Branch-and-cut methods for the Network Design Problem with Vulnerability Constraints

Luis Gouveia[1], Martim Joyce-Moniz*[2,3] and Markus Leitner[2]

[1] Universidade de Lisboa, Faculdade de Ciências, Departamento de Estatística e Investigação Operacional, CMAF-CIO, Lisbon, Portugal.
legouveia@fc.ul.pt
[2]University of Vienna, Department of Statistics and Operations Research, Faculty of Business, Economics and Statistics, Vienna, Austria.
markus.leitner@univie.ac.at
[3]GERAD and Polytechnique Montréal, Montreal, Canada.
martim.joyce-moniz@gerad.ca

May 16, 2017

Abstract

The aim of Network Design Problem with Vulnerability Constraints (NDPVC), introduced by Gouveia and Leitner [EJOR, 2017], is to design survivable telecommunications networks that impose length bounds on the communication paths of each commodity pair, before and after the failure of any k links. This problem was proposed as an alternative to the Hop-Constrained Survivable Network Design Problem (kHSNDP), which addresses similar issues, but imposes very conservative constraints, possibly leading to unnecessarily expensive solution or even rendering instances infeasible. It was shown that the cost of the optimal solutions of the NDPVC never exceeds that of the related kHSNDP. However, results reported by Gouveia and Leitner [EJOR, 2017] using the standard methods of a general-purpose integer linear (ILP) solver, along with several ILP formulations, show that such methods fail to solve most instances in the benchmarking test set.

In this paper, we propose three branch-and-cut methods, based on graph-theoretical characterizations introduced by Gouveia and Leitner [EJOR, 2017], which are significantly more efficient in solving the NDPVC. With the proposed new methods, we are able to solve substantially more instances of the NDPVC and therefore able to provide a more complete comparison of its solutions to those of the kHSNDP.

Keywords: Networks, Integer Programming, Survivable Network Design, Hop-constraints, OR in Telecommunications, Benders Decomposition

1 Introduction

Every telecommunication network should be able to withstand a reasonable amount of technical equipment failures, without compromising the ability of any two points to communicate. Thereby, the network optimization literature contains many works related to the resistance of network to failures (or network *survivability*), as one of the main criteria for designing reliable communication networks (see, e.g., Kerivin and Mahjoub [16]). A network is said to be survivable if every pair of nodes has a path for communication, even after the failure of a predefined number of nodes or links. Quality of service is another relevant criterion when designing telecommunication networks (see, e.g., Klincewicz [17]). One example of a quality-of-service parameter is *jitter*, which is defined as the time difference between the maximum and minimum delay among all data packets flowing in a network [26]. The delay of each data packet depends on the propagation delay on the links that it traverses, on the path from its source to its

*corresponding author.

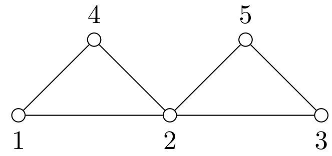


Figure 1: A feasible solution for the NDPVC with $k = 2$, $\mathcal{R} = \{\{1, 3\}\}$, $H_{13} = 2$, $H'_{13} = 3$, that does not contain two edge-disjoint paths of length 2 nor of length 3 between nodes 1 and 3, cf. Exoo [9].

destination, as well as on the queuing and transmission delays on each intermediate nodes. Designing networks with upper bounds on the number of links (hops) on each routing path has been the usual way of imposing constraints on the delay, and ensuring quality of service, see, e.g., Balakrishnan and Altinkemer [1], LeBlanc and Reddoch [18]. These bounds are commonly referred to as *hop constraints*.

In the last years, researchers have started to focus on network optimization problems that combine both network survivability and hop constraints. In the Hop-Constrained Survivable Network Design Problem (kHSNDP), the objective is to design a minimum-cost subgraph that has k edge- (node-) disjoint paths of at most H edges, between the source and target nodes of each commodity. Different integer programming formulations and solution algorithms have been proposed for the two variants of the kHSNDP, see [4, 12, 22]. Recently, Gouveia and Leitner [11] proposed and studied the *Network Design Problem with Vulnerability Constraints* (NDPVC), which is similar to the edge-disjoint variant of the kHSNDP but imposes less restrictions on the underlying network topology, and as a consequence, may lead to cheaper solutions. The formal definition of the NDPVC is as follows: let $G = (V, E)$ be an undirected graph with nonnegative edge costs $c_e \geq 0$, for all $e \in E$, and $k \in \mathbb{N}$ a parameter specifying the network survivability. In addition, consider a set of commodities $\mathcal{R} \subseteq V \times V$. The objective is to find a minimum-cost subgraph of G, such that for each pair $\{s, t\} \in \mathcal{R}$, it contains a path of length at most $H_{st} \in \mathbb{N}$, and after the removal of any $k - 1$ edges from it, the resulting subgraph contains a path of length at most $H'_{st} \in \mathbb{N}$ ($H_{st} \leq H'_{st}$).

The main motivation for this new problem is that hop-constrained variants of Menger's theorem [23] do not hold in general, and as a consequence the NDPVC is not equivalent to the kHSNDP, even when $H_{st} = H'_{st}$ for all $\{s, t\} \in \mathcal{R}$. More importantly, the following has been observed in [11]: if I is an arbitrary, feasible instance of the NDPVC, with optimal cost $v(I)$, then either (i) there is no feasible solution of the kHSNDP for I (an example of this situation is illustrated in Figure 1), or (ii) $v(I) \leq v'(I)$, where $v'(I)$ is the optimal cost of the kHSNDP for I. Furthermore, there exist instances such that $v(I) < v'(I)$, where $\frac{v'(I)}{v(I)}$ can be arbitrarily large.

The notation that follows will be used to model the NDPVC (see [11]). Let $x_e \in \{0, 1\}, e \in E$, be decision variables indicating whether or not an edge $e \in E$ is used in a solution, and let $E(\mathbf{x})$ denote the set of edges such that $x_e = 1$. Moreover, consider $\mathcal{F}_{st} = \{\mathbf{x} \in \{0, 1\}^{|E|} \mid \exists\ (s, t)\text{-path } P \text{ in } E(\mathbf{x}) \text{ s.t. } |P| \leq H_{st}\}$, the set of feasible incidence vectors of $\mathbf{x}$ containing a path of length at most H_{st} for each commodity pair $\{s, t\} \in \mathcal{R}$; and $\mathcal{B}_{st} = \{\mathbf{x} \in \{0, 1\}^{|E|} \mid \forall F \subset E,\ |F| = k - 1,\ \exists\ (s, t)\text{-path } P \text{ in } E(\mathbf{x}) \setminus F \text{ s.t. } |P| \leq H'_{st}\}$, the set of feasible incidence vectors of $\mathbf{x}$ ensuring the required redundancy, i.e., the existence of a path of length at most H'_{st} for each commodity pair $\{s, t\} \in \mathcal{R}$ after removing $k - 1$ edges. Three distinct graph-theoretical characterizations of $\mathcal{B}_{st}$ are described in [11], and repeated below. Furthermore, several integer linear programming (ILP) formulations based on these characterizations have been proposed in the same work. The first one, CH1, is trivial and leads to a straightforward but less efficient formulation; CH2 and CH3, on the other hand, are the base of ILP formulations with fewer variables, and whose linear programing (LP) relaxation usually provides better bounds to the optimal value.

Characterization 1 (CH1 [11])**.** *Let $P \subset E_{st}$ be the primary (s, t)-path of length $\leq H_{st}$ for a given commodity $\{s, t\} \in \mathcal{R}$. Then, a valid set of backup edges $\hat{E}$ is established by ensuring that:*

$$\forall e \in P, \quad \exists\ (s, t)\text{-path } P'[e] \subset E'_{st} \setminus \{e\},\ \text{s.t. } |P'[e]| \leq H'_{st}.$$

Thereby, $H_{st} \leq H'_{st}$ and $\hat{E} = \left(\bigcup_{e \in P} P'[e]\right) \setminus P$.

Characterization 2 (CH2 [11])**.** *Let $P \subset E_{st}$ be the primary (s,t)-path of length l, $l \leq H_{st}$, for a given commodity $\{s,t\} \in \mathcal{R}$. Then, a valid set of backup edges $\hat{E}$ is established by ensuring that there exist l additional (s,t)-paths $P'_i \subset E'_{st}$, $i = 1,2,\ldots,l$, of length at most H'_{st}, $H'_{st} \geq H_{st}$, such that at most $l-1$ of them contain the same edge from P, i.e.:*

$$\exists\ (s,t)\text{-paths } P'_i \subset E'_{st},\ i = 1,2,\ldots,l, \text{s.t. } |P'_i| \leq H'_{st} \text{and } \forall e \in P : \sum_{i=1}^{l} |P'_i \cap \{e\}| \leq l-1.$$

Thereby, $\hat{E} = \left(\bigcup_{i=1}^{l} P'_i\right) \setminus P$.

Characterization 3 (CH3 [11])**.** *Let $P = \{e_1, e_2, \ldots, e_l\} \subset E_{st}$ be the primary (s,t)-path of length l, $l \leq H_{st}$ for a given commodity $\{s,t\} \in \mathcal{R}$, such that $e_i = \{u_{i-1}, u_i\}$, $u_i \in V_{st}$, $i = 1,2,\ldots,l$, $u_0 = s$, and $u_l = t$. Then, a valid set of backup edges $\hat{E}$ is established by ensuring that:*

$$\exists\ (s,t)\text{-paths } P'_i \subset E'_{st}, i = 1,2,\ldots,l, \text{ s.t. } |P'_i| \leq H'_{st} \text{ and } P'_i \cap \{e_i\} = \emptyset.$$

Thereby, $H_{st} \leq H'_{st}$ and $\hat{E} = \left(\bigcup_{i=1}^{l} P'_i\right) \setminus P$.

Extensive computational experiments are reported in [11] with the twofold purpose of comparing the performance of the proposed models within a general-purpose ILP solver (IBM ILOG CPLEX), and of comparing the solutions obtained by solving the NDPVC to those obtained by solving the kHSNDP. For these experiments, two classes of benchmark instances were created: one class where the nodes in the graph are distributed in a grid and the other where the nodes are randomly distributed, leading to a total of 3150 instances. The results of these experiments show that two formulations based on CH2 and CH3 are the most promising ones, as they are able to solve a higher number of the proposed instances; these two models will be revisited in Section 2.2.

However, the results also show that, in order to solve the NDPVC efficiently, one needs to design more sophisticated methods that go beyond developing a "good" formulation and using a general-purpose ILP solver such as CPLEX. Even with the best model, CPLEX only solves 781 of the 2070 grid instances, and 250 of the 1080 random instances, within the time limit of two hours. These results have complicated the other aforementioned objective in [11], of making a more complete comparison between the cost of the optimal solutions of the NDPVC and the cost of the optimal solutions of the kHSNDP. For 450 grid instances, the cost of the optimal solution of the NDPVC is shown to be smaller than that of the corresponding solution given by the kHSNDP, while for 547 grid instances the cost is shown to be the same. However, these results still leave out half of the grid instances for which no conclusion could be achieved. This situation is even worse for the random instances: for more than 70% of the instances, the authors could not obtain a conclusive comparison between the solutions of both problems.

Scientific contribution The results in [11] motivate the need for more efficient solution algorithms for the NDPVC. In this paper, we propose three different branch-and-cut methods, which are based on decompositions of formulations inspired by CH1, CH2 and CH3. The first branch-and-cut algorithm is based on a new integer programming formulation based on layered graphs. The other two branch-and-cut methods proposed in this article are based on new theoretical results that reveal that the two best-performing formulations from Gouveia and Leitner [11] remain valid when we relax the flow variables to be continuous, instead of binary as originally proposed. These results are central for these two methods, in the sense that they enable the use of a standard Benders decomposition method without the need to resort to, e.g., combinatorial Benders cuts, that may significantly deteriorate their performance. In addition, these results also imply that one can decide in polynomial time whether a given set of selected edges is a feasible solution to the NDPVC, when $k = 2$. Observe, in contrast, that the analogous problem, the edge-disjoint variant of the 2HSNDP is known to be NP-hard (this follows from the complexity results in Li et al. [21]). We also introduce an ILP-based heuristic for the NDPVC which is used to initialize the three decomposition algorithms with good initial feasible solutions. Our results show that the proposed methods are significantly more efficient in solving the NDPVC, in that they are able to solve many more instances than the standard solving methods of commercial ILP solvers such as CPLEX.

Notation In addition to the notation already presented, we consider the following notation that will be used in the remainder of the paper. Let $T = \{s \in V \mid \exists\{s,t\} \in \mathcal{R}, s < t\}$ denote the set of commodity origins (sources), and $T(s) = \{t \in V \mid \{s,t\} \in \mathcal{R},\ s < t\}$ denote the set of commodity destinations (targets) of origin $s \in T$. We define $H_{\max}(s) = \max_{t \in T(s)} H_{st}$, and $H'_{\max}(s)$ analogously. Moreover, let $A = \{(i,j) \mid \{i,j\} \in E\}$ denote the arc set obtained from bi-directing edge set E. For a subset $W \subset V'$ of nodes of some graph $G' = (V', E')$, we will use $\delta(W) = \{\{i,j\} \in E' \mid i \in W, j \notin W\}$ to denote the cutset of W with respect to G'. Let, furthermore, $d_{ij} \in \mathbb{N}$ denote the minimum distance between nodes i and j in G (measured in the number of hops). Then, for each commodity pair $\{s,t\} \in \mathcal{R}$, $A_{st} = \{(i,j) \in A \mid d_{si} + d_{jt} + 1 \leq H_{st}\}$ and $A'_{st} = \{(i,j) \in A \mid d_{si} + d_{jt} + 1 \leq H'_{st}\}$ are the sets of arcs feasible for establishing a primary or secondary connection of commodity $\{s,t\}$, respectively. Similarly, $E_{st} = \{\{i,j\} \in E \mid (i,j) \in A_{st} \vee (j,i) \in A_{st}\}$ and $E'_{st} = \{\{i,j\} \in E \mid (i,j) \in A'_{st} \vee (j,i) \in A'_{st}\}$ are the sets of eligible primary and secondary edges for commodity $\{s,t\}$ while $V_{st} = \{i \in V \mid \exists\{i,j\} \in E_{st}\}$ and $V'_{st} = \{i \in V \mid \exists\{i,j\} \in E'_{st}\}$ are the eligible nodes within a primary or secondary connection, respectively. Also, consider $E_s = \bigcup_{t \in T(s)} E_{st}$, and $E'_s = \bigcup_{t \in T(s)} E'_{st}$. Finally, for $e = \{i,j\} \in E$ and a set of arcs A' let $A'[e] = A' \setminus \{(i,j),(j,i)\}$.

Paper layout In Section 2, we describe the three proposed branch-and-cut methods. Next, in Section 3, we describe the primal heuristic. In Section 4, we present and analyze the results of computational experiments, before drawing conclusions in Section 5.

2 Branch-and-cut methods

In this section, we propose three decomposition approaches, based on the characterizations originally given in Gouveia and Leitner [11], and recalled in the previous section. Section 2.1 details a branch-and-cut method based on a new, layered graph formulation for the NDPVC, whose validity follows CH1. In Section 2.2, we describe Benders decomposition methods for two formulations based on CH2 and CH3, which were proposed in [11]. Note, that a similar approach could also have been explored for a straightforward formulation based on CH1, and also presented in the same paper. The results from [11] clearly indicate, however, that such a method would not be competitive with the ones that will be presented in Section 2.2. More precisely, it was shown in [11] that despite the fact that this formulation contains more variables and constraints than the formulations based on CH2 and CH3, its LP relaxation bounds are significantly weaker (in practice) than the ones obtained by the formulation based on CH3. We will therefore refrain from discussing a decomposition method based on this model and instead focus on a new, layered graph formulation based on CH1.

2.1 Layered graph formulation and branch-and-cut method based on CH1

Gouveia, Simonetti, and Uchoa [13] showed how to solve the solve the hop- and diameter-constrained minimum spanning tree problem as Steiner tree problems (with only few additional constraints) on appropriately defined layered graphs. They also showed that their formulation dominates all other formulations from the literature, both theoretically, and with respect to the computational performance of a correspondingly developed branch-and-cut approach. Since then, layered graphs have been used to derive theoretically strong and computationally well-performing models for a number of network design problems with hop-, diameter-, or more general resource constraints, see, e.g., [15, 19, 25]. An effective branch-and-cut algorithm based on multiple layered graphs (one per root node) has been recently proposed by Gouveia et al. [14] for the hop-constrained Steiner tree problem with multiple roots. The formulation introduced in this section significantly generalizes this idea by considering multiple and disjoint paths per commodity, different hop limits for primary and backup paths, and terminal-dependent hop limits.

