Computers and Operations Research 92 (2018) 155–169

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Branch-and-price approaches for the network design problem with relays

Barış Yıldız[a,*], Oya Ekin Karaşan[b], Hande Yaman[b]

[a] Koç University, Department of Industrial Engineering, Sarıyer, İstanbul 34450, Turkey
[b] Bilkent University, Department of Industrial Engineering, Bilkent, Ankara 06800, Turkey

ARTICLE INFO

Article history:
Received 7 August 2016
Revised 2 October 2017
Accepted 5 January 2018
Available online 6 January 2018

Keywords:
Relay
Regenerator location
Routing
Branch-and-Price
Branch-and-Price-and-Cut

ABSTRACT

With different names and characteristics, relays play a crucial role in the design of transportation and telecommunication networks. In transportation networks, relays are strategic locations where exchange of drivers, trucks or mode of transportation takes place. In green transportation, relays become the refuelling/recharging stations extending the reach of alternative fuel vehicles. In telecommunication networks, relays are regenerators extending the reach of optical signals. We study the network design problem with relays and present a multi-commodity flow formulation and a branch-and-price algorithm to solve it. Motivated by the practical applications, we investigate the special case where each demand has a common designated source. In this special case, we can show that there exists an optimal design that is a tree. Using this fact, we replace the multi-commodity flow formulation with a tree formulation enhanced with Steiner cuts. Employing a branch-and-price-and-cut schema on this formulation, we are able to further extend computational efficiency to solve large problem instances.

1. Introduction

Several facilities on transportation and telecommunication networks are relay points. In transportation, relays play an important role as strategically located facilities on the network where the exchange of drivers, trucks and trailers takes place. In multimodal transportation operations, they function as linkages where the mode of transportation is switched (Ali et al., 2002). Relay network architectures are proposed to alleviate the high turnover rate problem for the truck drivers and to lower the operational costs (Ali et al., 2002; Üster and Kewcharoenwong, 2011; Üster and Maheshwari, 2007; Vergara and Root, 2013).

Recent advances in the alternative fuel vehicle (AFV) technologies pose big opportunities and challenges for the transportation sector. The lack of refuelling/recharging infrastructure for AVF is one of the main barriers to harvesting the potential benefits of these novel technologies (Bapna et al., 2002; Melaina and Bremson, 2008; Melaina, 2003; Romm, 2006). Motivated by this urgent need and high installation cost for the refueling/recharging stations, the refueling station location problem has begun to attract significant attention both from academia and industry (Capar et al., 2013; Kim and Kuby, 2012, 2013; Kuby and Lim, 2005, 2007; Kuby et al., 2009; MirHassani and Ebrazi, 2013; Wang and Lin, 2009, 2013; Wang and Wang, 2010; Yıldız et al., 2016). The refueling/recharging stations are essentially relay points that extend the reach of an AFV.

Besides transportation, relays also carry out significant functions in telecommunication networks. Indeed, network design problem with relays (NDR) is introduced by Cabral et al. (2007) and is motivated by a telecommunication network design project in Alberta. In telecommunication networks, signal quality degrades with the distance and relays function as repeaters, regenerators, amplifiers, etc. to enhance the reach of the signal. For example, though capable of carrying the bulk of the data over the internet, optical signals cannot travel more than some given distance before their quality degrades below some threshold level. To overcome this deficiency, regenerators, which are expensive devices, are needed to transmit optical signals over long distances. As such, regenerators are essential for the optical networks and there is a rich literature on the regenerator location problem (Chen et al., 2010, 2015; Jinno et al., 2009; Kewcharoenwong and Üster, 2014; Yang and Ramamurthy, 2005; Yetginer and Karaşan, 2003; Yıldız and Karaşan, 2015, 2017). The kinship between the network design with relays and regenerator placement problems also indicates the affiliation between the hub location problems and NDR. As an important example one can cite the hub covering location problem that is introduced by Campbell (1994). In this study the author considers a reach

* Corresponding author.
E-mail address: baris.yildiz@bilkent.edu.tr (B. Yıldız).

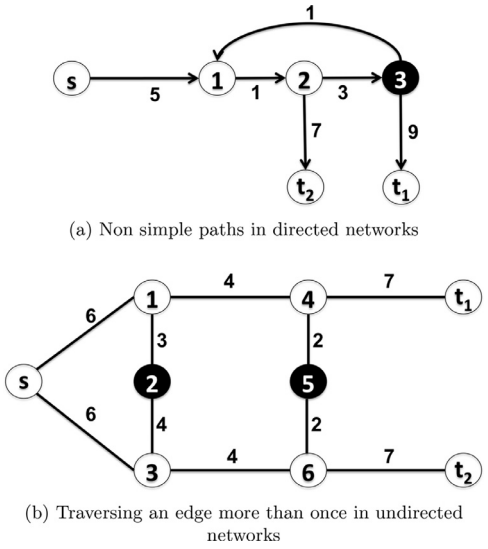
https://doi.org/10.1016/j.cor.2018.01.004
0305-0548/

limit for the commodities transported in the network and requires the non-hub nodes to reach their hub nodes and hub nodes to reach other hubs without violating the reach constraints. In such a setting, hubs essentially function as relays.

Considering the network design with relays, the edge design aspect has been mostly overlooked in the transportation literature even though some sort of edge construction constitutes a crucial part of the overall network design for many practical applications. One such example is the relay network design for freight transportation. To abide by the traffic regulations and alleviate the problem of high turnover rates for the truck drivers, load/truck exchange stations are located in the transportation networks (Ali et al., 2002). When logistics firms consider choosing the best locations for such relay locations they also need to consider which lanes they will operate which is akin to setting up edges in their transportation network. Similarly for the multi-modal/intermodal transportation, inclusion of new modes of transportation or extensions to the current transportation network requires the joint consideration of relay locations and edge designs. The inclusion of alternative fuel vehicles in transportation networks also requires a comprehensive planning of road and refueling infrastructure extensions in a concerted way (Leitner et al., 2015).

A classical network design instance is represented by an undirected graph corresponding to the physical transportation or telecommunication network. A given set of origin-destination (OD) pairs should be routed through this network. There is an associated establishment cost for every link/edge and once an edge is chosen in the design, it can be used in the routing of any number of OD pairs. The aim is to choose a subset of edges in the most cost-effective manner so as to enable the routing of every OD pair. The network design problem with relays shares all these characteristics with a classical network design problem. However, the commodities have also reach limitations and while traversing an edge the reach diminishes by an amount proportional to the length of this edge. Ultimately, relays should be located in order to extend the reach and there is an associated cost to enhancing a node in the graph with the relay capability. The route for each OD pair should be such that two consecutive nodes from the set of terminal and relay nodes on this route should be within the reach limitation. The aim is to choose a subset of edges and a subset of nodes so as to enable the proper communication of all the OD pairs in the most cost-effective manner. Variations of this general form are also studied in the literature. In the directed network design problem with relays (DNDR), the underlying graph is directed. In the single-source network design problem with relays (NDR-S), the origin is the same for each OD pair.

NDR is an interesting problem not only because it is pervasive in real world applications but also because it adds extra challenges to classical network design problems. For instance, in DNDR an optimal solution can contain paths with loops. In Fig. 1(a) a simple example is given to illustrate this fact. There are two OD pairs, (s, t_1) and (s, t_2) and the number given on each edge is the length for that edge. Suppose all edge costs are zero and all node costs are 1. In this example the threshold length is 9 and the only optimal solution is to put a relay on node 3, use the path $s-1-2-3-t_1$ to connect the OD pair (s, t_1) and employ the non-simple path $s-1-2-3-1-2-t_2$ to connect s to t_2. It may also happen that the path for an OD pair in an optimal solution of NDR traverses the same edge more than once. Fig. 1(b) shows such an example. Assuming the same cost structure, the first OD pair (s, t_1) follows the path $s-1-2-1-4-5-4-t_1$ and the second OD pair (s, t_2) follows $s-1-2-1-4-5-6-t_2$. Nodes 2 and 5 are chosen as the relay nodes. Note that this solution is the unique optimal solution and in this solution the edges {1, 2} and {4, 5} are traversed back and forth. The fact that OD pairs may use paths with cycles makes NDR a very challenging network design problem.



(a) Non simple paths in directed networks

(b) Traversing an edge more than once in undirected networks

Fig. 1. Examples of non-simple paths in directed and undirected graphs with threshold set to 9.

Various network design problems are closely related with NDR. One such example is the Steiner tree problem (STP) (Hwang et al., 1992). When all pairwise communications are required for a given subset of nodes and the threshold value is arbitrarily large, NDR reduces to STP. NDR also generalizes the node-weighted Steiner tree problem (NSTP) (Segev, 1987) and the Steiner tree problem with hop constraints (STPH) (Voß, 1999). Note that, for a given set of terminal nodes considering all pairs connectivity, assuming the threshold and edge lengths equal to one and setting relay costs as the weight of the nodes, NDR becomes the same problem as NSTP. Similarly, NDR where the threshold is the allowed number of hops, edge lengths are all equal to one, and relay costs are set to some big number (such as the sum of the costs of all edges) solves STPH. Observe that, the regenerator location problem (RLP) (Chen et al., 2010) is also a special case of NDR where all edge costs are assumed to be zero.

For the solution of NDR, Cabral et al. (2007) present a path-based integer programming formulation and propose a column generation approach to obtain a lower bound. They also obtain feasible solutions by solving the formulation with a restricted set of columns. In addition, they propose four construction heuristics. They use randomly generated grid graphs to test their solution methods on single-source instances. Kulturel-Konak and Konak (2008) propose a multi-commodity flow formulation and present a heuristic algorithm that integrates local search and genetic algorithm. Konak (2012) separates NDR into two problems, one to find paths and the other to locate relays. He presents an efficient heuristic algorithm in which feasible paths for each OD pair are generated by a genetic algorithm and a set covering problem is then solved with these paths to locate relays in the network. Considering a directed graph instead of an undirected one, Li et al. (2012) propose a node-arc and an arc-path formulation for DNDR. To solve the node-path formulation they suggest a branch-and-price algorithm for which the pricing problem is an NP-Hard problem called the minimum cost path problem with relays. They use a slightly modified version of the algorithm by Laporte and Pascoal (2011) to solve it. In the computational experiments, for the sake of simplicity, the authors only consider simple paths in the pricing phase and omit those with cycles. In their computational studies they consider single-source instances. Unfortunately, it is not possible to use an algorithm that solves DNDR directly to solve NDR by simply modifying the problem data. A recent study

by Leitner et al. (2015) considers NDR problems with non-simple paths and presents multi-commodity flow and cut-set formulations to solve it. They propose a branch-and-price and a branch-and-price-and-cut algorithm and present computational results.

