_1$ and $\widetilde{P}_2$ contain the arcs (u,v') and (v,u') of $\widetilde{G}_{st}$. Since $\widetilde{P}_1$ and $\widetilde{P}_2$ are arc-disjoint, $\widetilde{P}_1$ contains say (u,v') and $\widetilde{P}_2$ (v,u'). Thus $\widetilde{P}_1$ and $\widetilde{P}_2$ are respectively of the form (s,u,v',t) and (s,v,u',t). This implies that $P_1 = (su, uv, vt)$ and $P_2 = (sv, vu, ut)$. Let $P'_1 = (su, ut)$ and $P'_2 = (sv, vt)$. Clearly P'_1 and P'_2 are edge-disjoint and of length 2. Thus, we associate $\widetilde{P}_1$ and $\widetilde{P}_2$ with them, respectively, which ends the proof of the lemma. □

As a consequence of Lemma 3.1, for $L \in \{2,3\}$ 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 $\widetilde{H}_{st}$, $\{s,t\} \in D$, 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 $\{s,t\} \in D$.

3.2 Separated Flow Formulation

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

Let $F \subseteq E$ be a subgraph of G. 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

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

and

$$f_a^{st} \leq x(e), \text{ for all } a \in \widetilde{A}_{st}(e), \ \{s,t\} \in D, \ e \in E. \tag{3.2}$$
$$f_a^{st} \leq 1, \text{ for all } a = (u,u'), \ u \in V \setminus \{s,t\}, \ \{s,t\} \in D. \tag{3.3}$$
$$f_a^{st} \geq 0, \text{ for all } a \in \widetilde{A}_{st}, \ \{s,t\} \in D. \tag{3.4}$$

Also x^F satisfies the following inequalities

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

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 capacity on every arc corresponding to e is 0. Inequalities (3.4)-(3.5) are called *trivial inequalities*.
We have the following theorem which will be given without proof.

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

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

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

3.3 Separated Path Formulation

Since the well known work by Rardin and Choe [36], it is known that flows can also be modelled by paths. Every solution of the problem can hence be represented by directed st-paths in graphs $\widetilde{G}_{st}$, for all $\{s,t\} \in D$.

Let $\{s,t\} \in D$ and $\mathcal{P}_{st}$ be the set of st-dipaths in $\widetilde{G}_{st}$. Given a directed path $\widetilde{P} \in \mathcal{P}_{st}$, we denote by $\Gamma_{\widetilde{P}}^{st} = (\gamma_{\widetilde{P},a}^{st})_{a \in \widetilde{A}_{st}}$ the *incidence vector* of $\widetilde{P}$ that is the vector given by $\gamma_{\widetilde{P},a}^{st} = 1$ if $a \in \widetilde{P}$ and $\gamma_{\widetilde{P},a}^{st} = 0$ otherwise. Given a subgraph H of G and a set of subgraphs $\widetilde{H}_{st}$ of $\widetilde{G}_{st}$, $\{s,t\} \in D$, we let $\mu_{\widetilde{H}_{st}}^{st} \in \mathbb{R}^{\mathcal{P}_{st}}$ be the 0-1 vector such that $\mu_{\widetilde{H}_{st}}^{st}(\widetilde{P}) = 1$ if $\widetilde{P} \in \mathcal{P}_{st}$ is in $\widetilde{H}_{st}$ and $\mu_{\widetilde{H}_{st}}^{st}(\widetilde{P}) = 0$ otherwise.

If H induces a solution of the kHNDP, then x^H and $(\mu^{st}_{\widetilde{H}_{st}})_{\{s,t\}\in D}$ satisfy the following inequalities.

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

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

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

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

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

where $\mu^{st}\in\mathbb{R}^{\mathcal{P}_{st}}$ and $x\in\mathbb{R}^E$.
Inequalities (3.7) express the fact that the subgraph $\widetilde{H}_{st}$ 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 inter program*

$$\begin{aligned} min\{cx;\ &\textit{subject to } (3.8)-(3.11),\ x\in\mathbb{Z}^E_+, \mu^{st}\in\mathbb{Z}^{\mathcal{P}_{st}}_+,\\ &\textit{for all } \{s,t\}\in D\}. \end{aligned} \tag{3.12}$$

Formulation (3.12) is called *separated path formulation* and denoted by $k\text{HNDP}^{Sep}_{Path}$. Remark that for a 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 non trivial inequalities is

$$\sum_{\{s,t\}\in D}|\delta(s)| + \sum_{\{s,t\}\in D}|\delta(t)| - \sum_{\{s,t\}\in D}|[s,t]| - d(n-3)$$

if $L=2$ and

$$2d|E| - \sum_{\{s,t\}\in D}|\delta(s)| - \sum_{\{s,t\}\in D}|\delta(t)| + \sum_{\{s,t\}\in D}|[s,t]| - d(n-3)$$

if $L=3$, which is polynomial.
Hence, as for the undirected path-based formulation $k\text{HNDP}^{U}_{Path}$, its linear relaxation can hence be solved in polynomial time using linear programming and by enumerating all the variables and constraints of the problem.

Still using Rardin and Choe [36], 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), since L-st-paths in the original graph G corresponds 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 of k under the arc capacities given by x. By the Max flow-Min cut Theorem, this 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.

By Corollary 3.1, if a subgraph H of G induces a solution of the kHNDP, then the subgraph $\widetilde{H}_{st}$ contains at least k arc-disjoint st-dipaths, for all $\{s,t\} \in D$, and conversely. Thus, for any solution H of the kHNDP, the following inequalities are satisfied by $y_{st}^{\widetilde{H}_{st}}$, for all $\{s,t\} \in D$,

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

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

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

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

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

where $y_{st} \in \mathbb{R}^{\widetilde{A}_{st}}$ for all $\{s,t\} \in D$ and $x \in \mathbb{R}^E$.

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 called *trivial inequalities*.

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

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

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

This formulation is called *separated cut formulation* and denoted by $k\text{HNDP}_{Cut}^{Sep}$. It contains

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

variables if $L = 2$ and

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

variables if $L = 3$ (remind that $d = |D|$).
However, the number of constraints is exponential since the directed st-cuts are in exponential number in $\widetilde{G}_{st}$, for all $\{s,t\} \in D$. As it will turn out in Section 6, its linear programming relaxation can be solved in polynomial time using a cutting plane algorithm.

In the next section, we introduce another 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 kHNDP

Let $G = (V, E)$ be an undirected graph, $L \in \{2, 3\}$, $k \geq 2$ two integers, and D the demand set. 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.