To this end, for each source $s \in T$, we define a layered graph $G^s_{\mathrm{L}} = (V^s_{\mathrm{L}}, A^s_{\mathrm{L}})$; the node set V^s_{L} is defined as $V^s_{\mathrm{L}} = \{s_0\} \cup \{t_h \mid t \in T(s), h \in \{1,2,\ldots,H'_{st}\} \cup \{i_h \mid i \in V'_{st} \setminus (\{s\} \cup T(s)), h \in \{1,2,\ldots,H'_{\max}(s)\}\}$ and the arc set is defined as $A^s_{\mathrm{L}} = \{(i_h, j_{h+1}) \mid \{i_h, j_{h+1}\} \subseteq V^s_{\mathrm{L}}, (i,j) \in E'_s\}$. In addition to the edge design variables $\mathbf{x}$, the layered graph formulation (1)–(5), which we refer to as $\mathrm{H}^{\mathrm{L}}_{\mathrm{I}}$, uses variables $X^{sh}_{ij} \in \{0,1\}$, $\forall s \in T$, $\forall (i_{h-1}, j_h) \in \tilde{A}^s_{\mathrm{L}}$ that indicate whether arc (i,j) is used at position h on either a primary or a backup path, originated in source s.

$$\min \quad \sum_{e \in E} c_e x_e \tag{1}$$

$$\text{s.t.} \quad \sum_{(i_{h-1}, j_h) \in \delta^-(S)} X_{ij}^{s,h} \geq 1 \qquad s \in T,\ S \subset V_{\mathrm{L}}^s(H_{\max}(s)),\ s_0 \in S,\ \exists t \in T(s) : \bigcup_{h=0}^{H_{st}} t_h \cap S = \emptyset \tag{2}$$

$$\sum_{(i_{h-1}, j_h) \in \delta^-(S)[e]} X_{ij}^{s,h} \geq 1 \qquad s \in T,\ e \in E_s,\ S \subset V_{\mathrm{L}}^s,\ s_0 \in S,\ \exists t \in T(s) : \bigcup_{h=0}^{H'_{st}} t_h \cap S = \emptyset \tag{3}$$

$$X_{ij}^{sh} \leq x_e \qquad s \in T,\ e = \{i,j\} \in E'_s, h \in \{1, 2, \ldots, H'_{\max}(s)\} \tag{4}$$

$$X_{ij}^{sh} \in \{0,1\} \qquad s \in T,\ (i_{h-1}, j_h) \in A_{\mathrm{L}}^s \tag{5}$$

Inequalities (2) are directed connectivity constraints ensuring the existence of a primary path of length at most H_{st} for each $s \in T$ and $t \in T(s)$. Here, $V_{\mathrm{L}}^s(H_{\max}(s)) = \{i_h \in V_{\mathrm{L}}^s \mid 0 \leq h \leq H_{\max}(s)\}$ for each $s \in T$, i.e., considered cutsets S are constrained to the first $H_{\max}$ layers. Cut constraints (3) ensure the existence of a path from each $s \in T$ to each $t \in T(s)$ after removing all arcs from A_{L}^s that correspond to a single original edge, as stated by CH1. For each source $s \in T$ and every subset of nodes $S \subset V_{\mathrm{L}}^s$, we use notation $\delta^-(S)[e] = \{(i_{h-1}, j_h) \in A_{\mathrm{L}}^s \setminus \{(u_l, v_{l+1}) \in A_{\mathrm{L}}^s \mid e = \{u,v\}\} \mid i_{h-1} \notin S, j_h \in S\}$ to denote all ingoing arcs of node set S on a subgraph of G_{L}^s in which all arcs corresponding to edge $e \in E_s$ have been removed. Similar sets of connectivity constraints ensuring disjoint paths have, e.g., been used for network design problem considering node-disjoint paths in [7, 20]. Finally, constraints (4) link arc variables on the layered graph to original edge design variables.

It is well known that although the number of cutset constraints (2) and (3) may be exponential, they can be efficiently separated. This motivates the development of a branch-and-cut method, which we refer to as BC1; see Section 4 for more details.

2.2 Benders decomposition methods based on CH2 and CH3

In this section, we first revisit the two best-performing formulations from [11], which are denoted as $\mathrm{H}_2^{\mathrm{AS}}$ and H_3, and which are based on CH2 and CH3, respectively. In Section 2.2.1, we will then prove two theoretical results related to these formulations and describe Benders decomposition methods that are enabled by these results.

Formulations $\mathrm{H}_2^{\mathrm{AS}}$ and H_3 use the, previously mentioned, undirected edge design variables $\mathbf{x}$ and can be viewed in the following form:

$$\min \quad \sum_{e \in E} c_e x_e \tag{6}$$

$$\text{s.t.} \quad \mathbf{x} \in \mathcal{F}_{st} \cap \mathcal{B}_{st} \qquad \forall \{s,t\} \in \mathcal{R} \tag{7}$$

$$\mathbf{x} \in \{0,1\}^{|E|} \tag{8}$$

As $\mathrm{H}_2^{\mathrm{AS}}$ and H_3 only differ in the sets of variables and constraints used to model $\mathcal{B}_{st}$, we will next detail their common part modeling $\mathcal{F}_{st}$ before giving all details of their formulations for $\mathcal{B}_{st}$.

Formulation for $\mathcal{F}_{st}$ used in $\mathrm{H}_2^{\mathrm{AS}}$ and H_3 [11] Let $y_{ij}^{st,h} \in \{0,1\}$ be the routing variables for the primary path, indicating whether or not arc $(i,j) \in A_{st}$ is used at position $h \in \{1, 2, \ldots, H_{st}\}$, in the path from source s to target t, for all $\{s,t\} \in \mathcal{R}$. For each $\{s,t\} \in \mathcal{R}$, formulations $\mathrm{H}_2^{\mathrm{AS}}$ and H_3 ensure that $\mathbf{x} \in \mathcal{F}_{st}$ through constraints (9)–(13).

$$\sum_{(s,j) \in A_{st}} y_{sj}^{st,1} = 1 \tag{9}$$

$$\sum_{(i,j) \in A_{st}} y_{ij}^{st,h} = \sum_{(j,i) \in A_{st}} y_{ji}^{st,h+1} \qquad i \in V_{st} \setminus \{s,t\},\ h \in \{1, \ldots, H_{st} - 1\} \tag{10}$$

$$\sum_{h=1}^{H_{st}} \sum_{(i,t)\in A_{st}} y_{it}^{st,h} = 1 \tag{11}$$

$$\sum_{h=1}^{H_{st}} (y_{ij}^{st,h} + y_{ji}^{st,h}) \le x_e \qquad e = \{i,j\} \in E_{st} \tag{12}$$

$$y_{ij}^{st,h} \in \{0,1\} \qquad (i,j) \in A_{st},\ h \in \{1,\dots,H_{st}\} \tag{13}$$

Constraints (9)-(11) define the flow system for variables $\mathbf{y}$, whereas constraints (12) link the latter with variables $\mathbf{x}$.

Formulation for $\mathcal{B}_{st}$ used in $\mathrm{H}_2^{\mathrm{AS}}$ **[11]** For every commodity $\{s,t\} \in \mathcal{R}$, formulation $\mathrm{H}_2^{\mathrm{AS}}$ ensures that $\mathbf{x} \in \mathcal{B}_{st}$ through constraints (14)–(20), thereby using additional routing variables for the backup paths, $\tilde{z}_{ij}^{st,h} \in \{0,1,\dots,H_{st}\}$, that indicate the number of backup paths for commodity $\{s,t\} \in \mathcal{R}$ that use arc $(i,j) \in A'_{st}$ at position $h \in \{1,2,\dots,H'_{st}\}$.

$$\sum_{(s,j)\in A'_{st}} \tilde{z}_{sj}^{st,1} = \sum_{h=1}^{H_{st}} \sum_{(i,t)\in A_{st}} h y_{it}^{st,h} \tag{14}$$

$$\sum_{(i,j)\in A'_{st}} \tilde{z}_{ij}^{st,h} = \sum_{(j,i)\in A'_{st}} \tilde{z}_{ji}^{st,h+1} \qquad i \in V'_{st} \setminus \{s,t\},\ \forall h \in \{1,\dots,H'_{st}-1\} \tag{15}$$

$$\sum_{h=1}^{H'_{st}} \sum_{(i,t)\in A'_{st}} \tilde{z}_{it}^{st,h} = \sum_{h=1}^{H_{st}} \sum_{(i,t)\in A_{st}} h y_{it}^{st,h} \tag{16}$$

$$\sum_{h=1}^{H_{st}} \left(y_{ij}^{st,h} + y_{ji}^{st,h}\right) + \sum_{h=1}^{H'_{st}} \left(\tilde{z}_{ij}^{st,h} + \tilde{z}_{ji}^{st,h}\right) \le \sum_{h=1}^{H_{st}} \sum_{(i',t)\in A_{st}} h y_{i't}^{st,h} \qquad \{i,j\} \in E_{st} \tag{17}$$

$$\sum_{h=1}^{H'_{st}} \left(\tilde{z}_{ij}^{st,h} + \tilde{z}_{ji}^{st,h}\right) \le H_{st} x_e \qquad e = \{i,j\} \in E'_{st} \setminus E_{st} \tag{18}$$

$$\sum_{h=1}^{H_{st}} \left(y_{ij}^{st,h} + y_{ji}^{st,h}\right) + \sum_{h=1}^{H'_{st}} \left(\tilde{z}_{ij}^{st,h} + \tilde{z}_{ji}^{st,h}\right) \le H_{st} x_e \qquad e = \{i,j\} \in E_{st} \tag{19}$$

$$\tilde{z}_{ij}^{st,h} \in \{0,1,\dots,H_{st}\} \qquad (i,j) \in A'_{st},\ h \in \{1,\dots,H'_{st}\} \tag{20}$$

Constraints (14)-(16) define the conservation of the flow $\tilde{\boldsymbol{z}}^{\boldsymbol{st}}$, whose quantity must match the length of the primary path between s and t. Inequalities (17) ensure the existence of a path connecting nodes s and t, after the failure of each edge in E_{st}, by preventing that all units of $\tilde{\boldsymbol{z}}^{\boldsymbol{st}}$ flow are routed through the same edge, belonging to the primary path. Constraints (18) are linking constraints between variables $\tilde{\boldsymbol{z}}$ and $\mathbf{x}$; for edges in E_{st}, these can be strengthened as in (19).

Formulation for $\mathcal{B}_{st}$ used in H_3 **[11]** Formulation H_3 ensures that $\mathbf{x} \in \mathcal{B}_{st}$ using constraints (21)–(25) and by considering the disaggregated routing variables for the backup paths, $\hat{z}_{ij}^{stl,h}$, which are set to one if arc $(i,j) \in A'_{st}$ is used at position $h \in \{1,2,\dots,H'_{st}\}$ in the backup path of commodity $\{s,t\} \in \mathcal{R}$ with index $l \in \{1,2,\dots,H_{st}\}$, and to zero if otherwise.

$$\sum_{(s,j)\in A'_{st}} \hat{z}_{sj}^{stl,1} = \sum_{h=l}^{H_{st}} \sum_{(i,t)\in A_{st}} y_{it}^{st,h} \qquad l \in \{1,\dots,H_{st}\} \tag{21}$$

$$\sum_{(i,j)\in A'_{st}} \hat{z}_{ij}^{stl,h} = \sum_{(j,i)\in A'_{st}} \hat{z}_{ji}^{stl,h+1} \qquad i \in V'_{st} \setminus \{s,t\}, l \in \{1,\dots,H_{st}\}, h \in \{1,\dots,H'_{st}-1\} \tag{22}$$

$$\sum_{h=1}^{H'_{st}} \sum_{(i,t)\in A'_{st}} \hat{z}_{it}^{stl,h} = \sum_{h=l}^{H_{st}} \sum_{(i,t)\in A_{st}} y_{it}^{st,h} \qquad l \in \{1,\ldots,H_{st}\} \tag{23}$$

$$y_{ij}^{st,l} + y_{ji}^{st,l} + \sum_{h=1}^{H'_{st}} \left(\hat{z}_{ij}^{stl,h} + \hat{z}_{ji}^{stl,h} \right) \leq x_e \qquad e = \{i,j\} \in E'_{st},\ l \in \{1,\ldots,H_{st}\} \tag{24}$$

$$\hat{z}_{ij}^{stl,h} \in \{0,1\} \qquad (i,j) \in A'_{st},\ l \in \{1,\ldots,H_{st}\},\ h \in \{1,\ldots,H'_{st}\} \tag{25}$$

Constraints (21)-(23) route one unit of flow $\hat{\boldsymbol{z}}^{\boldsymbol{stl}}$ between s and t, if the primary path of that commodity has at least l arcs. Finally, inequalities (24) are strong linking constraints, that guarantee the condition in CH3.

2.2.1 Benders decomposition methods for $\mathrm{H}_2^{\mathrm{AS}}$ and H_3

The results in [11] suggest that $\mathrm{H}_2^{\mathrm{AS}}$ and H_3 perform better than the other proposed formulations, in the sense that they were able to solve a larger number of instances of the NDPVC when implemented in CPLEX. Of these two models, formulation H_3 was able to solve the highest number of the considered instances. As observed by the authors, the main explanation for this behavior is the fact that this formulation produces, by far, the best LP bounds. On the other hand, formulation $\mathrm{H}_2^{\mathrm{AS}}$, the second-best option for solving instances of the problem, benefits from the fact that it has fewer variables than H_3. Nevertheless, the results imply that both approaches suffer from their large number of (flow) variables and constraints. Thus, it seems natural to develop a Benders decomposition method [3] from these two formulations, in order to more efficiently solve the NDPVC. Benders decomposition explores special substructures in mathematical programming formulations, often allowing for significant speed-ups in their solving. At each iteration of this method, the value of a subset of all the variables is fixed, yielding a subproblem or set of subproblems that are easier to solve. If the solution to the master problem is not feasible or optimal for a given subproblem, a Benders inequality is generated that cuts off the given, tentative, solution and the master problem is resolved. The Benders decomposition method has been used successfully to solve a wide array of optimization problems; see, e.g., Rahmaniani et al. [24] for a detailed literature review or [4, 5, 10] for recent applications to network design problems.

The structure of formulations $\mathrm{H}_2^{\mathrm{AS}}$ and H_3 makes them attractive to use in the context of Benders decomposition, by keeping the design variables $\mathbf{x}$ in the Benders master problem and validating the feasibility of a current candidate solution through solving several independent subproblems, one for each commodity $\{s,t\} \in \mathcal{R}$. To this end, all variables and constraints corresponding to abstract constraints (7) are projected out of the master problem and these subproblems are defined by constraints (9)–(20) for $\mathrm{H}_2^{\mathrm{AS}}$ and by (9)–(13) together with (21)–(25) for H_3.

Since the standard Benders method is based on LP duality theory, it is, however, not directly applicable to the obtained subproblems due to the integrality requirements of the involved flow variables $\mathbf{y}$, $\tilde{\boldsymbol{z}}$, and $\hat{\boldsymbol{z}}$, respectively. We observe that for H_3 a valid Benders decomposition algorithm could be developed by additionally using so-called combinatorial Benders cuts [8]. For $\mathrm{H}_2^{\mathrm{AS}}$, however, as flow variables $\tilde{\boldsymbol{z}}$ are general integers (and not binary), this approach is not applicable without considering additional variables (e.g., through discretization). Regardless, in either case, we expect that the requirement of using those typically rather weak combinatorial Benders cuts would significantly deteriorate the performance of such an algorithm.

Instead of pursuing this idea, in the following we will prove that the integrality of the flow variables in the subproblems associated to $\mathrm{H}_2^{\mathrm{AS}}$ and H_3 can be relaxed, thus enabling the application of a Benders decomposition algorithm in its standard, and typically better-performing form.