In our study we use the notion of a directed virtual network that originates from previous studies Yıldız and Karaşan (2017) and Yıldız et al. (2016). The virtual network is derived from the physical one; it has the same nodes as the physical network but the arcs of the virtual network correspond to simple directed paths in the original graph with lengths not more than the threshold. We propose a multi-commodity flow and a tree formulation on this virtual network to solve NDR and NDR-S problems, respectively. Since the number of arcs on the virtual network grows exponentially with the size of the original network and we have a variable for each of these arcs in both formulations, we propose branch-and-price algorithms to solve our formulations.

We first present our multi-commodity flow based formulation and the branch-and-price algorithm to solve it. In order to expedite the branch-and-price algorithm, we strengthen our formulation with optimality cuts and employ graph transformations. Then, we focus our efforts to the special case NDR-S.

The special attention for NDR-S problem in the literature stems from both practical and theoretical reasons. On the practical side, for many transportation and telecommunication applications there is a special source from which some commodities or signals are disseminated to other terminal nodes as we have in server-client network architectures considered by Cabral et al. (2007). Sometimes these special nodes are hubs where commodities/signals are congregated or exchanged. In a transportation setting the common source could be a depot from which some goods are distributed to their final destinations; it could be the center where entities from terminal nodes are collected and mode of transportation is switched. When the number of OD pairs grows, NDR becomes harder to solve. In solving these large problems, NDR can be decomposed into several NDR-S problems. For instance, one can relax NDR in a Lagrangian manner to obtain single source problems. As such, having an efficient algorithm to solve NDR-S is of interest to devise efficient solution algorithms for the general NDR problem. On the theoretical side, NDR-S also has some quite interesting properties. We can show that there is an optimal design which is a tree both in the original and the virtual networks. Exploiting these properties we propose an improved formulation that uses the cut formulation of Steiner trees. We derive valid inequalities and optimality cuts and implement a branch-and-price-and-cut algorithm to solve our tree formulation that contains an exponential number of constraints involving an exponential number of variables.

2. Definitions and notation

In this section, we provide definitions and notation pertinent throughout the paper. Additional definitions and notation will be listed on a need basis.

Throughout the text $G = (V, E)$ represents our physical network with associated data $l_e \geq 0$ for $e \in E$ corresponding to edge length, $c_e \geq 0$ for $e \in E$ corresponding to edge design cost and $h_i \geq 0$ for $i \in V$ corresponding to relay design cost. K will represent the set of OD pairs. For an OD pair k, the origin and the destination nodes are denoted by $\mathcal{O}(k)$ and $\mathcal{D}(k)$, respectively. The relay free communication range is a given threshold value $d_{max} > 0$. In other words, two nodes of distance at most d_{max} in G can communicate without any relays. We assume without loss of generality that $l_e \leq d_{max}$ for every $e \in E$ since any edge violating this condition can simply be deleted from G.

We define $A = \{(i, j) \cup (j, i) : \{i, j\} \in E\}$ as the arc set induced by the edges in G. For the pair of arcs $a_1 = (i, j), a_2 = (j, i)$ that are induced by the edge $e = \{i, j\}$ we denote $\epsilon(a_1) = \epsilon(a_2) = e$. We assume that $l_a = l_{\epsilon(a)}$ for every $a \in A$.

A directed path is a sequence of arcs $(a_1, \ldots, a_\eta)$ with $a_i = (n_{i-1}, n_i) \in A$ for $i = 1, \ldots, \eta$ and $n_i \in N$ for $i = 0, \ldots, \eta$. It is called simple if it does not repeat any node. The length of a directed path p is denoted by $l(p)$ and it is the sum of the lengths of arcs contained in it, i.e., $l(p) = \sum_{a \in p} l_a$. The formulations that we present next depend on the notion of *path-segments* introduced to the literature in Yıldız and Karaşan (2017) and Yıldız et al. (2016). A *path-segment* p is a directed simple path with total length not more than d_{max}. Its origin and destination nodes are denoted as $o(p)$ and $d(p)$, respectively. Due to the symmetry of our length function, if p is a path-segment, so is p', the directed path obtained by reversing all the arcs on p. We define $\mathcal{P}$ as the set of all path-segments.

A *route* $\pi = (p_1 \ldots, p_\eta)$ is an ordered union of path-segments p_i, $i = 1, \ldots, \eta$ where $d(p_i) = o(p_{i+1})$ for $i = 1, \ldots, \eta - 1$. We call a route *feasible* for an OD pair k, if $o(p_1) = \mathcal{O}(k), d(p_\eta) = \mathcal{D}(k)$ and $d(p_i)$ for $i = 1, \ldots, \eta - 1$ is a relay location, i.e., there exists a relay node at the end of each path-segment except the last one.

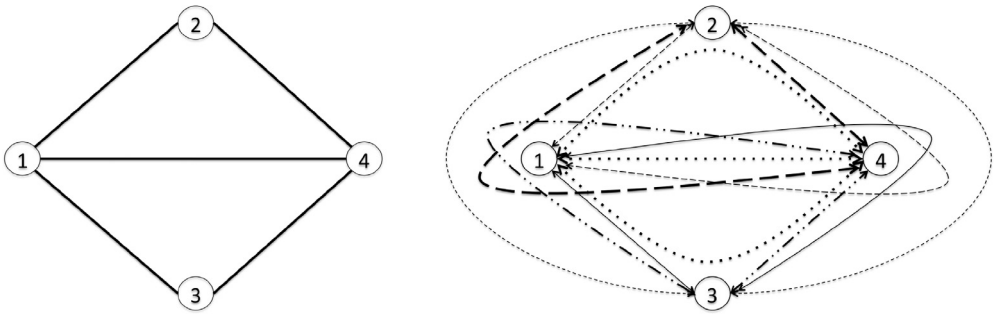
An NDR problem instance is specified by a physical network G with associated data l, c and h, OD pair set K, and a threshold value d_{max}. The aim is to establish a feasible route for every $k \in K$ such that the total of edge design and relay design cost is minimized.

A physical network G and a given threshold value d_{max} induce a *virtual network* $\Gamma = (V, \mathcal{A})$ where $\mathcal{A} = \{(i, j, p) : p \in \mathcal{P}, o(p) = i, d(p) = j\}$. By this construction, a feasible route in the physical network corresponds to a directed path in the virtual network. Conversely, any directed path from $\mathcal{O}(k)$ to $\mathcal{D}(k)$ in Γ corresponds to a feasible route for $k \in K$ provided that every intermediate node on this path is equipped with the relay property. Our formulations rely on the correspondence between the physical and the virtual networks. Fig. 2 presents an example for a virtual graph. In this example we consider a quite simple input graph with four nodes and five edges with unit lengths in Fig. 2(a). Considering a d_{max} value of two, we obtain the virtual network depicted in Fig. 2(b). In Fig. 2(b) the parallel arcs between the same node pairs are drawn with the same style and they are depicted in a way to show the physical path-segments they represent. For example the double headed dashed lines on the rightmost and leftmost boundaries of the figure, represent the parallel virtual network arcs between nodes 2 and 3 corresponding to the path-segments ((2, 4), (4, 3)), ((2, 1), (1, 3)), ((3, 4), (4, 2)) and ((3, 1), (1, 2)). As we see in Fig. 2, the induced virtual graph for a quite simple input graph and moderate d_{max} value, can get quite dense. If we merge the parallel arcs in the virtual network into one edge, we obtain the communication graph, which is introduced by Chen et al. (2010) to solve the RLP. The communication graph has been used in Leitner et al. (2015) to derive an alternative exact approach to the NDR.

A solution for an NDR problem in the physical network $G = (V, E)$ is a pair $\langle R, T \rangle$ where $R \subseteq V$ is the set of relay locations and $T \subseteq E$ is the set of edges included in the design. The relay locations and the edges in the resulting design enable feasible routes for each OD pair $k \in K$. A solution for an NDR problem $\langle R, T \rangle$ induces a directed subgraph of the virtual network $\Gamma = (V, \mathcal{A})$ on which for every $k \in K$, there exists a directed path from $\mathcal{O}(k)$ to $\mathcal{D}(k)$ with every intermediate node in R.

3. Solving NDR

In this section, we present a branch-and-price algorithm to solve the general problem NDR. We first list some structural properties satisfied by certain optimal solutions. Then we provide our multi-commodity flow formulation strengthened by optimality cuts based on these properties and detail the algorithm.



(a) Physical network with unit edge lengths.

(b) Virtual network for $d_{max} = 2$. The matching line styles are parallel virtual arcs with the same source and destination nodes

Fig. 2. Virtual network example.

3.1. Properties of NDR solutions

The following result plays a key role in strengthening our formulations.

Lemma 1. *There exists an optimal solution $\langle R^*, T^*\rangle$ to NDR such that for all $k \in K$, there exists a feasible route π^k using edges $T^k \subseteq T^*$ for which T^k is a tree.*

Proof. Consider an arbitrary $k \in K$. Among all feasible routes from $\mathcal{O}(k)$ to $\mathcal{D}(k)$ pick one using the fewest number of edges T^k from T^*. Assume T^k has a cycle. Since each path-segment in π^k is simple, in order for T^k to create a cycle, there should exist two distinct path-segments $\hat{p}$ and $\hat{q}$ of π^k sharing a particular node, say i. Without loss of generality assume that $\hat{p} = (p_1, p_2)$ comes before $\hat{q} = (q_1, q_2)$ where p_1, p_2, q_1, q_2 are path-segments with $d(p_1) = d(q_1) = o(p_2) = o(q_2) = i$. Since the route traverses a directed cycle, $q_1' \neq p_2$.

