Algorithms and Complexity Group | Institute of Logic and Computation | TU Wien, Vienna, Austria

Technical Report AC-TR-18-001

first version: Feb 2018
last updated: Nov 2018

Exact approaches for the directed network design problem with relays

Markus Leitner, Ivana Ljubić, Martin Riedler, and Mario Ruthmair

www.ac.tuwien.ac.at/tr

Exact approaches for the directed network design problem with relays

Markus Leitner[a], Ivana Ljubić[b], Martin Riedler[c,*], Mario Ruthmair[a]

[a] *University of Vienna, Department of Statistics and Operations Research, Vienna, Austria*
[b] *ESSEC Business School of Paris, Cergy-Pontoise, France*
[c] *Institute of Logic and Computation, TU Wien, Vienna, Austria*

Abstract

We study the directed network design problem with relays (DNDPR) whose aim is to construct a minimum cost network that enables the communication of a given set of origin-destination pairs. Thereby, expensive signal regeneration devices need to be placed to cover communication distances exceeding a predefined threshold. Applications of the DNDPR arise in telecommunications and transportation.

We propose two new integer programming formulations for the DNDPR. The first one is a flow-based formulation with a pseudo-polynomial number of variables and constraints and the second is a cut-based formulation with an exponential number of constraints. Fractional distance values are handled efficiently by augmenting both models with an exponentially-sized set of infeasible path constraints. We develop branch-and-cut algorithms and also consider valid inequalities to strengthen the obtained dual bounds and to speed up convergence. The results of our extensive computational study on diverse sets of benchmark instances show that our algorithms outperform the previous state-of-the-art method based on column generation.

Keywords: Integer Programming, Networks, Layered Graphs, Telecommunications

1. Introduction

The directed network design problem with relays (DNDPR) was introduced by Li et al. [19] for modeling the design of networks when the maximum distance a commodity (i.e., signal) can travel is bounded from above by some threshold. This distance limit can be surpassed by locating special, commodity regenerating equipment (relays) at intermediate network nodes. Applications of this problem arise in the design of transportation and telecommunication networks [19]. In the latter, signals deteriorate after traveling a certain distance and thus there is the need to regenerate them before a predefined maximum distance is exceeded. Thus, comparably expensive regenerating devices (e.g., repeaters) need to be installed, see, e.g., Cabral et al. [2], Chen et al. [3], Yıldız and Karaşan [29], in order to avoid signal loss or falsification of the transmitted information.

*Corresponding author

In the design of optical telecommunication networks, for example, commodities correspond to node pairs that need to communicate with each other, but the quality of the optical signal degrades with the distance, so that after a certain distance the signal has to be amplified, which is done by deploying regenerator devices at some nodes of the network [4]. Edge costs are directly proportional to edge lengths (multiplied by some factor that corresponds to cable costs per unit of distance) whereas relay costs correspond to the installation and purchasing costs of regenerator devices. Such devices are usually very expensive (see, e.g., [24] for further details). In the design of fiber optic networks, Wavelength Division Multiplexing (WDM) is used to divide the bandwidth of a single fiber into different wavelength channels so that there is no interference between transmissions on different wavelengths. A signal from a source node to its destination is sent using the wavelength routing through a *lightpath* which is an end-to-end connection over a dedicated communication channel (circuit) that traverses one or more links and uses one WDM channel per link. The circuit guarantees the full bandwidth of the channel and allows for a data rate of 10 or even 40 giga-bits per second (Gbps), see, e.g., [25] for further details. When deploying regenerators in such a network, the signal is converted from optical to electric and back to optical, each time a regenerator is used in the routing path from a source to its destination (such paths are commonly referred to as *translucent lightpaths)*. When translucent lightpaths are not allowed to contain cycles (which can be due to the signal interference, or due to the fact that each lightpath has to be uniquely defined per source-destination pair), one has to explicitly impose *simple paths* for the wavelength routing.

The DNDPR is defined on a digraph $G = (V, A, c, w, d)$ with relay costs $c\colon V \to \mathbb{Q}_{\geq 0}$, arc costs $w\colon A \to \mathbb{Q}_{\geq 0}$, and arc distances $d\colon A \to \mathbb{Q}_{\geq 0}$. Moreover, a distance bound $\lambda_{\max}$ and a set of commodity pairs $\mathcal{K}$ are given. For each commodity $(u, v) \in \mathcal{K}$, nodes u and v are called its source and target, respectively. The goal of the DNDPR is to place relays on a subset of the nodes $V' \subseteq V$ and to select a subset of arcs $A' \subseteq A$ such that:

1. The subgraph induced by A' contains for each $(u, v) \in \mathcal{K}$ a directed (simple) path from u to v not exceeding the distance limit between u and the first relay, any two consecutive relays, and the last relay and v, and
2. the cost induced by installing relays and arcs, defined as

$$\sum_{v \in V'} c_v + \sum_{a \in A'} w_a$$

   is minimized.

A problem instance and its optimal solution are given in Figure 1.

The DNDPR is closely related to the previously introduced and well-studied network design problem with relays (NDPR) (see, e.g., [2]). The major difference between the two problems is in the way how routing paths are defined: whereas only simple paths are allowed in case of the DNDPR, solutions of the NDPR may contain cycles. This latter property renders NDPR solutions infeasible when it comes to the design of translucent optical networks. For an example see Figure 2.

To simplify notation, we will in the following use $S = \{u \mid (u, v) \in \mathcal{K}\}$ to denote the set of commodity sources and $T^u = \{v \mid (u, v) \in \mathcal{K}\}$ to denote all

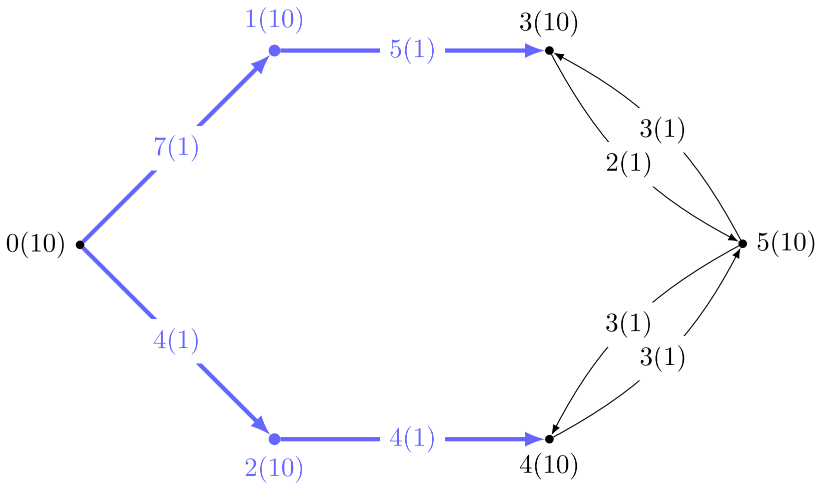


Figure 1: Example instance with two commodities $\mathcal{K} = \{(0,3),(0,4)\}$ and $\lambda_{\max} = 7$. Arc distances are provided next to the arcs, relay and arc costs are given in parentheses. Relays and arcs used in the optimal solution are marked bold and blue.

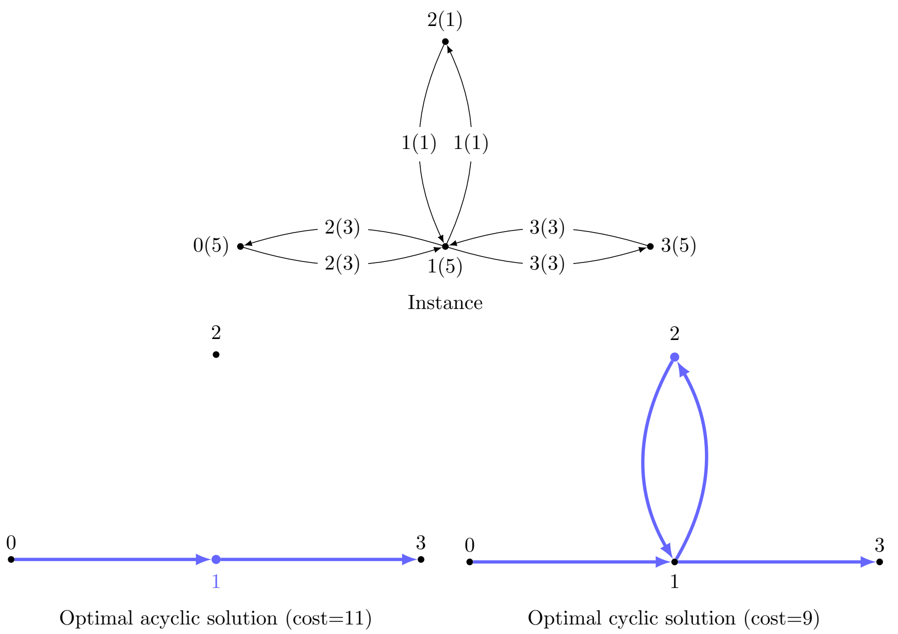


Figure 2: Symmetric instance together with an acyclic and a general solution for $\lambda_{\max} = 4$ and $\mathcal{K} = \{(0,3)\}$. Arc distances are provided next to the arcs, relay and arc costs are given in parentheses. Relays and arcs used in the optimal solution are marked bold and blue. Note that in the acyclic solution we place a relay at node 1, while in the cheaper cyclic solution we place a relay at node 2.

targets that need to be reached from source $u \in S$. Additionally, $\delta^-(W) = \{(j,i) \mid i \in W, (j,i) \in A\}$ and $\delta^+(W) = \{(i,j) \mid i \in W, (i,j) \in A\}$ will be used to denote the sets of incoming and outgoing arcs for node sets $W \subset V$. By slightly abusing notation, we write $\delta^-(i)$ and $\delta^+(i)$ instead of $\delta^-(\{i\})$ or $\delta^+(\{i\})$ for singletons $W = \{i\}$.

Related work. The DNDPR has been introduced in Li et al. [19] where a compact *node-arc formulation* and an *arc-path formulation* with an exponential number of variables have been proposed. Two branch-and-price (B&P) algorithms based on the latter formulation have been developed that differ in the way the pricing subproblem is solved. A metaheuristic based on tabu search has been recently proposed in Li et al. [20].

Several related studies consider the undirected variant of the problem, i.e., the NDPR. The NDPR has been introduced in Cabral et al. [2] where the proposed B&P approach turned out to be quite inefficient (even for small instances) due to the high complexity of the associated pricing subproblem. Therefore, Cabral et al. [2] have focused on construction heuristics that were able to tackle larger problem instances in comparably short time. More efficient B&P approaches for the NDPR have been given in Leitner et al. [18] and Yıldız et al. [30]. In addition to these exact approaches, several metaheuristics have been developed for approximately solving larger problem instances: genetic algorithms (Kulturel-Konak and Konak [16], Konak [15]), tabu search (Lin et al. [21]), and variable neighborhood search (Xiao and Konak [28]).

Existing methods for the NDPR cannot by applied in a straightforward way to the DNDPR, since NDPR solutions may contain cycles (or even traverse a single edge in both directions for one commodity). Besides, asymmetric arc costs and arcs existing in a single direction only would require some adaptations.

Node-arc formulation. The node-arc formulation (1) introduced in Li et al. [19] is used for comparison purposes in our computational study. Therefore, we briefly summarize it in the following. Its basic idea is to keep track of the distance from the last relay (or the source of the respective commodity) in order to forbid subpaths exceeding the distance bound. Four sets of variables are used. Binary arc and node variables x_a, $\forall a \in A$, and y_i, $\forall i \in V$, mark the selected arcs and relays, respectively. For each commodity $k = (u,v) \in \mathcal{K}$ and node $i \in V$, continuous variable v_i^k tracks the distance of node i to the preceding relay or the source u of that commodity (in case the path from u to i does not contain relays). Finally, multi-commodity flow variables f_a^k, $\forall k \in \mathcal{K}$, $\forall a \in A$, are used to enforce connectivity of each commodity pair. Formulation (NA) reads as follows:

$$\min \sum_{i \in V} c_i y_i + \sum_{a \in A} w_a x_a \tag{1a}$$

$$\text{s.t.} \sum_{a \in \delta^+(i)} f_a^k - \sum_{a \in \delta^-(i)} f_a^k = \begin{cases} 1 & \text{if } k = (i,j) \\ -1 & \text{if } k = (j,i) \\ 0 & \text{otherwise} \end{cases} \quad \forall k \in \mathcal{K}, \forall i \in V, \tag{1b}$$

$$f_{ij}^k \le x_{ij} \qquad \forall k \in \mathcal{K}, \forall (i,j) \in A, \tag{1c}$$

$$v_i^k + d_{ij} f_{ij}^k - \lambda_{\max}(1 - f_{ij}^k + y_j) \le v_j^k \qquad \forall k \in \mathcal{K}, \forall (i,j) \in A, \tag{1d}$$

$$
\begin{aligned}
& v_i^k + d_{ij} f_{ij}^k \leq \lambda_{\max} && \forall k \in \mathcal{K}, \forall (i,j) \in A, && \text{(1e)} \\
& 0 \leq v_i^k \leq \lambda_{\max}(1 - y_i) && \forall k \in \mathcal{K}, \forall i \in V, && \text{(1f)} \\
& v_u^{uv} = 0 && \forall (u,v) \in \mathcal{K}, && \text{(1g)} \\
& f_{ij}^k \in \{0,1\} && \forall k \in \mathcal{K}, \forall (i,j) \in A, && \text{(1h)} \\
& y_i \in \{0,1\} && \forall i \in V, && \text{(1i)} \\
& 0 \leq x_{ij} \leq 1 && \forall (i,j) \in A. && \text{(1j)}
\end{aligned}
$$

Flow conservation constraints (1b) together with linking constraints (1c) ensure the existence of a directed path from u to v for each commodity pair $(u,v) \in \mathcal{K}$. Constraints (1d) ensure that the value of variable v_j^k is at least the distance from the last relay or from the source, respectively, along the path connecting commodity $k \in \mathcal{K}$. The distance limit is enforced using inequalities (1e) and (1f). The latter inequalities also link distance and relay variables. Observe that binary (rather than continuous) flow variables are needed to prevent flow splittings which would yield incorrect values of distance variables v. The main disadvantage of this model is its relatively weak linear programming (LP) relaxation bound, resulting from the (potentially) large coefficient $\lambda_{\max}$ required in constraints (1d).