For each $\{s,t\} \in \mathcal{R}$, let $M_{st}^3(\bar{\mathbf{x}})$ be the integral Benders subproblem defined by constraints (9)–(13) together with (21)–(25) for a given candidate vector $\bar{\mathbf{x}}$ of variable values for $\mathbf{x}$ (i.e., variables $\mathbf{x}$ are replaced by their current values $\bar{\mathbf{x}}$ in constraints (12) and (24)). Let furthermore, $M_{st}^{3\mathrm{R}}(\bar{\mathbf{x}})$ be the linear relaxation of $M_{st}^3(\bar{\mathbf{x}})$ in which constraints (13) and (25) are replaced by their relaxed variants (26) and (27), respectively.

$$y_{ij}^{st,h} \geq 0 \qquad (i,j) \in A_{st}, h \in \{1,...,H_{st}\} \tag{26}$$

$$\hat{z}_{ij}^{stl,h} \geq 0 \qquad (i,j) \in A'_{st}, l \in \{1,...,H_{st}\}, h \in \{1,...,H'_{st}\} \tag{27}$$

Theorem 1. *Consider an arbitrary commodity $\{s,t\} \in \mathcal{R}$ and let $\bar{\mathbf{x}} \in \{0,1\}^{|E|}$ denote a current set of values for variables $\mathbf{x} \in \{0,1\}^{|E|}$. Then there exists a feasible solution for $M^3_{st}(\bar{\mathbf{x}})$, if and only if there exists a feasible solution for $M^{3\mathrm{R}}_{st}(\bar{\mathbf{x}})$.*

Proof. Notice that each solution of $M^3_{st}(\bar{\mathbf{x}})$ is also feasible for $M^{3\mathrm{R}}_{st}(\bar{\mathbf{x}})$ as the latter is a relaxation of the former. It therefore remains to show that there exists a feasible solution of $M^3_{st}(\bar{\mathbf{x}})$ whenever $M^{3\mathrm{R}}_{st}(\bar{\mathbf{x}})$ is feasible. Thus, let $\bar{\mathbf{y}}$ and $\bar{\mathbf{z}}$ denote the values of variables $\mathbf{y}$ and $\hat{\mathbf{z}}$ in an arbitrary solution of $M^{3\mathrm{R}}_{st}(\bar{\mathbf{x}})$. Let furthermore, $\bar{G}_{st} = (\bar{V}_{st}, \bar{E}_{st})$ denote the subgraph of G induced by all edges $e = \{i,j\} \in E$ such that $\bar{x}_e = 1$ in the considered solution to $M^{3\mathrm{R}}_{st}(\bar{\mathbf{x}})$, i.e., $\bar{E}_{st}$ is the set of edges for which $\sum_{h=1}^{H_{st}} \left(\bar{y}^{st,h}_{ij} + \bar{y}^{st,h}_{ji}\right) + \sum_{h=1}^{H'_{st}} \left(\bar{z}^{st,h}_{ij} + \bar{z}^{st,h}_{ji}\right) > 0$ holds. We will show that

(i) $\bar{G}_{st}$ contains a path $P \subset \bar{E}_{st}$ between s and t (an (s,t)-*path*) with length at most H_{st}, and

(ii) $\bar{G}_{st}$ contains an (s,t)-path $P'[e] \subset \bar{E}_{st} \setminus \{e\}$ of length at most H'_{st} for each $e \in P$.

and consequently, $\bar{G}_{st}$ satisfies CH1 for commodity $\{s,t\} \in \mathcal{R}$, which implies that $M^3_{st}(\bar{\mathbf{x}})$ contains a feasible solution.

case (i): From constraints (9)-(12) we conclude that each feasible solution to $M^{3\mathrm{R}}_{st}(\bar{\mathbf{x}})$ must contain an (s,t)-path $P = \{\{s = v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{q-1}, v_q = t\}\}$ of length at most H_{st}, i.e., $q \le H_{st}$, with $\bar{y}^{st,h+1}_{v_h,v_{h+1}} > 0$, such that $\{v_h, v_{h+1}\} \in E_{st}$ for each $h \in \{0, \ldots, q-1\}$). Together with the integrality of $\bar{\mathbf{x}}$ and linking constraints (12) these nonzero flow values imply that $\{v_h, v_{h+1}\} \in \bar{E}_{st}$ for each $h \in \{0, \ldots, q-1\}$ and thus $P \subset \bar{E}_{st}$.

case (ii): Consider an arbitrary $e = \{i,j\}$ included in the (primary) path P whose existence is shown above. Since, $\sum_{h=0}^{H_{st}} (\bar{y}^{st,h}_{uv} + \bar{y}^{st,h}_{vu}) > 0$ holds for each $\{u,v\} \in P$ it is sufficient to prove the existence of $P'[e]$ in $\bar{G}_{st}$ for the following three cases: (a) $\sum_{h=1}^{H_{st}} \left(\bar{y}^{st,h}_{ij} + \bar{y}^{st,h}_{ji}\right) < 1$; (b) there exists $l \in \{1, \ldots, H_{st}\}$ such that $\bar{y}^{st,l}_{ij} + \bar{y}^{st,l}_{ji} = 1$; and (c) $\sum_{h=1}^{H_{st}} \left(\bar{y}^{st,h}_{ij} + \bar{y}^{st,h}_{ji}\right) = 1$ but there does not exist $l \in \{1, \ldots, H_{st}\}$ such that $\bar{y}^{st,l}_{ij} + \bar{y}^{st,l}_{ji} = 1$.

case (a): In that case, constraints (21)–(23) imply the existence of (s,t)-path $P'[e]$ of length at most $H_{st} \le H'_{st}$ such that $e \notin P$ and for which $\sum_{h=1}^{H_{st}} \left(\bar{y}^{st,h}_{uv} + \bar{y}^{st,h}_{vu}\right) > 0$ for each $\{u,v\} \in P'[e]$. Together with linking constraints (12) the latter argument implies that $P'[e] \subset \bar{E}_{st}$.

case (b): Here, constraints (21)–(23) and (24) imply that there exists at least one (s,t)-path $P'[e]$ of length at most H'_{st} with $e \notin P'[e]$ and such that $\sum_{h=1}^{H'_{st}} \left(\bar{z}^{stl,h}_{uv} + \bar{z}^{stl,h}_{vu}\right) > 0$ holds for each $\{u,v\} \in P'[e]$. Thus, constraints (24) and the integrality of $\bar{\mathbf{x}}$ ensure that $P'[e] \in \bar{E}_{st}$.

case (c): Let $h' = \mathrm{argmin}_{h \in \{1,\ldots,H_{st}\}} \{\bar{y}^{st,h}_{ij} + \bar{y}^{st,h}_{ji} > 0\}$, i.e., the smallest hop value such that there exists a non-zero primary flow on the considered edge $e = \{i,j\}$. Let, furthermore $h'' = \mathrm{argmin}_{h \in \{1,\ldots,H_{st}\}} \{\sum_{(i',t) \in A_{st}} \bar{y}^{st,h}_{i't} > 0\}$ be the length of a shortest path induced by the fractional flow values $\bar{\mathbf{y}}$. We observe that constraints (9)–(11) ensure that $h' \le h''$ since $\sum_{h=1}^{H_{st}} \left(\bar{y}^{st,h}_{ij} + \bar{y}^{st,h}_{ji}\right) = 1$. Thus, $\sum_{h=h'}^{H_{st}} \sum_{(i',t) \in A_{st}} \bar{y}^{st,h}_{i't} = 1$ due to equations (11) and therefore $\sum_{(s,j') \in A'_{st}} \bar{z}^{sth',1}_{sj'} = 1$, i.e., one unit of flow associated to backup path with index h' needs to be routed from s to t due to (21)–(23). Since $\bar{y}^{st,h'}_{ij} + \bar{y}^{st,h'}_{ji} > 0$, constraints (24) ensure that $\sum_{h=1}^{H'_{st}} \left(\bar{z}^{sth',h}_{ij} + \bar{z}^{sth',h}_{ji}\right) < 1$. As a consequence, there must exist a path $P'[e]$ of length at most H'_{st}, such that $e \notin P'[e]$, for which $\sum_{h=1}^{H'_{st}} \left(\bar{z}^{sth',h}_{uv} + \bar{z}^{sth',h}_{vu}\right) > 0$ for each $\{u,v\} \in P'[e]$. Using constraints (24) and the integrality of $\bar{\mathbf{x}}$ we conclude that $P'[e] \in \bar{E}_{st}$.

□

To prove a similar result for $\mathrm{H}^{\mathrm{AS}}_2$ given in Theorem 2, let $M^2_{st}(\bar{\mathbf{x}})$ be the integral Benders subproblem defined by constraints (9)–(20) for each candidate vector $\bar{\mathbf{x}}$ (i.e., variables $\mathbf{x}$ are replaced by their current values $\bar{\mathbf{x}}$ in constraints (12), (18) and (19)) and let $M^{2R}_{st}(\bar{\mathbf{x}})$ be its linear relaxation obtained by replacing (13) and (20) by (26) and (28).

$$\tilde{z}_{ij}^{st,h} \geq 0 \qquad (i,j) \in A'_{st},\ h \in \{1,\dots,H'_{st}\} \tag{28}$$

Theorem 2. *Consider an arbitrary commodity $\{s,t\} \in \mathcal{R}$ and let $\bar{\mathbf{x}} \in \{0,1\}^{|E|}$ denote a current set of values for variables $\mathbf{x} \in \{0,1\}^{|E|}$. Then there exists a feasible solution for $M^2_{st}(\bar{\mathbf{x}})$, if and only if there exists a feasible solution for $M^{2\mathrm{R}}_{st}(\bar{\mathbf{x}})$.*

Proof. Similar to Theorem 1 it suffices to show that there exists a feasible solution of $M^2_{st}(\bar{\mathbf{x}})$ whenever $M^{2\mathrm{R}}_{st}(\bar{\mathbf{x}})$ is feasible (the other direction is trivial). Thus, let $\bar{\mathbf{y}}$ and $\bar{\mathbf{z}}$ denote the values of variables $\mathbf{y}$ and $\tilde{\mathbf{z}}$ in an arbitrary solution of $M^{2\mathrm{R}}_{st}(\bar{\mathbf{x}})$. Let furthermore, $\bar{G}_{st} = (\bar{V}_{st}, \bar{E}_{st})$ denote the subgraph of G induced by all edges $e = \{i,j\} \in E$ such that $\bar{x}_e = 1$ in the considered solution to $M^{2\mathrm{R}}_{st}(\bar{\mathbf{x}})$, i.e., $\bar{E}_{st}$ is the set of edges for which $\sum_{h=1}^{H_{st}} \left(\bar{y}_{ij}^{st,h} + \bar{y}_{ji}^{st,h}\right) + \sum_{h=1}^{H'_{st}} \left(\bar{z}_{ij}^{st,h} + \bar{z}_{ji}^{st,h}\right) > 0$ holds. Since the models for the primary path in $M^{2\mathrm{R}}_{st}(\bar{\mathbf{x}})$ and $M^{3\mathrm{R}}_{st}(\bar{\mathbf{x}})$ are identical we conclude (by the same arguments as in the proof of Theorem 1) that $\bar{G}_{st}$ contains a (primary) path $P \subset \bar{E}_{st}$ such that $|P| \leq H_{st}$.

It remains to show that for each $e \in P$, $\bar{G}_{st}$ contains a path $P'[e] \subset \bar{E}_{st} \backslash \{e\}$ of length at most H'_{st}. We first observe that the existence of this path follows from the arguments used in the proof of Theorem 1 if $\sum_{h=1}^{H_{st}} \left(\bar{y}_{ij}^{st,h} + \bar{y}_{ji}^{st,h}\right) < 1$. Thus, let $\sum_{h=1}^{H_{st}} \left(\bar{y}_{ij}^{st,h} + \bar{y}_{ji}^{st,h}\right) = 1$ and let $\bar{l} = \sum_{h=1}^{H_{st}} \sum_{(i',t)\in A_{st}} h\bar{y}_{i't}^{st,h}$. Constraints (14)–(16) ensure that $\bar{l}$ units of flow are routed along a set of paths of length at most H'_{st} each from s to t (possibly with fractional flow values). Linking constraints (19) and integrality of $\bar{\mathbf{x}}$ ensures that each edge contained in such a path is contained in $\bar{E}_{st}$. From constraints (17) and our assumption regarding $\bar{\mathbf{y}}$ we conclude, however, that $\sum_{h=1}^{H'_{st}} \left(\bar{z}_{ij}^{st,h} + \bar{z}_{ji}^{st,h}\right) < \bar{l}$, i.e., not all flow $\bar{\mathbf{z}}$ can be routed via edge e. Consequently, there must exist at least one path $P'[e]$ in $\bar{G}_{st}$ that does not use edge e. □

The relevance of Theorems 1 and 2 is that the standard Benders method can, now, be directly applied to Formulations $\mathrm{H}_2^{\mathrm{AS}}$ and H_3. In the master problem we iteratively fix the values of $\mathbf{x}$ to $\bar{\mathbf{x}}$. The Benders subproblems (which are separable by commodity) are then defined by $M^{2\mathrm{R}}_{st}(\bar{\mathbf{x}})$ and $M^{3\mathrm{R}}_{st}(\bar{\mathbf{x}})$ for each commodity $\{s,t\} \in \mathcal{R}$.

In order to solve the NDPVC, we propose two branch-and-cut methods, BC2 and BC3, where a Benders decomposition of Formulations $\mathrm{H}_2^{\mathrm{AS}}$ and H_3 respectively is solved at each node of the branch-and-bound tree.

3 Primal heuristic

In this section, we propose a matheuristic for the NDPVC that will be (optionally) used to provide initial solutions to the three branch-and-cut methods described in the previous section. The heuristic has two phases. In the first phase, a randomized greedy heuristic is used to generate several candidate solutions. To keep the required computing time relatively low, only the best of these solutions (the one that was first found among those, in case of ties) is subsequently improved via local search in the second phase. The underlying procedure of our matheuristic is shown in Algorithm 1, where Lines 1-12 refer to the construction phase, and Line 13 refers to the improvement phase. In the end, the method either returns a feasible solution, defined by its set of edges Ω^*, or an empty set if the considered instance is infeasible.

In each iteration of the construction phase, we consider a random order of the set of commodities (Line 3), which is provided to the `Greedy` procedure, that creates a feasible solution (Line 4). In our experiments, we set a fixed number of ten iterations for this phase (Line 2). At the end of each iteration, the cheapest solution is stored (Lines 9 and 10).

Algorithm 2 details the `Greedy` procedure mentioned in Line 4 of Algorithm 1. This procedure iteratively designs the primary and backup paths of each commodity, following the order given as input. For each commodity a greedy approach is employed that first selects a primary path, and then, for each edge used in that path, iteratively identifies a backup path that does not use it. In order to obtain each path, a hop-constrained shortest path problem is solved (function `Find_HCSP`). The latter is essentially a resource-constrained shortest path problem (RCSP), with a single resource, and a consumption of one unit associated with every arc in the network. A well-known dynamic programming algorithm for the

Algorithm 1: Matheuristic for the NDPVC.

$\Omega^* \leftarrow \emptyset$
for *each start of construction phase* **do**
  $\Delta \leftarrow$ random order of $\mathcal{R}$
  $\Omega \leftarrow$ `Greedy`(Δ)
  if $\Omega = \emptyset$ **then**
    Problem instance is infeasible
    return $\emptyset$
  end
  if $\sum_{e \in \Omega} c_e < \sum_{e \in \Omega^*} c_e$ **then**
    $\Omega^* \leftarrow \Omega$
  end
end
$\Omega^* \leftarrow$ `Local_Search`(Ω^*)
return Ω^*

Algorithm 2: `Greedy`(Δ).

input: Ordered set of all commodities, Δ.
$\Omega \leftarrow \emptyset$
$\bar{c}_e \leftarrow c_e,\ e \in E$
forall $\{s, t\} \in \Delta$ **do**
  $P \leftarrow$ `Find_HCSP`$(s, t, H_{st}, \bar{c})$ // *cheapest* (s,t)*-path with at most* H_{st} *hops w.r.t. costs* $\bar{\mathbf{c}}$
  if $\sum_{e \in P} \bar{c}_e < \infty$ **then**
    $\Omega \leftarrow \Omega \cup P$
    $\bar{c}_e \leftarrow 0,\ e \in P$
    forall $e \in P$ **do**
      $\bar{c}_e \leftarrow \infty$
      $P'[e] \leftarrow$ `Find_HCSP`$(s, t, H'_{st}, \bar{c})$
      if $\sum_{e \in P'[e]} \bar{c}_e < \infty$ **then**
        $\Omega \leftarrow \Omega \cup P'[e]$
      else
        Problem instance is infeasible
        return $\emptyset$
      end
      $\bar{c}_e \leftarrow 0$
    end
  else
    Problem instance is infeasible
    return $\emptyset$
  end
end
return Ω

RCSP, originally proposed in [2], runs in time $\mathcal{O}(|V||E|)$ for the considered case, i.e., one resource and nonnegative edge weights.

Recall that, for a given set of edges, the subproblem of finding a subset of edges containing a primary path, and the required backup paths for one commodity, is independent from the same subproblem for other commodities. Thus, choosing a particular subset of edges at a given iteration of the greedy algorithm does not compromise the existence of a feasible paths for commodities considered in subsequent iterations. Furthermore, as it will be pointed out in Theorem 3, choosing and fixing a primary path before deciding upon the required backup paths, always yields a feasible solution, whenever such a solution exists. Thus, Theorem 3 shows that Algorithm 2 always finds a feasible solution for the NDPVC, provided that there exists at least one for the given instance.

Theorem 3. *Let $P \in \mathcal{F}_{st}$ be an arbitrary (s,t)-path of length at most H_{st} for commodity $\{s,t\} \in \mathcal{R}$, i.e., a primary path for that commodity. Assume that there exists an edge $e \in P$ for which no backup path exists, i.e., there does not exist an (s,t)-path $P'[e] \subseteq E'_{st} \setminus \{e\}$ of length at most H'_{st}. Then, $\mathcal{B}_{st} = \emptyset$, i.e., the problem instance is infeasible because there does not exists another primary path, say $\hat{P} \subset E_{st}$, $|P| \leq H_{st}$, such that there exists a backup path $\hat{P}'[e] \subset E'_{st} \setminus \{e\}$, $|\hat{P}'[e]| \leq H'_{st}$, for each $e \in \hat{P}$.*

Proof. Let $P \subset E_{st}$, $|P| \leq H_{st}$, be a primary path (s,t)-path for which the necessary backup paths do not exist. Assume indirectly that there exists an alternative primary path $\hat{P} \subset E_{st}$, $|\hat{P}| \leq H_{st}$, $P \neq \hat{P}$, for which the required backup paths do exist. Let $e \in P$ be an edge such that there does not exist an (s,t)-path $P'[e] \subseteq E'_{st} \setminus \{e\}$ of length at most H'_{st}. We first observe that $e \notin \hat{P}$ immediately contradicts our assumption, since in this case $\hat{P}$ is an (s,t)-path of length at most $H_{st} \leq H'_{st}$.