Since both $\hat{p}$ and $\hat{q}$ are path-segments, we have

$$l(p_1) + l(p_2) \leq d_{max} \quad \text{and} \quad l(q_1) + l(q_2) \leq d_{max} \tag{1}$$

We must have

$$l(p_1) + l(q_1) > d_{max}, \tag{2}$$

since otherwise, we can replace the portion of the $\mathcal{O}(k)$–$\mathcal{D}(k)$ path from $o(p_1)$ to $d(q_2)$ with (p_1, q_1', q_1, q_2) removing all edges on p_2, the edges on the intermediate segments between $\hat{p}$ and $\hat{q}$ and thus violating our assumption of π^k using the minimum number of edges from T^*. Note that the case $p_2 = \emptyset$ implies node $i \in R^*$ for which (p_1, q_2) is a feasible route from $o(p_1)$ to $d(q_2)$ again contradicting the minimum usage of edges.

By a similar line of reasoning, we must have

$$l(p_2) + l(q_2) > d_{max}, \tag{3}$$

However, inequalities (1)–(3) cannot simultaneously hold. □

Lemma 1 has several obvious implications.

Corollary 1. *There exists an optimal solution $\langle R^*, T^*\rangle$ to NDR such that for all $k \in K$, there exists a feasible route π^k with the following properties:*

(i) *Each node $i \in V$ appears at most twice on π^k,*
(ii) *Each arc $a \in A$ appears at most once on π^k,*
(iii) *Non-consecutive path-segments on π^k are disjoint,*
(iv) *If two consecutive path-segments say $\hat{p} = (p_1, p_2)$ and $\hat{q} = (q_1, q_2)$ on π^k with $d(p_2) = o(q_1)$ share an intermediate node $i \in V$, with $d(p_1) = d(q_1) = o(p_2) = o(q_2) = i$, then $p_2 = q_1'$.*

We would like to note that Lemma 1 does not hold for DNDR as the example in Fig. 1(a) attests to.

3.2. Multi-commodity flow formulation (MCF)

Considering OD pairs as separate commodities and routing each one from its source to the destination by establishing a feasible route, i.e., a directed path in the virtual layer, gives a multi-commodity flow formulation for NDR. Similar multi-commodity flow formulations are used in Yıldız and Karaşan (2017) and Yıldız et al. (2016) for different application settings.

We define the following decision variables for this formulation:

$$r_i = \begin{cases} 1, & \text{if node } i \in V \text{ is a relay point,} \\ 0, & \text{otherwise,} \end{cases}$$

$$w_e = \begin{cases} 1, & \text{if edge } e \in E \text{ is used in the network design,} \\ 0, & \text{otherwise,} \end{cases}$$

$$v_p^k = \begin{cases} 1, & \text{if path-segment } p \in \mathcal{P} \text{ is used by the OD pair } k \in K, \\ 0, & \text{otherwise.} \end{cases}$$

We name these variables as *relay, edge* and *flow variables*, respectively. We use the following additional notation. For node $i \in V$, $\delta^+(i)$ and $\delta^-(i)$ are the sets of path-segments that start and end at node i, respectively. For brevity of notation, we use $v^k(\mathcal{P}') = \sum_{p \in \mathcal{P}'} v_p^k$ for $\mathcal{P}' \subseteq \mathcal{P}$ and $k \in K$.

The multi-commodity formulation that we refer to as MCF is:

$$\min \sum_{i \in V} h_i r_i + \sum_{e \in E} c_e w_e \tag{4}$$

$$\text{s.t. } v^k(\delta^+(i)) - v^k(\delta^-(i)) = \begin{cases} 1 & \text{if } i = \mathcal{O}(k) \\ -1 & \text{if } i = \mathcal{D}(k) \\ 0 & \text{otherwise} \end{cases} \quad k \in K, i \in V, \tag{5}$$

$$v^k(\delta^-(i)) \leq r_i \qquad k \in K, i \in V \setminus \mathcal{D}(k), \tag{6}$$

$$v_p^k \leq w_{\epsilon(a)} \qquad k \in K, p \in \mathcal{P}, a \in p, \tag{7}$$

$$r_i \in \{0, 1\} \qquad i \in V, \tag{8}$$

$$w_e \in \{0, 1\} \qquad e \in E, \tag{9}$$

$$v_p^k \geq 0 \qquad k \in K, p \in \mathcal{P}. \tag{10}$$

The objective is to minimize the total relay placement and edge design cost. Constraints (5) are flow balance constraints to route

each commodity from its origin to its destination by a concatenation of path-segments. Constraints (6) ensure that path-segments end at either a relay node or the destination node. Constraints (7) are the edge design constraints that enforce that an edge used by an active path-segment should be included in the solution. Finally, constraints (8)–(10) are the domain restrictions for the variables. Although all decision variables are assumed to be binary, we can relax the integrality requirement for the flow variables since once relay and edge design variables are fixed, flows on different routes for an OD pair can be consolidated on just one arbitrarily chosen path (among those that carry flow) without any change in the objective function value.

Lemma 1 enables us in strengthening constraints (7) and decreasing the size of the formulation. In particular,

Corollary 2. *Let $k \in K$ and $a \in A$. The inequality*

$$\sum_{p \in \mathcal{P}: a \in p} v_p^k \leq w_{\epsilon(a)} \tag{11}$$

is an optimality cut for (4)–(10).

Proof. Corollary 1(ii) states that MCF has an optimal solution where two path-segments employed by the same OD pair do not share an arc. Thus we can sum constraints (7) over all path-segments without losing optimality. □

During our experimentations, we replaced constraints (7) with constraints (11) in model MCF.

Note that even though Lemma 1 is not valid for a DNDR instance, our original MCP model (4)–(10) can solve the directed network design problem with relays by simply replacing (4) with:

$$\min \sum_{i \in V} h_i r_i + \sum_{a \in A} c_a w_a$$

and (7) with:

$$v_p^k \leq w_a \quad \forall k \in K, p \in \mathcal{P}, a \in p$$

where c_a is the cost of including an arc $a \in A$ and w_a is the binary decision variable for the arc design.

3.3. Solving MCF

Since the number of flow variables grows exponentially with the network size, for realistic size problems it is not practical to generate all these variables a-priori and solve MCF directly as a mixed integer program. To overcome this problem we employ a column generation procedure to solve the linear relaxation of MCF that we denote as MCF-LP. To solve MCF we devise a branch-and-price algorithm.

To solve MCF-LP, we start with a restriction that contains a subset of flow variables and add the remaining variables when needed. The problem that determines the flow variables to add to achieve optimality is the pricing problem. Below, we explain how we solve the pricing problem, how we choose the initial subset of flow variables, the branching rule, the search rule and other implementation details.

Pricing problem

After solving the restricted MCF-LP that contains only a subset of the flow variables, we look for columns (flow variables) with negative reduced costs. We solve the pricing problem to find such a column or conclude that none exists. Let α_i^k, $-\beta_i^k$ and $-\gamma_a^k$ denote the dual variables associated with the constraints (5),(6), and (11), respectively. Then the reduced cost for a flow variable v_p^k can be calculated as follows:

$$\bar{v}_p^k = \begin{cases} \alpha_{d(p)}^k - \alpha_{o(p)}^k + \sum_{a \in p} \gamma_a^k, & \text{if } d(p) = \mathcal{D}(k) \\ \alpha_{d(p)}^k - \alpha_{o(p)}^k + \sum_{a \in p} \gamma_a^k + \beta_{d(p)}^k, & \text{otherwise.} \end{cases} \tag{12}$$

The pricing problem is to check for each $k \in K$ and $(o(p), d(p), p) \in \mathcal{A}$ whether $\bar{v}_p^k$ is negative. Let $A^c = \{(i, j) : (i, j, p) \in \mathcal{A}$ for some path-segment $p\}$ denote the set of *plausible* node pairs. We would like to recall that given a directed graph with costs and resources associated with each arc, the constrained shortest path problem (CSP) (Garey and Johnson, 1979) seeks a minimum cost path from a given source node to a given destination node with a side constraint on the total resource of the path. Since we have $\gamma_a^k \geq 0$ for all $a \in A$, $k \in K$ and we look for a negative reduced cost column, the pricing problem for OD pair $k \in K$ and a plausible pair $(i, j) \in A^c$ is actually a CSP instance from node i to node j on a graph $G^k = (V, A)$ in which the resource is $\tau_a = l_a$ and the cost is $\bar{c}_a = \gamma_a^k$ for all $a \in A$ and the resource limit is $\tau = d_{max}$.

In this approach, one needs to solve $O(|K||A^c|)$ CSP problems to solve the pricing problem. This can in fact be done in a more efficient way by solving $O(|K|)$ CSP problems as follows. Consider the pricing graph $\widetilde{G}^k = (\widetilde{V}, \widetilde{A})$, where:

1. $\widetilde{V} = V \cup \{\tilde{s}\} \cup \{\tilde{t}\}$,
2. $\widetilde{A} = A \cup \{(\tilde{s}, i) : i \in V\} \cup \{(i, \tilde{t}) : i \in V\}$,
3. for arc $a \in A$, the cost is $\bar{c}_a = \gamma_a^k$ and the resource is $\tau_a = l_a$,
4. for arc $a \in \widetilde{A} \setminus A$, the resource $\tau_a = 0$,
5. for arc $(\tilde{s}, i), i \in V$ the cost is $\bar{c}_{\tilde{s},i} = -\alpha_i^k$,
6. for arc $(i, \tilde{t}), i \in V$, the cost is $\bar{c}_{i,\tilde{t}} = \alpha_i^k$ if $\mathcal{D}(k) = i$ and $\bar{c}_{i,\tilde{t}} = \alpha_i^k + \beta_i^k$ otherwise.

Fig. 3 illustrates a small example for the generation of the pricing graph. For the input graph shown in Fig. 3(a) and the dual variable values α, β and γ, the related pricing graph for the OD pair $k = (i, m)$ is depicted in Fig. 3(b). The following lemma establishes the validity of the proposed solution approach for the pricing problem.