Overview and contributions. Two mixed integer linear programming (MILP) formulations that are based on considering one layered graph per source are introduced in Section 2. The first one is a flow-based formulation with a pseudo-polynomial number of variables and constraints, whereas the second one uses an exponential number of connectivity constraints. Fractional distance values are handled efficiently by augmenting both models with an exponentially-sized set of infeasible path constraints. Subsequently, different families of symmetry breaking constraints and valid inequalities are introduced. Section 3 describes components and variants of a branch-and-cut (B&C) algorithm based on the second formulation, introduces preprocessing routines, and details a heuristic used to obtain initial solutions. Benchmark instances used in our study are described in Section 4 where we also verify the effectiveness of our algorithms by extensive computational experiments. Finally, implications of our study to the practice of management are highlighted in Section 4.6.

2. Formulations

Properties of feasible solutions. In the DNDPR, the routing of each single commodity is done following a simple path, see [19]. Hence, the in-degree of each node in the routing path is at most one. An optimal solution is a union of all routing paths over all commodity pairs, and hence, the in-degree of a node in this solution can be as large as the number of commodities. On the other hand, if all commodities share a common source node, then it is not difficult to see that there always exists an optimal solution in which the in-degree of each node is at most one, i.e., such that the set A' of selected arcs forms an arborescence.

Theorem 1. *If $S = \{u\}$, there exists an optimal DNDPR solution which corresponds to a Steiner arborescence rooted at u, whose leaves are a subset of the nodes from T^u.*

Theorem 1 enables the interpretation of an optimal solution as a union of several Steiner arborescences (one per source). It does not help, however, to handle the distance constraints and the installation of relays to respect the threshold $\lambda_{\max}$. To deal with these issues, we propose to exploit layered graphs introduced below.

The new formulations presented in this article use extended, so-called layered graphs. The basic idea of layered graphs is to introduce multiple copies for each node and arc of an original graph along one or multiple dimensions (e.g., time or distance) to implicitly model certain constraints and to obtain stronger mathematical models. In our case, layered graphs are used to encode the distances. If all distance values are integral, only feasible paths with respect to the distance bound $\lambda_{\max}$ are generated. On the contrary, paths (slightly) violating the distance bound may be contained in our layered graphs in the more general case of fractional distance values. These paths will be excluded from solutions via additional inequalities. Picard and Queyranne [26] were among the first to consider layered graphs and used them for solving the time-dependent traveling salesman problem. More recent successful applications of layered graphs are, e.g., given in Godinho et al. [8], Gouveia et al. [11, 12, 13], Gouveia and Ruthmair [10], Ljubić and Gollowitzer [22], Ruthmair and Raidl [27]; see Gouveia et al. [14] for a survey on this topic. Using an approximate layered graph and handling infeasible paths by cutting planes is similar to the approach used in Dash et al. [6], however, we use a static relaxed layered graph instead of an iteratively derived one.

For the DNDPR, we construct layered digraph $G_{\mathrm{L}} = (V_{\mathrm{L}}, A_{\mathrm{L}})$ whose node set V_{L} is recursively defined by sets V_{L}^l, $\forall l \in \{0, 1, \ldots, \lambda_{\max}\}$. Thereby, $V_{\mathrm{L}} = V_{\mathrm{L}}^{\lambda_{\max}}$ and each subset V_{L}^l contains all nodes that can be reached with a total distance of at most l starting from a node at layer zero, i.e.,

$$
\begin{aligned}
V_{\mathrm{L}}^0 &= \{i_0 \mid i \in V\} \\
V_{\mathrm{L}}^l &= \{j_l \mid i_m \in V_{\mathrm{L}}^{l-1}, (i,j) \in \delta^+(i), m + \lfloor d_{ij} \rfloor = l\} \cup V_{\mathrm{L}}^{l-1}.
\end{aligned}
$$

Arc set A_{L} connects layered node copies $i_l, j_m \in V_{\mathrm{L}}$ for which $(i,j) \in A$ and the difference of the layers corresponds to the arc distance rounded down to the nearest integer, i.e., $m - l = \lfloor d_{ij} \rfloor$. Furthermore, arcs (i_l, i_0) are included for each node $i_l \in V_{\mathrm{L}}$ not at layer zero, i.e., when $l > 0$. As the latter arcs correspond to using a relay, we will call them *relay arcs*. Formally, arc set $A_{\mathrm{L}} = A_{\mathrm{L}}^{\mathrm{a}} \cup A_{\mathrm{L}}^{\mathrm{r}}$ where $A_{\mathrm{L}}^{\mathrm{r}}$ is the set of relay arcs and $A_{\mathrm{L}}^{\mathrm{a}}$ is the set of arcs derived from the original graph:

$$
\begin{aligned}
A_{\mathrm{L}}^{\mathrm{r}} &= \{(i_l, i_0) \mid i_l \in V_{\mathrm{L}}, l > 0\} \\
A_{\mathrm{L}}^{\mathrm{a}} &= \{(i_l, j_m) \mid i_l, j_m \in V_{\mathrm{L}}, (i,j) \in A, \lfloor d_{ij} \rfloor = m - l\}.
\end{aligned}
$$

Figure 3 shows the layered graph corresponding to the instance given in Figure 1 as well as the embedding of the optimal solution in the layered graph. Thereby, relay arcs $A_{\mathrm{L}}^{\mathrm{r}}$ are depicted in dashed lines and the remaining arcs in solid lines. Bold blue arcs indicate those that are included in the considered solution.

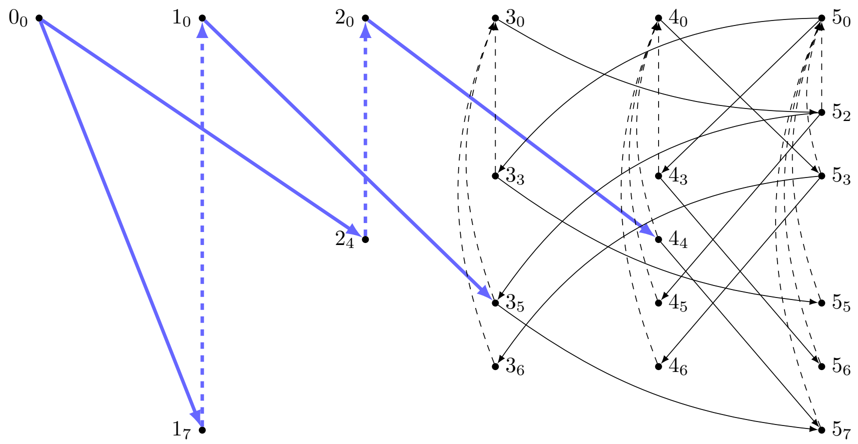


Figure 3: Layered graph $G_{\mathrm{L}} = (V_{\mathrm{L}}, A_{\mathrm{L}})$ for $\lambda_{\max} = 7$ corresponding to the instance in Figure 1. The optimal solution is marked bold and blue. Dashed lines indicate relay arcs.

2.1. Multi-commodity Flow Formulation

The *layered multi-commodity flow formulation* ($\mathrm{L}_{\mathrm{MCF}}$) is based on flow variables $f_a^{uv} \geq 0$, $\forall (u,v) \in \mathcal{K}$, $\forall a \in A_{\mathrm{L}}$. As in the node-arc formulation, variables $y_i \in \{0,1\}$, $\forall i \in V$, indicate whether a relay is placed at some node and variables x_a, $\forall a \in A$, indicate whether an arc is included in a solution. Observe that for $(u,v) \in \mathcal{K}$, flow variables $\mathbf{f}^{uv}$ corresponding to arcs leaving any copy of target node v can be fixed to zero. Similarly, arcs incident to a copy of source node u on a non-zero layer can be set to zero. Instead of formulating the corresponding constraints we omit variables with respect to these arcs by using the notation

$$\hat{A}_{\mathrm{L}}^{uv} = \bigcup_{u_l \in V_{\mathrm{L}} : l > 0} \delta^+(u_l) \cup \bigcup_{u_l \in V_{\mathrm{L}}} \delta^-(u_l) \cup \bigcup_{v_l \in V_{\mathrm{L}}} \delta^+(v_l)$$

and deriving a formulation using flow variables f_a^{uv} for arcs from $A_{\mathrm{L}} \setminus \hat{A}_{\mathrm{L}}^{uv}$ only, given a commodity $(u,v) \in \mathcal{K}$. Formulation ($\mathrm{L}_{\mathrm{MCF}}$) reads then as follows:

$$\min \sum_{i \in V} c_i y_i + \sum_{a \in A} w_a x_a \tag{2a}$$

$$\text{s.t.} \sum_{a \in \delta^+(u_0)} f_a^{uv} = 1 \qquad \forall (u,v) \in \mathcal{K}, \tag{2b}$$

$$\sum_{a \in \delta^-(i_l)} f_a^{uv} - \sum_{a \in \delta^+(i_l)} f_a^{uv} = 0 \qquad \forall (u,v) \in \mathcal{K}, \forall i_l \in V_{\mathrm{L}} : i \notin \{u,v\}, \tag{2c}$$

$$\sum_{v_l \in V_{\mathrm{L}}} \sum_{a \in \delta^-(v_l) \setminus A_{\mathrm{L}}^{\mathrm{r}}} f_a^{uv} = 1 \qquad \forall (u,v) \in \mathcal{K}, \tag{2d}$$

$$\sum_{i_l \in V_{\mathrm{L}}} \sum_{a \in \delta^-(i_l) \setminus A_{\mathrm{L}}^{\mathrm{r}}} f_a^{uv} \leq 1 \qquad \forall (u,v) \in \mathcal{K}, \forall i \in V \setminus \{u,v\}, \tag{2e}$$

$$\sum_{a = (i_l, i_0) \in A_{\mathrm{L}}^{\mathrm{r}}} f_a^{uv} \leq y_i \qquad \forall (u,v) \in \mathcal{K}, \forall i \in V, \tag{2f}$$

$$\sum_{a = (i_l, j_m) \in A_{\mathrm{L}}^{\mathrm{a}}} f_a^{uv} \leq x_{ij} \qquad \forall (u,v) \in \mathcal{K}, \forall (i,j) \in A, \tag{2g}$$

$$\sum_{a\in\sigma} f_a^{uv} \leq |\sigma| - 1 \qquad \forall (u,v)\in\mathcal{K}, \forall \sigma \in \mathcal{P}_{\text{inf}}, \quad \text{(2h)}$$

$$y_i \in \{0,1\} \qquad \forall i \in V, \quad \text{(2i)}$$

$$x_a \in \{0,1\} \qquad \forall a \in A, \quad \text{(2j)}$$

$$f_a^{uv} \in \{0,1\} \qquad \forall (u,v)\in\mathcal{K}, \forall a \in A_{\text{L}} \setminus \hat{A}_{\text{L}}^{uv}. \quad \text{(2k)}$$

For each commodity, flow balance constraints (2b)–(2d) together with linking constraints (2g) ensure connectivity between the source and exactly one copy of the target node. Inequalities (2e) ensure that this connection contains at most one copy of each intermediate node, i.e., that the associated path in the original graph is simple. Constraints (2f) link the relay arcs to the relay variables. Recall that all fractional distances are rounded down in the construction of layered graph G_{L}. As a consequence G_{L} may contain paths whose length exceeds the distance limit $\lambda_{\max}$. Such paths are clearly infeasible, as they do not contain any relay arc. Let $\mathcal{P}_{\text{inf}}$ denote this set of infeasible paths that may occur in the layered graph. To forbid the usage of paths from $\mathcal{P}_{\text{inf}}$, we introduce *infeasible path constraints* (2h), cf. Ascheuer et al. [1]. These constraints, which are only considered if arcs with fractional distance values exist, are separated dynamically, see Section 3.3 for details.

2.2. Cut Formulation

In contrast to formulation (L_{MCF}), which considers one variable for each commodity pair and layered graph arc, the *layered cut formulation* (L_{CUT}) uses one layered graph variable $z_a^u \in \{0,1\}$ for each source $u \in S$ and layered graph arc $a \in A_{\text{L}}$. By means of an exponential number of connectivity constraints, each set of variables $\mathbf{z}^u$ will model an arborescence rooted at $u \in S$ that reaches all targets $v \in T^u$, cf. Gouveia et al. [12] where a similar idea has been used in the context of a Steiner tree problem with multiple root nodes.