Thus, $e \in \hat{P}$. Since all required backup paths exist for $\hat{P}$ by assumption, we know that there exists an (s,t)-path $\hat{P}'[e] \subseteq E'_{st} \setminus \{e\}$ of length at most H'_{st}. Consequently, $\hat{P}'[e]$ is also a valid backup path for edge e in path P, thus contradicting our initial assumption that such a path does not exist. □

Algorithm 3 describes the general structure of the improvement phase, that is referred to in Line 13 of Algorithm 1. At each iteration a restricted NDPVC is solved (Line 4), that only considers a subset of commodities $\bar{\mathcal{R}}$, chosen in function `Choose_Commodities` (Lines 1 and 8), while all edges used to establish valid connections (i.e., primary and backup paths) for the remaining commodities are fixed to be included in the solution. In case a cheaper solution is identified, the incumbent solution is replaced by this new solution.

Algorithm 3: `Local_Search`(Ω^*).

input: Solution Ω^*.
1 $\bar{\mathcal{R}} \leftarrow$ `Choose_Commodities`
2 **while** $\bar{\mathcal{R}} \neq \emptyset$ *and stopping criteria not met* **do**
3 $\quad \hat{E} \leftarrow \Omega^* \setminus \{e \in \Omega^* \mid \Psi_e = \bar{\mathcal{R}}\}$ // *fixed edges*
4 $\quad v \leftarrow \min\{\sum_{e \in E \setminus \Omega^*} c_e x_e - \sum_{e \in \Omega^* \setminus \hat{E}} c_e(1-x_e) \wedge \mathbf{x} \in \mathcal{F}_{st} \cap \mathcal{B}_{st},\ \forall \{s,t\} \in \bar{\mathcal{R}}, \mathbf{x} \in \{0,1\}^{|E|}, x_e = 1, \forall e \in \hat{E}\}$
5 $\quad$ **if** $v < 0$ **then**
6 $\quad\quad$ `Update_Solution` Ω^*
7 $\quad$ **end**
8 $\quad \bar{\mathcal{R}} \leftarrow$ `Choose_Commodities`
9 **end**

In each iteration, the restricted NDPVC is solved with the general-purpose ILP solver CPLEX. Based on the results of preliminary experiments, an appropriate (small) adaptation of formulation H_3, originally proposed by Gouveia and Leitner [11] (see also Section 2.2.1), is used. In order to reduce the solving time of the restricted problem, we fix all variables corresponding to edges $e \in \hat{E}$ to one. In addition, and to further ensure that each iteration of Algorithm 3 is not too time consuming, we do not necessarily solve each restricted problem to optimality, but instead set a time limit of two minutes, and set CPLEX's settings prioritizing the search for feasible solutions over proving optimality (`MIP Emphasis := Feasibility`).

While the current incumbent is defined by its set of edges Ω^* in Algorithm 3, we also store and update a mapping $\Psi : \Omega^* \rightarrow 2^{\mathcal{R}}$, i.e., for each edge $e \in \Omega^*$, Ψ_e is the set of commodities that use edge e for one if its paths. Together with parameter $N \in \mathbb{N}$, this mapping is used in procedure `Choose_Commodities` to identify a promising subset of commodities $\bar{\mathcal{R}}$. To this end, we solve the optimization problem $\bar{\mathcal{R}} \leftarrow \arg\max\{\sum_{e \in \Omega^*} c_e \alpha_e : |\Psi_e| \alpha_e \leq \sum_{\{s,t\} \in \Psi_e} \beta^{st}, \sum_{\{s,t\} \in \mathcal{R}} \beta^{st} \leq N, (\boldsymbol{\alpha}, \boldsymbol{\beta}) \in \{0,1\}^{|\Omega^*| \times |\mathcal{R}|}\}$. Thereby, for each commodity $\{s,t\} \in \mathcal{R}$, variable $\beta_{st} \in \{0,1\}$ indicates whether commodity $\{s,t\}$ will included in set $\bar{\mathcal{R}}$, while variable $\alpha_e \in \{0,1\}$ indicates whether edge $e \in E$ can be removed from Ω^* for a particular choice of $\bar{\mathcal{R}}$. We restrict the cardinality of set $\bar{\mathcal{R}}$ to at most N commodities and avoid to repeatedly choose the same set(s) of commodities by adding constraint $\sum_{\{s,t\} \in \bar{\mathcal{R}}'} \beta^{st} - \sum_{\{s,t\} \notin \bar{\mathcal{R}}'} \beta^{st} \leq |\bar{\mathcal{R}}'| - 1$, for every subset of commodities $\bar{\mathcal{R}}'$ chosen in previous iterations. To achieve a good balance between the size of the search space and the time needed for solving the restricted subproblem (Line 4) we initially set N to be the smallest value that may yield an improvement, i.e., $N \leftarrow \min_{e \in \Omega^*} |\Psi_e|$. Parameter N is increased to $\min_{e \in \Omega^*} \{|\Psi_e| : |\Psi_e| > N\}$ if no solution to the optimization problem solved in function `Choose_Commodities` is found, or after five consecutive non-improving iterations of Algorithm 3. Even through preliminary experiments indicated that the optimization problem solved in `Choose_Commodities` is generally solved very quickly, we still set a solving time limit of 30 seconds, and set CPLEX's `MIP Emphasis` to `Feasibility`. Algorithm 3 is terminated after at most 10 minutes or after 15 consecutive, non-improving iterations.

Note that we could alternatively solve the restricted problems via the Benders decomposition method described in the previous section. However, since the integrality of the routing variables is not enforced in that case, solutions with fractional values for the flow variables may be obtained; this makes it complicated to identify the minimal set of edges necessary to route all paths associated with each commodity. As seen above, this identification is crucial in future iterations, in order to select a "promising" subset of commodities $\bar{\mathcal{R}}$ in the function `Choose_Commodities`. Therefore, it is important to obtain solutions with single-path routing for every path. We can only guarantee this by imposing the integrality of the routing variables, and solving the restricted problems with CPLEX's standard branch-and-bound procedure.

4 Computational experiments

In this section, we present the results of computational experiments, conducted with the objective of: a) understanding under which configurations the branch-and-cut methods proposed in Section 2 are more efficient in solving instances of the NDPVC, b) comparing the efficiency of these branch-and-cut methods, with that of CPLEX's standard solving methods, and of c) continuing the study initiated by Gouveia and Leitner [11], of comparing the solutions obtained by solving the NDPVC to those of the kHSNDP. The results of this tests are presented and analyzed in Sections 4.1 to 4.4, respectively.

For our experiments, we resorted to the instances proposed in [11]. These instances are divided in two classes, based on their graph topology: a first class based on grid graphs with chords, and a second one where only the cheapest edges from a randomly-generated complete graph are kept. For the sake of simplicity, we refer to these class of instances as *grid instances* and *random instances*, respectively. Each of these classes of instances is, in turn, partitioned in two subclasses: grid instances consist of subclasses C and D, which differ in the way commodities are selected; and random instances are divided in subclasses EU and RE, which differ in the way the costs of each edge are set. Tables 1a and 1b detail the most important parameters of each instance set, namely number of nodes ($|V|$), number of edges ($|E|$), number of commodities ($|\mathcal{R}|$), and set size (#). For more details on the construction of these instances, we redirect the reader to [11]. Following [11], we consider uniform hop limits over all commodities in our experiments, i.e., $H = H_{st} = H_{uv}$ and $H' = H'_{st} = H'_{uv}$ for each pair of commodities $\{s, t\}$ and $\{u, v\}$. As such, we additionally present in Tables 1a and 1b the minimum, maximum and average value of $H_{\min}$ for each set, where $H_{\min}$ is the smallest commodity-independent hop limit that may enable a feasible solution of an instance. For each instance, we test the following cases: $H \in \{H_{\min}, H_{\min}+1, H_{\min}+2\}$ and $H' \in \{H, H+1, H+2\}$. Therefore, nine different instances are considered for each original instance yielding a total of 3150 test instances - 2070 grid instances and 1080 random instances.

All our experiments were performed on a single core within a cluster of computers, each consisting of 20 cores (2.3GHz). To each experiment, we set a time limit of 7 200 CPU-seconds and a memory limit of 3GB. All formulations and methods were implemented in C++, using IBM ILOG CPLEX 12.7. The latter has been used in its standard settings, except when used in the initial heuristic; in those cases, we emphasize feasibility when solving the subproblems, see Section 3. In BC2 and BC3, we resort to the built-in Benders decomposition method of CPLEX 12.7.

For the branch-and-cut algorithm based on formulation $H_{\mathrm{I}}^{\mathrm{L}}$, BC1, we also consider variant $\mathrm{BC1}_{\mathrm{I}}$, in which the model is initialized through additional constraints (29)–(31) in order to potentially decrease the number of separated cutset constraints (2) and (3) and/or iterations of the cutting plane loop. In addition, these constraints may help the solver to identify general-purpose cuts.

$$\sum_{h=1}^{H_{st}} \sum_{(i_{h-1}, t_h) \in \delta^-(t_h)} X_{it}^{sh} \geq 1 \qquad s \in T, t \in T(s) \tag{29}$$

Table 1: Summary of parameters for each instance set.

(a) Instance sets C and D.

Set	$\|V\|$	$\|E\|$	$\|\mathcal{R}\|$	#	$H_{\min}$ min	$H_{\min}$ avg	$H_{\min}$ max
C-1	100	342	5	20	3	4.7	7
C-2	100	342	10	20	4	5.4	7
C-3	400	1482	5	20	3	4.7	7
C-4	400	1482	10	20	4	5.1	7
C-5	400	1482	20	20	4	5.8	7
C-6	900	3422	5	20	3	4.7	7
C-7	900	3422	10	20	4	5.4	7
C-8	900	3422	20	20	4	5.6	7
C-9	900	3422	30	20	4	5.9	7
D-1	25	72	10	10	3	3.8	4
D-2	49	156	10	10	4	5.2	6
D-3	100	342	10	10	7	8.2	9
D-4	100	342	45	10	6	8.1	9
D-5	400	1482	10	10	12	14.6	18

(b) Instance sets E and RE.

Set	$\|V\|$	$\|E\|$	$\|\mathcal{R}\|$	#	$H_{\min}$ min	$H_{\min}$ avg	$H_{\min}$ max
EU-1	50	122	10	5	6	7.6	9
EU-2	50	122	45	5	7	8.6	11
EU-3	50	245	10	5	4	4.6	6
EU-4	50	245	45	5	4	4.8	6
EU-5	75	277	10	5	5	5.6	6
EU-6	75	277	45	5	6	8.6	12
EU-7	75	555	10	5	3	4.4	6
EU-8	75	555	45	5	4	4.6	5
EU-9	100	495	10	5	5	5.2	6
EU-10	100	495	45	5	6	7.2	9
EU-11	100	990	10	5	3	4.0	5
EU-12	100	990	45	5	3	4.8	6
RE-1	50	122	10	5	3	4.6	6
RE-2	50	122	45	5	4	5.2	6
RE-3	50	245	10	5	2	2.8	3
RE-4	50	245	45	5	3	3.0	3
RE-5	75	277	10	5	3	3.8	4
RE-6	75	277	45	5	4	4.2	5
RE-7	75	555	10	5	2	2.6	3
RE-8	75	555	45	5	2	2.8	3
RE-9	100	495	10	5	3	3.6	5
RE-10	100	495	45	5	4	4.0	4
RE-11	100	990	10	5	2	2.0	2
RE-12	100	990	45	5	3	3.0	3

$$\sum_{h=1}^{H'_{st}} \sum_{(i_{h-1},t_h)\in\delta^-(t_h)} X_{it}^{sh} \geq 2 \qquad s \in T, t \in T(s) \tag{30}$$

$$\sum_{(i_{h-1},j_h)\in\delta^-(j_h):i\neq k} X_{ij}^{sh} \geq X_{jk}^{sh} \qquad s \in S, (j_{h-1},k_h) \in \tilde{A}_{\mathrm{L}}, h \geq 2 \tag{31}$$

Constraints (29) and (30) are indegree constraints for terminals, while (31) are a well-known set of compact connectivity constraints for layered graph formulations. Connectivity cuts (2) and (3) are separated using a push-relabel algorithm for the maximum flow problem [6]. Thereby, we only search for violated constraints (3) that ensure the required redundancy if no simple connectivity cuts (2) are violated by the current LP solution.

For both types of cuts, we add a small epsilon to each flow capacity in order to favor sparse cuts (i.e. containing a small number of variables). Additionally, we try to identify orthogonal cuts for each source by setting to one, the capacities of all arcs that are contained in a cut (for this source) already added in the current iteration of the cutting-plane loop.

To keep the number of added constraints at an acceptable level, we only add constraints (2) and (3) if they are violated by a value of more than 0.1. Finally, for each source, in our separation routine we only consider inequalities (3) for edges such that the sum of the current LP-relaxation values of their corresponding arcs on the first H layers is at least 0.1, i.e., only those that are likely to be used "significantly" within primary paths in the current solution.

Table 2: Average running times (t^{g}_{avg} and t^{g+ls}_{avg}) in seconds and gaps to the best-known solution ((gap^{g}_{avg} and gap^{g+ls}_{avg})) in percent, for the greedy heuristic (g) and the greedy heuristic with subsequent local search (g+ls).

(a) Instance sets C and D

Set	t^{g}_{avg}	t^{g+ls}_{avg}	gap^{g}_{avg}	gap^{g+ls}_{avg}
C-1	0	204	19	1
C-2	0	403	21	6
C-3	0	129	21	1
C-4	0	211	19	3
C-5	0	441	21	5
C-6	0	90	19	0
C-7	0	249	22	2
C-8	1	360	22	4
C-9	1	469	19	4
D-1	0	58	13	3
D-2	0	240	16	3
D-3	0	518	17	8
D-4	1	351	24	13
D-5	2	647	18	10
Avg	0	301	20	4

(b) Instance sets E and RE.

Set	t^{g}_{avg}	t^{g+ls}_{avg}	gap^{g}_{avg}	gap^{g+ls}_{avg}
EU-1	0	329	10	4
EU-2	0	226	7	6
EU-3	0	300	15	8
EU-4	0	263	17	13
EU-5	0	310	13	7
EU-6	1	487	13	8
EU-7	0	379	18	10
EU-8	1	375	17	11
EU-9	0	388	13	7
EU-10	1	483	11	6
EU-11	0	437	12	5
EU-12	1	519	12	6
RE-1	0	190	18	6
RE-2	0	150	15	11
RE-3	0	171	16	5
RE-4	0	168	25	17
RE-5	0	258	18	6
RE-6	0	152	20	12
RE-7	0	274	17	6
RE-8	0	233	27	15
RE-9	0	289	20	7
RE-10	1	271	19	10
RE-11	0	206	20	7
RE-12	1	346	23	12
Avg	0	300	16	9

4.1 Heuristic solution quality

In this section, we analyze the performance of the matheuristic proposed in Section 3. Table 2 provides a first overview of its performance, with separate focus given to each phase in it: the construction or greedy phase (represented by the superscript g) and the full heuristic (indicated by the superscript g+ls). Besides reporting average run times, we also present average gaps between the cost of the best-known solution (UB^*) and that of the solution obtained at the end of each phase (UB^{g}_{avg} and UB^{g+ls}_{avg} for the greedy phase and the full algorithm, respectively), i.e., for each instance we have $gap^{g} = \frac{UB^{g}-UB^*}{UB^{g}}$ and $gap^{g+ls} = \frac{UB^{g+ls}-UB^*}{UB^{g+ls}}$, while gap^{g}_{avg} and gap^{g+ls}_{avg} are the averages over the corresponding subsets of instances. These numbers are grouped by instance subset in Table 2. Additional insights can be obtained from Table 4 in Appendix A, where the same data is grouped by considered hop limits.

We first observe that the greedy heuristic is extremely fast (at most two seconds) in finding feasible solutions, which are on average 20% and 16% more expensive than the best solution known for grid and random instances, respectively. One additional advantage that immediately results from Theorem 3 is that the greedy heuristic provides an efficient way to determine whether an instance is feasible or not.

The results in Table 2 clearly indicate that the second phase (i.e., local search) of the proposed matheuristic is typically able to significantly improve the solution constructed by the greedy heuristic. This effect seems to be more pronounced on grid instances for which the final gap to the overall best-known solution is comparably small (4% on average); for the random instances, it is slightly larger (9% on average). Finally, the average running time of five minutes for both classes of instances suggests that the termination criteria most often met by the heuristic is the limit on iterations without improvement.