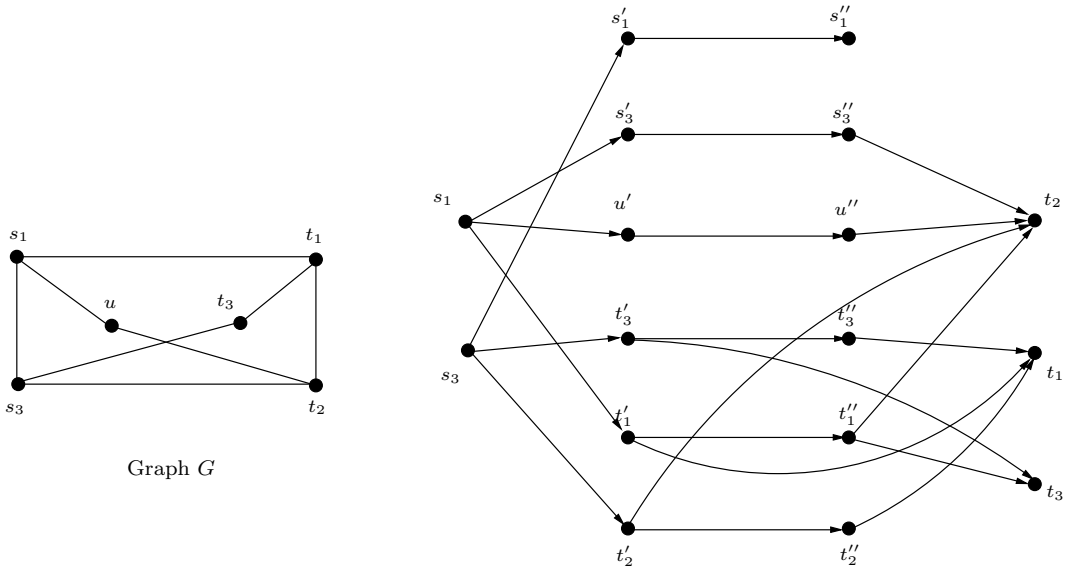
In this section, we will introduce a new formulation for the kHNDP which is supported by a directed graph $\widetilde{G} = (\widetilde{V}, \widetilde{A})$ obtained from G as follows. Let N' and N'' be two copies of V. We denote by u' and u'' the nodes of N' and N'' corresponding to a node $u \in V$. Let $\widetilde{V} = S_D \cup N' \cup N'' \cup T_D$. For every node $u \in V$, we add in $\widetilde{G}$ k arc (u', u''). For each $\{s, t\} \in D$, with $s \in S_D$ and $t \in T_D$, we apply the following procedure.

i) For an edge $e = st$, we add in $\widetilde{G}$ an arc (s, t') and k arc (t', t) ;

ii) For an edge $su \in E$, $u \in V \setminus \{s, t\}$, we add an arc (s, u') in $\widetilde{G}$;

iii) For an edge $vt \in E$, $v \in V \setminus \{s, t\}$, we add an arc (v'', t).

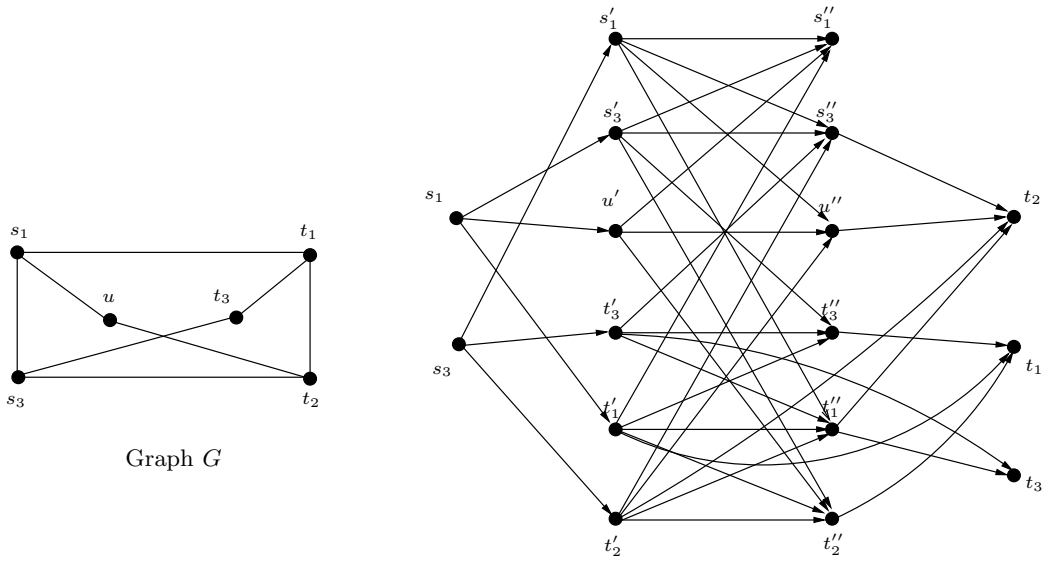
If $L = 3$, for each edge $e = uv \in E$, we also add two arcs (u', v'') and (v', u'') (see Figures 4 and 5 for examples with $L = 2$ and $L = 3$).



Graph G

Graph $\widetilde{G}$

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



Graph G

Graph $\widetilde{G}$

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

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

If $\widetilde{G} = (\widetilde{V}, \widetilde{A})$ is the graph 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, correponds 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 remain 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 $\{s_1, t\}, ..., \{s_p, t\}$, with $p \geq 2$, then one may replace in T_D t by p nodes $t_1, ..., t_p$ and in D each demand $\{s_i, t\}$ by $\{s_i, t_i\}$, $i = 1, ..., p$.

We have the following result.

Lemma 4.1 *Let $L \in \{2, 3\}$. If each node $t \in T_D$ appears in exacly 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.*

Proof. The proof will be given for $L = 3$. It follows the same lines for $L = 2$.

i) Let $\{s, t\} \in D$, P_1 and P_2 be two edge-disjoint 3-st-paths and $\widetilde{P}_1$ and $\widetilde{P}_2$ be the two st-dipaths of $\widetilde{G}$ corresponding to P_1 and P_2. We will show that $\widetilde{P}_1$ and $\widetilde{P}_2$ are arc-disjoint. Suppose the