Similar to formulation (L_{MCF}), for $u \in S$, we can eliminate $\mathbf{z}^u$ variables associated to arcs incident to a copy of source node u on a non-zero layer. In other words, for a given $u \in S$, we define

$$\hat{A}_{\text{L}}^{u} = \bigcup_{u_l \in V_{\text{L}}: l>0} \delta^+(u_l) \cup \bigcup_{u_l \in V_{\text{L}}} \delta^-(u_l)$$

and work only with z_a^u variables from $A_{\text{L}} \setminus \hat{A}_{\text{L}}^{u}$. Notice that, in contrast to formulation (L_{MCF}), the arcs leaving target nodes cannot be removed from this model. The (L_{CUT}) formulation reads as follows:

$$\min \sum_{i\in V} c_i y_i + \sum_{a\in A} w_a x_a \quad \text{(3a)}$$

$$\text{s.t.} \sum_{a\in\delta^-(W)} z_a^u \geq 1 \quad \forall u \in S, \forall W \subseteq V_{\text{L}} \setminus \{u_0\}, \exists v \in T^u : \{v_l \in V_{\text{L}}\} \subseteq W, \quad \text{(3b)}$$

$$\sum_{i_l\in V_{\text{L}}} \sum_{a\in\delta^-(i_l)\setminus A_{\text{L}}^{\text{r}}} z_a^u = 1 \qquad \forall u \in S, \forall i \in T^u, \quad \text{(3c)}$$

$$\sum_{i_l\in V_{\text{L}}} \sum_{a\in\delta^-(i_l)\setminus A_{\text{L}}^{\text{r}}} z_a^u \leq 1 \qquad \forall u \in S, \forall i \in V \setminus (T^u \cup \{u\}), \quad \text{(3d)}$$

$$\sum_{a=(i_l,i_0)\in A^{\mathrm{r}}_{\mathrm{L}}} z^u_a \leq y_i \qquad \forall u \in S, \forall i \in V, \quad (3e)$$

$$\sum_{a=(i_l,j_m)\in A^{\mathrm{a}}_{\mathrm{L}}} z^u_a \leq x_{ij} \qquad \forall u \in S, \forall (i,j) \in A, \quad (3f)$$

$$\sum_{a\in\sigma} z^u_a \leq |\sigma| - 1 \qquad \forall u \in S, \forall \sigma \in \mathcal{P}_{\mathrm{inf}}, \quad (3g)$$

$$y_i \in \{0,1\} \qquad \forall i \in V, \quad (3h)$$

$$x_a \in \{0,1\} \qquad \forall a \in A, \quad (3i)$$

$$z^u_a \in \{0,1\} \qquad \forall u \in S, \forall a \in A_{\mathrm{L}} \setminus \hat{A}^u_{\mathrm{L}}. \quad (3j)$$

Connectivity constraints (3b) state that every subset of nodes containing all copies of some target node must be connected to the corresponding source. As there exist exponentially many of these constraints, we will add them on the fly in a cutting plane approach, see Section 3.3 for details. Constraints (3c) and (3d) prevent nodes from being visited more than once, i.e., each target node is visited exactly once and each non-target node is visited at most once. Thus, together with constraints (3b) they ensure that, for every source $u \in S$, the subgraph induced by all arcs $a \in A_{\mathrm{L}}$ such that $z^u_a = 1$ is an arborescence rooted at u_0 that contains exactly one copy of each node $v \in T^u$. The layered graph variables are linked to the relay node and arc variables on the original graph by inequalities (3e) and (3f), respectively. Infeasible path constraints (3g) are considered in the case of fractional distances to ensure that paths violating the distance constraint are not used, see Section 3.3 for their separation.

In the upcoming polyhedral comparison, we compare the strength of the two proposed formulations, concerning the quality of their LP relaxation bounds. In doing so, we ignore infeasible path constraints (2h) and (3g), as these valid inequalities are only used for cutting off infeasible integer solutions in case of fractional distances and not for strengthening the LP relaxation of our models. Additionally, since the two sets of inequalities are defined in different variable spaces it is not obvious how they relate to each other in this context.

Theorem 2. *Formulations (L_{MCF}) and (L_{CUT}) without infeasible path constraints* (2h) *and* (3g)*, respectively, are equally strong, i.e., the LP relaxation values of the two models coincide.*

Proof. Let $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{f}^*)$ be an optimal LP solution of the ($\mathrm{L}_{\mathrm{MCF}}$) model. We show how to construct a feasible solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}})$ of the ($\mathrm{L}_{\mathrm{CUT}}$) model with the same objective value. We set $\tilde{\mathbf{x}} = \mathbf{x}^*$, $\tilde{\mathbf{y}} = \mathbf{y}^*$, and

$$\tilde{z}^u_a = \max_{(u,v)\in\mathcal{K}} f^{uv}_a \qquad \forall u \in S, \forall a \in A_{\mathrm{L}}.$$

Following this definition, it is not difficult to see that flow-based capacity constraints (2f) and (2g) imply constraints (3e) and (3f), respectively. Consider a node $u \in S$. The flow-balance constraints (2d) are slightly different from the classical ones, due to the aggregation of the incoming flow at the target node $v \in T^u$. In this constraint the incoming flow is aggregated over all copies $v_l \in V_{\mathrm{L}}$ of the target node $v \in T^u$. This can be interpreted as a flow-balance constraint in a modified layered graph, say G^{uv}_{L}, in which a target node t_v is introduced for each node $v \in T^u$, and arcs (v_l, t_v) with infinite capacity are added to this graph. Hence, in such a modified graph, the flow-balance constraints (2b)–(2d) guarantee existence of a path from u_0 to t_v, for each $t \in T^u$. By the max-flow min-cut theorem, this implies that cut-set inequalities (3b) are satisfied. Degree-constraints (3c) and (3d) are not satisfied by an arbitrary flow $\mathbf{f}^*$, but the flow can be rerouted (without changing the capacities given by $\mathbf{x}^*$ and $\mathbf{y}^*$) so that these constraints are always satisfied.

Consider now an optimal LP solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}})$ of the $(\mathrm{L_{CUT}})$ model. For each commodity pair $(u, v) \in \mathcal{K}$, we consider the graph G_{L}^{uv} described above, with arc capacities cap_a defined as:

$$\mathrm{cap}_a = \tilde{z}_a^u, \forall a \in A_{\mathrm{L}} \qquad \mathrm{cap}_a = \infty, \forall a = (v_l, t_v), v_l \in V_{\mathrm{L}}.$$

By the max-flow min-cut theorem applied to G_{L}^{uv}, it follows that for each $(u, v) \in \mathcal{K}$, one can send one unit of flow from u_0 to t_v in G_{L}^{uv} using $\tilde{\mathbf{z}}$ (and hence $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{x}}$) as capacities. Since $f_a^{uv} \leq \tilde{z}_a^u$ holds for each $a \in A_{\mathrm{L}}$ and $(u, v) \in \mathcal{K}$, constraints (2e)–(2g) are implied by (3d)–(3f) which concludes the proof. □

We also propose to extend the $(\mathrm{L_{CUT}})$ model by considering *flow-balance constraints* (4). For each source $u \in S$, they ensure that an outgoing arc of layered graph node i_l, $i \notin T^u \cup \{u\}$, has to be used in the arborescence associated to source u if at least one incoming arc is chosen as well:

$$\sum_{a \in \delta^-(i_l)} z_a^u \leq \sum_{a \in \delta^+(i_l)} z_a^u \qquad \forall u \in S, \forall i_l \in V_{\mathrm{L}} : i \notin T^u \cup \{u\}. \tag{4}$$

While the flow-balance constraints are not necessary to ensure validity of $(\mathrm{L_{CUT}})$, there exist cases in which they strengthen the associated LP relaxation. Figure 4 shows one of the rather typical situations in which this happens due to involved relay arcs. By adding flow-balance inequalities, the LP solution becomes integral and corresponds to the optimal solution shown in Figure 3. Observe that the two incoming arcs to node 5_7 belong to different variable sets in the corresponding LP solution to $(\mathrm{L_{MCF}})$ for which reason similar constraints are not strengthening there. Besides strengthening the LP relaxation of $(\mathrm{L_{CUT}})$, the flow-balance constraints also help to improve convergence by reducing the number of violated connectivity cuts (3b).

2.3. Symmetry Breaking Constraints

By construction of the layered graph, it can sometimes happen that multiple feasible embeddings of rooted arborescences, one for each $u \in S$, exist. Each such embedding results in the same solution in the original graph, and hence, symmetries may be introduced in our $(\mathrm{L_{MCF}})$ and $(\mathrm{L_{CUT}})$ models. Since these symmetries may deteriorate the performance of branch-and-bound (B&B) based approaches, we next introduce two families of symmetry breaking constraints. One typical situation arises if the routing paths of two different commodities contain a common node that needs to be used as a relay by only one of them. Let $i \in V$ be a node at which a relay has to be installed and let $(u, v) \in \mathcal{K}$ be a commodity that does not need to use i as a relay along its routing path. Assume that the distance to the previous relay (or the commodity source) of i along the path connecting u and v is equal to l. Furthermore, let (i, j) be the outgoing arc of node i on this path. Then, two feasible routing paths in G_{L} exist:

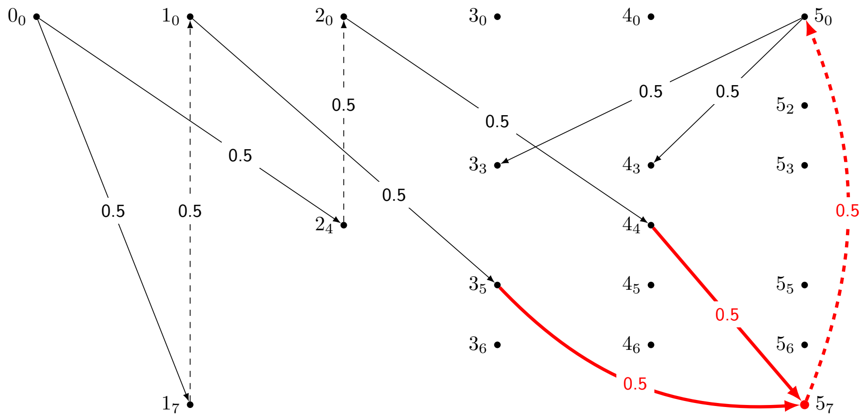


Figure 4: Optimal LP solution of ($\mathrm{L_{CUT}}$) on the layered graph for the input in Figure 1 without flow-balance constraints. Only arcs with associated non-zero variables are shown, labeled with the respective LP values. The violation of flow-balance constraints at node 5_7 is marked bold and red.

1. If the relay at i is not used, then layered graph arc $(i_l, j_{l+d_{ij}})$ belongs to the arborescence rooted at u, and is used to connect u_0 to some copy of v in V_L;
2. Alternatively, if the relay at i is used, the subpath given by $\{(i_l, i_0), (i_0, j_{d_{ij}})\}$ is used instead.

To get rid of symmetries implied by such ambiguities, we force that in every feasible routing path installed relays are used whenever possible.

In case of ($\mathrm{L_{CUT}}$), this is enforced by constraints (5) that forbid the use of non-relay arcs emanating from some node i_l, $l > 0$, if a relay is installed at node i:

$$\sum_{a=(i_l,j_m)\in A_\mathrm{L}\setminus A_\mathrm{L}^\mathrm{r}:l>0} z_a^u \le M_i^u \cdot (1 - y_i) \qquad \forall u \in S, \forall i \in V, i \neq u. \tag{5}$$

Thereby, M_i^u is a (tight) upper bound on the out-degree of node i in the arborescence rooted at $u \in S$ which is defined as follows:

$$M_i^u = \begin{cases} \min(|T^u|, |\delta^+(i)|) & \text{if } i \notin T^u \\ \min(|T^u| - 1, |\delta^+(i)|) & \text{otherwise.} \end{cases}$$

For ($\mathrm{L_{MCF}}$) we use the stronger variant of the above symmetry breaking constraints

$$\sum_{a=(i_l,j_m)\in A_\mathrm{L}\setminus A_\mathrm{L}^\mathrm{r}:l>0} f_a^{uv} \le 1 - y_i \qquad \forall (u,v) \in \mathcal{K}, \forall i \in V, i \neq u \tag{6}$$

that exploit the fact that the binary flow variables are disaggregated per commodity. Thus, the outflow of each node is at most one.

3. Algorithmic Framework

This section describes all implementation details that are relevant to ensure a good performance of our approaches. These include: (1) preprocessing techniques that aim to reduce the number of variables that have to be considered, (2) further valid inequalities, (3) the separation routines of all families of inequalities that are added dynamically, (4) customized branching priorities, and (5) a heuristic to obtain initial solutions.

3.1. Preprocessing

Reductions may be possible for nodes $i \in V$ whose in-degree or out-degree is equal to one at some layer. If $\delta^-(i_l) = \{(j_p, i_l)\}$ for a node $i_l \in V_{\mathrm{L}}$, $i \notin S$, we can remove a possibly existing outgoing arc (i_l, j_m) and all associated variables since using it would induce a cycle of length two in the original graph. Similarly, incoming arcs (j_p, i_l) can be eliminated if $\delta^+(i_l) = \{(i_l, j_m)\}$ in case $i \notin T$. In case a non-source node becomes unreachable (i.e., all incoming arcs are removed), this node and all its outgoing arcs can be removed as well (cf. *simple path reductions* introduced in De Boeck and Fortz [7]). Similarly, a non-target node without outgoing arcs can be removed together with all its incoming arcs. Additional reductions can be made if nodes are unreachable from a particular target, or have no remaining flow or layered arc variables associated to incoming (outgoing, respectively) arcs for a particular commodity or source node. In these cases, we eliminate the flow variables associated with that commodity or the layered arc variables associated with some source, respectively. These procedures are iteratively applied in several elimination rounds until no further reductions occur.

3.2. Valid Inequalities

In this section, we describe two further families of valid inequalities for formulation ($\mathrm{L}_{\mathrm{CUT}}$). Though both are implied by the layered graph connectivity constraints (3b) considering them before separating the latter inequalities typically turns out to be beneficial for the performance of our B&C approaches.

Connectivity constraints on G. Connectivity constraints (7) on the original graph are analogous to inequalities (3b) on the layered graph:

$$\sum_{a \in \delta^-(W)} x_a \geq 1 \qquad \forall W \subset V : \exists (u,v) \in \mathcal{K}, u \notin W, v \in W. \tag{7}$$

They ensure that each node set that separates source and target of a commodity must have at least one incoming arc. From the max-flow min-cut theorem, one can easily conclude that any solution satisfying constraints (7) contains a path from u to v for every commodity $(u, v) \in \mathcal{K}$. This path may, however, contain relay-free subpaths whose distance exceeds $\lambda_{\max}$. Thus, they are not sufficient to guarantee a feasible solution. A main advantage compared to the layered graph connectivity constraints (3b) is that they are specified on the arc design variables of the original graph Thus, each such cut influences all commodities. Since the number of connectivity constraints (7) is exponential, we separate them dynamically; see Section 3.3 for details.

Two-cycle inequalities. Constraints (8) ensure that an outgoing arc of some layered graph node can only be used if at least one incoming arc whose source is different from the outgoing arc's target is used as well:

$$\sum_{a'=(p_r,i_l)\in\delta^-(i_l):p\neq j} z^u_{a'} \geq z^u_a \qquad \forall u \in S, \forall a = (i_l, j_m) \in A_\mathrm{L} : i \neq u. \tag{8}$$

Two-cycle inequalities (8) are implied by layered graph connectivity constraints (3b) and do not strengthen the formulation [9]. Similar to cut constraints (7), they are, however, beneficial for reducing the number of dynamically separated cut-set inequalities (3b).