Lemma 2. *Let $k \in K$ and $\widetilde{G}^k = (\widetilde{V}, \widetilde{A})$, $\bar{c}$ and τ be as defined above. There exists a negative reduced cost column for commodity k if and only if there exists a negative cost path from $\tilde{s}$ to $\tilde{t}$ in $\widetilde{G}^k$ with resource consumption not more than the threshold d_{max}.*

Proof. It is enough to observe that for each path $\tilde{p} = ((\tilde{s}, o(p)), p, (d(p), \tilde{t}))$ from $\tilde{s}$ to $\tilde{t}$ in $\widetilde{G}^k$, the cost is equal to $\bar{c}_{\tilde{s},o(p)} + \sum_{a \in p} \bar{c}_a + \bar{c}_{d(p),\tilde{t}}$, which is equal to $-\alpha_{o(p)}^k + \sum_{a \in p} \gamma_a^k + \alpha_{d(p)}^k$ if $d(p) = \mathcal{D}(k)$ and to $-\alpha_{o(p)}^k + \sum_{a \in p} \gamma_a^k + \alpha_{d(p)}^k + \beta_{d(p)}^k$, otherwise. In both cases, these quantities give the reduced cost of the column associated with path segment p and commodity k. In addition, the resource constraint ensures that the path segment has length at most d_{max}. □

Observe that as an immediate result of Lemma 2, we can safely terminate the column generation procedure and declare the current best solution as an optimal solution for MCF-LP, if CSP solutions return a path with a nonnegative cost for all $k \in K$ in the respective pricing graphs.

Since the CSP is NP-Hard (Garey and Johnson, 1979) and we need to solve $|K|$ CSPs at each column generation iteration, to save time we first try to find the negative reduced cost variables by a heuristic approach and resort to the exact solution methodology if the heuristic method fails. For that purpose we use the heuristic algorithm HS_q used also in Yildiz and Karasan (2014). For each plausible pair $(i, j) \in A^c$, HS_q stores the first q shortest paths from i to j in G and checks them first to detect a negative reduced cost column. Since HS_q requires to solve a q-shortest path problem once in the beginning of the branch-and-price algorithm and there are very efficient algorithms that solve it on graphs with nonnegative lengths (de Azevedo et al., 1994), this heuristic can drastically improve the performance of the pricing phase in exchange of a quite moderate increase in computer memory cost.

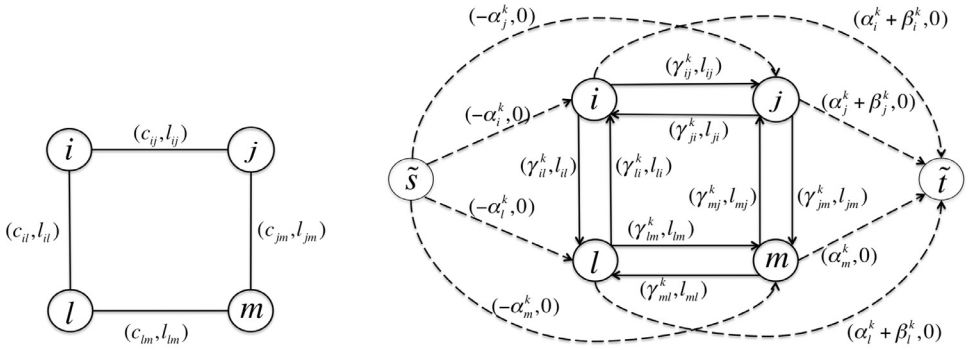


Fig. 3. Pricing graph corresponding to the detection of negative reduced cost variables for the OD pair (i, m). The artificial arcs are depicted with dashed lines.

Initial variable pool:

Defining variables for path-segments instead of whole paths diverts from the widely used path based formulations for which column generation techniques have been applied very successfully for a wide range of problems (Lübbecke and Desrosiers, 2005). Path-segments as variables require a more careful approach to determine the initial variable pool of flow variables (Yıldız and Karaşan, 2017). Let p_{ij} be the trivial path-segment that contains only the arc $(i, j) \in A$. We can define the initial variable pool as $V_0 = \{v^k_{p_{ij}} : k \in K, (i, j) \in A\}$. Note that, a solution for the MCF-LP, considering only the flow variables in V_0, contains enough information to derive all the needed dual variable values to properly construct the pricing problem.

Branching and search:

One of the key points in developing a branch-and-price algorithm is to define a branching rule that eliminates non-integral solutions while preserving the special structure of the pricing problem (Barnhart et al., 2000). In our case, since the flow variables are not required to be integral, standard branching rules can be applied to non-integral relay and edge variables without any change in the structure of the pricing problem. In our implementations, based on the results we get in preliminary tests we give priority to relay variables and branch on edge variables only when all relay variables are integral. For the search of the branch-and-bound tree we use depth first strategy that speeds up the re-optimization after adding a branching cut.

Heuristic:

For any branch-and-bound algorithm, obtaining a good feasible solution at the beginning can be very useful to speed up the solution procedure. In order to get such a good solution we do the following. At the root node, at each iteration of the column generation phase we check whether the solution for the current restricted MCF-LP is integral or not. If it is integral and has a lower cost than the best integer solution found so far we store it as the best integer solution. If no such solution is found we solve MCF as an integer program with only the columns generated at the root node. Our numerical experiments have shown that the integral solution obtained by this procedure is often of high quality and can speed up the branch-and-price algorithm significantly. Since we generate a significant number of new columns during our search in the branch-and-bound tree, it is also useful to repeat this solution procedure at some nodes in the branch-and-bound phase to obtain a better heuristic solution. In our implementation we resort to this heuristic every 300 nodes if more than 500 columns have been added after our last heuristic attempt. We give a time limit of 1800 s to our mixed integer linear program (MIP) solver and consider the best incumbent solution if it reaches to this time limit without a proof of optimality.

Variable pool management:

In most column generation/branch-and-price implementations managing the variable pool is crucial. Our preliminary studies have indicated that keeping all the columns in the model and not taking them out based on the frequency they appear in the optimal bases gave the best performance so we abide by this strategy during all our computational experiments.

We present the results of our computational experiments with this branch-and-price algorithm in Section 5.

4. Solving NDR-S

Obviously our MCF formulation can be used directly to solve NDR-S. However, as a general solution approach it ignores the special structure of the NDR-S problem, which can be exploited to improve computational efficiency. Facilitated by this special problem structure, we present a more efficient solution approach for NDR-S in this section.

4.1. Properties of NDR-S solutions

Let D be the set of *terminal* nodes, i.e., $D = \{i \in V : \mathcal{O}(k) = i \text{ or } \mathcal{D}(k) = i, k \in K\}$ for a given NDR instance. In the variation NDR-S, there exists a special source node $s \in V$ such that $s = \mathcal{O}(k)$ for all $k \in K$.

NDR-S requires that node s reaches all terminal nodes in $D \backslash \{s\}$. This calls for a solution that induces a connected graph in the physical network and a rooted connected directed graph in the virtual network (ignoring isolated nodes and edges with zero costs). In fact, we can even show that there exists an optimal solution inducing a tree in the physical layer and a rooted tree in the virtual layer.

Theorem 1. *Given an NDR-S instance, there exists an optimal solution $\langle R^*, T^* \rangle$ such that T^* is a tree.*

Proof. Let $T^i \subseteq T^*$ for each $i \in K$ be the set of edges used in the route π^i from s to i. By Lemma 1, we may assume without loss of generality that T^i is a tree for each $i \in K$. Now we shall constructively show that we can add these trees one by one without creating cycles and maintaining optimality. Assume $\widetilde{T} = \cup_{i \le m} T^i$ for some $1 \le m < |K|$ is free of cycles. We shall show that $\widetilde{T} \cup T^{m+1}$ can be made cycle free without losing optimality.

Assume to the contrary that when the edge sets in $\widetilde{T}$ and T^{m+1} are combined, we create a cycle. In other words, visualizing $\widetilde{T}$ rooted at s, there exist two nodes u and v such that π^{m+1} touches these two nodes but does not use the unique path from u to v in

$\widetilde{T}$. Among all potential such node pairs u and v, pick one for which the unique paths from u to v in $\widetilde{T}$ and T^{m+1} only intersect at these nodes. Without loss of generality we assume that the unique $s-u$ paths in $\widetilde{T}$ and T^{m+1} are identical (possibly void corresponding to the case $s=u$).

We know that each node in $\widetilde{T}$ (similarly in T^{m+1}), has a feasible route from s otherwise we can remove it along with all its incident edges without violating feasibility. Consider the subtree $\widetilde{T}_{uv}$ consisting of the unique $u-v$ path in $\widetilde{T}$ as well as any subtree hanging from the nodes on this path. Now, there may be some nodes on $\widetilde{T}_{uv}$ that are reachable from s only through routes touching node v. Their routes first have to traverse from u to v, possibly get regenerated through a subtree rooted at v and then come back. Note that since v and all its nodes in its subtree have this property, such nodes do exist. Let W be the set of all such nodes along with all the nodes on their unique subtrees hanging from $\widetilde{T}_{uv}$ and $S=\widetilde{T}_{uv}\setminus W$. In other words, if node i is a node on the unique $u-v$ path, and if j is a node in its subtree which is reachable from s through v only, any node on the subtree rooted at i (including i itself) will belong to set W. By our construction, (S, W) properly partitions the node set of $\widetilde{T}_{uv}$ and there exists a unique edge, say $\tilde{e}\in\widetilde{T}_{uv}$, with one endpoint in S and the other endpoint in W.