contrary that is $\widetilde{P}_1$ and $\widetilde{P}_2$ intersect on an arc $a \in \widetilde{A}$ of the form either (s,t'), (s,u'), (v'',t), (u',v'') or (u',u''), with $u' \in N'$ and $v'' \in N''$. If a is of the form (s,t'), (s,u'), (v'',t) or (u',v''), then it corresponds to an edge e of G of the form either st, su, vt or uv. It then follows that P_1 and P_2 both contain edge e, a contradiction. If $\widetilde{P}_1$ and $\widetilde{P}_2$ intersect on an arc of the form (u',u''), then they also contain arcs of the form (s,u') and (u'',t). Thus, P_1 and P_2 also contain simultaneously the edges su and ut, a contradiction. Thus, $\widetilde{P}_1$ and $\widetilde{P}_2$ are arc-disjoint.
ii) Let $\widetilde{P}_1$ and $\widetilde{P}_2$ be two arc-disjoint st-dipaths of $\widetilde{G}$ and suppose that P_1 and P_2, the 3-st-paths of G corresponding to $\widetilde{P}_1$ and $\widetilde{P}_2$, are not edge-disjoint. Thus P_1 and P_2 intersect on edges of the form either st, su, vt or uv, with $u, v \neq s, t$.
If both P_1 and P_2 contain edge st, then each path $\widetilde{P}_1$ and $\widetilde{P}_2$ contains at least one arc among those corresponding to st in $\widetilde{G}$, that is (s,t'), (s',t'') or (t',s''). If $\widetilde{P}_1$ and $\widetilde{P}_2$ contain (s',t''), then they should also contain arc (s,s'). Since $[s,s'] = \emptyset$, this is impossible. In a similar way, we show that $\widetilde{P}_1$ and $\widetilde{P}_2$ cannot contain (t',s''). Hence, $\widetilde{P}_1$ and $\widetilde{P}_2$ both contain arc (s,t'), a contradiction.
If P_1 and P_2 intersect on su, then each path $\widetilde{P}_1$ and $\widetilde{P}_2$ contains either (s,u'), (s',u'') or (u',s''). Since $[s,s'] = \emptyset = [s'',t]$, $\widetilde{P}_1$ and $\widetilde{P}_2$ should both use arc (s,u'), a contradiction.
If P_1 and P_2 intersect on vt, then $\widetilde{P}_1$ and $\widetilde{P}_2$ contain either (v',t''), (t',v'') or (v'',t). As $[t'',t] = \emptyset$, $\widetilde{P}_1$ and $\widetilde{P}_2$ cannot use arc (v',t''). Moreover, if $\widetilde{P}_1$ or $\widetilde{P}_2$ contains (t',v''), then it also contains arc (v'',t). Hence, $\widetilde{P}_1$ and $\widetilde{P}_2$ both contain arc (v'',t), a contradiction.
In consequence, $P_1 \cap P_2 = \{uv\}$, $u, v \neq s, t$. This implies that $\widetilde{P}_1$ and $\widetilde{P}_2$ are respectively of the form $(su', u'v'', v''t)$ and $(sv', v'u'', u''t)$, and $P_1 = (su, uv, vt)$ and $P_2 = (su, vu, ut)$. Let $P'_1 = (su, ut)$ and $P'_2 = (sv, vt)$. Clearly P'_1 and P'_2 are edge-disjoint. Since they are of length 2, we simply associate $\widetilde{P}_1$ and $\widetilde{P}_2$ with them, which ends the proof of the lemma. □

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 we have the following corrollary.

Corollary 4.1 *Let H be a subgraph of G and $\widetilde{H}$ the subgraph of $\widetilde{G}$ obtained by considering all the arcs of $\widetilde{G}$ corresponding to the edges of H toghether 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 (kDNDP). 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 kDNDP.*

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. Let $x \in \mathbb{R}^E$ and $y \in \mathbb{R}^{\widetilde{A}}$. If $\widetilde{F} \subseteq \widetilde{A}$ induces a solution of the kDNDP and $F \subseteq E$ is the set of edges of G corresponding to the arcs of $\widetilde{F}$, then x^F and $y^{\widetilde{F}}$ satisfy the following inequalities

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

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

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

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

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*

$$min\{cx;\ \textit{subject to } (4.1) - (4.4),\ x \in \mathbb{Z}^E_+, y \in \mathbb{Z}^{\widetilde{A}}_+\}. \tag{4.5}$$

Formulation (4.5) will be called *aggregated formulation* and 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 in $\widetilde{H}$ if e is not taken in H.
This formulation contains $|E| + |\widetilde{A}| = |E| + |V| + \sum_{s \in S_D} |\delta(s)| + \sum_{t \in T_D} |\delta(t)|$ variables if $L = 2$ and $|E| + |\widetilde{A}| = 3|E| + |V| + \sum_{s \in S_D} |\delta(s)| + \sum_{t \in T_D} |\delta(t)|$ variables if $L = 3$. The number of constraints is exponential since the *st*-dicuts are in exponential 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) so is.
In the next section, we present a comparitive study of 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 are compact and hence have a polynomial number of variables and constraints, while the undirected and separated path formulations have an exponential number of variables. However, the aggregated graph contains less arcs than the union of the separated graphs. Thus, the aggregated formulation contains less variables than the separated formulations, in particular, the separated cut and flow 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 between them. In fact, as mentionned in Section 3.2, by Max flow-Min cut Theorem, the separated flow and cut formulations produce the same LP-bound. Also, by Rardin and Choe [36], 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, we show that the undirected flow formulation can be obtained by projecting the undirected path formulation.

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

Proof. To prove the result, we use the same argument as [11] (see proof of Proposition 2.2). We first augment the undirected path formulation by additonnal flow variables and then show that the projection of Formulation (2.10) of these extra variables is defined by constraints (2.11)-(2.17). W.o.l.g, we will suppose that $L = 3$.

First, let $G' = (V, A)$ be the directed graph obtained from G by replacing every edge uv by two arcs (u, v) and (v, u). Let $f^{st} \in \mathbb{R}^A$, be a $0-1$ vector, for a given demand $\{s, t\} \in D$. For all $\{s, t\} \in D$, we let $f^{st}(a) = 0$ for every arc a entering s and leaving t. For a given arc $(i, j) \in A$, with $i, j \neq s, t$, a 3-st-path containing (i, j) is of the form (s, i, j, t). We will denote by μ^{st}_{ij}, the path variable corresponding to that path. For an arc (s, i), $i \in V \setminus \{s, t\}$, a 3-$st$-path containing the arc (s, i) is of the form (s, i, t) and the path variable corresponding to that path will be denoted by μ^{st}_i. Finally, the path (s, t) corresponding to the arc between s and t, will be denoted by μ^{st}_{st}. We let $\mu^{st}_{ij} = f^{st}_{ij}$ for all arc $(i, j) \in A$. It is not hard to see that the undirected path formulation can be augmented by f^{st} variables in the following way.

$$\sum_{P \in \mathcal{P}_{st}} \mu^{st}(P) = k, \text{ for all } \{s, t\} \in D \tag{5.1}$$

$$\sum_{P \in \mathcal{P}_{st},\ a \in P} \mu^{st}(P) = f^{st}_a, \text{ for all } a \in [s, u] \cup [v, t],\ \{s, t\} \in D, \tag{5.2}$$

$$\sum_{P \in \mathcal{P}_{st},\ uv \in P} \mu^{st}(P) = f^{st}_{uv} + f^{st}_{vu}, \text{ for all edge } uv \in E, \text{ with } u, v \neq s, t,\ \{s, t\} \in D, \tag{5.3}$$

$$f^{st}_{uv} + f^{st}_{vu} \leq x(uv), \text{ for all edge } uv \in E, \tag{5.4}$$

$$x(uv) \leq 1, \text{ for all edge } uv \in E, \tag{5.5}$$

$$\mu^{st}(P) \geq 0, \text{ for every } P \in \mathcal{P}_{st}, \{s, t\} \in D. \tag{5.6}$$