Since the number of flow-balance constraints (4) and two-cycle inequalities (8) is pseudo-polynomial, two implementation variants are considered in our computations: (1) adding them exhaustively to the initial formulation, and (2) separating them dynamically. Details of the used separation procedures are given below in Section 3.3.

3.3. Separation

In this section we describe the separation procedure used in our B&C approach for formulation ($\mathrm{L_{CUT}}$) that dynamically adds layered connectivity constraints (3b), flow balance constraints (4), connectivity constraints on the original graph (7), and two-cycle elimination constraints (8). Two variants of the separation are considered in this paper. In the following, the version to which we refer as $\mathrm{L_{CUT}}$-d is described in detail, and minor modifications for the other variant, denoted by $\mathrm{L_{CUT}}$-s, are provided below. The overall separation procedure for $\mathrm{L_{CUT}}$-d is outlined in Algorithm 3.1.

1 separate cut-set inequalities (7) on the original graph
2 separate flow-balance constraints (4)
3 separate two-cycle inequalities (8)
4 **if** *no flow-balance constraints and two-cycle inequalities added* **then**
5   separate cut-set inequalities (3b) on the layered graph

Algorithm 3.1: Separation procedure for $\mathrm{L_{CUT}}$-d.

First, possibly violated cut-set constraints on the original graph are identified using the maximum flow algorithm by Cherkassy and Goldberg [5] (cf. Step 1 of Algorithm 3.1). Thereby, so-called nested cuts (see, e.g., Ljubić et al. [23]) are considered which means that the capacities of all arcs included in just added cuts are set to one and the flow computation is subsequently repeated to possibly find further violated inequalities. This procedure is applied for each commodity pair until no further violated inequalities are found. Before proceeding with the next commodity pair, all arc capacities are reset (to the original values induced by the current LP solution). Since this procedure may yield identical cuts for different commodity pairs, we employ duplicate detection and stop separating cuts for the current commodity as soon as a duplicate is identified. To keep track of the already added cuts and check for duplicates we use hash sets. The order in which the commodities are separated is perturbed in each separation call, i.e., we consider the commodities in a random order based on a fixed seed value.

Once this first separation routine terminates, we add violated flow-balance (4) and two-cycle inequalities (8) (cf. Steps 2 and 3 of Algorithm 3.1). Separation of these constraints is performed by inspecting the LP values of relevant variables for all not yet added inequalities. Complete separation of the two-cycles turned out to be too inefficient. Therefore, we resort to a slightly simpler approach that adds those inequalities if the following condition is violated:

$$\sum_{a' \in \delta^-(i_l)} z^u_{a'} \geq z^u_a \qquad \forall u \in S, \forall a = (i_l, j_m) \in A_{\mathrm{L}} : i \neq u.$$

Finally, we separate layered graph connectivity cuts (3b) in case neither flow-balance nor two-cycle inequalities have been added in the previous step (cf. Step 5 of Algorithm 3.1). Such a conditional separation is beneficial for avoiding too many (possibly redundant) constraints in the model. As before, violated cuts are identified by maximum flow computations using the algorithm from [5]. To detect a violated inequality of type (3b), for each $(u, v) \in \mathcal{K}$, we construct the layered graph G^{uv}_{L} as described in the proof of Theorem 2, and calculate the maximum flow between u_0 and the target node t_v, using z^u_a values as arc capacities for the arcs from A_{L} and ∞ for the arcs adjacent to t_v. If the obtained flow is less than one, the violated cut is added to the model. Again, we consider nested cuts, duplicate handling, and fixed-seed randomization for the order in which the commodity pairs are processed.

In the second implementation variant of our B&C approach, denoted by $\mathrm{L}_{\mathrm{CUT}}$-s, flow-balance constraints and two-cycle inequalities are not separated. Instead, since there is only a pseudo-polynomial number of them, these cuts are added a priori to the model. In addition, the layered cut-set inequalities are separated unconditionally, i.e., the overall separation procedure for $\mathrm{L}_{\mathrm{CUT}}$-s consists of Steps 1 and 5 of Algorithm 3.1 performed sequentially.

To avoid separating too many inequalities, we only add cut-set inequalities if they are violated by a value of at least 0.5. Flow-balance constraints and two-cycle inequalities, however, are separated without such a threshold.

Infeasible path cuts. Constraints (2h) and (3g) are only considered if the input instance contains fractional distance values, and separated if the current candidate solution vector is integral and satisfies all other types of dynamically separated inequalities. In this case, violated constraints corresponding to (inclusion-wise) minimal infeasible paths starting at source or relay nodes are identified by breadth-first search.

3.4. Branching

Several properties of feasible solutions are enforced on the layered graph. The objective function, however, depends solely on the variables corresponding to the original graph. Moreover, decisions concerning the arcs available in the original graph directly influence the layered graph variables. Hence, it is reasonable to focus on the former for branching decisions. Regarding the two types of variables for the original graph—edge and relay variables—it is natural to prioritize the relay variables since placing a relay is usually much more expensive than installing a connection along an arc. To stay consistent with the literature we do not use custom branching priorities for the node-arc formulation.

3.5. Initial Heuristic

Before starting the B&C algorithm, we compute a feasible solution and hand it over to the MILP solver as initial primal bound. The heuristic resembles Prim's algorithm for spanning trees: it computes optimal paths for one commodity at a time and sets the costs of used arcs and relays to zero before it proceeds with the next source-target pair. When only a single commodity pair is given the DNDPR is known as the minimum cost path problem with relays (MCPPR). This latter problem can be solved exactly using an efficient dynamic programming (DP) algorithm proposed by Laporte and Pascoal [17]. However, different to the DNDPR, the MCPPR also allows connecting commodities by non-simple paths. We therefore adjust the DP algorithm from Laporte and Pascoal [17] by keeping track of already visited nodes in each state, in order to disallow extensions that form cycles.

To improve the basic algorithm, we consider some extensions. Observe that the design of the heuristic entails a strong influence of the order in which the commodities are processed. We attempt to reduce this influence by considering ten different permutations based on fixed-seed randomization and then keep the best solution.

Additionally, within the DP algorithm for the MCPPR we not only order the labels by non-decreasing cost but also break ties by favoring paths that reach the considered node at smaller distance. Once the best solution among the ten runs has been identified, we run the DP algorithm once again with the costs of all used relays and arcs set to zero. In certain cases this helps to avoid redundancies resulting in a smaller cost. We note that similar heuristic ideas based on sequential upgrades of a partial solution have been successfully used in [18, 19].

4. Computational Study

In this section we present computational results for the considered algorithms and variants. We start by giving details on the computational environment as well as the used test instances and the motivation for their selection. Finally, we present the obtained results.

Our algorithms are implemented in C++ using CPLEX 12.7.1 as a general-purpose MILP solver. All experiments have been performed in single thread mode with default parameter settings except for the described modifications. Experiments have been executed on an Intel Xeon E5-2670v2 machine with 2.5 GHz. The computation time limit has been set to 7 200 seconds.

In the following we compare the four solution approaches described in Table 1. Note that in both L_{CUT}-s and L_{CUT}-d, cut-set inequalities (3b) and (7) are separated dynamically. In L_{CUT}-s, flow-balance constraints (4) and two-cycle inequalities (8) are added initially to the model while in L_{CUT}-d these two sets are separated dynamically as described in Section 3.3.

4.1. Instances

Benchmark instances used by the authors of [19] are no longer available (personal communication with X. Li). Due to the lack of other existing benchmark instances for the DNDPR we constructed new ones based on

Table 1: Overview of the tested algorithms with their abbreviations. Column "base" denotes the inequalities of the core model, column "static" provides valid inequalities that are added to the initial formulation, and column "separation" provides valid inequalities as well as inequalities of the base formulation that are separated dynamically.

		Inequalities		
Abbreviation	Model	Base	Static	Separation
NA	(NA) from [19]	(1b)–(1j)	-	-
$\mathrm{L}_{\mathrm{MCF}}$	($\mathrm{L}_{\mathrm{MCF}}$)	(2b)–(2k)	(6)	(2h)
$\mathrm{L}_{\mathrm{CUT}}$-s	($\mathrm{L}_{\mathrm{CUT}}$)	(3b)–(3j)	(4), (5), (8)	(3b), (3g), (7)
$\mathrm{L}_{\mathrm{CUT}}$-d	($\mathrm{L}_{\mathrm{CUT}}$)	(3b)–(3j)	(5)	(3b), (3g), (4), (7), (8)

existing instances for the NDPR. We consider three sets of benchmark instances: (1) asymmetric instances derived from Cabral et al. [2], (2) symmetric instances derived from Konak [15], and (3) a set of newly generated symmetric instances. In the following we shortly outline the construction procedures for the original instances which involve undirected graphs, and then discuss our adjustments to obtain directed graphs. These instances are available at `https://www.ac.tuwien.ac.at/research/problem-instances/#Directed_Network_Design_Problem_with_Relays`.

Cabral instances. Cabral et al. [2] generated instances in which the underlying graph is a square grid graph (i.e., each node is connected to its direct vertical and horizontal neighbors). Integral edge costs and distances are chosen uniformly at random from the interval $[10, 30]$ and the distance limit $\lambda_{\max}$ is equal to 70. The relay costs are selected uniformly at random from $\{\lambda_{\max}, \lambda_{\max}+1, \dots, 2\lambda_{\max}\}$. All instances are based on grid graphs with a rows and b columns (i.e., with $|V| = ab$ nodes and $|E| = 2ab - a - b$ edges). The small instance set consists of nine such graphs with $(a, b) \in \{(4, 5), (5, 5), (6, 5), (7, 5), (8, 5), (9, 5), (10, 5), (11, 5), (12, 5)\}$. For the large set (a, b) is chosen from $\{10, 20, 30, 40, 50\} \times \{5, 10, 15, 20\}$. Each set contains 10 instances for each graph and each considered number of commodities $|\mathcal{K}| \in \{5, 10\}$ (by random sampling of the commodities). Observe that the graph with $(a, b) = (10, 5)$ is contained in both sets but the sampled instances are not identical. In particular, all commodities of each instance have the same source node, i.e., $|S| = 1$. We remark that duplicate commodities exist in some of the instances from Cabral et al. [2] which can be removed in a preprocessing step.

We obtained directed instances by replacing each edge by two directed arcs. Distance and cost of the first arc are equivalent to those of the edge. The values for the second arc are chosen uniformly at random from the interval $[10, 30]$. Instances for which the specified number of commodities does not match the actual number have been corrected by inserting sufficiently many new commodities while preserving the property that $|S| = 1$. Note that [19] uses the same approach to construct their instances although they use new base graphs instead of those from [2]. Thus, our new instance set is comparable in size and structure. The basic instance properties ($|V|$, $|A|$, $|\mathcal{K}|$, and $\lambda_{\max}$) are shown in Tables 3 and 4. We consider all of the 180 small instances and all of the large instances except for those with $b = 20$ as those turned out to be too challenging to provide meaningful insights. Instead, we included further instances with fifteen commodities whose base graphs were also generated by Cabral et al. [2] but remained unpublished. Hence, the large set consists in total of 450 instances. In our result tables we mark the subsets of large instances considered by Li et al. [19] with the set indices used in their paper.

Konak instances. Konak [15] generated instances by first placing $|V| \in \{40, 50, 60, 80, 160\}$ nodes at random integer coordinates $(x, y) \in [0, 100] \times [0, 100]$. Initially all node pairs $i, j \in V$ are connected by arcs with lengths d_{ij} set to the Euclidean distance, while the costs are either equal to the arc length (type I) or equal to $\lambda_{\max} - d_{ij}$ (type II). Edges with length beyond the distance limit are omitted. This leads to instance sizes ranging from $|V| = 40$ and $|E| = 198$ up to $|V| = 160$ and $|E| = 3624$. Relay costs are selected uniformly at random from $\{0, 1, \ldots, 100\}$. Using $\lambda_{\max} \in \{30, 35\}$ and $|\mathcal{K}| \in \{5, 10\}$, 20 instances have been generated for each of the two types. Each of these instances typically contains multiple sources and targets.

Directed instances are obtained by replacing each edge by two directed arcs. However, this time both arcs have the cost and distance of the original edge. This is done to keep the instance Euclidean and also to preserve the direct or indirect correlation of edge costs and distances. See Table 5 for an overview.

The newly generated third instance group uses a similar construction principle as the instances by Konak [15] and is specifically designed to reflect a practical application from telecommunications. Further details are given in Section 4.6 below.

4.2. Comparison to the State of the Art

As indicated above, we could not obtain the instances used in the previous literature. However, the Cabral instances are comparable in structure and size to those tested in [19], which allows us to use them to obtain at least an intuition on how our exact algorithms compare to those presented in [19]. As reference point we employ the node-arc formulation. We compare speedups between NA and the best B&P approach from the literature as well as the algorithms based on our layered graph formulations. The respective values are computed by t_{NA}/t_{alg} for $alg \in \{\mathrm{B\&P}, \mathrm{L_{MCF}}, \mathrm{L_{CUT}}\text{-s}, \mathrm{L_{CUT}}\text{-d}\}$. Since the Cabral instances feature 10 instances of each type, results have been aggregated by computing averages. The results are shown in Table 2. Observe that we provide this comparison only for the small Cabral instances because there are no NA results in Li et al. [19] for the large set. The speedup values of our algorithms are slightly worse than those from the literature for the smallest instances but they are considerably better for $a \geq 6$. Also note that on our instances the node-arc formulation sometimes terminated prematurely due to the time limit. Allowing the algorithm to finish—as done in [19]—would have resulted in even larger speedups.

Without the original benchmark set, a precise comparison is impossible. Yet these results indicate that our algorithms are at least as fast as those from the existing literature and most likely outperform them significantly.

4.3. LP Relaxation Bounds

In the following, we compare the quality of lower bounds that can be obtained by the three algorithms from Table 1: $\mathrm{L_{MCF}}$, $\mathrm{L_{CUT}}$ (note that both $\mathrm{L_{CUT}}$-s and

Table 2: Speedup ratio to the node-arc formulation. Values have been obtained by dividing the computation time of the node-arc formulation through the computation time of the respective algorithm. The first column has been obtained by extracting the respective results from [19]. The highest speedup per row is marked bold.