We now know that s has a route to each $w\in W$ visiting node v. Let $\pi^w=(\pi_1^w,\pi_2^w)$ denote this route and assume v appears once on π_1^w as a terminal node. Let $p(w)$ be the portion of the last path-segment on π_1^w terminating at node v. Similarly, for $\mathcal{D}(m+1)$, consider the route π^{m+1} which passes through v and let q be the portion of the path-segment on this route while visiting v for the first time. Note that $l_q<l_{p(w)}$ for each $w\in W$ otherwise it would be possible for s to reach $\mathcal{D}(m+1)$ without using the edges on the unique path from u to v on T^{m+1}. But then, s can reach each $w\in W$ by first reaching v following the route π^{m+1} on T^{m+1} and then following the route π_2^w from v to w on $\widetilde{T}_{uv}$. So if edge $\tilde{e}$ is deleted no node will lose its reachability from node s and one cycle in $\widetilde{T}\cup T^{m+1}$ can be removed. One can repeat these arguments to eliminate all cycles. □

Theorem 2. *Given an NDR-S instance, there exists an optimal solution $\langle R^*, T^*\rangle$ such that the union of path-segments used in the design form a rooted tree in the virtual network Γ with root s spanning nodes in $R^*\cup D$.*

Proof. For each $i\in D\setminus\{s\}$, let π^i be a feasible route from s to i. Consider the subgraph of the virtual network Γ induced by the path-segments used by the routes π^i, $i\in D\setminus\{s\}$. Let $\overline{\Gamma}=(\overline{V},\overline{\mathcal{A}})$ be this subgraph. By definition of feasible routes, $\overline{V}=D\cup R^*$ so $\overline{\Gamma}$ will be the desired rooted tree if no node in $\overline{V}$ has two incoming arcs from $\overline{\mathcal{A}}$. Assume to the contrary that node $i\in\overline{V}$ has two incoming arcs a_1 and a_2 from $\overline{\mathcal{A}}$. Then node i is necessarily in R^* and any one of the arcs a_1 or a_2 could be removed without violating feasibility. □

For the sake of generality, the proofs for Theorems 1 and 2 do not assume strictly positive edge costs. However, if this is the case, then no optimal solution would induce a cycle in the physical network. In other words, we can say that:

Corollary 3. *Given an NDR-S instance, if $c_e>0$ for each $e\in E$, then every optimal solution induces a tree in the physical network and a rooted tree in the virtual network.*

4.2. Tree formulation (TF)

Now we are ready to present our Tree Formulation (TF) for NDR-S. Using the results of Theorem 2, we can provide a single-commodity formulation. TF uses the same relay and edge design variables of the MCF formulation but considers flow variables without commodity superscripts. These variables are called *single-flow variables*:

$$\nu_p=\begin{cases}1, & \text{if path-segment } p\in\mathcal{P} \text{ is established,}\\ 0, & \text{otherwise.}\end{cases}$$

Given $S\subset V$, $\mathcal{P}(S)=\{p\in\mathcal{P}: o(p),d(p)\in S\}$ denotes the set of all path-segments with both endpoints in set S. For $p\in\mathcal{P}$, let $\mathcal{E}(p)=\{e\in E: e=\epsilon(a) \text{ for } a\in p\}$. We let $\nu(P')=\sum_{p\in P'}\nu_p$, for $P'\subseteq P$ and $r(V')=\sum_{i\in V'}r_i$, for $V'\subseteq V$. The tree formulation (TF) is given as :

$$\min \sum_{i\in V}h_i r_i+\sum_{e\in E}c_e w_e \tag{13}$$

$$\text{s.t. } \nu(\mathcal{P})=|D|+r(V\setminus D)-1, \tag{14}$$

$$\nu(\mathcal{P}(S))\le|S\cap D|+r(S\setminus D)-1 \qquad S\subset V: S\cap D\neq\emptyset, \tag{15}$$

$$\nu(\mathcal{P}(S))\le r(S\setminus\{j\}) \qquad S\subset V: S\cap D=\emptyset, j\in S, \tag{16}$$

$$\nu(\delta^-(i))=\begin{cases}1 & \text{if } i\in D\\ r_i & \text{if } i\in V\setminus D\end{cases} \qquad \forall i\in V\setminus\{s\}, \tag{17}$$

$$r_i\ge\begin{cases}\sum_{p\in\mathcal{P}:(i,j,p)\in\mathcal{A}}\nu_p+\sum_{p\in\mathcal{P}:(j,i,p)\in\mathcal{A}}\nu_p & \text{if } i\notin D\\ \sum_{p\in\mathcal{P}:(i,j,p)\in\mathcal{A}}\nu_p & \text{otherwise}\end{cases} \quad (i,j)\in A^c, \tag{18}$$

$$\sum_{p\in\delta^-(i):e\in\mathcal{E}(p)}\nu_p\le w_e \qquad i\in V, e\in E, \tag{19}$$

$$\nu_p\ge 0 \qquad p\in\mathcal{P}, \tag{20}$$

$$r_i\in\{0,1\} \qquad i\in V, \tag{21}$$

$$w_e\in\{0,1\} \qquad e\in E. \tag{22}$$

Similar to the MCF formulation, the objective function is to minimize the network design cost. Note that, by Theorem 2, we know that there exists an optimal solution that induces a rooted tree in the virtual network spanning all terminal nodes and relay nodes. Constraints (14)–(16) ensure that the union of path-segments form a tree in the virtual network. Constraints (17) force that all the terminal nodes are visited in this rooted tree and that a non-terminal node is visited if and only if it is enhanced as a relay. Recall that $\mathcal{A}$ is the virtual network arc set and A^c is the set of plausible node pairs. Constraints (18) are regeneration constraints. They make sure that all the internal nodes of the virtual rooted tree are enhanced with relay capabilities. Note that, these constraints are stronger for the non-terminal nodes for which there can be no incoming or outgoing arcs if they are not chosen as relays. Since all routes start in s, for the sake of simplicity and without loss of generality, we assume that $h_s=0$ and allow for source node to be chosen as a relay point. Constraints (19) are the edge design constraints enforcing all the edges used by established path-segments to be included in the design. Note that instead of writing this constraint separately for each path-segment we can sum over all path-segments sharing the same destination and obtain a stronger inequality. The validity of this inequality is also guaranteed by constraints (17). Finally (20)–(22) are the variable domain restrictions.

4.3. Strengthening TF

4.3.1. Connectivity cuts for the relays

We seek for an optimal solution that is a rooted tree at node s in the virtual network having relay locations as internal nodes and terminal nodes as leaves. Any terminal or relay location that is not in the relay free communication range of s needs to reach to a relay location within its range. Using this knowledge, we get a tighter formulation by adding the following cuts to TF:

$$\sum_{j:(i,j)\in A^c} r_j \geq \begin{cases} 1 & \text{if } i \in D \\ r_i & \text{if } i \in V \setminus D \end{cases} \qquad i \in V \setminus \{s\} : (s,i) \notin A^c. \tag{23}$$

4.3.2. Steiner tree cuts in the physical network

TF seeks an optimal solution where the arcs of the virtual network corresponding to single-flow variables induce a rooted tree and where the design cost of the edge and relay variables is minimum. The correctness of this formulation is ensured by Theorem 2. Due to Theorem 1, we can also restrict the design to induce a tree in the physical network. Since all terminal nodes need to be in this tree for connectivity requirements, this tree is actually a Steiner tree that spans terminal nodes in D. Using this fact, we can generate further optimality cuts to eliminate some fractional solutions.

We obtain a strengthened tree formulation (STF) by adding the relay connectivity cuts (23) and the following Steiner tree constraints to TF:

$$w(E) = u(V) - 1, \tag{24}$$

$$w(E(S)) \leq u(S \setminus \{j\}) \qquad S \subset V, j \in S, \tag{25}$$

$$u_i \geq r_i \qquad i \in V \setminus D, \tag{26}$$

$$u_i = 1 \qquad i \in D, \tag{27}$$

where we define the *Steiner variables* as:

$$u_i = \begin{cases} 1, & \text{if node } i \in V \text{ is included in the design,} \\ 0, & \text{otherwise.} \end{cases}$$

Constraints (24) and (25) are the well known Steiner tree constraints where for $S \subset V$, $E(S) = \{\{i,j\} \in E : i,j \in S\}$. Since these constraints provide the complete characterization of the related spanning tree polytope in the case of binary u_i variables (Edmonds, 1970), they are strengthened by tightening the bounds on the Steiner variables. For this reason we add constraints (26) and (27) that force all terminal nodes and all non-terminal relay locations to be included in the resulting Steiner tree.

4.4. Solving STF

Since STF contains a large number of variables and constraints which exponentially grow with the problem size it is not possible to solve STF directly for realistic size problems and a simultaneous row and column generation is needed. We denote the linear relaxation of STF with STF-LP.

To solve STF-LP, we start with a subset of single-flow variables and add the remaining variables iteratively in a column generation phase. Once there is no variable to add we check for violated inequalities. If we find one we add it to the model and return to the column generation phase again.

During the column generation phase we solve the pricing problem to detect any negative reduced cost variables. For the row generation phase we solve a separation problem to detect any violated inequalities. Here note that, constraints (15) and (16) could be violated by a solution with integral relay and edge design variables and to be able to properly calculate the reduced costs of the single-flow variables we need to add any violated inequalities after each column generation iteration. Below, we discuss this issue in more detail, explain how we solve the pricing and the separation problems, how we choose the initial subset of flow variables, the branching rule, the search rule and other implementation details.

Initial variable and constraint pool:

For the initial variable pool we use the trivial single-flow variables each associated with an arc in the input network, i.e., we have $V_0 = \{v_{p_{ij}} : (i,j) \in A\}$ as the initial set of columns. For the initial constraints we consider Constraints (14),(17)–(19),(23),(24),(26) and (27) and the relaxed domain restrictions.

Separation problem:

After solving the initial problem with a subset of variables and constraints, we look for violated inequalities by solving the separation problem. We use the separation procedure proposed by Lee et al. (1996), which is designed for a closely related Steiner tree problem, for both the physical and virtual network cycle cancelation constraints. For the sake of brevity, we only explain this separation procedure for the physical network, i.e. separation of constraints (25), but it is very easy to see that the same procedure can be readily applied to the virtual network cycles as well.

Let (r^*, w^*, u^*, v^*) be the optimal solution for STF-LP where $V^* = \{i \in V : u_i^* > 0\}$, $E^* = \{e \in E : w_e^* > 0\}$ are the sets of Steiner and edge design variables with positive values. We first generate the separation graph $\hat{G} = (\hat{V}, \hat{A})$, where:

1. node set $\hat{V} = V^* \cup E^* \cup \{\hat{s}, \hat{t}\}$,
2. arc set $\hat{A} = \{(\hat{s}, e) : e \in E^*\} \cup \{(i, \hat{t}) : i \in V^*\} \cup \{(e,i), (e,j) : e = \{i,j\} \in E^*\}$,
3. arc capacities are infinite except for those arcs:
   - leaving from $\hat{s}$, with capacities $\hat{c}_{(\hat{s},e)} = w_e^*$,
   - reaching to $\hat{t}$, with capacities $\hat{c}_{(i,\hat{t})} = u_i^*$.

To solve the separation problem, for each $j \in V^*$, we change the arc capacity for $(j, \hat{t})$ to zero and solve a maximum flow problem from $\hat{s}$ to $\hat{t}$. If the solution value is less than $w(E)$, the minimum capacity cut gives a violated inequality where S is the set of indices of Steiner variables in the minimum cut and j is the index of the Steiner variable that is excluded in the first sum in (25). For more details about the validity of this separation procedure we refer the reader to Lee et al. (1996).