With the new notations, Inequalities (5.1) and (5.2) become

$$\mu^{st}_{st} = f^{st}_{st}, \text{for all } \{s, t\} \in D, \tag{5.7}$$

$$\mu^{st}_{ij} + \mu^{st}_{ji} = f^{st}_{ij} + f^{st}_{ji}, \text{for all } (i, j) \in A,\ i, j \neq s, t, \{s, t\} \in D, \tag{5.8}$$

$$\mu^{st}_{st} + \sum_{i \in V \setminus \{s,t\}} \mu^{st}_i + \sum_{\substack{(i,j) \in A \\ i,j \neq s,t}} \mu^{st}_{ij} = k, \text{for all } \{s, t\} \in D, \tag{5.9}$$

$$\mu^{st}_i + \sum_{\substack{(i,j) \in A \\ i,j \neq s,t}} \mu^{st}_{ij} = f^{st}_{si}, \text{ for all } i \in V \setminus \{s, t\},\ \{s, t\} \in D, \tag{5.10}$$

$$\mu^{st}_j + \sum_{\substack{(j,i) \in A \\ i,j \neq s,t}} \mu^{st}_{ji} = f^{st}_{jt}, \text{ for all } j \in V \setminus \{s, t\},\ \{s, t\} \in D. \tag{5.11}$$

By combining (5.7), (5.8) and (5.10), we obtain

$$\mu^{st}_i = f^{st}_{si} - \sum_{\substack{(i,j) \in A \\ i,j \neq s,t}} \mu^{st}_{ij}, \text{ for all } i \in V \setminus \{s, t\},\ \{s, t\} \in D. \tag{5.12}$$

By substituting μ^{st}_i by (5.12) in (5.11), we get the flow conservation constraint (2.11) for every node $i \neq s, t$, for all $\{s, t\} \in D$. Similarly, by combining (5.12) and (5.9), we obtain the flow conservation constraint (2.11) associated with node t, and by combining (5.11) and (5.9), we get

the flow conservation constraint associated with node s.

Now, by combining $\mu_j^{st} \geq 0$ with (5.11), we get Inequalities (2.12).
Thus, together with Inequalities (5.4)-(5.6), we obtain the undirected flow formulation (2.18), and the result holds. □

Theorem 5.2 implies that the linear programming relaxation of Formulations (2.18) 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 permits to associate 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 $(V_0, ..., V_{L+1})$ 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 proof of Lemma 4.1 in [4]. □

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

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

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 $\overline{x}(C) \leq \overline{y}_{st}(\widetilde{C})$.

Proof. See proof of Lemma 4.2 in [4]. □

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 value 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 $\overline{Z}_{Nat} \leq \overline{Z}^{Sep}_{Cut}$. To do this, we consider an optimal solution $\overline{\Gamma} = (\overline{x}, \overline{y}_{s_1t_1}, ..., \overline{y}_{s_dt_d})$ of the linear programming relaxation of kHNDP$^{Sep}_{Cut}$. We are going to show that $\overline{x}$ also induces a solution of kHNDP$_{Nat}$. 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 $(V_0, ..., V_{L+1})$, 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 $\widetilde{G}_{st}$. Since each arc of $\widetilde{C}$ corresponds to a unique edge of C and vice-versa, and since $\overline{y}_{st}(a) \leq \overline{x}(e)$, for all $a \in \widetilde{A}_{st}(e)$, $e \in E$, we have that $\overline{x}(C) \geq \overline{y}(\widetilde{C})$. As $\overline{\Gamma}$ is solution of the kHNDP$^{Sep}_{Cut}$ 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 kHNDP$_{Nat}$ yielding that $\overline{Z}_{Nat} \leq \overline{Z}^{Sep}_{Cut}$.

Now we show that $\overline{Z}_{Nat} \geq \overline{Z}^{Sep}_{Cut}$. For this, we consider an optimal solution $\overline{x}$ of the linear programming relaxation of kHNDP$_{Nat}$. 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 $\overline{\Gamma} = (\overline{x}, \overline{y}_{s_1t_1}, ..., \overline{y}_{s_dt_d})$ induces a solution of kHNDP$^{Sep}_{Cut}$. To do this, we consider an st-dicut $\widetilde{C}$ of $\widetilde{G}_{st}$, for a given demand $\{s,t\} \in D$. We distinguish two cases. First, if $\widetilde{C}$ contains an arc of the form (u,u'), $u \in V \setminus \{s,t\}$, then, since $|[u,u']| \geq k$ and $\overline{y}_{st}(a) = 1$, for all $a \in [u,u']$, we have that $\overline{y}_{st}(C) \geq k$.
Now 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, since $\overline{x}$ is solution of kHNDP$_{Nat}$ and hence satisfies all the st-cut and L-st-path-cut inequalities, we have that $\overline{y}_{st}(\widetilde{C}) \geq k$. Also, it is not hard to see that $\overline{\Gamma}$ satisfies inequalities (3.14) and (3.15), and thus, induces a solution of the linear programming relaxation of kHNDP$^{Sep}_{Cut}$ with cost $\overline{Z}_{Nat}$. Therefore, we get $\overline{Z}_{Nat} \geq \overline{Z}^{Sep}_{Cut}$, 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 before, we introduce a procedure, that we will call 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 as follows.

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 neither of the form (u', u'') with $u' \in N'$ and $u'' \in N''$, nor 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 5.3 *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 $(V_0, ..., V_{L+1})$ 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 proof of Lemma 4.1 in [4]. □

Lemma 5.4 *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 $\overline{x}$ by*

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

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 their linear programming relaxation 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 relaxation of the natural and aggregated formulations. Then, $\overline{Z}_{Nat} = \overline{Z}_{Ag}$.*

Proof. We will show first that $\overline{Z}_{Ag} \geq \overline{Z}_{Nat}$. 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 $(V_0, ..., V_{L+1})$, 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 5.3, the arc set $\widetilde{C}$ induces an *st*-dicut of $\widetilde{G}$. Since each arc of $\widetilde{C}$ corresponds to a unique edge of C and vice versa, and since $\overline{y}(a) \leq \overline{x}(e)$, for all $a \in \widetilde{A}(e)$, $e \in E$, we have that $\overline{x}(C) \geq \overline{y}(\widetilde{C})$. Moreover, $\overline{y}$ satisfies all the *st*-dicut inequalities. Thus, we have that $\overline{x}(C) \geq \overline{y}(\widetilde{C}) \geq k$, and $\overline{x}$ induces a solution of the linear programming relaxation of kHNDP$_{Nat}$, yielding that $\overline{Z}_{Ag} \geq \overline{Z}_{Nat}$.