	Speedup ratio			
Instance	B&P2 (Li et al.)	L_{MCF}	L_{CUT}-s	L_{CUT}-d
04A05B70L05K	**8.4**	1.3	3.1	3.2
04A05B70L10K	**55.9**	9.6	35.7	19.8
05A05B70L05K	**20.8**	8.2	15.7	14.1
05A05B70L10K	**107.3**	14.4	53.6	44.9
06A05B70L05K	7.1	14.0	**25.4**	22.3
06A05B70L10K	61.1	67.8	**164.4**	98.7
07A05B70L05K	**31.8**	15.0	19.2	16.2
07A05B70L10K	34.6	224.1	**476.3**	289.8
08A05B70L05K	9.3	194.9	**216.3**	81.7
08A05B70L10K	92.0	286.1	**543.7**	218.6
09A05B70L05K	9.9	97.1	**110.9**	69.0
09A05B70L10K	40.5	217.8	**391.6**	237.8
10A05B70L05K	40.9	305.6	**319.8**	122.1
10A05B70L10K	33.6	467.7	**1337.3**	683.9
11A05B70L05K	25.1	266.0	**306.5**	123.0
11A05B70L10K	45.4	731.0	**1500.4**	555.7
12A05B70L05K	5.2	**597.4**	528.4	215.0
12A05B70L10K	110.1	538.7	**1164.3**	405.9

L_{CUT}-d provide the same lower bounds), and NA. Note that we ignore infeasible path constraints in this comparison since we only use them to cut off infeasible integer solutions in cases of fractional distances and not to strengthen the LP bounds. When computing LP bounds we deactivate CPLEX presolving, general purpose heuristics, and general purpose cuts. In addition, no threshold value is set for the separation of cut-set inequalities (3b) and (7).

LP gaps are computed as $(UB^* - LB)/UB^*$ where UB^* is the best known upper bound and LB is the lower bound obtained by the LP relaxation. Tables 3 and 4 report results obtained on the Cabral instances, and Table 5 provides results for the Konak instances.

Cabral instances. We observe that the algorithms based on (L_{CUT}) yield the strongest bounds. L_{MCF} follows closely behind but mostly delivers strictly weaker bounds. The reason for this is the fact that the Cabral instances consider only a single source node. This means that algorithms based on (L_{CUT}) use precisely one set of variables with respect to the layered graph on which they model an arborescence. The multi-commodity flow formulation, on the other hand, uses one set of variables per commodity pair. In this situation the cut model benefits from aggregating per source which enables it to obtain a stronger bound by means of flow-balance constraints (4). Moreover, the cut formulation yields a much smaller model here with respect to the number of variables. In general, both algorithms based on layered graph models deliver excellent bounds well below 5 %. The only exception are the larger instances with fifteen commodities where the bounds are slightly larger or even missing due to hitting the time limit. For the reasons mentioned above we observe that the performance of the multi-commodity flow formulation is much more susceptible to an increasing number of commodities than the cut formulation. As expected, the node-arc formulation is the weakest model with significantly

Table 3: LP gaps for the small directed Cabral instances. Each line represents the average across ten instances. The strongest bounds per row are marked bold.

	Properties				LP gap [%]		
Instance	$\|V\|$	$\|A\|$	$\lambda_{\max}$	$\|\mathcal{K}\|$	$\mathrm{L}_{\mathrm{MCF}}$	$\mathrm{L}_{\mathrm{CUT}}$	NA
04A05B70L05K	20	62	70	5	0.2	**0.0**	27.6
04A05B70L10K	20	62	70	10	0.2	**0.0**	35.0
05A05B70L05K	25	80	70	5	0.8	**0.0**	31.4
05A05B70L10K	25	80	70	10	0.1	**0.0**	34.4
06A05B70L05K	30	98	70	5	0.5	**0.0**	36.8
06A05B70L10K	30	98	70	10	0.6	**0.0**	34.9
07A05B70L05K	35	116	70	5	0.1	**0.0**	40.5
07A05B70L10K	35	116	70	10	0.7	**0.1**	40.6
08A05B70L05K	40	134	70	5	0.1	**0.0**	45.1
08A05B70L10K	40	134	70	10	1.0	**0.1**	40.2
09A05B70L05K	45	152	70	5	0.1	**0.0**	42.9
09A05B70L10K	45	152	70	10	0.7	**0.0**	39.8
10A05B70L05K	50	170	70	5	0.1	**0.0**	46.2
10A05B70L10K	50	170	70	10	0.9	**0.0**	43.9
11A05B70L05K	55	188	70	5	0.5	**0.0**	46.2
11A05B70L10K	55	188	70	10	0.2	**0.1**	42.5
12A05B70L05K	60	206	70	5	0.5	**0.1**	43.3
12A05B70L10K	60	206	70	10	0.8	**0.1**	42.6

worse bounds than the layered graph models. On the other hand, we obtain lower bounds for all instances in relatively short CPU times due to the small size of the model.

Konak instances. Compared to the Cabral instances we face much denser graphs here. Moreover, we are now dealing with multiple source nodes instead of just a single one. This means that now also the ($\mathrm{L}_{\mathrm{CUT}}$) formulation requires multiple sets of layered graph variables. Under these circumstances we still obtain strong LP bounds but not as strong as on the Cabral instances, see Table 5. For the larger instances of type I it becomes challenging to solve the LP relaxation to optimality as indicated by dashes in the table. The instances with indirectly correlated costs (type II) turned out to be much easier to solve. Here $\mathrm{L}_{\mathrm{MCF}}$ as well as $\mathrm{L}_{\mathrm{CUT}}$ provide results for all instances before exceeding the time limit. The results indicate that the bound strength is excellent if the computations can be completed. As before, we observe that the bounds provided by $\mathrm{L}_{\mathrm{CUT}}$ are at least as strong as those of $\mathrm{L}_{\mathrm{MCF}}$. The node-arc formulation (NA) yields much weaker bounds but also terminates significantly faster. Therefore, NA gives the only bounds for the two largest instances of type I where the time limit is reached for the layered graph models. In contrast to the layered graph models, the node-arc formulation seems to work much better on type I instances than on type II instances where the bounds are much worse, roughly by a factor of two.

4.4. Overall Performance

We continue by evaluating the performance of the MILP runs on the instances by Cabral et al. [2] and Konak [15].

Cabral instances. Our layered graph models are able to solve all 180 small instances to proven optimality and 334 of the 450 large instances. The MILP

Table 4: LP gaps for the large directed Cabral instances. Each line considers ten instances. Averages for the gaps are computed only with respect to the instances for which all algorithms terminated within the time limit. Column #tl denotes the number of instances that terminated due to the time limit. The strongest bounds per row are marked bold. Superscripts next to the instance names refer to the comparable instance group in Li et al. [19].

	Properties				Gap [%]			#tl		
Instance	$\lvert V\rvert$	$\lvert A\rvert$	$\lambda_{\max}$	$\lvert\mathcal{K}\rvert$	L_{MCF}	L_{CUT}	NA	L_{MCF}	L_{CUT}	NA
10A05B70L05K	50	170	70	5	0.4	**0.0**	47.5	**0**	**0**	**0**
10A05B70L10K	50	170	70	10	1.0	**0.0**	44.8	**0**	**0**	**0**
10A05B70L15K	50	170	70	15	1.3	**0.0**	42.2	**0**	**0**	**0**
10A10B70L05K[19]	100	360	70	5	1.6	**0.0**	48.4	**0**	**0**	**0**
10A10B70L10K[25]	100	360	70	10	0.8	**0.0**	45.1	**0**	**0**	**0**
10A10B70L15K	100	360	70	15	2.6	**0.0**	45.6	**0**	**0**	**0**
10A15B70L05K[20]	150	550	70	5	2.0	**0.3**	52.9	**0**	**0**	**0**
10A15B70L10K	150	550	70	10	3.3	**0.0**	50.1	**0**	**0**	**0**
10A15B70L15K	150	550	70	15	3.9	**0.1**	46.7	**0**	**0**	**0**
20A05B70L05K[21]	100	350	70	5	0.3	**0.0**	53.2	**0**	**0**	**0**
20A05B70L10K[26]	100	350	70	10	0.4	**0.1**	48.6	**0**	**0**	**0**
20A05B70L15K	100	350	70	15	0.1	**0.0**	48.7	**0**	**0**	**0**
20A10B70L05K	200	740	70	5	0.7	**0.0**	51.8	**0**	**0**	**0**
20A10B70L10K	200	740	70	10	2.0	**0.0**	50.6	**0**	**0**	**0**
20A10B70L15K	200	740	70	15	3.6	**0.1**	49.9	**0**	**0**	**0**
20A15B70L05K	300	1130	70	5	1.4	**0.0**	52.4	**0**	**0**	**0**
20A15B70L10K	300	1130	70	10	3.7	**0.1**	52.2	**0**	**0**	**0**
20A15B70L15K	300	1130	70	15	6.6	**2.5**	53.2	2	2	**0**
30A05B70L05K[22]	150	530	70	5	**0.0**	**0.0**	55.3	**0**	**0**	**0**
30A05B70L10K[27]	150	530	70	10	0.1	**0.0**	53.2	**0**	**0**	**0**
30A05B70L15K	150	530	70	15	0.3	**0.0**	51.3	**0**	**0**	**0**
30A10B70L05K	300	1120	70	5	1.0	**0.1**	54.2	**0**	**0**	**0**
30A10B70L10K	300	1120	70	10	1.4	**0.0**	54.2	**0**	**0**	**0**
30A10B70L15K	300	1120	70	15	1.9	**0.1**	53.1	4	**0**	**0**
30A15B70L05K	450	1710	70	5	0.7	**0.0**	56.0	**0**	**0**	**0**
30A15B70L10K	450	1710	70	10	1.6	**0.5**	53.6	5	**0**	**0**
30A15B70L15K	450	1710	70	15	-	-	-	10	3	**0**
40A05B70L05K[23]	200	710	70	5	0.1	**0.0**	56.6	**0**	**0**	**0**
40A05B70L10K[28]	200	710	70	10	0.3	**0.0**	55.7	**0**	**0**	**0**
40A05B70L15K	200	710	70	15	0.1	**0.0**	53.0	**0**	**0**	**0**
40A10B70L05K	400	1500	70	5	0.4	**0.0**	56.4	**0**	**0**	**0**
40A10B70L10K	400	1500	70	10	1.6	**0.4**	55.1	2	**0**	**0**
40A10B70L15K	400	1500	70	15	5.9	**4.7**	54.8	7	**0**	**0**
40A15B70L05K	600	2290	70	5	0.5	**0.0**	56.1	**0**	**0**	**0**
40A15B70L10K	600	2290	70	10	3.3	**1.9**	54.8	7	3	**0**
40A15B70L15K	600	2290	70	15	-	-	-	10	10	**0**
50A05B70L05K[24]	250	890	70	5	0.1	**0.0**	58.1	**0**	**0**	**0**
50A05B70L10K[29]	250	890	70	10	0.2	**0.0**	56.6	**0**	**0**	**0**
50A05B70L15K	250	890	70	15	0.1	**0.0**	54.9	**0**	**0**	**0**
50A10B70L05K	500	1880	70	5	0.2	**0.0**	56.6	**0**	**0**	**0**
50A10B70L10K	500	1880	70	10	1.0	**0.3**	55.2	3	**0**	**0**
50A10B70L15K	500	1880	70	15	-	-	-	10	**0**	**0**
50A15B70L05K	750	2870	70	5	0.5	**0.0**	56.4	**0**	1	**0**
50A15B70L10K	750	2870	70	10	**0.0**	**0.0**	54.4	9	7	**0**
50A15B70L15K	750	2870	70	15	-	-	-	10	10	**0**

Table 5: LP gaps for the directed Konak instances. Missing gap values correspond to runs that did not complete within the time limit. Bold values indicate the tightest bounds per type and instance.

					LP gap [%]					
	Properties				Type I			Type II		
Instance	$\lvert V\rvert$	$\lvert A\rvert$	$\lambda_{\max}$	$\lvert\mathcal{K}\rvert$	L_{MCF}	L_{CUT}	NA	L_{MCF}	L_{CUT}	NA
040N_05K_30L	40	396	30	5	**21.2**	**21.2**	39.2	**37.9**	**37.9**	77.3
040N_05K_35L	40	544	35	5	**4.7**	**4.7**	25.1	**0.6**	**0.6**	75.2
040N_10K_30L	40	396	30	10	22.9	**21.4**	41.3	**31.3**	**31.3**	76.6
040N_10K_35L	40	544	35	10	7.2	**6.3**	26.5	6.9	**4.9**	72.3
050N_05K_30L	50	558	30	5	**0.8**	**0.8**	31.6	**0.0**	**0.0**	76.5
050N_05K_35L	50	744	35	5	**0.0**	**0.0**	29.6	**0.0**	**0.0**	83.6
050N_10K_30L	50	558	30	10	20.1	**16.1**	48.8	**0.4**	**0.4**	79.6
050N_10K_35L	50	744	35	10	12.4	**9.4**	36.5	**0.0**	**0.0**	83.0
060N_05K_30L	60	610	30	5	**8.8**	**8.8**	51.7	**5.4**	**5.4**	84.9
060N_05K_35L	60	824	35	5	**2.6**	**2.6**	36.9	**0.0**	**0.0**	79.7
060N_10K_30L	60	610	30	10	**13.4**	**13.4**	51.1	**7.2**	**7.2**	82.1
060N_10K_35L	60	824	35	10	**4.8**	**4.8**	36.7	**0.0**	**0.0**	79.3
080N_05K_30L	80	1282	30	5	**1.7**	**1.7**	17.9	**1.2**	**1.2**	71.5
080N_05K_35L	80	1706	35	5	**0.0**	**0.0**	14.0	**0.2**	**0.2**	75.2
080N_10K_30L	80	1282	30	10	**4.3**	-	25.4	**0.6**	**0.6**	66.9
080N_10K_35L	80	1706	35	10	**4.4**	-	21.3	**0.0**	**0.0**	75.1
160N_05K_30L	160	5546	30	5	**0.3**	-	21.1	**2.1**	**2.1**	85.2
160N_05K_35L	160	7248	35	5	**6.2**	-	21.3	**3.2**	**3.2**	83.0
160N_10K_30L	160	5546	30	10	-	-	**32.1**	**2.4**	**2.4**	81.1
160N_10K_35L	160	7248	35	10	-	-	**29.5**	**0.2**	**0.2**	78.7

runs are consistent with the LP results, however, L_{MCF} is now much closer to the cut model despite its worse bounds—at least for the small instances, see Table 6. Optimality gaps are computed by $(UB^* - LB)/UB^*$ where UB^* is the best known upper bound and LB is the lower bound obtained by the investigated algorithm.