Note that the above exact separation procedure requires solving several maximum flow problems. Indeed we can do better than this by first trying a simple yet quite efficient heuristic before resorting to the exact solution for the separation problem. The heuristic algorithm which we call as the *connectivity heuristic* (HS_C) depends on the following observation. If the separation graph $\hat{G}$ is not connected and none of its connected components contains all the terminal nodes (including s), then we can detect violations as follows. Let $\mathcal{C}$ be the collection of connected components of $\hat{G}$ each including a terminal node. For $S \subset V$, let $\delta(S) = \{e \in E : |e \cap S| = 1\}$. Then the following cuts can be added to the model to remove this infeasible solution.

$$w(\delta(C)) \geq 1 \qquad C \in \mathcal{C}. \tag{28}$$

It is obvious that Theorem 1 establishes the validity of these cuts for the physical layer and Theorem 2 enables a very similar procedure to be applied in the virtual layer. Thus, using the heuristic HS_C, we have the chance to solve the separation problem with a simple graph search in many cases especially at the beginning of the algorithm where a very limited number of constraints are in the model.

Pricing problem:

After the row generation (cutting plane) phase ends, we look for columns (single-flow variables) with negative reduced costs to add to the model. We solve the pricing problem to find such a column or conclude that none exists. Let $-\alpha, -\beta, -\gamma, -\lambda, \theta, -\mu$ be the dual variables associated with constraints (14)–(19), respectively. Since we are looking for a tree rooted at s, the reduced cost $\bar{v}_p$ of a flow variable v_p is infinity if $d(p) = s$ and if this is not the case it can be calculated as follows:

$$\bar{v}_p = \begin{cases} \alpha + \sum_{\substack{S\subset V,\\ S\cap D\neq\emptyset \\ o(p),d(p)\in S}} \beta_S + \sum_{\substack{S\subset V,\\ S\cap D=\emptyset \\ o(p),d(p)\in S}} \sum_{j\in S} \gamma_S^j + \lambda_{d(p)} & \\ \quad + \theta_{o(p),d(p)} + \sum_{e\in\mathcal{E}(p)} \mu^e_{d(p)}, & \text{if } d(p) \in D, \\ \alpha + \sum_{\substack{S\subset V,\\ S\cap D\neq\emptyset \\ o(p),d(p)\in S}} \beta_S + \sum_{\substack{S\subset V,\\ S\cap D=\emptyset \\ o(p),d(p)\in S}} \sum_{j\in S} \gamma_S^j + \lambda_{d(p)} & \\ \quad + \theta_{o(p),d(p)} + \theta_{d(p),o(p)} + \sum_{e\in\mathcal{E}(p)} \mu^e_{d(p)}, & \text{otherwise.} \end{cases} \tag{29}$$

Note that, as a complicating factor, the above calculation involves some dual variables associated with constraints that are not in the current model. Fortunately, this does not cause a problem since we can set the values of the dual variables associated with the constraints that are not in the model to zero. Here note that, since we start with all the trivial single-flow variables and do not remove any of them throughout the algorithm, addition of the Steiner cuts does not cause infeasibility and reduced cost calculations are not adversely affected.

Recall that we are looking for negative reduced cost variables and we have $\mu^e_{d(p)} \geq 0$, for each edge $e \in E$ and path-segment $p \in \mathcal{P}$. Thus, the pricing problem for each plausible pair $(i, j) \in A^c$ is actually a CSP instance from node i to node j on original network $G = (V, A)$ in which the resource for each arc $a \in A$ is l_a, the cost for each arc $a \in A$ is μ^e_j and the resource limit is d_{max}. As we have discussed during the presentation of the branch-and-price algorithm for the MCF formulation in Section 3.3, we first try the HS_q heuristic to detect any negative reduced cost variables and resort to the exact solution only when HS_q cannot find one.

Implementation details:

For the branch-and-price-and-cut, we use essentially the same branching and search strategies previously discussed for the branch-and-price in Section 3.3 and for the sake of brevity we do not repeat the same explanations here. For the variable and constraint pool management our preliminary studies have shown that our branch-and-price-and-cut algorithm performs best when we do not remove any column or row. During the branch-and-price-and-cut algorithm we search for violated inequalities in the root node, in the last branch-and-bound node left in the queue and when we find an integral solution. For the cut generation, except for the case we have an integer solution at hand, we terminate it when the increase in the objective function value is less than 0.01 after the inclusion of the last set of cuts.

Heuristic:

Mimicking the heuristic solution approach we have for the branch-and-price algorithm, at the root node of the branch-and-bound tree, at each iteration of the column generation phase we check whether the solution for the current restricted STF-LP is integral or not. If it is integral, there is no violated constraint and this solution has a lower cost than the best integer found so far then we store it as the best integer solution. If no such solution is found, we solve STF as an integer program with only the columns generated at the root node. To solve this restricted STF more efficiently, we employ a branch-and-cut approach in which the violated constraints are added iteratively using the same separation procedure we presented above. Our numerical experiments have shown that the integral solution obtained by this procedure is often of high quality and can speed up the branch-and-price-and-cut algorithm significantly.

Note that we can also solve NDR-S by solving the TF formulation with a branch-and-price-and-cut algorithm that is similar to the one presented for STF. With an aim to asses the benefits of using the information that the solution of the problem is a tree in the input graph, we consider this simpler algorithm in our computational experiments as well.

4.5. Enhancing MCF for NDR-S problems

When we consider NDR-S problems, we can add the Steiner tree cuts to the MCF formulation as well and obtain a stronger formulation. Let SMCF be the MCF formulation strengthened with the Steiner tree cuts. We need a branch-and-price-and-cut algorithm to solve this new formulation. Since the Steiner tree cuts for MCF do not include the flow variables, the pricing problem is not affected by the inclusion of these cuts. As a result, with a slight modification of the branch-and-price algorithm explained in detail in Section 3.3, it is straightforward to implement this new branch-and-price-and-cut algorithm and for the sake of brevity we do not provide the details here.

5. Computational study

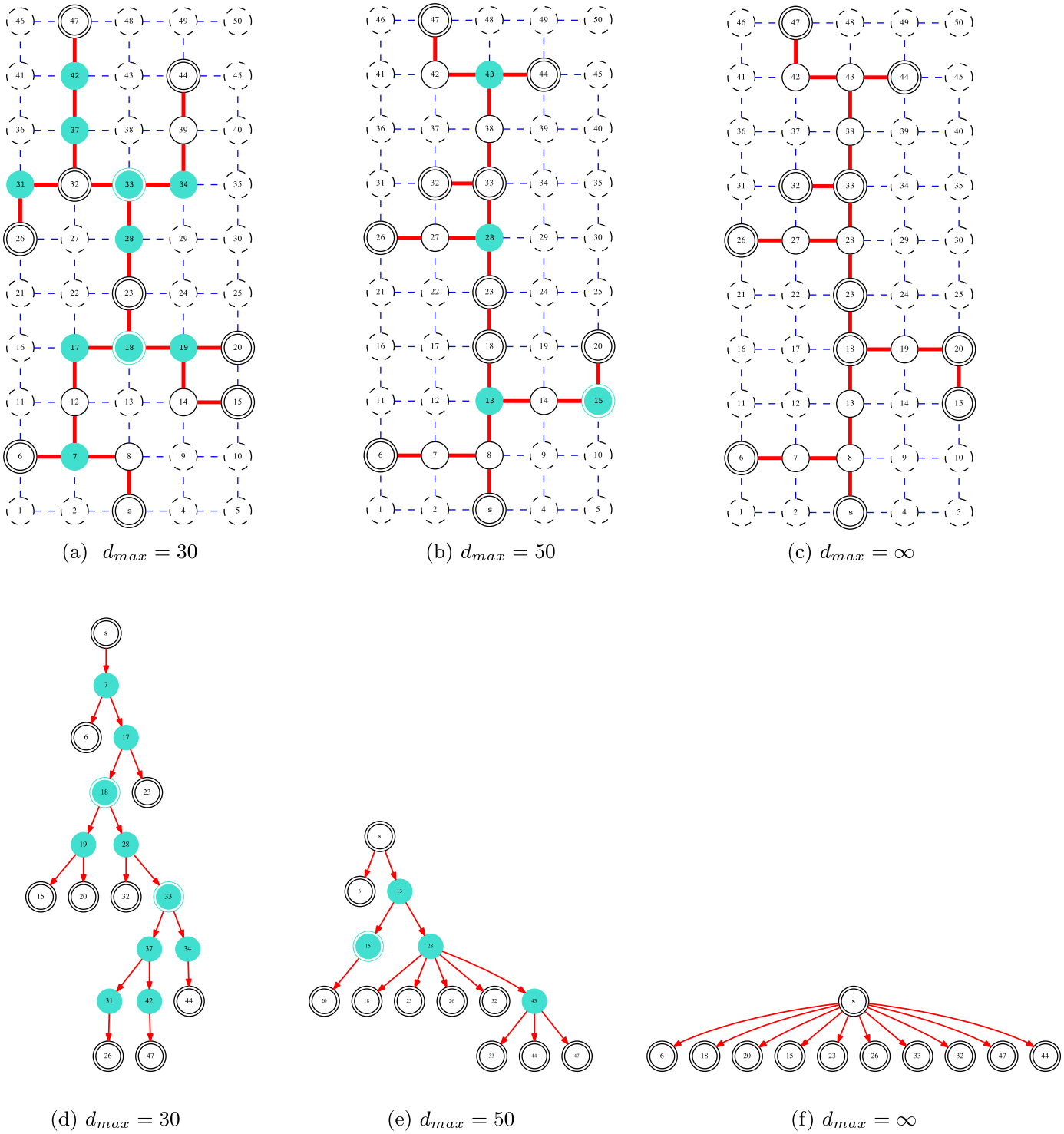
Comprehensive numerical experiments are conducted to test the performance of the algorithms proposed in this study.Three common network topologies from the literature are considered: grid networks by Cabral et al. (2007), Steiner instances B and C from the OR-Library (Beasley, 1990) and instances by Konak (2012). We implemented all the algorithms using Java under Linux and CPLEX 12.5 and all experiments are done on the same machine: Intel 2 Xeon E5-2609 4C 2.4 GHz CPU and 8GB RAM.