Now, we are going to show that $\overline{Z}_{Nat} \geq \overline{Z}_{Ag}$. To do this, 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 $\overline{Z}_{Nat}$, implying that $\overline{Z}_{Nat} \geq \overline{Z}_{Ag}$. Let $\overline{y}$ be the vector of $\mathbb{R}^{\widetilde{A}}$ obtained from $\overline{x}$ as described in Lemma 5.4, and let $\overline{\Gamma} = (\overline{x}, \overline{y})$. We claim that $\overline{y}$ satisfies all the st-dicut inequalities (4.1). To prove it, we consider an st-dicut $\widetilde{C} = \delta^+(\widetilde{W})$ of $\widetilde{G}$ and distinguish three cases. W.l.o.g, we will suppose that $\widetilde{W} \cap S = \{s\}$ and $(\widetilde{V} \setminus \widetilde{W}) \cap T = \{t\}$. 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 $(\widetilde{W} \setminus (S \setminus \{s\})) \cup (T \setminus \{t\})$.

Case 1.
If $\widetilde{C}$ does not contains 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 5.4, the edge set C is either an st-cut or an L-st-path-cut, and $\overline{x}(C) \leq \overline{y}(C)$. Since $\overline{x}$ is a solution of kHNDP$_{Nat}$ and hence, $\overline{x}(C) \geq k$, we also have that $\overline{y}(\widetilde{C}) \geq k$.

Case 2.
If $\widetilde{C}$ contains an arc of the form (u', u''), for some $u \in V$, or of the form (t', t), for some $t \in T_D$, then it contains k arcs of the form (u', u'') or (t', t). Since $\overline{y}(a) = 1$, for all $a \in [u', u''] \cup [t', t]$, we have that $\overline{y}(\widetilde{C}) \geq k$.

Case 3.
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. Since $\widetilde{C}$ 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 that $\delta^+(\widetilde{W}') = \widetilde{C} \setminus \{(s', u'')\}$. 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 st-dicut, and since $[t'', t] = \emptyset$, we have that $\delta^+(\widetilde{W}'') = \widetilde{C} \setminus \{(v', t'')\}$. 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, based on Case 1, we have that $\overline{y}(\widetilde{C}') \geq k$. Since $\widetilde{C}' \subset \widetilde{C}$, we have that $\overline{y}(\widetilde{C}) \geq \overline{y}(\widetilde{C}')$ and get $\overline{y}(\widetilde{C}) \geq k$.

Therefore, $\overline{y}$ satisfies every st-dicut inequality (4.1) and, since $\overline{y}(a) = \overline{x}(e)$, 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$_{Ag}$, with cost $\overline{Z}_{Nat}$. Thus, $\overline{Z}_{Nat} \geq \overline{Z}_{Ag}$, 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. This results is summurized in Corollary 5.1 and in Figure 6.

Corollary 5.1 *Formulations (2.5), (2.10), (2.18), (3.6), (3.12), (3.18) and (4.5) produce the same LP-bound for the kHNDP.*

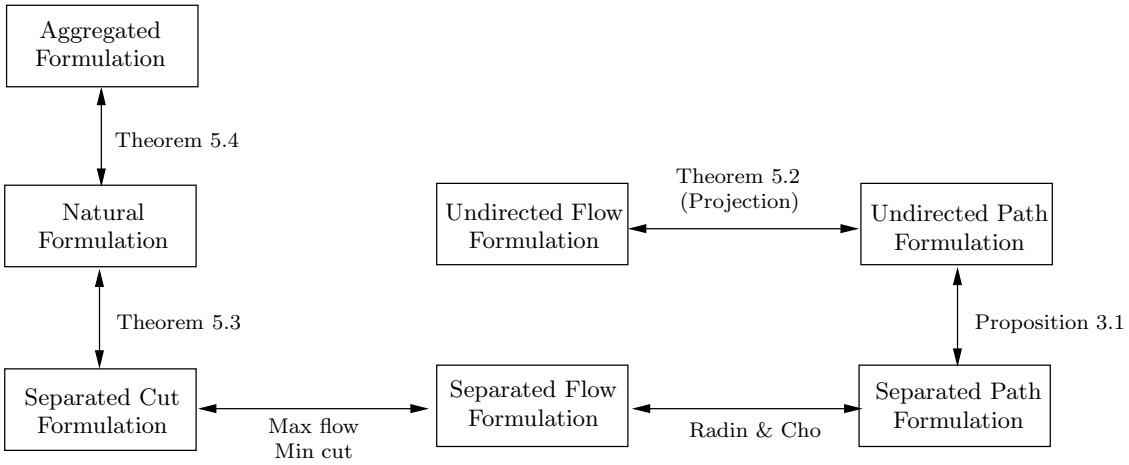


FIGURE 6 – Comparison of LP-bounds of the all the formulations

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 the problem for each formulation by using CPLEX 12.2 and Concert Technology, implemented in C++, on a DELL Workstation T3500 with an Intel Xeon Quad-Core 2.26GHz and 3Go of RAM. For our instances, we use graphs from TSPLIB [37] (euclidean complete graphs) with randomly generated demand sets. We consider single-source multi-destination instances and multi-source multi-destination instances. The number of nodes of the graphs varies from 21 up 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 formultation. 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 by using a linear program, since 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, since 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 [18]).

The results given in the following tables are obtained for $k = 3$ and for $L = 2$ and $L = 3$.
Each instance is given by the number of nodes of the graph preceded by the type of demand, indicated by 'r' for rooted demands and 'a' for arbitrary demands. The other entries of the various tables are :
In all the tables, the instances indicated with "*" are instances for which the algorithm has reached the maximum CPU time, 5 hours, while instances with "**" are those for which the

$\|V\|$	:	number of nodes of the graph ;
$\|D\|$	:	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 of the Branch-and-Cut tree ;
NSub	:	number of subproblems in the Branch-and-Cut tree ;
TT	:	total CPU time in hours :min :sec.

algorithm runs out of ressources (lack of memory). For all theses instances, we give the best results obtained at the end of the execution of the algorithm.

The first test is done for the separated flow formulation. Tables 1a and 1b give the results for this formulation for $L = 2$ and $L = 3$, respectively, and with $k = 3$.

$\|V\|$	$\|D\|$	COpt	Gap	NSub	TT
r 21	15	7138	2.52	69	0 :00 :01
r 21	17	7790	5.38	245	0 :00 :01
r 21	20	8762	6.85	818	0 :00 :03
a 21	10	8313	0	1	0 :00 :01
a 21	11	8677	0	1	0 :00 :01
r 30	15	12512	0	1	0 :00 :01
r 30	20	14215	1.81	93	0 :00 :02
r 30	25	15610	2.54	195	0 :00 :04
a 30	10	12124	1.04	122	0 :00 :01
a 30	15	15868	1.74	124	0 :00 :01
r 48	20	21586	2.26	174	0 :00 :03
r 48	30	29326	9.71	14653	0 :00 :32
r 48	40	37458	13.57	1061135	0 :19 :23
a 48	15	32097	1.48	90	0 :00 :01
a 48	20	44400	0.23	105	0 :00 :03
a 48	24	52619	0.16	108	0 :00 :02
r 52	20	14093	1.89	95	0 :00 :01
r 52	30	17643	5.11	2290	0 :00 :20
r 52	40	21041	7.72	97749	0 :02 :53
r 52	50	24619	8.97	547195	0 :23 :38
a 52	20	18480	1.12	482	0 :00 :02
a 52	26	24125	0.15	37	0 :00 :02