On the small instances we observe a clear difference between the static and the dynamic variant of the cut formulation L_{CUT}-s and L_{CUT}-d, respectively. The reasons for the advantage of the static approach are the sparseness and the size of the input graphs. Both lead to rather small models and the overhead for adding the valid inequalities in advance is manageable. Therefore, the slowdown for solving the LP relaxations is negligible but we can reduce the number of cut iterations in each node of the B&C tree significantly. Similarly, we also observe that much fewer B&B nodes—about 17 % on average—are needed until optimality can be proven. L_{MCF}, however, performs better in this respect: It solves the majority of instances already at the root node and the two “outliers” with 19 and 23 B&B nodes, respectively. The cut formulation, albeit being stronger, solves 52 instances fewer at the root node and requires up to 67 B&B nodes when adding flow-balance and two-cycle inequalities statically. The reasons for this seem to be that the fractional solutions of L_{MCF} are closer to being feasible and that L_{MCF} interacts better with the solver since no further inequalities are added in the solution process. Although the computation times of L_{MCF} and L_{CUT}-s are quite similar, we still observe that the former is considerably more sensitive to changes in the number of commodities due to the resulting increase in model size, see also Table 2. The node-arc formulation is significantly outperformed and cannot even solve all instances to optimality within the time limit.

Table 6: Results for the small directed Cabral instances. Each line represents the average across ten instances. Column #opt provides the number of optimally solved instances.

	Gap [%]				CPU time [s]				#opt			
Instance	L_{MCF}	L_{CUT}-s	L_{CUT}-d	NA	L_{MCF}	L_{CUT}-s	L_{CUT}-d	NA	L_{MCF}	L_{CUT}-s	L_{CUT}-d	NA
04A05B70L05K	**0.0**	**0.0**	**0.0**	**0.0**	**< 1**	**< 1**	**< 1**	1	**10**	**10**	**10**	**10**
04A05B70L10K	**0.0**	**0.0**	**0.0**	**0.0**	1	**< 1**	**< 1**	8	**10**	**10**	**10**	**10**
05A05B70L05K	**0.0**	**0.0**	**0.0**	**0.0**	1	**< 1**	**< 1**	4	**10**	**10**	**10**	**10**
05A05B70L10K	**0.0**	**0.0**	**0.0**	**0.0**	1	**< 1**	**< 1**	16	**10**	**10**	**10**	**10**
06A05B70L05K	**0.0**	**0.0**	**0.0**	**0.0**	1	**< 1**	1	11	**10**	**10**	**10**	**10**
06A05B70L10K	**0.0**	**0.0**	**0.0**	**0.0**	2	**1**	**1**	111	**10**	**10**	**10**	**10**
07A05B70L05K	**0.0**	**0.0**	**0.0**	**0.0**	**1**	**1**	**1**	10	**10**	**10**	**10**	**10**
07A05B70L10K	**0.0**	**0.0**	**0.0**	**0.0**	3	**1**	2	639	**10**	**10**	**10**	**10**
08A05B70L05K	**0.0**	**0.0**	**0.0**	**0.0**	**1**	**1**	2	167	**10**	**10**	**10**	**10**
08A05B70L10K	**0.0**	**0.0**	**0.0**	**0.0**	3	**2**	5	991	**10**	**10**	**10**	**10**
09A05B70L05K	**0.0**	**0.0**	**0.0**	**0.0**	**1**	**1**	**1**	78	**10**	**10**	**10**	**10**
09A05B70L10K	**0.0**	**0.0**	**0.0**	**0.0**	4	**2**	4	891	**10**	**10**	**10**	**10**
10A05B70L05K	**0.0**	**0.0**	**0.0**	**0.0**	**1**	**1**	2	302	**10**	**10**	**10**	**10**
10A05B70L10K	**0.0**	**0.0**	**0.0**	3.2	9	**3**	6	4126	**10**	**10**	**10**	8
11A05B70L05K	**0.0**	**0.0**	**0.0**	**0.0**	**1**	**1**	3	382	**10**	**10**	**10**	**10**
11A05B70L10K	**0.0**	**0.0**	**0.0**	6.8	6	**3**	8	4168	**10**	**10**	**10**	6
12A05B70L05K	**0.0**	**0.0**	**0.0**	1.9	**2**	**2**	4	906	**10**	**10**	**10**	9
12A05B70L10K	**0.0**	**0.0**	**0.0**	6.6	9	**4**	11	4666	**10**	**10**	**10**	6

Most of the observations with respect to the small instance set directly transfer to the large one, see Table 7. Considering the two algorithms based on (L_{CUT}) we now observe an advantage for the dynamic variant. Due to the larger graph sizes it is no longer beneficial to add all the strengthening inequalities to the initial formulation. In total the cut formulations solve the highest number of instances to optimality: 318 in the static and 334 in the dynamic variant. The multi-commodity flow formulations follows closely behind with 303 instances solved to proven optimality. In general, we observe that the flow formulation works quite well for instances with only five commodities, frequently even outperforming the cut formulation. However, as the number of commodities increases, the performance starts to deteriorate. We observe significant advantages for the cut formulations in these cases. In particular, for the largest instances the algorithm based on (L_{MCF}) often delivers no bounds while both algorithms based on (L_{CUT}) still terminate with comparatively tight gaps. This appears to be a natural consequence of the larger model size of (L_{MCF}) that depends to a higher degree on the number of commodities. We omit results for NA for reasons of space and since the results are by far not competitive to our layered graph approaches.

Konak instances. The results of solving the MILP formulations are provided in Tables 8 and 9. In accordance with the experiments on the LP bounds, type II instances are again easier to solve than type I instances. L_{MCF} and L_{CUT}-d solve all instances of type II to optimality. L_{CUT}-s, on the other hand, cannot solve the largest two instances of this set to optimality. Similarly, L_{CUT}-s solves fewer instances of type I to optimality than the dynamic variant and leaves larger gaps whenever both terminate prematurely due to the time limit. As the graphs are denser here than the 4-grid graphs of the Cabral instances, it is no longer beneficial to add all valid inequalities in advance. Thus, dynamic separation helps to reduce the size of the LP relaxations. While being slightly slower on

Table 7: Results for the large directed Cabral instances. Each line represents the average across ten instances. Column #opt provides the number of optimally solved instances. Superscripts next to the instance names refer to the comparable instance group in Li et al. [19].

	Gap [%]			CPU time [s]			#opt		
Instance	$\mathrm{L}_{\mathrm{MCF}}$	$\mathrm{L}_{\mathrm{CUT}}$-s	$\mathrm{L}_{\mathrm{CUT}}$-d	$\mathrm{L}_{\mathrm{MCF}}$	$\mathrm{L}_{\mathrm{CUT}}$-s	$\mathrm{L}_{\mathrm{CUT}}$-d	$\mathrm{L}_{\mathrm{MCF}}$	$\mathrm{L}_{\mathrm{CUT}}$-s	$\mathrm{L}_{\mathrm{CUT}}$-d
10A05B70L05K	**0.0**	**0.0**	**0.0**	**1**	**1**	2	**10**	**10**	**10**
10A05B70L10K	**0.0**	**0.0**	**0.0**	8	**3**	8	**10**	**10**	**10**
10A05B70L15K	**0.0**	**0.0**	**0.0**	48	**7**	15	**10**	**10**	**10**
10A10B70L05K[19]	**0.0**	**0.0**	**0.0**	**15**	20	37	**10**	**10**	**10**
10A10B70L10K[25]	**0.0**	**0.0**	**0.0**	71	**38**	79	**10**	**10**	**10**
10A10B70L15K	**0.0**	**0.0**	**0.0**	1200	**108**	212	**10**	**10**	**10**
10A15B70L05K[20]	**0.0**	**0.0**	**0.0**	**70**	125	184	**10**	**10**	**10**
10A15B70L10K	1.0	**0.0**	**0.0**	2505	**638**	732	8	**10**	**10**
10A15B70L15K	2.9	**0.0**	**0.0**	4606	**1961**	2701	5	**10**	**10**
20A05B70L05K[21]	**0.0**	**0.0**	**0.0**	**5**	6	11	**10**	**10**	**10**
20A05B70L10K[26]	**0.0**	**0.0**	**0.0**	62	**28**	40	**10**	**10**	**10**
20A05B70L15K	**0.0**	**0.0**	**0.0**	88	**32**	74	**10**	**10**	**10**
20A10B70L05K	**0.0**	**0.0**	**0.0**	**46**	96	172	**10**	**10**	**10**
20A10B70L10K	1.1	**0.0**	**0.0**	4101	**1332**	1731	5	**10**	**10**
20A10B70L15K	2.8	2.1	**1.7**	4886	3170	**3146**	4	**7**	**7**
20A15B70L05K	**0.0**	**0.0**	**0.0**	**423**	1589	1179	**10**	**10**	**10**
20A15B70L10K	3.1	4.1	**1.8**	6159	6086	**5231**	2	2	**7**
20A15B70L15K	16.7	8.2	**8.0**	7200	**6877**	6937	0	**1**	**1**
30A05B70L05K[22]	**0.0**	**0.0**	**0.0**	**13**	14	32	**10**	**10**	**10**
30A05B70L10K[27]	**0.0**	**0.0**	**0.0**	149	**77**	155	**10**	**10**	**10**
30A05B70L15K	**0.0**	**0.0**	**0.0**	567	**222**	303	**10**	**10**	**10**
30A10B70L05K	**0.0**	**0.0**	**0.0**	398	**380**	512	**10**	**10**	**10**
30A10B70L10K	0.5	0.6	**0.4**	4574	3444	**2506**	6	8	**9**
30A10B70L15K	31.4	2.7	**1.8**	7200	5811	**5548**	0	4	**5**
30A15B70L05K	**0.1**	2.1	0.8	**1192**	2829	2092	**9**	7	**9**
30A15B70L10K	21.6	5.5	**4.4**	6028	6414	**6008**	**3**	2	**3**
30A15B70L15K	90.1	9.9	**9.7**	7200	7200	7200	0	0	0
40A05B70L05K[23]	**0.0**	**0.0**	**0.0**	32	**26**	45	**10**	**10**	**10**
40A05B70L10K[28]	**0.0**	**0.0**	**0.0**	468	**195**	221	**10**	**10**	**10**
40A05B70L15K	**0.0**	**0.0**	**0.0**	841	**271**	374	**10**	**10**	**10**
40A10B70L05K	**0.0**	0.5	**0.0**	**407**	2184	1474	**10**	8	**10**
40A10B70L10K	11.2	3.6	**2.7**	6079	5982	**5455**	3	4	**5**
40A10B70L15K	90.0	8.0	**7.2**	6736	**6254**	6329	1	**2**	**2**
40A15B70L05K	**0.1**	1.8	1.1	**2061**	4232	4063	**9**	6	6
40A15B70L10K	80.5	10.0	**9.1**	**6949**	7200	7200	**1**	0	0
40A15B70L15K	100.0	**10.6**	11.9	7200	7200	7200	0	0	0
50A05B70L05K[24]	**0.0**	**0.0**	**0.0**	57	**51**	86	**10**	**10**	**10**
50A05B70L10K[29]	**0.0**	**0.0**	**0.0**	525	**254**	359	**10**	**10**	**10**
50A05B70L15K	**0.0**	**0.0**	**0.0**	1583	**884**	939	**10**	**10**	**10**
50A10B70L05K	**0.0**	1.2	0.5	**628**	2453	2379	**10**	8	8
50A10B70L10K	40.4	4.2	**3.1**	6496	6241	**5799**	**3**	2	**3**
50A10B70L15K	100.0	8.9	**8.3**	7200	7025	**6958**	0	**1**	**1**
50A15B70L05K	**0.3**	2.3	1.5	**2971**	4108	3820	**9**	6	7
50A15B70L10K	90.0	**8.1**	12.1	**6690**	7200	7115	**1**	0	**1**
50A15B70L15K	100.0	**13.9**	16.1	7200	7200	7200	0	0	0

the largest type II instances, $\mathrm{L}_{\mathrm{MCF}}$ outperforms the $\mathrm{L}_{\mathrm{CUT}}$ approaches on the type I instances. There it solves seven more instances to optimality and features considerably smaller computation times on the remaining ones. However, for the largest two instances it fails to provide any non-trivial bounds. $\mathrm{L}_{\mathrm{CUT}}$-d terminates still in the root note, but at least provides reasonable bounds. It is noticeable that $\mathrm{L}_{\mathrm{MCF}}$ proves optimality for 17 out of 20 type II instances and 6 out of 20 type I instances already at the root note, mostly with non-zero LP gap. $\mathrm{L}_{\mathrm{CUT}}$-s performs quite similar in this respect and solves 2 type I and 12 type II instances at the root node. $\mathrm{L}_{\mathrm{CUT}}$-d, on the other hand, achieves this only for a single type II instance. Again, we presume that $\mathrm{L}_{\mathrm{MCF}}$ interacts better with the solver due to having most information available from the beginning. Since these instances feature fractional distances, infeasible path constraints have to be separated to ensure feasibility. The indirectly correlated instances of type II require such cuts only in rare cases. $\mathrm{L}_{\mathrm{CUT}}$-s and $\mathrm{L}_{\mathrm{MCF}}$ solve all but two instances without infeasible path constraints and those two instances with just one cut each. $\mathrm{L}_{\mathrm{CUT}}$-d requires cuts for one additional instance and uses no more than three such cuts. The directly correlated instances of type I require a higher number of infeasible path cuts. This can be explained by the fact that cost reductions can be achieved by exhausting the distance limit. Among the optimally solved instances $\mathrm{L}_{\mathrm{MCF}}$ requires no more than 27 infeasible path cuts while five instances can be solved without them. $\mathrm{L}_{\mathrm{CUT}}$-s solves only few of the type I instances to optimality, three of them without infeasible path cuts and the remaining five with at most 9. $\mathrm{L}_{\mathrm{CUT}}$-d solves four instances without infeasible path cuts and the remaining ones with at most 18. With respect to the instances that terminated due to the time limit the maximum number of added infeasible path cuts is 34 for $\mathrm{L}_{\mathrm{CUT}}$-d, the highest among all algorithms. The node-arc formulation is not competitive and features large gaps even on the smaller instances. In contrast to the results of the LP runs it provides better results on the type II instances, like the layered graph algorithms.