5.1. Single-source instances

5.1.1. Cabral instances

Since the NDR problem is first introduced by Cabral et al. (2007), most of the literature on this problem considers the instances presented in this study. In order to test our algorithms we consider original Cabral instances as well as new ones with higher number of OD pairs that we obtain by imitating the random problem generation procedure described in Cabral et al. (2007). Cabral instances are single-source instances that have a grid structure with a rows and b columns. The costs and lengths of the edges are randomly generated from a uniform distribution $U[10, 30]$. For these problem instances $d_{max} = 70$ and the relay costs are randomly chosen from a uniform distribution $U[70, 140]$.

Before proceeding with the computational results, we first look at NDR-S solutions on some small size network instances to show the complexity of the problem and to illustrate the correspondence between the solutions in the physical and virtual networks. These examples clearly show that solving NDR in a sequential manner, i.e., first solving a Steiner tree problem and then placing regenerators in the resulting design, may not produce high quality solutions. In Fig. 4, we consider a 5×10 grid network with 10 terminal nodes. Fig. 4(a)–(c) depict the NDR-S solutions on the input graphs for d_{max} values of 30, 50 and infinity, respectively. The dashed lines show the edges present in the grid graph, whereas the thick(red) lines show those edges included in the optimal solution. The terminal nodes are shown with double circles and nodes enhanced with relay capabilities are highlighted (depicted in turquoise color). The dashed circles show the nodes in the grid graph that are not



(a) $d_{max} = 30$ (b) $d_{max} = 50$ (c) $d_{max} = \infty$

(d) $d_{max} = 30$ (e) $d_{max} = 50$ (f) $d_{max} = \infty$

Fig. 4. Grid graph solution illustrations.

included in the design. Figs. 4(d)–(f) illustrate the respective solutions in the virtual networks as rooted trees. These instances clearly show that the optimal edge design may heavily depend on the locations of the relays. Note that when $d_{max} = \infty$ our problem reduces to a Steiner tree problem and Fig. 4(c) shows the minimum cost Steiner tree solution. This design is quite different than the design for the instance with $d_{max} = 30$. As we see in this small example, the tree structure generated in the physical network layer is significantly affected by the relay placement decisions. Therefore, when the relay costs are not negligible and d_{max} values are restrictive, solving the NDR problem by jointly solving relay placement and network design problems can provide much different results than a sequential solution approach. As a result, depending on the cost structure of the given problem instance, the difference between such a heuristic solution's objective function value and the true optimum can get arbitrarily large.

We first compare the computational performances of the solution approaches we propose in this study. Table 1 shows the re-

Table 1
Solution time comparisons for the proposed algorithms.

			Solution Time (seconds)				Solution Time Ratios		
a	b	$\|K\|$	MCF	SMCF	TF	STF	MCF	SMCF	TF
5	5	5	0.9	0.6	1.0	0.7	1.3	0.9	1.4
		10	2.6	1.3	1.2	0.7	3.7	1.8	1.7
		20	115.7	15.4	19.3	1.4	81.8	10.9	13.7
6	5	5	1.2	1.0	2.0	1.1	1.1	1.0	1.9
		10	8.9	3.5	5.9	1.6	5.7	2.3	3.8
		20	285.7	12.8	35.8	1.7	170.3	7.6	21.3
7	5	5	2.0	2.3	3.8	2.4	0.8	0.9	1.6
		10	18.3	6.4	10.1	2.3	7.9	2.8	4.4
		20	448.5	31.1	55.1	3.0	148.7	10.3	18.2
		30	4213.5	230.5	387.8	9.1	461.2	25.2	42.4
8	5	5	2.2	1.9	5.8	3.5	0.6	0.5	1.7
		10	51.9	14.6	45.0	7.0	7.4	2.1	6.4
		20	735.5	39.8	145.5	6.6	110.7	6.0	21.9
		30	> 7200.0	859.7	2376.7	17.9	> 402.9	48.1	133.0
9	5	5	3.1	5.6	20.3	6.1	0.5	0.9	3.3
		10	53.1	23.5	65.9	8.4	6.3	2.8	7.8
		20	1786.8	136.2	682.2	17.1	104.8	8.0	40.0
		30	> 7200.0	220.8	2148.6	19.1	> 376.2	11.5	112.3
10	5	5	3.1	3.3	58.5	40.6	0.1	0.1	1.4
		10	54.2	15.2	149.5	44.4	1.2	0.3	3.4
		20	1483.0	65.7	1011.4	70.3	21.1	0.9	14.4
		30	> 7200.0	317.9	3564.4	43.4	> 165.9	7.3	82.1
		40	> 7200.0	851.9	> 7200.0	63.7	> 113.1	13.4	> 113.1
11	5	5	11.6	18.6	675.8	42.9	0.3	0.4	15.8
		10	98.1	28.6	484.0	43.9	2.2	0.7	11.0
		20	2072.6	108.9	1881.9	28.7	72.1	3.8	65.5
		30	> 7200.0	1104.0	5819.3	122.0	> 59.0	9.0	47.7
		40	> 7200.0	3968.6	> 7200.0	226.3	> 31.8	17.5	> 31.8
12	5	5	5.0	8.9	294.2	40.6	0.1	0.2	7.2
		10	58.6	32.1	712.0	44.4	1.3	0.7	16.0
		20	2684.6	166.1	3187.7	70.3	38.2	2.4	45.3
		30	> 7200.0	2926.0	5938.2	43.4	165.9	67.4	136.8
		40	> 7200.0	802.4	> 7200.0	63.7	> 113.1	12.6	> 113.1

sults of numerical experiments for instances with up to 60 nodes and 40 OD pairs. In the table, a and b are the grid dimensions and $|K|$ is the number of OD pairs. For each setting, five random instances are created and the average solution times are reported for each solution method. The first four columns show the run times in seconds for the respective algorithms that solve MCF, SMCF, TF and STF formulations and the last three columns report the ratios of the solution times of algorithms for MCF, SMCF and TF to those of STF. A time limit of 7200 seconds is given to all the algorithms. For all instances, HS_6 heuristic is used to solve the related pricing problem in the branch-and-price and branch-and-price-and-cut algorithms.

Looking at Table 1, we immediately notice that including Steiner cuts drastically improves the performances of the solution procedures both for the multi-commodity flow and tree formulations. For the multi-commodity flow formulations, solution times with SMCF can on the average improve more than 20 times, whereas this ratio increases to 100 when we consider tree formulations. Comparing the computational performances of all these solution methods, the results presented in Table 1 clearly show that the branch-and-price-and-cut algorithm of STF is superior to other algorithms. The last three columns show that, on the average, this algorithm can be more than 461, 67 and 136 times faster than the algorithms that solve MCF, SMCF and TF, respectively. Note that these ratios could be much larger if we had not have the time limit and waited all algorithms to finish. Such a drastic performance superiority can be explained by two major advantages STF has over other formulations. The first one, that it shares with TF, is the single flow approach for which the increase in the number of OD pairs in the problem does not increase the number of variables whereas, the number of variables in the multi-commodity flow formulations depends directly on the number of OD pairs. As a result, while the number of OD pairs grows, the run times for STF do not get as much adversely affected as those for the multi-commodity flow formulations. The second advantage is the inclusion of the Steiner cuts, which it shares with the SMCF. The inclusion of these cuts results in a much tighter formulation that can be solved more efficiently.

Table 2
Results for original Cabral instances with STF.

a	b	K	Time	# Opt	Gap	R.Gap	BBN	Cols	Cuts
4	5	5	0.5	10	0.00	0.00	1.4	464.6	18.5
		10	0.6	10	0.00	0.01	2.0	499.9	16.5
5	5	5	0.7	10	0.00	0.01	2.4	670.3	38.8
		10	0.7	10	0.00	0.01	5.4	716.8	32.3
6	5	5	4.7	10	0.00	0.01	3.2	902.2	84.2
		10	1.9	10	0.00	0.02	5.4	976.0	87.6
7	5	5	1.5	10	0.00	0.01	3.0	980.8	81.6
		10	2.5	10	0.00	0.03	10.6	1211.0	111.9
8	5	5	5.6	10	0.00	0.02	10.4	1308.2	222.8
		10	3.5	10	0.00	0.03	12.2	1345.7	207.7
9	5	5	7.1	10	0.00	0.02	6.8	1462.3	211.8
		10	7.9	10	0.00	0.03	16.8	1634.1	198.5
10	5	5	29.6	10	0.00	0.04	21.2	1749.3	409.6
		10	45.9	10	0.00	0.04	40.4	1930.0	306.3
11	5	5	24.5	10	0.00	0.04	28.0	1813.0	431.1
		10	78.7	10	0.00	0.05	46.6	2210.7	459.4
12	5	5	59.1	10	0.00	0.02	29.4	2239.0	523.0
		10	117.8	10	0.00	0.07	81.0	2556.4	657.6

As Table 1 indicates, the performance of the branch-and-price-and-cut algorithm for STF is quite satisfactory for small and medium size problems. For a better assessment of its performance we studied original Cabral instances (Cabral et al., 2007) and larger grid networks with up to 100 nodes and 80 OD pairs. For each

Table 3
Results for larger size problem instances with STF.

a	b	|K|	Time	#Opt	Gap	R.Gap	BBN	Cols	Cuts
8	8	5	3.80	5	0.00	0.05	12.20	1256.20	199.40
		10	19.52	5	0.00	0.08	42.60	2838.20	535.80
		20	56.10	5	0.00	0.07	100.20	3115.60	875.20
		30	99.84	5	0.00	0.08	223.00	3216.20	1002.80
		40	494.94	5	0.00	0.08	1410.20	3275.60	1450.40
		60	3110.17	4	0.02	0.07	13643.80	3343.40	1514.60
9	9	5	100.98	5	0.00	0.08	44.60	3782.60	1474.20
		10	219.75	5	0.00	0.10	181.00	4145.80	1286.80
		20	689.04	5	0.00	0.11	647.80	5026.00	1329.00
		30	1135.57	5	0.00	0.11	1130.60	4939.00	1820.60
		40	2149.40	5	0.00	0.10	2488.20	4882.00	1938.00
		60	6680.92	1	0.06	0.09	11118.00	4914.20	1507.60
10	10	5	208.85	5	0.00	0.12	71.80	4109.00	1952.00
		10	936.79	5	0.00	0.11	277.80	5368.80	2280.00
		20	2259.04	5	0.00	0.11	1402.20	5922.00	2264.40
		30	4198.15	4	0.02	0.10	1882.25	5620.75	2434.25
		40	6373.17	1	0.09	0.12	3885.40	5998.40	1729.80
		60	6053.07	1	0.09	0.10	5050.60	6022.80	1452.40
		80	7201.20	0	0.12	0.12	10563.80	5969.60	967.00

Table 4
Results for OR library Steiner tree instances with STF.