(a) Results for $L = 2$

$\|V\|$	$\|D\|$	COpt	Gap	NSub	TT
r 21	15	5472	7.94	3547	0 :01 :48
r 21	17	5864	7.81	4514	0 :02 :36
r 21	20	6466	9.12	35306	0 :17 :16
a 21	10	6675	8.40	22086	0 :05 :33
a 21	11	6770	6.54	5635	0 :02 :48
r 30	15	10109	5.87	2345	0 :00 :01
r 30	20	11182	5.94	35705	3 :40 :59
*** r 30*	*25*	*12493*	*9.83*	*76735*	*4 :16 :12*
a 30	10	10254	4.26	7915	0 :01 :57
*** a 30*	*15*	*13309*	*7.07*	*53410*	*1 :29 :24*
* r 48	20	16684	8.75	94790	5 :00 :00
* r 48	30	21434	15.14	11894	5 :00 :00
* r 48	40	27968	20.88	1715	5 :00 :00
*** a 48*	*15*	*25416*	*18.34*	*9923*	*2 :31 :12*
* a 48	20	33310	19.60	5074	5 :00 :00
* a 48	24	41520	24.30	2094	5 :00 :00
r 52	20	11154	7.99	47581	2 :25 :23
* r 52	30	13818	11.69	11506	5 :00 :00
* r 52	40	16638	15.34	3571	5 :00 :00
* r 52	50	19911	20.21	1243	5 :00 :00
*** a 52*	*20*	*16012*	*9.39*	*6339*	*3 :54 :03*
* a 52	26	21817	17.70	2029	5 :00 :00

(b) Results for $L = 3$

TABLE 1 – Results for separated Flow Formulation with $k = 3$

First, by observing Table 1a, 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 1mn in almost all cases, and the gap between the LP-root node and the optimal solution is low (less than 5% for 15 instances over 22, and between 5% and 10% for 6 instances). For $L = 3$ (Table 2b), the separated flow formulation solves to optimality only 8 instances. For the remaining instances, upper bounds are obtained by CPLEX, with a relative gap of at most 24.30%.
The problem seems to be easier for this formulation when $L = 2$ than when $L = 3$. The same observation can be done for the other formulations. More instances are solved to optimality within

5 hours when $L = 2$ than when $L = 3$. This observation confirms the idea that the kHNDP is easier when $L = 2$ than when $L = 3$.

In the remaining, 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.

$\|V\|$	$\|D\|$	COpt	Gap	NSub	TT
r 21	15	7138	2.73	175	0 :00 :01
r 21	17	7790	5.34	122	0 :00 :01
r 21	20	8762	7.26	669	0 :00 :03
a 21	10	8313	0	1	0 :00 :01
a 21	11	8677	0	1	0 :00 :01
r 30	15	12512	0	1	0 :00 :01
r 30	20	14215	1.31	72	0 :00 :02
r 30	25	15610	2.52	437	0 :00 :04
a 30	10	12124	0	1	0 :00 :01
a 30	15	15868	1.21	85	0 :00 :01
r 48	20	21586	2.92	91	0 :00 :06
r 48	30	29326	9.64	12101	0 :00 :37
r 48	40	37458	13.90	1098180	0 :18 :00
a 48	15	32097	1.33	98	0 :00 :01
a 48	20	44400	0.18	68	0 :00 :02
a 48	24	52619	0.13	45	0 :00 :02
r 52	20	14093	1.09	31	0 :00 :01
r 52	30	17643	5.68	776	0 :00 :18
r 52	40	21041	7.88	60261	0 :02 :14
r 52	50	24619	9.70	607522	0 :20 :30
a 52	20	18480	1.20	511	0 :00 :02
a 52	26	24125	0.27	229	0 :00 :02

(a) Results for $L = 2$

$\|V\|$	$\|D\|$	COpt	Gap	NSub	TT
r 21	15	5472	7.80	2880	0 :01 :03
r 21	17	5864	7.76	4879	0 :01 :56
r 21	20	6466	9.35	22544	0 :07 :44
a 21	10	6675	8.41	20479	0 :04 :31
a 21	11	6770	6.66	9700	0 :03 :26
r 30	15	10109	5.96	1740	0 :01 :45
r 30	20	11182	7.20	23429	2 :04 :54
* r 30	25	12449	9.57	223183	5 :00 :00
a 30	10	10254	5.38	5423	0 :01 :30
*** a 30*	*15*	*13309*	*7.44*	*45591*	*0 :55 :42*
r 48	20	16684	9.39	71705	4 :11 :37
* r 48	30	21407	15.03	14044	5 :00 :00
*** r 48*	*40*	*27382*	*19.01*	*2677*	*3 :56 :05*
*** a 48*	*15*	*25527*	*18.69*	*9933*	*1 :59 :00*
*** a 48*	*20*	*35608*	*24.78*	*6200*	*4 :48 :26*
* a 48	24	42145	25.42	2185	5 :00 :00
r 52	20	11154	8.49	26377	2 :03 :38
* r 52	30	13739	11.14	12257	5 :00 :00
* r 52	40	16339	13.93	5142	5 :00 :00
* r 52	50	19304	17.86	1713	5 :00 :00
*** a 52*	*20*	*15721*	*7.64*	*6385*	*3 :24 :03*
* a 52	26	20952	14.60	2247	5 :00 :00

(b) Results for $L = 3$

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

$\|V\|$	$\|D\|$	NCut	COpt	Gap	NSub	TT
r 21	15	1020	7138	9.5	81	0 :00 :02
r 21	17	1144	7790	9.34	77	0 :00 :03
r 21	20	1642	8762	11.6	539	0 :00 :09
a 21	10	244	8313	3.55	328	0 :00 :03
a 21	11	278	8677	3.53	115	0 :00 :02
r 30	15	1741	12512	5.56	134	0 :00 :05
r 30	20	2587	14215	6.84	365	0 :00 :14
r 30	25	3002	15610	8.57	666	0 :00 :23
a 30	10	342	12124	5.2	256	0 :00 :05
a 30	15	481	15868	3.68	1145	0 :00 :20
r 48	20	4822	21586	8.16	263	0 :00 :55
r 48	30	36022	29326	15.22	30202	3 :18 :17
** r 48	40	62831	37802	17.48	9530	1 :46 :40
a 48	15	1103	32097	3.08	1391	0 :01 :57
* a 48	20	2501	44466	4.3	155694	5 :00 :00
* a 48	24	3667	52840	4.41	107911	5 :00 :00
r 52	20	6954	14093	6.21	158	0 :01 :09
r 52	30	18864	17643	10.72	3854	0 :16 :16
** r 52	40	55739	21179	13.47	10480	1 :49 :33
** r 52	50	55978	24791	13.94	11449	2 :30 :16
a 52	20	1016	18480	3.43	1960	0 :03 :11
a 52	26	1971	24125	4.11	21210	0 :47 :38