4.5. Evaluation of Algorithm Properties

In the following we evaluate the impact of specific instance properties and algorithmic components on the performance of the introduced approaches.

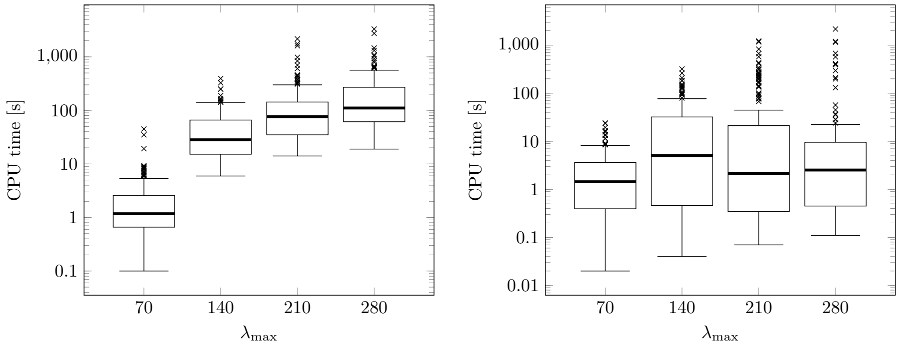
We start by evaluating the sensitivity of our layered graph algorithms to the distance limit. We decided to use the small Cabral instances which could all be solved to optimality with the original distance limit of $\lambda_{\max} = 70$. For our experiments we consider increased distance limits of 140, 210, and 280, while leaving the remaining instance characteristics the same. Increasing the limit further does not make sense because with $\lambda_{\max} \geq 280$ the solutions no longer contain any relays. Figure 5 shows box plots for the respective computation times. We observe a moderate slowdown, leveling off as we converge towards the relay-free scenario. In general, $\mathrm{L}_{\mathrm{CUT}}$-d appears to be slightly more resilient to the parameter change due to the smaller base model. The observed slowdown is to be expected since increasing the distance limit allows connecting the commodities with more complex paths consisting of a higher number of arcs. We conjecture that paths with a higher number of arcs impact most kinds of algorithms alike. In our case the number of layers increases, which leads to larger models. Column generation based approaches, on the other hand, suffer from the additional computational effort for solving the subproblems and

Table 8: Results for the directed Konak instances of type I.

Instance	Gap [%]				CPU time [s]			
	L_{MCF}	L_{CUT}-s	L_{CUT}-d	NA	L_{MCF}	L_{CUT}-s	L_{CUT}-d	NA
040N_05K_30L	**0.0**	**0.0**	**0.0**	**0.0**	**6**	103	120	1204
040N_05K_35L	**0.0**	**0.0**	**0.0**	3.4	**23**	575	329	7200
040N_10K_30L	**0.0**	**0.0**	**0.0**	11.3	**45**	1217	1357	7200
040N_10K_35L	**0.0**	5.2	**0.0**	16.3	**153**	7200	4202	7200
050N_05K_30L	**0.0**	**0.0**	**0.0**	7.3	**1**	23	15	7200
050N_05K_35L	**0.0**	**0.0**	**0.0**	**0.0**	**3**	234	71	5417
050N_10K_30L	**0.0**	7.7	1.9	42.6	**690**	7200	7200	7200
050N_10K_35L	**0.0**	13.5	7.4	31.3	**638**	7200	7200	7200
060N_05K_30L	**0.0**	**0.0**	**0.0**	38.0	**25**	282	213	7200
060N_05K_35L	**0.0**	**0.0**	**0.0**	23.1	**3**	211	280	7200
060N_10K_30L	**0.0**	8.3	7.8	46.7	**392**	7200	7200	7200
060N_10K_35L	**0.0**	7.8	7.9	31.3	**469**	7200	7200	7200
080N_05K_30L	**0.0**	**0.0**	**0.0**	10.9	**12**	2664	1040	7200
080N_05K_35L	**0.0**	5.4	**0.0**	6.3	**24**	7200	5500	7200
080N_10K_30L	**0.6**	6.9	4.9	19.3	7200	7200	7200	7200
080N_10K_35L	**3.4**	65.4	32.5	17.5	7200	7200	7200	7200
160N_05K_30L	**0.0**	76.4	7.7	18.5	**1123**	7200	7200	7200
160N_05K_35L	**6.0**	76.7	10.4	19.8	7200	7200	7200	7200
160N_10K_30L	100.0	78.1	**26.8**	31.7	7200	7200	7200	7200
160N_10K_35L	100.0	77.7	**26.6**	29.0	7200	7200	7200	7200

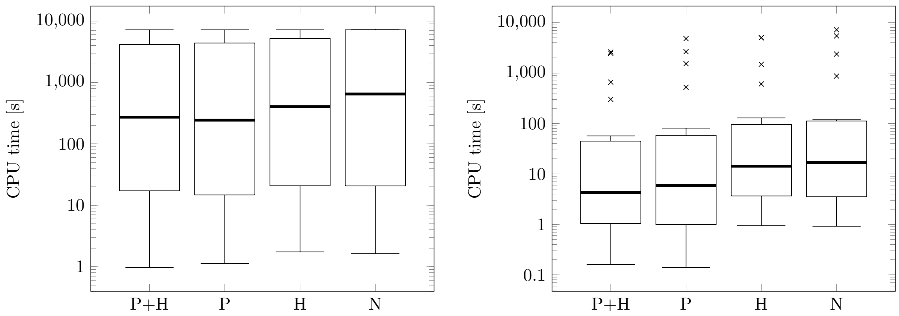
Table 9: Results for the directed Konak instances of type II.

Instance	Gap [%]				CPU time [s]			
	L_{MCF}	L_{CUT}-s	L_{CUT}-d	NA	L_{MCF}	L_{CUT}-s	L_{CUT}-d	NA
040N_05K_30L	**0.0**	**0.0**	**0.0**	**0.0**	**< 1**	2	13	7
040N_05K_35L	**0.0**	**0.0**	**0.0**	**0.0**	**1**	2	2	51
040N_10K_30L	**0.0**	**0.0**	**0.0**	**0.0**	**1**	16	36	352
040N_10K_35L	**0.0**	**0.0**	**0.0**	**0.0**	**5**	30	13	3420
050N_05K_30L	**0.0**	**0.0**	**0.0**	**0.0**	**< 1**	1	1	19
050N_05K_35L	**0.0**	**0.0**	**0.0**	**0.0**	**1**	3	2	4679
050N_10K_30L	**0.0**	**0.0**	**0.0**	49.9	**1**	8	42	7200
050N_10K_35L	**0.0**	**0.0**	**0.0**	62.5	**4**	27	18	7200
060N_05K_30L	**0.0**	**0.0**	**0.0**	44.1	**3**	73	88	7200
060N_05K_35L	**0.0**	**0.0**	**0.0**	**0.0**	**1**	3	3	1039
060N_10K_30L	**0.0**	**0.0**	**0.0**	70.7	**57**	1903	621	7200
060N_10K_35L	**0.0**	**0.0**	**0.0**	62.9	**3**	11	8	7200
080N_05K_30L	**0.0**	**0.0**	**0.0**	34.3	**8**	18	14	7200
080N_05K_35L	**0.0**	**0.0**	**0.0**	44.2	**13**	27	20	7200
080N_10K_30L	**0.0**	**0.0**	**0.0**	42.7	**26**	48	39	7200
080N_10K_35L	**0.0**	**0.0**	**0.0**	59.2	**33**	334	56	7200
160N_05K_30L	**0.0**	**0.0**	**0.0**	73.5	301	1428	**246**	7200
160N_05K_35L	**0.0**	**0.0**	**0.0**	73.4	661	1681	**153**	7200
160N_10K_30L	**0.0**	45.0	**0.0**	71.8	2616	7200	**2041**	7200
160N_10K_35L	**0.0**	50.8	**0.0**	69.3	2466	7200	**620**	7200



(a) L_{MCF} (b) L_{CUT}-d

Figure 5: Sensitivity of L_{MCF} and L_{CUT}-d to increasing values of $\lambda_{\max}$ on the Cabral instances. Both box plots use a logarithmic scale.



(a) Type I (b) Type II

Figure 6: Impact of preprocessing and initial heuristic for L_{MCF} on the Konak instances. Both box plots use a logarithmic scale. The four boxes are labeled as follows: "P+H" uses both preprocessing and initial heuristic, "P" uses only preprocessing, "H" uses only the initial heuristic, "N" uses neither preprocessing nor the initial heuristic.

an increased number of pricing iterations (cf. [18], where we conduct a similar experiment for a B&P algorithm for the NDPR).

In Figure 6 we evaluate the impact of the introduced preprocessing techniques and the initial heuristic on the performance of L_{MCF}. We omit the results for the algorithms based on (L_{CUT}) for brevity as they lead to similar conclusions. The box plots indicate that both techniques are beneficial though preprocessing appears to be a little more important due to providing larger speedups. However, when looking at the detailed results we observe that the initial heuristic is significant for proving optimality. Compared to L_{MCF} with preprocessing and initial heuristic we can solve 5 fewer instances of type I when not using the heuristic, 4 fewer instances of type I when not using preprocessing, and 6 fewer instances of type I as well as 1 fewer instance of type II when using neither preprocessing nor the initial heuristic.

4.6. Managerial Insights

In this section we provide more insights into managerial implications of our study. We focus on the design of translucent optical WDM networks (cf. Section 1) and analyze some of their major key performance indicators (KPIs).

To this end, we consider a set of instances in which cost parameters are set to mimic a realistic setting in which commodities correspond to node pairs that need to communicate with each other, and arc costs are directly proportional to the arc lengths (multiplied by some factor that corresponds to cable costs per unit of distance). The cost parameters in our study are chosen so as to reflect this difference between arc and regenerator costs, the latter ones being often significantly more expensive than the arc costs (cf., e.g., [24]). The Konak instances of type I seem to be particularly relevant from this practical perspective due to their direct correlation between arc distances and arc costs. Because of the limited size of the Konak data set, we have generated further instances using the method proposed by Konak [15].

We sampled a set of five base instances for each $|V| \in \{20, 25, 30, 35, 40\}$ and $|K| \in \{5, 7, 10\}$ with $\lambda_{\max} = 30$. The relay costs were selected according to a normal distribution with $\mu = 200$ and a standard deviation of 25 to model a base cost for purchasing the relay and a slightly varying construction cost. Then, we varied the distance limit $\lambda_{\max} \in \{30, 35, 40, 45\}$ with arc costs set to $c_{ij} = d_{ij} \cdot (1 + (\lambda_{\max} - 30)/100)$, i.e., a cost increase of 1 % per unit increase in the distance limit. This is intended to model a scenario in which transmission over increased distances is only possible when relying on material of higher quality which is in turn more expensive. In total, we consider 300 new instances ranging from 60 to 696 arcs.

In what follows, we provide a cost-effectiveness analysis by comparing the costs for installing regenerators and arcs in the network. In addition, one of the major concerns when designing WDM optical networks is the power consumption. The power consumption is directly correlated with the number of deployed regenerators. We therefore analyze the energy efficiency of obtained solutions in function of the size of the network, the input demand, and the operating range of regenerators.

Cost-effectiveness analysis. The major goal of this analysis is to find out what are the potential benefits of investing into regenerators with a higher operating range. We measure these benefits in terms of the savings of the overall solution costs, when compared to the nominal solution, in which the regenerators of the minimum range ($\lambda_{\max} = 30$ in our setting) are purchased. Figure 7 provides a detailed overview of the overall costs of optimal DNDPR solutions. The results are sorted according to increasing value of $\lambda_{\max}$ (starting with $\lambda_{\max} = 30$ and ranging up to $\lambda_{\max} = 45$). In addition, we separately show the costs needed to install regenerators, and those for installing the links in the network, respectively. For each fixed value of $\lambda_{\max}$, the number of commodities grows from 5 to 10.

We observe that the overall solution cost significantly decreases when regenerators with a higher range are deployed (i.e., increasing the value of $\lambda_{\max}$ from 30 to 45 allows for a reduction of the overall cost of up to 50 %). This can be explained by the fact that a larger distance limit allows to reduce the number of relays. Through the increased freedom in placing the relays they can be shared to a higher degree by using more direct connections. Consequently, also fewer arcs have to be installed but the impact of savings due to economizing regenerators dominates as a consequence of their higher cost. The portion of the costs spent on the regenerators decreases between 20 (5K) and 23 % (10K) when increasing the distance limit from 30 to 45. We also observe that the increase in the total costs is less sensitive to the increase in the number of commodities when regenerators with higher range are deployed. This can be particularly important in realistic scenarios in which the future demand is uncertain but it is expected to increase. In such a scenario, networks with regenerators of higher range are expected to be more resilient to the increasing demand, so that fewer upgrades might be needed in a later stage, once the uncertain demand is revealed.

Hence, our analysis clearly indicates that the decision makers need to carefully explore the cost structure of the underlying solutions and to consider different available technologies before deciding on the type of regenerators to be deployed in the network. Sometimes, investing in more expensive hardware may result in an overall reduction of the capital expenditures (CapEx), as is shown in Figure 7.