We also observe that the solution quality obtained by the greedy heuristic seems to decrease when the number of commodities increases. This can be observed clearly for instances of test set RE (the odd-index instances have 10 commodities, the even-index instances have 45 commodities). In addition, it seems that when considering instances with many commodities, the developed local search also fails to improve the initial solutions as much as for cases with comparably few commodities. For example, for instances of test set RE with 10 commodities the average $gap^{\mathrm{g}}_{\mathrm{avg}}$ is 18% and the value of $\frac{gap^{\mathrm{g+ls}}_{\mathrm{avg}}}{gap^{\mathrm{g}}_{\mathrm{avg}}}$ is 34%; for instances of test set RE with 45 commodities these values are 22% and 60%, respectively.

These results are not surprising, given the structure of both phases of the matheuristic. While, the overall solution quality achieved seems quite acceptable on average (in particular given the relatively simple design of the heuristic) the obtained results also suggest several ideas for further performance improvements: (i) increasing the number of iterations associated with the first phase, benefiting from its negligible run time, in order to explore other solution neighborhoods; (ii) improving the selection of commodities to be resolved in each iteration of the second phase (method `Choose_Commodities` in Algorithm 3), and (iii) increasing the number of possible iterations of the second phase without improvement, taking in mind that the reported average run time of the heuristic is only half the imposed time limit.

4.2 Influence of primal heuristic and initialization constraints

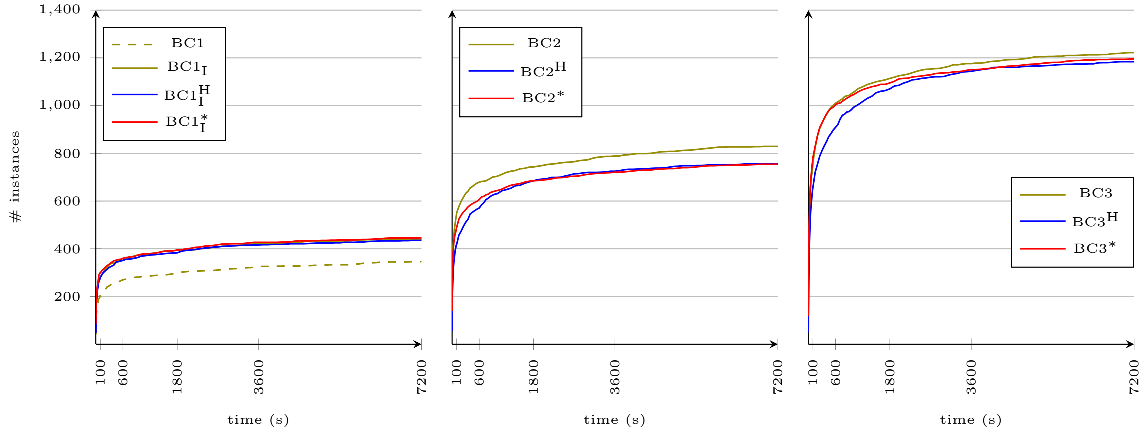
In this section, we aim to identify the best-performing variants of each proposed branch-and-cut method. Thus, we will study (i) the effect the initialization constraints (29)–(31) on BC1 and (ii) the influence of providing an initial solution to each of the branch-and-cut methods by the matheuristic from Section 3. In addition, we will also analyze whether further improvements of this initial heuristic would significantly improve the performance of our branch-and-cut methods.

To distinguish the different configurations of our methods, we use the following notation: BCx indicates the basic branch-and-cut method based on the formulation using CHx, $x \in \{1, 2, 3\}$, while BCx^{H} is the variant of BCx in which the solution obtained by the matheuristic is provided to the solver as initial solution. Finally, BCx^* refers to a variant of BCx in which the overall best-known solution is given as initial solution to CPLEX. Naturally, the latter is not a valid method to solve the NDPVC. As mentioned above, it serves, however, as a benchmark that provides insights on the performance of a branch-and-cut method when initialized with an almost-perfect initial heuristic. Finally, for BC1 we also consider variant BC1_{I} in which constraints (29)–(31) are initially added to the model.

Obtained results are summarized by Figures 2 to 5 while more detailed numerical results are also given in Tables 5 to 8, see Appendix A. Thereby, Figures 2 and 4 depict cumulative numbers of instances of grid and random instances, respectively, solved within a certain CPU time for the considered branch-and-cut variants. Similarly, Figures 3 and 5 show cumulative numbers of instances whose optimality gap (gap_{opt}), is within a certain value. This gap is computed as $gap_{\mathrm{opt}} = \frac{UB-LB}{UB}$, where UB and LB are, respectively, the best lower and upper bound obtained by the corresponding method for the particular instance. In order to keep the results consistent, we assign a run time of 7 200 seconds to instances that exceeded the time limit, and an optimality gap of 100% to instances for which no feasible solution could be identified within the given time limit. Additionally, we assign both a run time of 7 200 seconds and an optimality gap of 100% to instances that exceeded the memory limit, before CPLEX finishes preprocessing the model and starts the solving procedure.

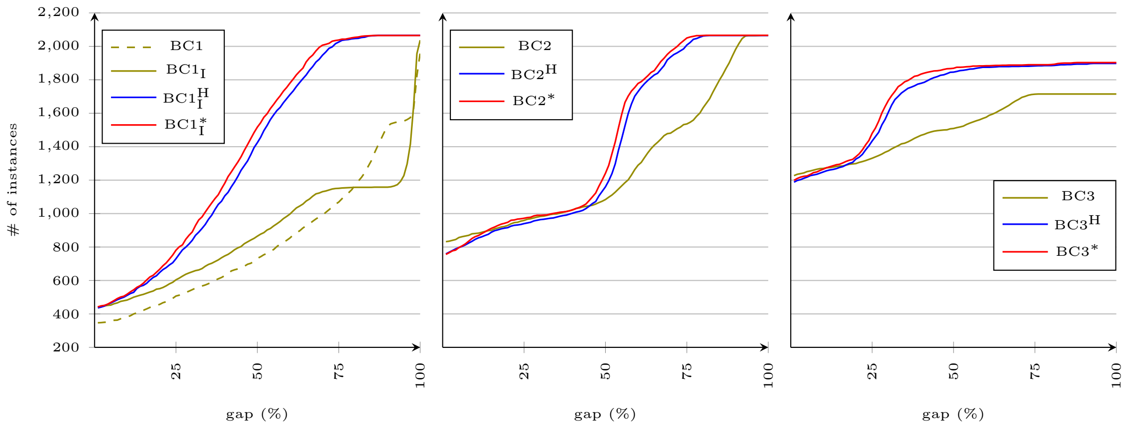
Figures 2a and 4a reveal that including initialization constraints (29)–(31), i.e., considering BC1_{I}, significantly reduces the running times of BC1, without compromising the observed optimality gaps (see Figures 3a and 5a, as well as Tables 5 and 7). Concerning the variants of BC1, we will therefore focus our further analysis on BC1_{I}.

Figures 2 to 5 show that the impact of using a primal heuristic is similar for all three branch-and-cut methods. For grid instances, we observe from Figure 2 that providing an initial solution (even an almost-perfect one as in variants BCx^*) does not seem to improve the efficiency of the branch-and-cut algorithms. In fact, for these instances, we can observe that in its basic configuration, BC1_{I} has a performance similar to BC$1_{\mathrm{I}}^{\mathrm{H}}$, whereas BC2 and BC3 even perform better than BC2^{H} and BC3^{H}, respectively. One could be tempted to believe that the running time of the primal heuristic is the bottleneck here; however, the similar results obtained for BCx^* confirm that this is not case. Instead, we postulate that for some instances in sets C and D, providing CPLEX with a feasible solution - however good it might be - impacts negatively the node selection of the branch-and-bound tree.



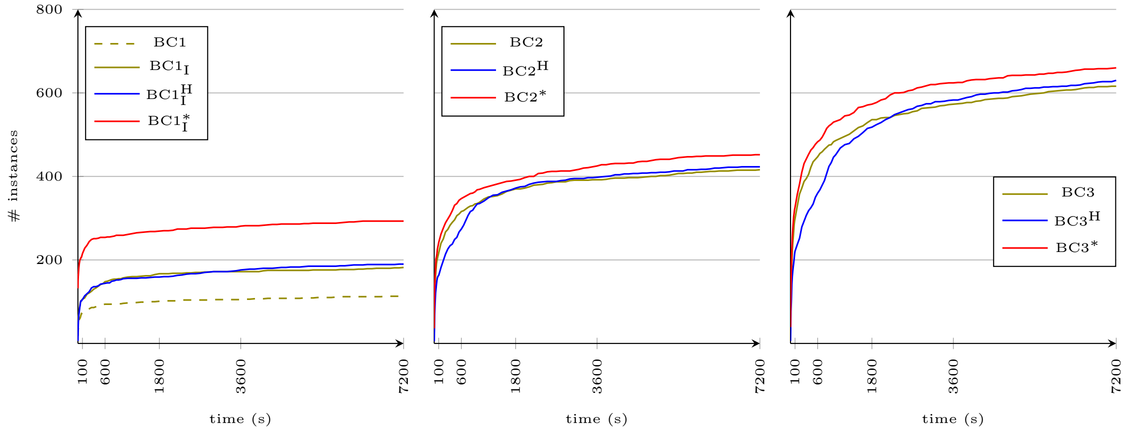
(a) BC1 (b) BC2 (c) BC3

Figure 2: Cumulative numbers of instances of test sets C and D, solved within a certain CPU time. Results are grouped in subfigures according to the branch-and-cut methods (BCx). Results are also shown for variants initialized by the solution from the matheuristic (BCx^H), or with the best known solution (BCx^*). For the BC1, the impact of using initialization constraints (29)–(31) is also shown ($BC1_I$).



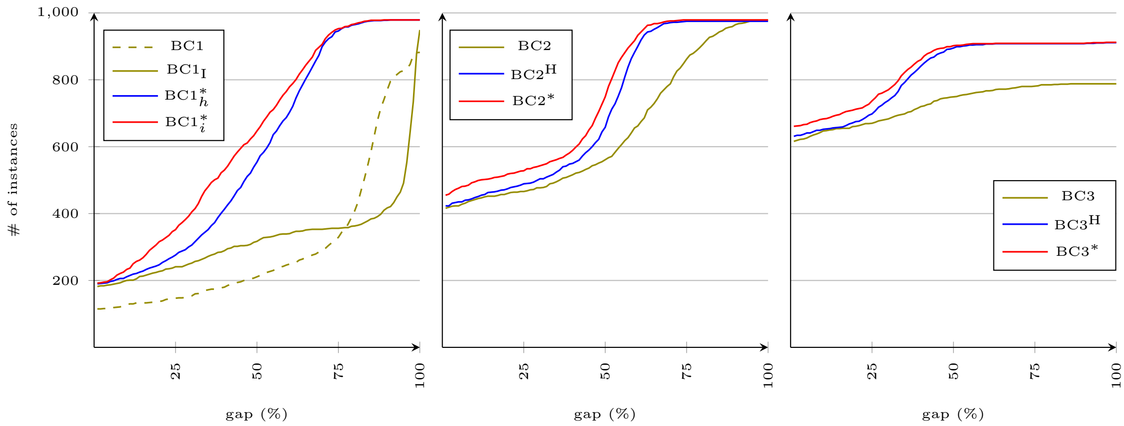
(a) BC1 (b) BC2 (c) BC3

Figure 3: Cumulative numbers of instances of test sets C and D, for which the remaining optimality gap is within a certain value. Results are grouped in subfigures according to the branch-and-cut methods (BCx). Results are also shown for variants initialized by the solution from the matheuristic (BCx^H), or with the best known solution (BCx^*). For the BC1, the impact of using initialization constraints (29)–(31) is also shown ($BC1_I$).



(a) BC1 (b) BC2 (c) BC3

Figure 4: Cumulative numbers of instances of test sets EU and RE, solved within a certain CPU time. Results are grouped in subfigures according to the branch-and-cut methods (BCx). Results are also shown for variants initialized by the solution from the matheuristic (BCx^H), or with the best known solution (BCx^*). For the BC1, the impact of using initialization constraints (29)–(31) is also shown ($BC1_I$).



(a) BC1 (b) BC2 (c) BC3

Figure 5: Cumulative numbers of instances of test sets EU and RE, for which the remaining optimality gap is within a certain value. Results are grouped in subfigures according to the branch-and-cut methods (BCx). Results are also shown for variants initialized by the solution from the matheuristic (BCx^H), or with the best known solution (BCx^*). For the BC1, the impact of using initialization constraints (29)–(31) is also shown ($BC1_I$).

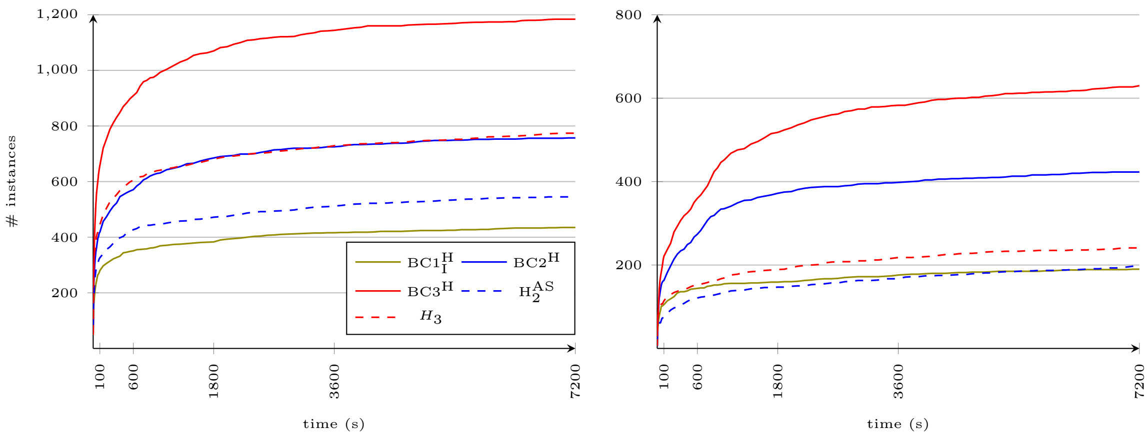
The benefits of a primal heuristic become nevertheless clear when analyzing the optimality gaps depicted in Figures 3 and 5: when a good initial feasible solution is provided, the optimality gaps at the end of the run tend to be substantially lower. This suggests that CPLEX's built-in heuristics are not particularly effective in creating good feasible solutions for the NDPVC. We also observed that for more difficult instances, they even often struggle to find a single solution.

Finally, we conclude that for grid instances, $\mathrm{BC1_I^H}$, $\mathrm{BC2^H}$, and $\mathrm{BC3^H}$ have similar performances than their counterparts, $\mathrm{BC1_I^*}$, $\mathrm{BC2^*}$, and $\mathrm{BC3^*}$ respectively, both in terms of run times and optimality gaps. This suggests that for these type of instances, our proposed primal heuristic performs sufficiently good (for the use of providing an initial solution for a branch-and-cut procedure), since the final performance is almost identical to the one that can be observed when passing an overall best-known solution (which is in fact an optimal solution in many cases) to the branch-and-cut algorithm.

For instances of test sets EU and RE, providing an extremely good primal solution to CPLEX also has a positive impact to the run time: Methods $\mathrm{BCx^*}$, $x \in \{1, 2, 3\}$, are overall faster in solving random instances to optimality; $\mathrm{BC}x^{\mathrm{H}}$ and $\mathrm{BC}x$ present comparable run times, but $\mathrm{BC}x^{\mathrm{H}}$ performs better with respect to the remaining optimality gaps. These results suggest that for random instances, the performance of the branch-and-cut methods may be further improved through the development of a better heuristic. To this end, we note, however, that in contrast to $\mathrm{BC}x^{\mathrm{H}}$, variants $\mathrm{BC}x^*$ also benefit from the fact that they do not devote a portion of the runtime to computing the initial solution.

4.3 Comparison to the state of the art

In this section, we compare the performance of the proposed branch-and-cut methods to the performance of CPLEX's standard solving methods implementing the ILP formulations proposed by Gouveia and Leitner [11]. For this comparison, we consider only the best-performing branch-and-cut method for each characterization (i.e., $\mathrm{BC1_I^H}$, $\mathrm{BC2^H}$, and $\mathrm{BC3^H}$), as well as $\mathrm{H_2^{AS}}$ and $\mathrm{H_3}$, for which the best performance is reported in [11]. Figures 6 and 7 report numbers of solved instances within a given time and numbers of instances with a final optimality gap below some threshold for grid and random instances, respectively.