(a) Results for $L = 2$

$\|V\|$	$\|D\|$	NCut	COpt	Gap	NSub	TT
r 21	15	9747	5472	8.33	1055	0 :04 :12
r 21	17	17847	5864	8.24	2235	0 :13 :08
r 21	20	42145	6466	9.53	11061	1 :40 :31
* a 21	10	68044	6687	8.76	6151	5 :00 :00
a 21	11	40503	6770	6.8	2828	1 :46 :52
r 30	15	18158	10109	6.86	1070	0 :20 :12
r 30	20	39234	11182	7.97	5795	2 :16 :31
** r 30	25	72053	12546	11.53	8193	4 :23 :26
* a 30	10	40203	10287	6.03	8214	5 :00 :00
* a 30	15	56223	13919	11.68	3202	5 :00 :00
* r 48	20	64330	16847	10.85	1860	5 :00 :00
* r 48	30	72036	22368	18.89	1631	5 :00 :00
* r 48	40	74630	27920	20.74	1675	5 :00 :00
** a 48*	*15*	*36092*	-	-	290	*5 :00 :00*
** a 48*	*20*	*30890*	-	-	112	*5 :00 :00*
* a 48	24	27509	-	-	28	5 :00 :00
* r 52	20	58898	11439	11.28	2193	5 :00 :00
* r 52	30	75879	14084	13.92	1480	5 :00 :00
* r 52	40	73825	17123	18.22	1685	5 :00 :00
* r 52	50	65427	19783	20.33	2131	5 :00 :00
** a 52*	*20*	*36167*	-	-	*256*	*5 :00 :00*
** a 52*	*26*	*30049*	-	-	102	5 :00 :00

(b) Results for $L = 3$

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

$\|V\|$	$\|D\|$	NCut	COpt	Gap	NSub	TT
r 21	15	914	7138	9.5	128	0 :00 :02
r 21	17	1067	7790	9.34	99	0 :00 :02
r 21	20	1282	8762	11.6	773	0 :00 :08
a 21	10	222	8313	3.55	230	0 :00 :02
a 21	11	296	8677	3.53	297	0 :00 :03
r 30	15	1654	12512	5.56	61	0 :00 :05
r 30	20	2221	14215	6.84	347	0 :00 :09
r 30	25	2705	15610	8.57	626	0 :00 :16
a 30	10	280	12124	5.2	292	0 :00 :05
a 30	15	482	15868	3.68	1111	0 :00 :13
r 48	20	4543	21586	8.16	223	0 :00 :47
r 48	30	47884	29326	15.22	19749	3 :20 :52
*** r 48*	*40*	*64201*	*37681*	*17.21*	*9321*	*1 :45 :14*
a 48	15	1211	32097	3.08	2009	0 :02 :08
* a 48	20	2999	44495	4.36	178464	5 :00 :00
* a 48	24	3182	52714	4.18	132516	5 :00 :00
r 52	20	6018	14093	6.21	141	0 :00 :54
r 52	30	28506	17643	10.72	4215	0 :29 :14
*** r 52*	*40*	*48256*	*21157*	*13.38*	*13052*	*1 :40 :33*
*** r 52*	*50*	*50987*	*24758*	*13.83*	*13131*	*1 :55 :27*
a 52	20	989	18480	3.43	2246	0 :02 :03
a 52	26	1818	24125	4.11	9461	0 :14 :41

(a) Results for $L = 2$

$\|V\|$	$\|D\|$	NCut	COpt	Gap	NSub	TT
r 21	15	8114	5472	8.33	1006	0 :01 :37
r 21	17	11193	5864	8.24	1413	0 :03 :12
r 21	20	41111	6466	9.53	12378	1 :02 :42
* a 21	10	70115	6675	8.59	6890	5 :00 :00
a 21	11	57966	6770	6.8	3111	3 :18 :27
r 30	15	16740	10109	6.86	798	0 :08 :34
r 30	20	46730	11182	7.97	11148	3 :06 :47
*** r 30*	*25*	*73878*	*12656*	*12.3*	*7301*	*2 :48 :04*
a 30	10	44224	10254	5.73	8737	4 :29 :09
* a 30	15	67676	14320	14.16	4315	5 :00 :00
* r 48	20	60109	17096	12.15	3260	5 :00 :00
* r 48	30	83234	22135	18.04	2040	5 :00 :00
* r 48	40	92074	28906	23.44	2364	5 :00 :00
* a 48	15	42729	-	-	387	5 :00 :00
* a 48	20	35721	-	-	107	5 :00 :00
* a 48	24	28084	-	-	39	5 :00 :00
* r 52	20	58132	11272	9.97	3065	5 :00 :00
* r 52	30	87930	14357	15.55	2292	5 :00 :00
* r 52	40	100134	16468	14.97	1827	5 :00 :00
* 52	50	96923	19563	19.43	4102	5 :00 :00
* a 52	20	40251	-	-	381	5 :00 :00
* a 52	26	30799	-	-	113	5 :00 :00

(b) Results for $L = 3$

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

We start the comparison by $L = 2$. The comparison between Tables 1a, 2a, 3a and 4a shows first that, for $L = 2$, the separated flow and path formulations produce quit similar results, and that they achieve better results than the aggregated and natural formulations. The first two formulations are able to solve to optimality 100% of the instances, while the aggregated and natural formulations solve 77.27% of the instances to optimality. Also, for the instances solved to optimality, the total CPU time for the separated flow and path formulations is better than for aggregated 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 comparison, we choose the results 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 $G_i = COPt_i - COpt_{SFlow}$, where i is SPath, Agg and Nat, standing respectively for separated path formulation, aggregated and natural formulation.
The instances reported in the table are those for which at least one formulation does not give the optimal solution.

$\|V\|$	$\|D\|$	GSPath	GAgg	GNat
r 30	25	-44	53	163
a 30	15	0	610	1011
r 48	20	0	163	412
r 48	30	-27	934	701
r 48	40	-586	-48	938
a 48	15	111	∞	∞
a 48	20	2298	∞	∞
a 48	24	625	∞	∞
r 52	30	-79	266	539
r 52	40	-299	485	-170
r 52	50	-607	-128	-348
a 52	20	-291	∞	∞
a 52	26	-865	∞	∞

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

A negative value in Table 5 indicates that the formulation gives a better bound than that obtained by the separated flow formulation while a positive value indicates that the formulation produces a greater bound. From this table, we can see that the separated path formulation produces, for most cases, a better bound than the separated flow formulation. Also, this formulation produce better bounds than the natural and aggregated formulation.
The comparison between the natural and aggregated formulations shows that for 5 instances in Table 5 over 8, the aggregated formulation gives better bound.

We conclude this compuational study by making a comment on the separated cut formulation. This formulation performs bad 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 hours of CPU time, and this, even for $L = 2$, and for all of these instance, the algorithm does not produce an upper bound. This is explained by the long time spents by the algorithm in the separation of the cut constraints (3.13).