Energy efficiency analysis. From the environmental and operational perspective, one of the major concerns when designing WDM networks is the power consumption which is directly correlated with the number of deployed regenerator devices [31]. In the following we study the dependency between the number of deployed regenerators and their operational range.

Figure 8 reports the average number of arcs and regenerators in an optimal solution. The results are grouped according to the values of $\lambda_{\max}$, ranging between 30 and 45, and according to the number of commodities, ranging between 5 and 10.

The obtained results indicate that increasing the number of commodities has a stronger effect on the total size of the network (in terms of the number of arcs) than on the number of deployed regenerators. Furthermore, this effect does not depend on the range of regenerators. For example, by increasing the number of commodities by 100 % (i.e., from 5 to 10), the number of links in the network raises by around 50 %. On the contrary, the number of deployed regenerators remains relatively stable and raises only by a single unit. A much stronger effect on the number of deployed regenerators comes from their range. So, for example, the number of regenerators can be reduced by 50 %, by deploying regenerators of range $\lambda_{\max} = 45$, when compared to the number of regenerators needed with $\lambda_{\max} = 30$.

Overall, when it comes to operational expenditures (OpEx) associated to energy costs, our result shows that significant savings in OpEx can be achieved by purchasing regenerators of higher range. Hence, there is a clear trade-off between CapEx and potential savings in OpEx that has to be carefully examined before final decisions are made.

5. Conclusion

We introduced two exact solution approaches for the directed network design problem with relays (DNDPR) based on layered graphs. The first approach relies on a multi-commodity flow formulation ($\mathrm{L}_{\mathrm{MCF}}$) which is pseudo-polynomial in size, and the second is a branch-and-cut (B&C) approach based on the ($\mathrm{L}_{\mathrm{CUT}}$) model with an exponential number of constraints. Both models provide extremely tight linear programming (LP) bounds on the considered benchmark instances. Fractional distance values are handled efficiently by an exponentially-sized set of infeasible path constraints. We proposed additional

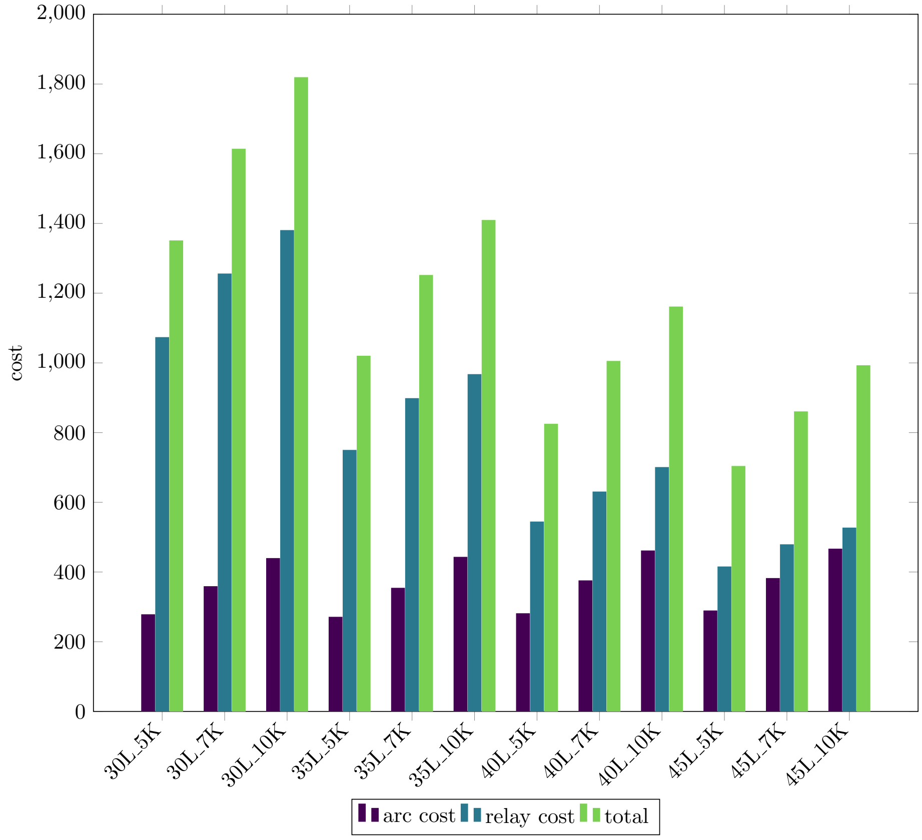


Figure 7: Cost-effectiveness analysis: distribution of the solution costs, separated by the cost for building the network (arc costs) and the cost for deploying relays. Each group of bars considers the mean of 25 instances, 5 for each $|V| \in \{20, 25, 30, 35, 40\}$.

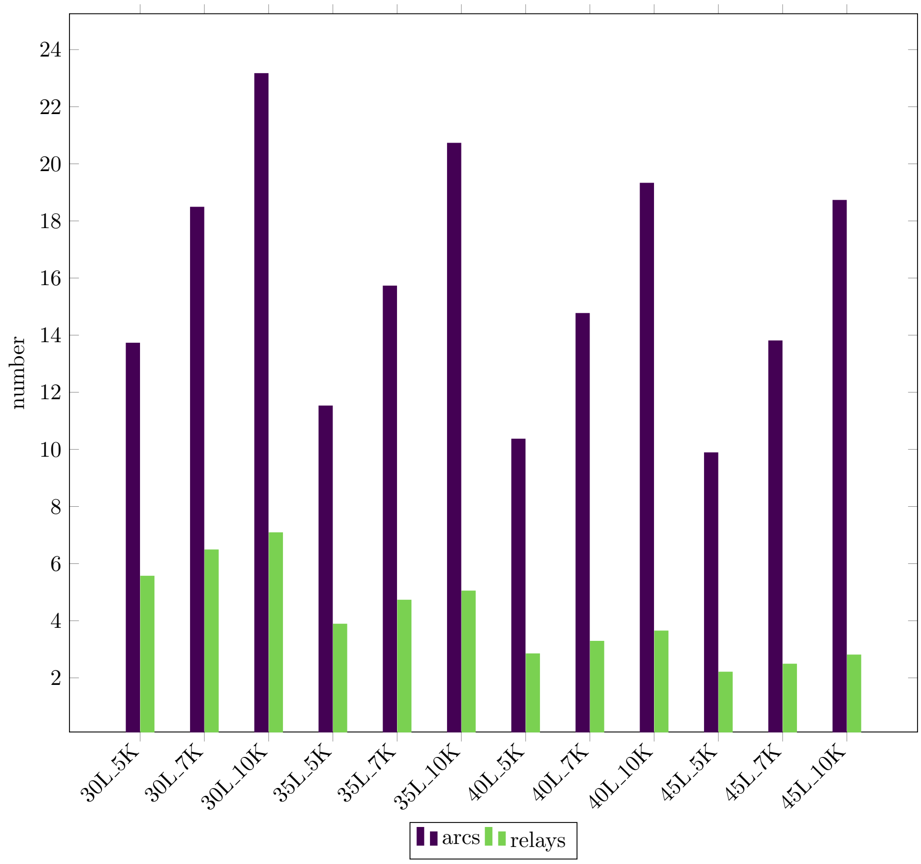


Figure 8: Solution structure of the newly generated Euclidean instances. Each group of bars considers the mean of 25 instances, 5 for each $|V| \in \{20, 25, 30, 35, 40\}$.

valid inequalities for strengthening the ($\mathrm{L_{CUT}}$) formulation and also for breaking symmetries induced by the layered graph structure. We investigated two approaches for adding the strengthening inequalities to the ($\mathrm{L_{CUT}}$) formulation: a static and a dynamic one. For small instances from Cabral et al. [2], representing sparse 4-grid graphs with very few commodities, it turns out that the overhead of adding more inequalities a priori to the model is negligible. On the contrary, for larger and denser instances from Cabral et al. [2] and Konak [15], this overhead does not pay off, and dynamic separation of strengthening inequalities is recommended.

The overall performance of the proposed layered graph approaches based on ($\mathrm{L_{MCF}}$) and ($\mathrm{L_{CUT}}$) is comparable. In general, we observed that the former performs slightly better on sparse graphs with very few commodities, whereas the latter can be used as an alternative for larger and denser graphs and when dealing with a larger number of commodities.

Using the existing node-arc formulation as base line for a comparison with the existing approaches, we showed that our exact approaches are significantly faster than the state of the art from [19].

Our managerial study sends a strong signal to decision makers that, even though the capital expenditures might be higher when acquiring regenerators of a higher range, there are significant long term savings in terms of operating expenditures that need to be taken into account. These savings are notably related to the energy consumption, network maintenance, and the network upgrade costs caused by an increasing future demand.

Acknowledgments

This work was partly supported by the Vienna Science and Technology Fund through project ICT15-014.

References

[1] N. Ascheuer, M. Fischetti, and M. Grötschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming*, 90(3):475–506, 2001.

[2] E. A. Cabral, E. Erkut, G. Laporte, and R. A. Patterson. The network design problem with relays. *European Journal of Operational Research*, 180 (2):834–844, 2007.

[3] S. Chen, I. Ljubić, and S. Raghavan. The regenerator location problem. *Networks*, 55(3):205–220, 2010.

[4] S. Chen, I. Ljubić, and S. Raghavan. The generalized regenerator location problem. *INFORMS Journal on Computing*, 27(2):204–220, 2015.

[5] B. V. Cherkassy and A. V. Goldberg. On implementing push-relabel method for the maximum flow problem. In *Proceedings of the 4^{th} International IPCO Conference on Integer Programming and Combinatorial Optimization*, volume 920 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 1995.

[6] S. Dash, O. Günlük, A. Lodi, and A. Tramontani. A time bucket formulation for the traveling salesman problem with time windows. *INFORMS Journal on Computing*, 24(1):132–147, 2012.

[7] J. De Boeck and B. Fortz. Extended formulation for hop constrained distribution network configuration problems. *European Journal of Operational Research*, 265(2):488–502, 2018.

[8] M. T. Godinho, L. Gouveia, and P. Pesneau. On a time-dependent formulation and an updated classification of ATSP formulations. In A. R. Mahjoub, editor, *Progress in Combinatorial Optimization*, pages 223–254. ISTE-Wiley, 2011.

[9] M.T. Godinho, L. Gouveia, and P. Pesneau. Natural and extended formulations for the time-dependent traveling salesman problem. *Discrete Applied Mathematics*, 164(1):138–153, 2014.

[10] L. Gouveia and M. Ruthmair. Load-dependent and precedence-based models for pickup and delivery problems. *Computers & Operations Research*, 63:56–71, 2015.

[11] 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(1-2):123–148, 2011.

[12] 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.

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

[14] L. Gouveia, M. Leitner, and M. Ruthmair. Layered graph approaches for combinatorial optimization problems. *Computers & Operations Research*, 102:22–38, 2019.

[15] A. Konak. Network design problem with relays: A genetic algorithm with a path-based crossover and a set covering formulation. *European Journal of Operational Research*, 218(3):829–837, 2012.

[16] S. Kulturel-Konak and A. Konak. A local search hybrid genetic algorithm approach to the network design problem with relay stations. In S. Raghavan, B. Golden, and E. Wasil, editors, *Telecommunications Modeling, Policy, and Technology*, volume 44 of *Operations Research/Computer Science Interfaces*, pages 311–324. Springer US, 2008.

[17] G. Laporte and M. M. B. Pascoal. Minimum cost path problems with relays. *Computers & Operations Research*, 38(1):165–173, 2011.

[18] M. Leitner, I. Ljubić, M. Riedler, and M. Ruthmair. Exact approaches for network design problems with relays. *INFORMS Journal on Computing*, To appear, 2018.

[19] X. Li, Y. P. Aneja, and J. Huo. Using branch-and-price approach to solve the directed network design problem with relays. *Omega*, 40(5):672–679, 2012.

[20] X. Li, S. Lin, S. Chen, Y. P. Aneja, P. Tian, and Y. Cui. An iterated metaheuristic for the directed network design problem with relays. *Computers & Industrial Engineering*, 113(Supplement C):35–45, 2017.

[21] S. Lin, X. Li, K. Wei, and C. Yue. A tabu search based metaheuristic for the network design problem with relays. In *Service Systems and Service Management (ICSSSM), 2014 11th International Conference on*, pages 1–6, 2014.

[22] I. Ljubić and S. Gollowitzer. Layered graph approaches to the hop constrained connected facility location problem. *INFORMS Journal on Computing*, 25(2):256–270, 2013.

[23] I. Ljubić, R. Weiskircher, U. Pferschy, G. W. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. *Math. Program.*, 105(2–3):427–449, 2006.

[24] G. B. Mertzios, I. Sau, M. Shalom, and S. Zaks. Placing regenerators in optical networks to satisfy multiple sets of requests. *IEEE/ACM Transactions on Networking*, 20(6):1870–1879, 2012.

[25] I. Nath, M. Chatterjee, and U. Bhattacharya. A survey on regenerator placement problem in translucent optical network. In *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*, pages 408–413, 2014.

[26] J. C. Picard and M. Queyranne. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research*, 26(1):86–110, 1978.

[27] M. Ruthmair and G.R. Raidl. A layered graph model and an adaptive layers framework to solve delay-constrained minimum tree problems. In O. Günlük and G.J. Woeginger, editors, *Proceedings of the 15th Conference on Integer Programming and Combinatorial Optimization (IPCO XV)*, volume 6655 of *LNCS*, pages 376–388. Springer, 2011.

[28] Y. Xiao and A. Konak. A variable neighborhood search for the network design problem with relays. *Journal of Heuristics*, 23(2-3):137–164, 2017.

[29] B. Yıldız and O. E. Karaşan. Regenerator location problem in flexible optical networks. *Operations Research*, 65(3):595–620, 2017.

[30] B. Yıldız, O. E. Karaşan, and H. Yaman. Branch-and-price approaches for the network design problem with relays. *Computers & Operations Research*, 92:155–169, 2018.

[31] Z. Zhu, X. Chen, F. Ji, L. Zhang, F. Farahmand, and J. P. Jue. Energy-efficient translucent optical transport networks with mixed regenerator placement. *Journal of Lightwave Technology*, 30(19):3147–3156, 2012.