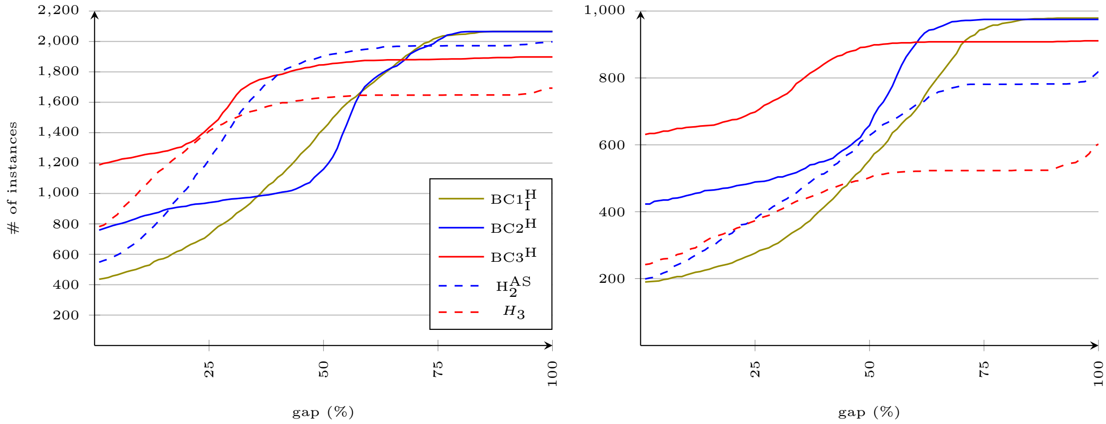


(a) Instance sets C and D

(b) Instance sets EU and RE

Figure 6: Cumulative numbers of instances solved within a certain CPU time, when solved by the branch-and-cut methods and by CPLEX's standard solving methods.

First and foremost, we observe that the branch-and-cut methods $\mathrm{BC2^H}$ and $\mathrm{BC3^H}$ perform significantly better than their respective counterparts, $\mathrm{H_2^{AS}}$ and $\mathrm{H_3}$, both in terms of number of instances solved to optimality (Figure 6) and optimality gaps (Figure 7). Furthermore, on the considered benchmark instances, $\mathrm{BC3^H}$ clearly outperforms all other methods in terms of its ability to solve instances to proven optimality. In the time limit of two hours, $\mathrm{BC3^H}$ is able to solve 1188 of the 2070 grid instances, more than 50% more instances than those solved by the second- and third-fastest methods, $\mathrm{H_3}$ and $\mathrm{BC2^H}$ respectively. From the more detailed results given in Tables 9 to 12 (see Appendix A), we observe that for two sets of grid instances, D-1 and D-2, $\mathrm{BC3^H}$ solves every instance in the time limit; for three



(a) Instance sets C and D

(b) Instance sets EU and RE

Figure 7: Cumulative numbers of instances for which the remaining optimality gap is within a certain value, when solved by the branch-and-cut methods and by CPLEX's standard solving methods.

other sets, C-1, C-3 and C-6 it finishes with an optimality gap of at most 2%. For random instances, this difference in efficiency is also substantial, with BC3^H solving almost 40% more instances than the second-fastest method, BC2^H. For five sets of random instances, EU-1, EU-3, RE-1, RE-3 and RE-5, BC3^H is able to solve every instance within the time limit; for another six, EU-2, EU-5, RE-2, RE-7, RE-9 and RE-11 it terminates with a gap of at most 2%.

As for BC2^H, even though it has a comparable efficiency to that of H_3 in solving grid instances, it clearly dominates the latter when it comes to solving random instances. We also observe that the performance of BC1$_I^H$ is significantly worse than that of BC3^H (or BC2^H) in the sense that it solves fewer grid and random instances. Still, we note that in Figure 7 more instances are reported having optimality gaps under 100% for BC1$_I^H$ (as well as for BC2^H), than for BC3^H; as these methods all use the same primal heuristic, this suggests that the latter might be more likely to surpass the memory limit than its slower counterparts. Finally, consider Tables 10 and 12 that report the results grouped by (H, H'). These results highlight a clear trend: the NDPVC tends to become harder with the increase of $\Delta_{H'}$.

4.4 Comparison to the kHSNDP

To achieve the final goal of our computationally study, we compare the solutions of the NDPVC, to those obtained by solving the related kHSNDP. This study has already been started in [11]. However, the comparably-poor performance of the developed methods prevented the authors from drawing conclusions for most of the instances. The superior methods in the present article allow us to extend their study and to provide additional, conclusive results for 811 grid instances and 405 random instances. Results are summarized in Tables 3a and 3b, grouped by test set, and in Tables 13a and 13b (see Appendix A), grouped by $(H, H') = (H_{\min} + \Delta_H, H_{\min} + \Delta_H + \Delta_{H'})$. We note that conclusions are drawn using the combination of the results from [11] and the new ones obtained through the approaches proposed in the present paper.

From Table 3a, we observe that the optimal solution of the NDPVC is cheaper then the one of the kHSNDP in at least almost 27% of the considered grid instances and equally expensive in at least 60% of the cases. For the remaining grid instances (11% of instances of set C and 20% of set D) our results do not allow to draw conclusions. On instance set C, the likelihood that a cheaper solution can be obtained by solving the NDPVC seems to increase with $\Delta_{H'}$. The difference between the NDPVC and the kHSNDP also seems to be pronounced when more dependencies between the commodities are present (i.e., on instance set D). Here, the solution of the NDPVC is cheaper for 231 instances and of equal cost for 131 instances (88 cases remain undecided).

For random instances, the relative proportion of instances whose NDPVC is cheaper seems to be slightly lower than for grid instances. Here, the NDPVC solutions of at least 23% of these instances are

Table 3: Overall number of cases in which solution to NDPVC is provably better (bt) and provably equal (eq) compared to the kHSNDP, and number of cases where kHSNDP is provably infeasible and NDPVC feasible (feas). Remaining cases could not be decided.

(a) Instance sets C and D.

Set	#	bt	feas	eq
C-1	180	56	0	124
C-2	180	44	0	125
C-3	180	26	0	151
C-4	180	56	0	122
C-5	180	41	0	83
C-6	180	24	0	155
C-7	180	36	0	134
C-8	180	39	0	124
C-9	180	21	0	85
D-1	90	44	0	46
D-2	90	58	0	32
D-3	90	59	0	28
D-4	90	37	0	3
D-5	90	33	0	22
Total	2070	574	0	1234

(b) Instance sets E and RE.

Set	#	bt	feas	eq
EU-1	45	7	0	38
EU-2	45	15	0	30
EU-3	45	24	0	21
EU-4	45	15	0	5
EU-5	45	22	0	21
EU-6	45	10	1	4
EU-7	45	12	0	23
EU-8	45	7	0	4
EU-9	45	21	0	17
EU-10	45	3	0	5
EU-11	45	12	0	12
EU-12	45	3	0	4
RE-1	45	9	0	36
RE-2	45	23	0	13
RE-3	45	7	0	38
RE-4	45	7	0	11
RE-5	45	10	0	35
RE-6	45	7	0	10
RE-7	45	8	0	34
RE-8	45	5	0	9
RE-9	45	7	0	35
RE-10	45	6	0	6
RE-11	45	4	0	40
RE-12	45	5	0	5
Total	1080	249	1	456

cheaper than those of kHSNDP and of equal costs for at least 42%. The results also identify one instance from test set D, for which the NDPVC is feasible and the kHSNDP is infeasible.

Finally, when comparing these results with those presented by Gouveia and Leitner [11], we see that the main gain that we obtain by using the newly proposed branch-and-cut algorithms, is in the identification of many more instances for which the optimal cost is the same for both problems. In particular, the number of such instances now reported is 133% higher than in [11]; in contrast, the relative gain in the number of instances proved to have a smaller optimal cost for the NDPVC is "only" of 44%. This is however not surprising. In fact, in order to prove that an instance has an optimal cost for the NDPVC that is lower than the optimal cost of the corresponding kHNSDP, it is sufficient to find a feasible solution of NDPVC for which this is true, whereas to prove the converse, one must be sure that the solution obtained for the NDPVC (with optimal cost equal to that of the related kHNSDP) is optimal. Therefore, since by using the branch-and-cut algorithms we are now able to solve more instances to optimality, it is natural that we are able to conclude the latter for many more instances than in [11].

5 Conclusions

The Network Design Problem with Vulnerability Constraints (NDPVC), proposed by Gouveia and Leitner [11] simultaneously addresses two fundamental criteria in the design of telecommunication networks: survivability and quality of service. Survivability is safeguarded by ensuring that any two points in the network can still communicate after the failure of a reasonable amount of technical equipment, e.g., links. Quality of service, on the other hand, is closely related to the number of links each communication packet must traverse, on the path from its origin to its destination. As such, in the NDPVC, one must design networks with a guaranteed maximum hop distance between each commodity pair, before and after the failure of a given number of links. The motivation for this problem is that is far less conservative than the well-known kHSNDP, which addresses the same two criteria, but often leads to more costly solutions and sometimes even fails to provide feasible ones. In fact, Gouveia and Leitner [11] show that for any feasible instance of the NDPVC, either the cost of its optimal solution does not exceed that of the related kHSNDP, or the related kHSNDP is infeasible. The authors also propose several ILP formulations for the NDPVC, based on three graph-theoretical characterizations of feasible backup systems. However, when used in combination with the standard solving methods of general-purpose ILP solvers, e.g., CPLEX, these formulations fail to solve to optimality the majority of the benchmarking instances. Consequently, for almost 60% of the instances, no optimal solution can be achieved.

In this paper, we propose branch-and-cut methods, based on decompositions of formulations inspired by the characterizations in [11]. The first method is a cutting plane algorithm for a new layered graph formulation. The other two methods are motivated by new theoretical results that show that the two best-performing formulations in [11] are still valid when relaxing the integrality of the flow variables, thus allowing them to be combined with Benders decomposition methods. In order to provide these methods with good initial feasible solutions, a primal heuristic is also proposed and tested.

The results of our computational experiments show that, whereas the cutting plane method based on the layered graph formulation does not perform particularly well, the Benders decomposition methods are significantly more efficient in solving the NDPVC than the methods used in [11]. Notably, the best performing Benders decomposition method solved, in the time limit of two hours, 71% more instances than the best-performing ILP formulation from [11].

Consequently, we are able to increase, by 39% of the total number of instances tested, the comparison proposed in [11], between the NDPVC and the kHSNDP. These new results indicate that for (at least) about one in four instances, the optimal solution of the NDPVC is cheaper than that of kHSNDP, and only rarely is the NDPVC feasible and the latter infeasible.

Our results also indicate that the solution quality obtained from the proposed matheuristic is good enough when used to initialize an exact method by means of a valid primal bound. Results also show, however, that the optimality gap of these solution is not negligible, thus indicating that the design of more effective, stand-alone, metaheuristics could be a topic worth investigating in future research. In addition, it might be relevant to identify and exploit (in integer programming formulations) solution characterizations that address the case of more than one link failure. To this end, we observe that the first and straightforward characterization from Gouveia and Leitner [11] is easy to extend in that manner, while this does not seem to be the case for the other two characterizations.

Acknowledgements

This work has been supported by the Portuguese National Funding from Fundação para a Ciência e a Tecnologia, under project UID/MAT/04561/2013 (L. Gouveia), by the Austrian Science Fund (FWF), under grant I892-N23 (M. Joyce-Moniz), and by the Vienna Science and Technology Fund (WWTF) through project ICT15-014 (M. Leitner). These supports are greatly acknowledged.

References

[1] A. Balakrishnan and K. Altinkemer. Using a hop-constrained model to generate alternative communication network design. *ORSA Journal on Computing*, 4(2):192–205, 1992.

[2] J.E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19(4):379–394, 1989.

[3] J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[4] Q. Botton, B. Fortz, L. Gouveia, and M. Poss. Benders decomposition for the hop-constrained survivable network design problem. *INFORMS Journal on Computing*, 25:13–26, 2013.

[5] H. Calik, M. Leitner, and M. Luipersbeck. A benders decomposition based framework for solving cable trench problems. *Computers & Operations Research*, 81:128–140, 2017.

[6] Boris V. Cherkassky and Andrew V. Goldberg. On implementing push-relabel method for the maximum flow problem. *Algorithmica*, 19:390–410, 1994.

[7] M. Chimani, M. Kandyba, I. Ljubić, and P. Mutzel. Orientation-based models for {0,1,2}-survivable network design: Theory and practice. *Mathematical Programming*, 124(1-2):413–440, 2010.

[8] G. Codato and M. Fischetti. Combinatorial benders cuts for mixed-integer linear programming. *Operations Research*, 54:756–766, 2006.

[9] G. Exoo. On a measure of communication network vulnerability. *Networks*, 12:405–409, 1982.

[10] B. Fortz and M. Poss. An improved benders decomposition applied to a multi-layer network design problem. *Operations research letters*, 37(5):359–364, 2009.

[11] L. Gouveia and M. Leitner. Design of survivable networks with vulnerability constraints. *European Journal of Operational Research*, 258(1):89–103, 2017.

[12] L. Gouveia, P. Patricio, and A. de Sousa. Compact models for hop-constrained node survivable network design: An application to MPLS. In *Telecommunications Planning: Innovations in Pricing, Network Design and Management*, volume 33 of *Operations Research/Computer Science Interfaces Series*, pages 167–180. Springer, 2006.

[13] L. Gouveia, L. Simonetti, and E. Uchoa. Modeling hop-constrained and diameter-constrained minimum spanning tree problems as Steiner tree problems over layered graphs. *Mathematical Programming*, 128:123–148, 2011.

[14] L. Gouveia, M. Leitner, and I. Ljubić. Hop constrained steiner trees with multiple root nodes. *European Journal of Operational Research*, 236(1):100–112, 2014.

[15] L. Gouveia, M. Leitner, and I. Ljubić. The two-level diameter constrained spanning tree problem. *Mathematical Programming*, 150(1):49–78, 2015.

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

[17] J. Klincewicz. Optimization issues in quality of service. In M.G.C. Resende and P.M. Pardalos, editors, *Handbook of Optimization in Telecommunications*, pages 435–458. Springer, 2006.

[18] L LeBlanc and R Reddoch. Reliable link topology/capacity design and routing in backbone telecommunication networks. In *First ORSA telecommunications SIG conference*, 1990.

[19] M. Leitner. Layered graph models and exact algorithms for the generalized hop-constrained minimum spanning tree problem. *Computers & Operations Research*, 65:1–18, 2016.

[20] M. Leitner. Integer programming models and branch-and-cut approaches to generalized {0,1,2}-survivable network design problems. *Computational Optimization and Applications*, 65:73–92, 2016.

[21] C.-L. Li, T. McCormick, and D. Simchi-Levi. The complexity of finding two disjoint paths with min-max objective function. *Discrete Applied Mathematics*, 26(1):105–115, 1990.

[22] A. R. Mahjoub, L. Simonetti, and E. Uchoa. Hop-level flow formulation for the survivable network design with hop constraints problem. *Networks*, 61:171–179, 2013.

[23] K. Menger. Zur allgemeinen Kurventheorie. *Fundamenta Mathematicae*, 10:96–115, 1927.

[24] R. Rahmaniani, T.G. Crainic, M. Gendreau, and W. Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 2016.

[25] M. Ruthmair and G. Raidl. A layered graph model and an adaptive layers framework to solve delay-constrained minimum tree problems. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 376–388. Springer, 2011.

[26] W. Sheikh and A. Ghafoor. Jitter-minimized reliability-maximized management of networks. *International Journal of Network Management*, 21:185–222, 2011.

A Further results of computational experiments

Table 4: Average running times ($t^{\text{g}}_{\text{avg}}$ and $t^{\text{g+ls}}_{\text{avg}}$) in seconds and gaps to the best known solution ($gap^{\text{g}}_{\text{avg}}$ and $gap^{\text{g+ls}}_{\text{avg}}$) in percent for the greedy heuristic (g) and the greedy heuristic with subsequent local search (g+ls). Results are grouped by $(H, H') = (H_{\min} + \Delta_H, H_{\min} + \Delta_H + \Delta_{H'})$.

(a) Instance sets C and D.

Set	$(\Delta_H, \Delta_{H'})$	$t^{\text{g}}_{\text{avg}}$	$t^{\text{g+ls}}_{\text{avg}}$	$gap^{\text{g}}_{\text{avg}}$	$gap^{\text{g+ls}}_{\text{avg}}$
C	(0,0)	0	70	19	2
	(0,1)	0	220	19	2
	(0,2)	0	390	21	4
	(1,0)	0	117	20	2
	(1,1)	0	279	20	3
	(1,2)	0	451	21	4
	(2,0)	0	181	20	2
	(2,1)	0	355	22	4
	(2,2)	1	495	22	5
D	(0,0)	1	211	21	8
	(0,1)	1	381	22	9
	(0,2)	1	461	19	8
	(1,0)	1	211	17	6
	(1,1)	1	348	16	6
	(1,2)	1	473	15	7
	(2,0)	1	250	17	7
	(2,1)	1	400	17	7
	(2,2)	1	522	16	7
Avg		0	301	20	4

(b) Instance sets E and RE.