7 Concluding remarks

In this paper, 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, which shows that, as expected, the resolution of 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 bound and the aggregated formulation produces, in some cases, good results. Finally, the results show that the separated cut formulation achieves bad results for both $L = 2$ and $k = 3$ and is, apparently, unusable from a practical point of view.

The experiments in this work let us suppose that some improvement may be needed in the resolution of the problem, especially for $L = 3$ (gaps relatively high). 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 by using further valid inequalities in the cutting plane phase. This latter improvement can be done especially with the aggregated graph transformation.

Références

[1] Cplex, "http ://www.ilog.com".

[2] F. Barahona and A. R. Mahjoub, "On two-connected subgraph polytopes", *Discrete Mathematics 147*, 1995, pp. 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 (1)*, 2012, pp. 13-32.

[4] F. Bendali, I. Diarrassouba, A. R. Mahjoub and J. Mailfert, "The k edge-disjoint 3-hop-constrained paths polytope", *Discrete Optimization 7 (4)*, 2010, pp. 222-233.

[5] Q. Botton, B. Fortz, L. Gouveia and M. Poss, "Benders decomposition for the hop-constrained survivable network design problem", *To appear in INFORMS Journal of Computing*, 2011.

[6] S. Chopra, "The k-edge connected spanning subgraph polyhedron", *SIAM Journal on Discrete Matematics 7*, 1994, pp. 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 editions, Kluwer Academic Publishers, Dordrecht, The Netherlands*, 1994, pp. 9-29.

[8] G. Dahl, "The Design of Survivable Directed Networks", *Telecommunication Systems 2*, 1992, pp. 349-377.

[9] G. Dahl, "The 2-hop spanning tree problem", *Operations Research Letters 23 (1-2)*, 1999, pp. 21-26.

[10] G. Dahl, "Notes on polyhedra associated with hop-constrained paths", *Operations Research Letters 25 (2)*, 1999, pp. 97-100.

[11] G. Dahl and L. Gouveia, "On the directed hop-constrained shortest path problem", *Operations Research Letters 32*, 2004, pp. 15-22.

[12] G. Dahl and B. Johannessen, "The 2-path network problem", *Networks 43 (3)*, 2004, pp. 190-199.

[13] G. Dahl, D. Huygens, A. R. Mahjoub and P. Pesneau, "On the k edge-disjoint 2-hop-constrained paths polytope", *Operation Research Letters 34 (5)*, 2006, pp. 577-582.

[14] M. Didi Biha and A. R. Mahjoub, "The k-edge connected subgraph problem I : Polytopes and critical extreme points", *Linear Algebra and its Applications 381*, 2004, pp. 117-139.

[15] M. Didi Biha and A. R. Mahjoub, "k-edge connected polyhedra on series-parallel graphs", *Operations Research Letters 19*, 1996, pp. 71-78.

[16] M. Didi Biha and A. R. Mahjoub, "Steiner k-edge connected subgraph polyhedra", *Journal of Combinatorial Optimization 4*, 2000, pp. 131-134.

[17] J. Fonlupt and A. R. Mahjoub, "Critical extreme points of the 2-edge connected spanning subgraph polytope", *Mathematical Programming 105*, 2006, pp. 289-310.

[18] B. Fortz, A. R. Mahjoub, S. T. McCormick and P. Pesneau, "Two-edge connected subgraphs with bounded rings : Polyhedral results and Branch-and-Cut", *Mathematical Programming 105 (1)*, 2006, pp. 85-111.

[19] B. Fortz, M. Labbe and F. Maffioli, "Solving the two-connected network with bounded meshes problem", *Operations Research Letters 48*, 2000, pp. 866-877.

[20] L. Gouveia, "Multicommodity flow models for spanning trees with hop constraints", *European Journal of Operational Research 95 (1)*, 1996, pp. 178-190.

[21] L. Gouveia, "Using variable redefinition for computing lower bounds for minimum spanning", *INFORMS Journal on Computing* 10 (2), 1998, pp. 180-188.

[22] L. Gouveia and P. icio and A. de Sousa and R. Valadas, "MPLS over WDM Network Design with Packet Level QoS Constraints based on ILP Models", *In Proceedings of INFOCOM 2003*, 2003.

[23] , L. Gouveia, L. Patricio and P. F. Sousa, "Compact Models for Hop-Constained Node Survivable Network Design : An Application to MPLS", *Telecommunications Network Planning : Innovations and Pricing, Network Design and Management*, ed. G. Anandaligam and S. Raghavan, Springer, 2005, pp. 167-180.

[24] , L. Gouveia, L. Patricio and P. F. Sousa, "Hop-Constrained Node Survivable Network Design : An Application to MPLS over WDM", *Networks and Spatial Economics 8 (1)*, 2008, pp. 3-21.

[25] L. Gouveia and C. Requejo, "A new Lagrangean relaxation approach for the hop-contrained minimum spanning tree problem", *European Journal of Operations Research 132 (3-1)*, 2001, pp. 539-552.

[26] M. Grötschel and C. L. Monma, "Integer polyhedra arising from certain network design problems with connectivity constraints", *SIAM Journal on Discrete Mathematics 3*, 1990, pp. 502-523.

[27] M. Grötschel, C. L. Monma and M. Stoer, "Polyhedral approches to network survivability", In F. Roberts, F. Hwang and C. L. Monma, eds, *Reliability of computer and Communication*

newtorks, Vol 5, Series Discrete Mathematics and Computer Science, AMS/ACM, 1991, pp. 121-141.

[28] M. Grötschel, C. L. Monma and M. Stoer, "Polyhedral and computational investigations arising for designing communication networks with high survivability requirements",*Operation Research 43*, 1995, pp. 1012-1024.

[29] 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 (1)*, 2007, pp. 116-133.

[30] D. Huygens and A. R. Mahjoub, "Integer programming formulation for the two 4-hop-constrained paths problem", *Networks 49 (2)*, 2007, pp. 135-144.

[31] D. Huygens, A. R. Mahjoub and P. Pesneau, "Two edge-disjoint hop-constrained paths and polyhedra", *SIAM Journal on Discrete Mathematics 18 (2)*, 2004, pp. 287-312.

[32] H. Kerivin and A. R. Mahjoub, "Design of Survivable Networks : A Survey", *Networks 46*, 2005, pp. 1-21.

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

[34] C.-W. Ko and C. L. Monma, "Heuristics for designing highly survivable communication networks", *Technical Report, Bellcore, Morristown, New Jersey*, 1989.

[35] S. Orlowski and R. Wessaly, "The Effect of Hop Limits on Optimal Cost in Survivable Network Design", *Operations Research/Computer Science Interfaces Series (33)*, 2006, pp. 151-166.

[36] R. Rardin and U. Choe, "Tighter relaxations of fixed charge network flow problems", *Technical Report J-79-18, Georgia Institue of Technology*, Atlanta, Georgia, 1979.

[37] TSPLIB, "http ://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/".