Set	$(\Delta_H, \Delta_{H'})$	$t^{\text{g}}_{\text{avg}}$	$t^{\text{g+ls}}_{\text{avg}}$	$gap^{\text{g}}_{\text{avg}}$	$gap^{\text{g+ls}}_{\text{avg}}$
EU	(0,0)	0	53	8	5
	(0,1)	0	313	16	9
	(0,2)	0	569	12	7
	(1,0)	0	131	16	8
	(1,1)	0	429	13	7
	(1,2)	1	570	10	7
	(2,0)	0	204	19	9
	(2,1)	0	506	13	9
	(2,2)	1	596	10	7
RE	(0,0)	0	12	11	5
	(0,1)	0	108	22	9
	(0,2)	0	328	18	9
	(1,0)	0	17	22	12
	(1,1)	0	212	20	8
	(1,2)	0	475	17	9
	(2,0)	0	38	26	13
	(2,1)	0	293	22	11
	(2,2)	0	547	18	11
Avg		0	300	16	9

Table 5: Numbers of instances solved to optimality ($\#_{\text{solved}}$), average running times (t_{avg}) in seconds, and average optimality gaps (gap_{opt}) in percent of branch-and-cut algorithms BCx, $x \in \{1, 2, 3\}$ and their variants using the primal heuristic (BCx^{H}) for instances of test sets C and D. For BC1, we also list results of variants with additional initialization constraints (BC1_{I}, BC1_{I}^{H}).

		$\#_{\text{solved}}$							t_{avg}							gap_{opt}						
Set	#	BC1	BC1_{I}	BC1_{I}^{H}	BC2	BC2^{H}	BC3	BC3^{H}	BC1	BC1_{I}	BC1_{I}^{H}	BC2	BC2^{H}	BC3	BC3^{H}	BC1	BC1_{I}	BC1_{I}^{H}	BC2	BC2^{H}	BC3	BC3^{H}
C-1	180	60	73	**74**	**136**	129	**175**	174	5128	4453	**4438**	**2171**	2491	**522**	743	38	35	**20**	**12**	14	**1**	**1**
C-2	180	16	**20**	19	**65**	52	**121**	114	6624	**6476**	6494	**4962**	5363	**2854**	3145	64	66	**39**	**35**	**35**	11	**10**
C-3	180	68	**77**	76	**131**	116	**165**	161	4732	**4281**	4309	**2362**	2824	**916**	1148	34	30	**18**	**15**	**15**	4	**2**
C-4	180	17	**28**	24	**51**	39	**120**	112	6563	**6253**	6357	**5436**	5748	**2746**	2982	57	53	**32**	38	**35**	12	**10**
C-5	180	1	**3**	**3**	**8**	**8**	**39**	38	7160	7097	**7088**	**6889**	6907	**5756**	5829	81	80	**48**	70	**53**	48	**25**
C-6	180	62	**77**	73	**125**	117	**164**	159	4931	**4268**	4399	**2512**	2884	**845**	1047	32	26	**17**	12	**10**	3	**2**
C-7	180	15	**22**	20	**45**	34	**108**	99	6647	**6461**	6529	**5481**	5972	**3240**	3516	59	54	**34**	40	**32**	18	**9**
C-8	180	6	**10**	9	**15**	12	**60**	57	7035	**6815**	6850	**6677**	6757	**4951**	5108	75	73	**42**	67	**52**	43	**24**
C-9	180	1	**4**	3	**7**	**7**	21	**22**	7160	**7069**	7095	**6929**	6961	6465	**6370**	87	85	**50**	80	**59**	67	**54**
D-1	90	66	78	**79**	**90**	**90**	**90**	**90**	2167	1326	**1266**	**9**	64	**6**	64	10	4	**3**	**0**	**0**	**0**	**0**
D-2	90	32	45	**49**	84	**85**	**90**	**90**	4899	4000	**3799**	**683**	883	**144**	363	40	34	**17**	3	**2**	**0**	**0**
D-3	90	5	7	**9**	**45**	42	**50**	49	6928	6709	**6617**	**3915**	4222	**4057**	4211	76	83	**45**	27	**25**	35	**16**
D-4	90	**0**	**0**	**0**	**28**	27	**22**	**22**	**7200**	**7200**	**7200**	**5360**	5401	**6087**	6093	97	96	**60**	38	**32**	70	**40**
D-5	90	**1**	**1**	**1**	**3**	**3**	**1**	**1**	**7120**	**7120**	**7120**	**6989**	7023	**7120**	**7120**	98	98	**65**	76	**62**	99	**90**
Ttl/Avg	2070	350	**445**	439	**833**	761	**1226**	1188	5941	5590	**5576**	**4108**	4310	**3313**	3446	61	59	**36**	35	**29**	32	**22**

Table 6: Numbers of instances solved to optimality ($\#_{\text{solved}}$), average running times (t_{avg}) in seconds, and average optimality gaps (gap_{opt}) in percent of branch-and-cut algorithms BCx, $x \in \{1, 2, 3\}$ and their variants using the primal heuristic (BCx^{H}) for instances of test sets C and D, grouped by $(H, H') = (H_{\min} + \Delta_H, H_{\min} + \Delta_H + \Delta_{H'})$. For BC1, we also list results of variants with additional initialization constraints (BC1_{I}, BC1_{I}^{H}).

			$\#_{\text{solved}}$							t_{avg}							gap_{opt}						
Set	$(\Delta_H, \Delta_{H'})$	#	BC1	BC1_{I}	BC1_{I}^{H}	BC2	BC2^{H}	BC3	BC3^{H}	BC1	BC1_{I}	BC1_{I}^{H}	BC2	BC2^{H}	BC3	BC3^{H}	BC1	BC1_{I}	BC1_{I}^{H}	BC2	BC2^{H}	BC3	BC3^{H}
C	(0,0)	180	85	**96**	94	**131**	120	**151**	150	**4038**	3489	3582	**2127**	2606	**1272**	1324	23	17	**10**	16	**13**	7	**4**
	(0,1)	180	45	**57**	55	**88**	77	**120**	115	**5630**	5085	5135	**3883**	4305	**2687**	2927	47	39	**24**	28	**26**	17	**13**
	(0,2)	180	15	**23**	21	**65**	51	**89**	74	**6565**	6383	6434	**4839**	5348	**4250**	4509	66	63	**39**	40	**37**	30	**20**
	(1,0)	180	50	**58**	55	**92**	85	**141**	140	**5390**	4975	5080	**3697**	3923	**1726**	1794	39	36	**21**	27	**22**	10	**5**
	(1,1)	180	15	**23**	21	**62**	51	**107**	101	**6661**	6379	6426	**5004**	5341	**3283**	3519	63	60	**36**	41	**35**	23	**15**
	(1,2)	180	4	**8**	**8**	**41**	33	**68**	66	6773	6914	**6928**	**5801**	6159	4951	**4815**	75	79	**48**	52	**44**	42	**24**
	(2,0)	180	25	**34**	32	**65**	62	**136**	134	**6332**	6044	6093	**4864**	4967	**2026**	2066	52	49	**26**	40	**31**	14	**9**
	(2,1)	180	7	12	**13**	**27**	24	**93**	90	**6952**	6783	6732	**6387**	6405	**3750**	3933	72	73	**41**	58	**45**	27	**17**
	(2,2)	180	0	**3**	2	**12**	11	**68**	66	6800	7120	**7149**	**6816**	6853	5191	**5001**	79	87	**52**	67	**53**	48	**28**
D	(0,0)	50	25	25	**26**	**41**	**41**	**40**	38	3032	**3610**	3587	**1434**	1614	2734	**2115**	35	44	**23**	12	**10**	29	**15**
	(0,1)	50	13	**18**	**18**	**27**	24	**29**	**29**	4209	4685	**4731**	**3556**	3904	4665	**3500**	48	57	**38**	31	**25**	54	**26**
	(0,2)	50	5	14	**15**	**22**	**22**	22	**23**	4716	**5741**	5558	**4193**	4382	6051	**4293**	50	68	**48**	38	**31**	77	**34**
	(1,0)	50	19	20	**21**	**37**	**37**	**35**	34	4117	**4404**	4342	**2026**	2059	3133	**2636**	40	51	**24**	16	**14**	34	**18**
	(1,1)	50	10	**15**	**15**	**24**	**24**	**26**	**26**	4471	**5267**	5180	**3854**	4007	5166	**3833**	46	65	**40**	34	**27**	63	**28**
	(1,2)	50	4	5	**6**	**20**	**20**	**22**	**22**	4784	**6507**	6475	**4450**	4565	6034	**4289**	53	77	**52**	40	**35**	80	**41**
	(2,0)	50	17	**18**	**18**	**37**	**37**	**34**	**34**	4281	**4740**	4704	2336	**2270**	3528	**3051**	44	57	**25**	15	**13**	39	**20**
	(2,1)	50	8	12	**14**	**24**	23	23	**24**	4665	**5824**	5548	**3965**	4091	5572	**3995**	47	68	**39**	33	**28**	71	**35**
	(2,2)	50	3	4	**5**	18	**19**	**22**	**22**	4451	6661	**6677**	**4707**	4777	6704	**4421**	**45**	80	53	41	**34**	90	**44**
Ttl/Avg		2070	350	**445**	439	**833**	761	**1226**	1188	5941	5590	**5576**	**4108**	4310	**3313**	3446	61	59	**36**	35	**29**	32	**22**

Table 7: Numbers of instances solved to optimality ($\#_{\text{solved}}$), average running times (t_{avg}) in seconds, and average optimality gaps (gap_{opt}) in percent of branch-and-cut algorithms BCx, $x \in \{1, 2, 3\}$ and their variants using the primal heuristic (BCx^{H}) for instances of test sets E and RE. For BC1, we also list results of variants with additional initialization constraints (BC1_{I}, $\text{BC1}_{\text{I}}^{\text{H}}$).

		$\#_{\text{solved}}$							t_{avg}							gap_{opt}						
Set	#	BC1	BC1_{I}	$\text{BC1}_{\text{I}}^{\text{H}}$	BC2	BC2^{H}	BC3	BC3^{H}	BC1	BC1_{I}	$\text{BC1}_{\text{I}}^{\text{H}}$	BC2	BC2^{H}	BC3	BC3^{H}	BC1	BC1_{I}	$\text{BC1}_{\text{I}}^{\text{H}}$	BC2	BC2^{H}	BC3	BC3^{H}
EU-1	45	17	**22**	**22**	**45**	**45**	**45**	**45**	4538	**4073**	4115	**49**	384	**110**	426	28	25	**18**	**0**	**0**	**0**	**0**
EU-2	45	**21**	**21**	**21**	**42**	**42**	40	**42**	3859	3842	**3840**	**731**	989	**1599**	1615	44	41	**24**	**1**	**1**	4	**1**
EU-3	45	12	**14**	**14**	**34**	32	44	**45**	5330	5059	**5049**	**2408**	2794	**470**	622	48	49	**29**	**9**	**9**	**0**	**0**
EU-4	45	**4**	**4**	**4**	**13**	**13**	21	**23**	**6560**	**6560**	**6560**	5589	**5487**	4383	**4301**	79	85	**51**	37	**32**	24	**14**
EU-5	45	14	17	**18**	26	**28**	41	**42**	5068	4676	**4653**	3167	**3069**	**1315**	1438	48	48	**29**	18	**16**	2	**1**
EU-6	45	**4**	**4**	**4**	**11**	9	10	**11**	6643	**6563**	6565	**5838**	5872	5759	**5740**	85	86	**57**	39	**38**	67	**42**
EU-7	45	4	5	**6**	14	**15**	**32**	31	6629	6410	**6380**	**5207**	5288	**2894**	3009	71	75	**43**	**36**	**36**	21	**10**
EU-8	45	**1**	**1**	**1**	**6**	**6**	**15**	14	**7040**	**7040**	**7040**	**6332**	6364	**5211**	5243	87	93	**58**	57	**42**	57	**34**
EU-9	45	8	8	**9**	**20**	**20**	36	**37**	5929	5924	**5900**	4205	**4195**	2227	**2219**	64	66	**38**	30	**29**	10	**4**
EU-10	45	**4**	**4**	**4**	**6**	**6**	**9**	**9**	6595	6561	**6560**	**6289**	6291	6028	**5987**	88	89	**60**	53	**42**	76	**57**
EU-11	45	5	**7**	**7**	12	**13**	**23**	**23**	6403	6097	**6090**	5397	**5321**	**4074**	4168	73	77	**47**	47	**35**	39	**19**
EU-12	45	**3**	**3**	**3**	**5**	**5**	**9**	7	6739	6725	**6720**	6517	**6430**	**6050**	6171	87	91	**65**	71	**52**	75	**63**
RE-1	45	19	**26**	24	**45**	**45**	**45**	**45**	4521	**3275**	3432	**129**	396	**64**	247	33	21	**14**	**0**	**0**	**0**	**0**
RE-2	45	6	**12**	11	**26**	**26**	38	**40**	6245	**5593**	5621	**3618**	3734	1969	**1819**	65	56	**31**	11	**7**	**2**	**2**
RE-3	45	20	24	**27**	**41**	40	**45**	**45**	4274	3425	**3194**	1151	**1124**	**92**	238	36	24	**12**	**4**	**4**	**0**	**0**
RE-4	45	4	8	**9**	**15**	**15**	27	**29**	6716	6090	**5966**	4908	**4858**	3404	**3330**	67	62	**36**	38	**35**	12	**8**
RE-5	45	16	**23**	**23**	35	**36**	**45**	**45**	5097	**3748**	4050	2239	**1995**	**173**	440	42	34	**18**	13	**8**	**0**	**0**
RE-6	45	4	**5**	**5**	**10**	**10**	22	**23**	6560	**6425**	6499	**5659**	5670	4172	**4063**	76	77	**43**	46	**39**	22	**14**
RE-7	45	13	**19**	**19**	**27**	26	42	**43**	5138	4345	**4256**	3379	**3280**	**935**	1034	51	43	**22**	22	**18**	3	**1**
RE-8	45	3	**7**	**7**	13	**15**	**20**	**20**	6720	**6100**	**6100**	5245	**4994**	**4557**	4574	75	75	**43**	51	**34**	35	**21**
RE-9	45	14	18	**21**	25	**27**	40	**42**	5115	4379	**4127**	3292	**3175**	1314	**1243**	50	44	**20**	28	**19**	8	**2**
RE-10	45	**3**	**3**	**3**	**6**	**6**	**14**	13	**6720**	**6720**	**6720**	6294	**6276**	5296	**5287**	80	86	**48**	53	**38**	46	**22**
RE-11	45	17	24	**25**	32	**34**	42	**44**	4488	3494	**3254**	2366	**2131**	1027	**956**	43	26	**16**	16	**11**	3	**1**
RE-12	45	0	**4**	**4**	7	**10**	12	**13**	7200	6772	**6711**	6305	**6032**	5508	**5374**	85	87	**49**	56	**37**	52	**38**
Ttl/Avg	1080	216	283	**291**	516	**524**	717	**731**	5839	5412	**5392**	4013	**4006**	**2860**	2898	63	61	**36**	31	**24**	23	**15**

Table 8: Numbers of instances solved to optimality ($\#_{\text{solved}}$), average running times (t_{avg}) in seconds, and average optimality gaps (gap_{opt}) in percent of branch-and-cut algorithms BCx, $x \in \{1, 2, 3\}$ and their variants using the primal heuristic (BCx^{H}) for instances of test sets E and RE, grouped by $(H, H') = (H_{\min} + \Delta_H, H_{\min} + \Delta_H + \Delta_{H'})$. For BC1, we also list results of variants with additional initialization constraints (BC1_{I}, BC1_{I}^{H}).

Set	$(\Delta_H, \Delta_{H'})$	#	$\#_{\text{solved}}$ BC1	BC1_{I}	BC1_{I}^{H}	BC2	BC2^{H}	BC3	BC3^{H}	t_{avg} BC1	BC1_{I}	BC1_{I}^{H}	BC2	BC2^{H}	BC3	BC3^{H}	gap_{opt} BC1	BC1_{I}	BC1_{I}^{H}	BC2	BC2^{H}	BC3	BC3^{H}
EU	(0,0)	60	46	48	**49**	**55**	**55**	**58**	**58**	1458	**1466**	1441	**725**	**725**	709	**353**	14	15	**7**	**3**	4	7	**1**
	(0,1)	60	12	**16**	15	**27**	26	**42**	41	4574	5500	**5505**	**4423**	4596	4094	**2964**	49	58	**37**	29	**26**	38	**14**
	(0,2)	60	3	**5**	**5**	**16**	15	**25**	24	5048	6614	**6669**	**5419**	5611	6537	**5028**	58	81	**57**	44	**40**	70	**31**
	(1,0)	60	18	19	**20**	43	**45**	**50**	**50**	4553	**4985**	4935	2225	**2119**	2060	**1484**	41	50	**25**	14	**10**	21	**5**
	(1,1)	60	4	**5**	**5**	**18**	**18**	34	**35**	5529	6613	**6620**	**5200**	5321	4873	**3717**	59	79	**50**	39	**33**	48	**19**
	(1,2)	60	**2**	**2**	**2**	**13**	12	20	**23**	5282	**6960**	**6960**	**5791**	5958	6980	**5273**	**62**	90	63	**49**	42	75	**35**
	(2,0)	60	10	12	**14**	**39**	38	**49**	**49**	5336	**5959**	5913	3087	**3017**	2555	**1886**	46	65	**30**	18	**13**	27	**10**
	(2,1)	60	**2**	**2**	**2**	13	**15**	30	**31**	5881	**6960**	**6960**	**5773**	5779	5258	**4274**	64	87	**54**	46	**37**	55	**25**
	(2,2)	60	0	**1**	**1**	**10**	**10**	17	**18**	5760	7089	**7103**	**6156**	6237	7104	**5724**	67	94	**66**	56	**45**	80	**44**
RE	(0,0)	60	46	**57**	**57**	**56**	**56**	**55**	**55**	**1804**	569	628	**593**	594	748	**731**	13	**1**	**1**	**2**	**2**	**2**	3
	(0,1)	60	14	**31**	29	**33**	**33**	**41**	39	**5753**	3759	3862	**3366**	3476	**2676**	2706	54	38	**20**	27	**23**	15	**11**
	(0,2)	60	1	5	**6**	**15**	**15**	29	**34**	6480	**6696**	6632	**5590**	5698	4698	**3783**	73	66	**41**	46	**36**	37	**15**
	(1,0)	60	28	**34**	**34**	47	**50**	58	**59**	**4255**	3290	3285	1794	**1553**	409	**326**	32	26	**13**	7	**6**	**0**	**0**
	(1,1)	60	4	8	**11**	**30**	**30**	39	**41**	**6887**	6320	6131	4108	**3911**	**2826**	2852	70	61	**34**	33	**23**	18	**9**
	(1,2)	60	1	**2**	**2**	**14**	**14**	29	**33**	6728	6991	**7025**	6020	**5920**	4420	**4150**	78	87	**51**	51	**37**	37	**19**
	(2,0)	60	22	30	**31**	52	**53**	**60**	**60**	**4606**	3835	3835	1277	**1169**	**114**	127	36	35	**15**	**2**	3	**0**	**0**
	(2,1)	60	2	5	**7**	26	**29**	47	**48**	**6905**	6733	6471	4440	**4178**	2367	**2331**	73	76	**36**	31	**20**	10	**5**
	(2,2)	60	**1**	**1**	**1**	9	**10**	**34**	33	6617	**7081**	**7081**	**6250**	**6250**	4684	**4448**	76	88	**54**	54	**39**	39	**19**
Ttl/Avg		1080	216	283	**291**	516	**524**	717	**731**	5839	5412	**5392**	4013	**4006**	**2860**	2898	63	61	**36**	31	**24**	23	**15**

Table 9: Numbers of instances solved to optimality ($\#_{\text{solved}}$), average running times (t_{avg}) in seconds, and average optimality gaps (gap_{opt}) in percent of best performing methods from Gouveia and Leitner [11] (H_2^{AS}, H_3) and best performing variants of branch-and-cut algorithms proposed in this work ($BC1_I^H$,BC2H, $BC3^H$) for instances of test sets C and D.

		$\#_{\text{solved}}$					t_{avg}					gap_{opt}				
Set	#	H_2^{AS}	H_3	$BC1_I^H$	$BC2^H$	$BC3^H$	H_2^{AS}	H_3	$BC1_I^H$	$BC2^H$	$BC3^H$	H_2^{AS}	H_3	$BC1_I^H$	$BC2^H$	$BC3^H$
C-1	180	84	123	74	129	**174**	4266	2647	4438	2491	**743**	9	5	20	14	**1**
C-2	180	21	45	19	52	**114**	6450	5745	6494	5363	**3145**	21	24	39	35	**10**
C-3	180	101	131	76	116	**161**	3413	2336	4309	2824	**1148**	8	4	18	15	**2**
C-4	180	27	63	24	39	**112**	6257	4902	6357	5748	**2982**	18	11	32	35	**10**
C-5	180	1	7	3	8	**38**	7160	6975	7088	6907	**5829**	31	46	48	53	**25**
C-6	180	109	142	73	117	**159**	3099	1798	4399	2884	**1047**	7	3	17	10	**2**
C-7	180	33	73	20	34	**99**	5974	4685	6529	5972	**3516**	18	11	34	32	**9**
C-8	180	9	27	9	12	**57**	6888	6244	6850	6757	**5108**	**24**	30	42	52	**24**
C-9	180	2	8	3	7	**22**	7122	6950	7095	6961	**6370**	**35**	55	50	59	54
D-1	90	**90**	**90**	79	**90**	**90**	394	364	1266	**64**	**64**	**0**	**0**	3	**0**	**0**
D-2	90	62	59	49	85	**90**	3046	3096	3799	883	**363**	6	10	17	2	**0**
D-3	90	9	9	9	42	**49**	6610	6599	6617	4222	**4211**	27	63	45	25	**16**
D-4	90	0	0	0	**27**	22	7200	7200	7200	**5401**	6093	69	97	60	**32**	40
D-5	90	1	1	1	**3**	1	7120	7120	7120	**7023**	7120	73	99	65	**62**	90
Ttl/Avg	2070	549	778	439	761	**1188**	5250	4787	5576	4310	**3446**	27	37	36	29	**22**

Table 10: Numbers of instances solved to optimality ($\#_{\text{solved}}$), average running times (t_{avg}) in seconds, and average optimality gaps (gap_{opt}) in percent of best performing methods from Gouveia and Leitner [11] (H_2^{AS}, H_3) and best performing variants of branch-and-cut algorithms proposed in this work ($BC1_I^H$,BC2H, $BC3^H$) for instances of test sets C and D, grouped by $(H, H') = (H_{\min} + \Delta_H, H_{\min} + \Delta_H + \Delta_{H'})$

			$\#_{\text{solved}}$					t_{avg}					gap_{opt}				
Set	$(\Delta_H, \Delta_{H'})$	#	H_2^{AS}	H_3	$BC1_I^H$	$BC2^H$	$BC3^H$	H_2^{AS}	H_3	$BC1_I^H$	$BC2^H$	$BC3^H$	H_2^{AS}	H_3	$BC1_I^H$	$BC2^H$	$BC3^H$
C	(0,0)	180	92	125	94	120	**150**	3712	2605	3582	2606	**1324**	8	**3**	10	13	4
	(0,1)	180	48	69	55	77	**115**	5392	4674	5135	4305	**2927**	17	14	24	26	**13**
	(0,2)	180	25	39	21	51	**74**	6348	5874	6434	5348	**4509**	25	26	39	37	**20**
	(1,0)	180	71	108	55	85	**140**	4563	3261	5080	3923	**1794**	11	6	21	22	**5**
	(1,1)	180	35	63	21	51	**101**	5944	4922	6426	5341	**3519**	19	21	36	35	**15**
	(1,2)	180	14	34	8	33	**66**	6672	6127	6928	6159	**4815**	27	34	48	44	**24**
	(2,0)	180	58	99	32	62	**134**	5009	3639	6093	4967	**2066**	13	13	26	31	**9**
	(2,1)	180	29	58	13	24	**90**	6272	5266	6732	6405	**3933**	22	26	41	45	**17**
	(2,2)	180	15	24	2	11	**66**	6716	6465	7149	6853	**5001**	29	44	52	53	**28**
D	(0,0)	50	24	23	26	**41**	38	3781	3955	3587	**1614**	2115	19	37	23	**10**	15
	(0,1)	50	17	19	18	24	**29**	5085	4872	4731	3904	**3500**	34	51	38	**25**	26
	(0,2)	50	14	13	15	22	**23**	5662	5769	5558	4382	**4293**	44	59	48	**31**	34
	(1,0)	50	23	24	21	**37**	34	3971	3894	4342	**2059**	2636	24	42	24	**14**	18
	(1,1)	50	17	18	15	24	**26**	4928	4865	5180	4007	**3833**	36	55	40	**27**	28
	(1,2)	50	14	11	6	20	**22**	5659	5945	6475	4565	**4289**	47	64	52	**35**	41
	(2,0)	50	22	22	18	**37**	34	4245	4183	4704	**2270**	3051	28	49	25	**13**	20
	(2,1)	50	18	17	14	23	**24**	4845	5167	5548	4091	**3995**	37	59	39	**28**	35
	(2,2)	50	13	12	5	19	**22**	5690	5815	6677	4777	**4421**	47	68	53	**34**	44
Ttl/Avg		2070	549	778	439	761	**1188**	5250	4787	5576	4310	**3446**	27	37	36	29	**22**

Table 11: Numbers of instances solved to optimality ($\#_{\text{solved}}$), average running times (t_{avg}) in seconds, and average optimality gaps (gap_{opt}) in percent of best performing methods from Gouveia and Leitner [11] (H_2^{AS}, H_3) and best performing variants of branch-and-cut algorithms proposed in this work ($BC1_I^H$,BC2H, $BC3^H$) for instances of test sets E and RE.

		$\#_{\text{solved}}$					t_{avg}					gap_{opt}				
Set	#	H_2^{AS}	H_3	$BC1_I^H$	$BC2^H$	$BC3^H$	H_2^{AS}	H_3	$BC1_I^H$	$BC2^H$	$BC3^H$	H_2^{AS}	H_3	$BC1_I^H$	$BC2^H$	$BC3^H$
EU-1	45	30	23	22	**45**	**45**	3383	3921	4115	**384**	426	4	14	18	**0**	**0**
EU-2	45	22	21	21	**42**	**42**	3695	3917	3840	**989**	1615	19	53	24	**1**	**1**
EU-3	45	16	20	14	32	**45**	4820	4359	5049	2794	**622**	16	19	29	9	**0**
EU-4	45	4	4	4	13	**23**	6560	6560	6560	5487	**4301**	53	80	51	32	**14**
EU-5	45	18	20	18	28	**42**	4678	4242	4653	3069	**1438**	17	28	29	16	**1**
EU-6	45	3	3	4	9	**11**	6720	6720	6565	5872	**5740**	70	90	57	**38**	42
EU-7	45	6	11	6	15	**31**	6379	5608	6380	5288	**3009**	34	52	43	36	**10**
EU-8	45	1	1	1	6	**14**	7040	7040	7040	6364	**5243**	75	92	58	42	**34**
EU-9	45	9	12	9	20	**37**	5915	5523	5900	4195	**2219**	26	39	38	29	**4**
EU-10	45	4	4	4	6	**9**	6560	6560	6560	6291	**5987**	80	91	60	**42**	57
EU-11	45	5	8	7	13	**23**	6403	5928	6090	5321	**4168**	41	61	47	35	**19**
EU-12	45	3	3	3	5	**7**	6720	6720	6720	6430	**6171**	87	90	65	**52**	63
RE-1	45	31	32	24	**45**	**45**	2645	2553	3432	396	**247**	4	5	14	**0**	**0**
RE-2	45	11	10	11	26	**40**	5643	5772	5621	3734	**1819**	25	52	31	7	**2**
RE-3	45	30	34	27	40	**45**	3404	2518	3194	1124	**238**	9	5	12	4	**0**
RE-4	45	9	12	9	15	**29**	6210	5982	5966	4858	**3330**	39	53	36	35	**8**
RE-5	45	22	27	23	36	**45**	3973	3269	4050	1995	**440**	13	8	18	8	**0**
RE-6	45	6	4	5	10	**23**	6443	6560	6499	5670	**4063**	46	68	43	39	**14**
RE-7	45	22	24	19	26	**43**	4481	3763	4256	3280	**1034**	19	17	22	18	**1**
RE-8	45	3	8	7	15	**20**	6720	6106	6100	4994	**4574**	57	65	43	34	**21**
RE-9	45	19	24	21	27	**42**	4458	3757	4127	3175	**1243**	18	24	20	19	**2**
RE-10	45	3	3	3	6	**13**	6720	6720	6720	6276	**5287**	65	81	48	38	**22**
RE-11	45	22	32	25	34	**44**	3940	2525	3254	2131	**956**	17	10	16	11	**1**
RE-12	45	0	2	4	10	**13**	7200	6970	6711	6032	**5374**	74	77	49	**37**	38
Ttl/Avg	1080	299	342	291	524	**731**	5446	5150	5392	4006	**2898**	38	49	36	24	**15**

Table 12: Numbers of instances solved to optimality ($\#_{\text{solved}}$), average running times (t_{avg}) in seconds, and average optimality gaps (gap_{opt}) in percent of best performing methods from Gouveia and Leitner [11] (H_2^{AS}, H_3) and best performing variants of branch-and-cut algorithms proposed in this work ($\text{BC1}_{\text{I}}^{\text{H}}$,BC2H, BC3^{H}) for instances of test sets E and RE, grouped by $(H, H') = (H_{\min} + \Delta_H, H_{\min} + \Delta_H + \Delta_{H'})$.

			$\#_{\text{solved}}$					t_{avg}					gap_{opt}				
Set	$(\Delta_H, \Delta_{H'})$	#	H_2^{AS}	H_3	$\text{BC1}_{\text{I}}^{\text{H}}$	BC2^{H}	BC3^{H}	H_2^{AS}	H_3	$\text{BC1}_{\text{I}}^{\text{H}}$	BC2^{H}	BC3^{H}	H_2^{AS}	H_3	$\text{BC1}_{\text{I}}^{\text{H}}$	BC2^{H}	BC3^{H}
EU	(0,0)	60	45	47	49	55	**58**	1874	1603	1441	725	**353**	9	10	7	4	**1**
	(0,1)	60	13	14	15	26	**41**	5750	5681	5505	4596	**2964**	42	50	37	26	**14**
	(0,2)	60	5	3	5	15	**24**	6665	6973	6669	5611	**5028**	57	73	57	40	**31**
	(1,0)	60	24	28	20	45	**50**	4631	4090	4935	2119	**1484**	27	39	25	10	**5**
	(1,1)	60	7	8	5	18	**35**	6544	6542	6620	5321	**3717**	47	64	50	33	**19**
	(1,2)	60	3	2	2	12	**23**	6930	7060	6960	5958	**5273**	62	83	63	42	**35**
	(2,0)	60	17	22	14	38	**49**	5388	4929	5913	3017	**1886**	29	49	30	13	**10**
	(2,1)	60	6	6	2	15	**31**	6710	6829	6960	5779	**4274**	52	74	54	37	**25**
	(2,2)	60	1	0	1	10	**18**	7164	7300	7103	6237	**5724**	66	90	66	45	**44**
RE	(0,0)	60	44	53	**57**	56	55	1928	1028	628	**594**	731	4	**1**	**1**	2	3
	(0,1)	60	21	30	29	33	**39**	5373	4113	3862	3476	**2706**	28	31	20	23	**11**
	(0,2)	60	3	7	6	15	**34**	6904	6685	6632	5698	**3783**	54	57	41	36	**15**
	(1,0)	60	37	39	34	50	**59**	3053	3038	3285	1553	**326**	7	9	13	6	**0**
	(1,1)	60	14	22	11	30	**41**	6037	5099	6131	3911	**2852**	33	42	34	23	**9**
	(1,2)	60	3	3	2	14	**33**	6944	6959	7025	5920	**4150**	56	62	51	37	**19**
	(2,0)	60	37	36	31	53	**60**	3168	3271	3835	1169	**127**	9	24	15	3	**0**
	(2,1)	60	16	19	7	29	**48**	6011	5727	6471	4178	**2331**	40	51	36	20	**5**
	(2,2)	60	3	3	1	10	**33**	6960	7000	7081	6250	**4448**	57	72	54	39	**19**
Ttl/Avg		1080	299	342	291	524	**731**	5446	5150	5392	4006	**2898**	38	49	36	24	**15**

Table 13: Overall number of cases from instance sets C and D in which solution to NDPVC is provably better (bt) and provably equal (eq) compared to the kHSNDP, and number of cases where kHSNDP is provably infeasible and NDPVC feasible (feas), grouped by $(H, H') = (H_{\min} + \Delta_H, H_{\min} + \Delta_H + \Delta_{H'})$. Remaining cases could not be decided.

(a) Instance sets C and D.

Set	$(\Delta_H, \Delta_{H'})$	#	bt	feas	eq
C	(0,0)	180	16	0	161
	(0,1)	180	91	0	82
	(0,2)	180	98	0	75
	(1,0)	180	5	0	159
	(1,1)	180	36	0	128
	(1,2)	180	53	0	102
	(2,0)	180	4	0	146
	(2,1)	180	19	0	130
	(2,2)	180	21	0	120
D	(0,0)	90	13	0	34
	(0,1)	90	45	0	2
	(0,2)	90	44	0	4
	(1,0)	90	26	0	15
	(1,1)	90	29	0	8
	(1,2)	90	28	0	9
	(2,0)	90	14	0	22
	(2,1)	90	19	0	17
	(2,2)	90	13	0	20
Total		2070	574	0	1234

(b) Instance sets E and RE.

Set	$(\Delta_H, \Delta_{H'})$	#	bt	feas	eq
EU	(0,0)	45	3	0	53
	(0,1)	45	39	1	12
	(0,2)	45	37	0	9
	(1,0)	45	20	0	24
	(1,1)	45	16	0	18
	(1,2)	45	14	0	16
	(2,0)	45	7	0	20
	(2,1)	45	8	0	17
	(2,2)	45	7	0	15
RE	(0,0)	45	1	0	56
	(0,1)	45	26	0	29
	(0,2)	45	27	0	28
	(1,0)	45	4	0	39
	(1,1)	45	13	0	22
	(1,2)	45	11	0	24
	(2,0)	45	7	0	24
	(2,1)	45	4	0	26
	(2,2)	45	5	0	24
Total		1080	249	1	456