Prob.	Num	V	E	|K|	Time	#Opt	Gap	R.Gap	BBN	Cols	Cuts
SteinB	1	50	63	9	1.5	5	0.00	0.03	3.8	1217.8	79.8
	2			13	1.9	5	0.00	0.03	6.6	1249.4	90.0
	3			25	1.4	5	0.00	0.00	4.2	1424.6	49.8
	4		100	9	6.9	5	0.00	0.03	7.4	5307.2	224.8
	5			13	2.8	5	0.00	0.01	4.2	4158.2	43.6
	6			25	4.5	5	0.00	0.01	5.4	4544.4	139.2
	7	75	94	13	2.9	5	0.00	0.02	9.8	1992.6	85.6
	8			19	4.2	5	0.00	0.03	7.4	1951.8	151.0
	9			38	12.6	5	0.00	0.02	99.0	1837.2	253.8
	10		150	13	10.1	5	0.00	0.02	6.2	6735.6	235.0
	11			19	24.9	5	0.00	0.03	11.4	7018.6	365.2
	12			38	24.8	5	0.00	0.02	11.8	8441.0	348.8
	13	100	125	17	2.6	5	0.00	0.00	3.4	2028.8	59.0
	14			25	5.6	5	0.00	0.02	9.4	2110.6	320.6
	15			50	41.1	5	0.00	0.02	206.6	2874.8	667.6
	16		200	17	31.1	5	0.00	0.04	18.2	7228.0	459.4
	17			25	92.2	5	0.00	0.05	31.4	10879.0	783.2
	18			50	458.8	5	0.00	0.03	86.6	11470.8	1215.2
SteinC	1	500	625	5	960.0	5	0.00	0.13	44.6	14707.0	1282.2
	2			10	3851.6	3	0.02	0.08	91.8	17618.8	1690.0
	3			83	7200.0	0	0.08	0.08	772.2	15461.2	1193.8
	4			125	7200.0	0	0.05	0.05	418.8	16529.8	2543.8
	5			250	7200.0	0	0.07	0.07	433.6	18392.0	3417.2
	6	500	1000	5	2743.1	5	0.00	0.02	5.0	50299.8	2791.6
	7			10	2786.1	5	0.00	0.02	5.4	58781.0	1795.2
	8			83	7200.0	0	0.08	0.08	21.6	42982.4	2617.8

problem setting we have ten instances for the Cabral problems and five random instances for the large grid networks. Since none of the other formulations are able to scale up to the large problem instances considered in these experiments, we only report the solutions for STF in Tables 2 and 3 for the Cabral instances and large grid networks, respectively. In these tables, the column *Time* is the average run time of the algorithm in seconds until it finds an optimal solution or the time limit of 7200 seconds is reached. The column *#Opt* shows the number of instances solved to optimality within the given time limit, *Gap* is the average optimality gap (percentage) and *R.Gap* is the average optimality gap (percentage) calculated at the root node of the branch and bound tree. The column *BBN* shows the number of branch-and-bound nodes explored by the algorithm whereas *Cols* and *Cuts* columns show the total number of columns and rows generated, respectively, throughout the solution procedure.

As we see in Tables 2 and 3, for most of the instances, the branch-and-price-and-cut algorithm solves STF to optimality and the optimality gaps are quite small for the instances for which the algorithm stopped due to the time limit. For these sets of instances, the linear relaxation of STF provides a quite tight lower bound which indicates the strength of the STF and accounts for the high computational efficacy. From Table 3 we can infer that the problem gets harder when the number of OD pairs increases. It is interesting to see that while the number of explored branch-and-bound nodes drastically increases with the number of OD pairs, the number of columns and rows generated do not increase as much.

5.1.2. Steiner problems from OR-library

As we have already argued, NDR problems are closely related with the Steiner tree problems, which are extensively studied in the literature. In order to further test and evaluate the computational performance of our branch-and-price-and-cut algorithm for STF on a different network topology than we have for the grid networks, we also consider the Steiner problem instances B and C from the OR-Library (Beasley, 1990). These network topologies are

Table 5
Results of the computational experiments for type-I problems with $c_e = l_e$.

				GA Results		MCF Results		
N	$\|K\|$	d_{max}	$\|E\|$	Average	Best	Sol.	Gap	RT
40	5	30	198	486.26	473.80	473.80	0.00	2.0
40	5	35	272	354.58	354.57	352.08	0.00	7.3
40	10	30	198	521.26	518.98	518.98	0.00	7.2
40	10	35	272	407.00	407.00	399.36	0.00	194.4
50	5	30	279	283.79	283.78	283.79	0.00	0.5
50	5	35	372	271.90	260.23	260.24	0.00	1.1
50	10	30	279	544.06	540.39	540.39	0.00	16.7
50	10	35	372	426.88	407.49	404.32	0.00	31.9
60	5	30	305	523.01	509.90	509.12	0.00	3.1
60	5	35	412	378.81	377.02	377.02	0.00	5.8
60	10	30	641	681.40	678.84	678.84	0.00	27.4
60	10	35	853	507.34	499.64	499.64	0.00	59.5
80	5	30	641	358.20	356.65	353.86	0.00	759.4
80	5	35	853	328.80	328.80	328.80	0.00	5495.0
80	10	30	641	471.07	471.00	460.36	0.16	7200.0
80	10	35	853	448.42	436.76	436.76	0.26	7200.0
160	5	30	2773	292.98	287.93	287.84	0.26	7200.0
160	5	35	3624	274.64	270.22	275.96	0.26	7200.0
160	10	30	2773	413.69	405.64	439.38	0.38	7200.0
160	10	35	3624	415.13	397.59	422.58	0.38	7200.0

Table 6
Results of the computational experiments for type-II problems with $c_e = d_{max} - l_e$.

				GA Results		MCF Results		
N	$\|K\|$	d_{max}	$\|E\|$	Average	Best	Sol.	Gap	RT
40	5	30	198	247.27	247.27	247.27	0.00	0.4
40	5	35	272	111.30	111.30	111.30	0.00	1.2
40	10	30	198	297.09	292.62	292.62	0.00	1.1
40	10	35	272	140.51	140.51	140.51	0.00	3.0
50	5	30	279	119.80	119.80	119.80	0.00	0.4
50	5	35	372	155.57	155.57	155.57	0.00	1.0
50	10	30	279	297.55	279.70	279.70	0.00	1.0
50	10	35	372	220.26	206.22	206.22	0.00	2.1
60	5	30	305	318.17	317.32	317.32	0.00	0.8
60	5	35	412	166.35	166.35	166.35	0.00	0.9
60	10	30	641	452.21	414.32	414.32	0.00	7.6
60	10	35	853	243.76	242.32	242.32	0.00	3.5
80	5	30	641	139.70	134.73	134.73	0.00	3.6
80	5	35	853	104.04	104.04	104.04	0.00	9.9
80	10	30	641	189.80	187.17	187.17	0.00	16.1
80	10	35	853	174.07	168.62	168.62	0.00	48.8
160	5	30	2773	82.35	78.61	78.61	0.00	601.6
160	5	35	3624	74.35	68.15	68.15	0.00	2881.8
160	10	30	2773	125.22	112.06	112.06	0.00	2744.7
160	10	35	3624	118.70	109.12	109.12	0.00	4555.1

considered for other extensions to the basic Steiner tree problem such as the Steiner tree problem with revenues, budget and hop constraints (Costa et al., 2009).

We abide by the cost structures and reach limitations of the Cabral instances when we generate random single-source instances from these network topologies. The root node s is chosen as the smallest number node in the list of terminals provided for each problem instance. In solving the pricing problems, we use HS_{15} and HS_5 for the group of instances B and C, respectively.

Table 4 summarizes the results with the branch-and-price-and-cut algorithm for these instances. For each network five random instances are considered. As we see in the table, STF can scale up to these large problem instances. Most of them are solved to optimality and the optimality gaps are quite low for the unsolved problems. We can see in Table 4 that the linear relaxation of the formulation provides quite tight bounds for these large size instances. Similarly we see that the number of OD pairs is a significant dimension of the problem that affects the performance of the algorithm.

5.2. Multi-source instances

In this subsection we consider NDR instances presented in Konak (2012) and compare the performance of our MCF algorithm with the genetic algorithm (GA) proposed in that paper. Our preliminary studies have shown that for the small and medium size instances MCF runs faster when we solve it as a compact mixed integer program by providing all the simple paths with lengths at most the given threshold value d_{max}. So in our computational experiments we solve MCF directly as a MIP for 40, 50, 60 and 80 node instances and solve it with the branch and price approach for instances with 160 nodes. Tables 5 and 6 present the results we obtain with MCF algorithm along with the results of the genetic algorithm for type-I and type-II problems, respectively (A. Konak, personal communication, March 28, 2017). Considering different practical applications, the correlations between the cost and length of an edge $e \in E$ are defined differently to generate type-I and type-II problem instances. For the type-I problem instances the edge costs and lengths are the same whereas for the

type-II problem instances they are inversely correlated and sum up to d_{max} for each edge. In both tables, columns N, $|K|$, d_{max}, $|E|$ indicate the number of nodes, number of OD pairs, maximum reach value and number of edges of the respective problem instance. For the GA results, the column *Average* indicates the average objective function value for the 10 runs of the algorithm and the following column *Best* reports the best solution value obtained in these runs. For the MCF algorithm results, *Sol.* is the value of the best feasible solution found in 7200 second time limit and the column *Gap* reports the optimality gap calculated for the reported MCF solution. The last column *RT* indicates the run time of the MCF algorithm.

One interesting result we see in Tables 5 and 6 is that in most of the instances GA is able to find the optimal solution in 10 trials. These results also attest to the efficiency of the MCF algorithm. For small and medium size problem instances our algorithm can find optimal solutions efficiently. In our experiments it is usually the case that MCF finds the optimal solution quite early and spends much of its time to obtain the proof of optimality. It is also interesting to see that when the cost and the length of the edges are inversely correlated, the performance of the MCF algorithm gets better. For the type-II problem instances forming path segments with multiple short arcs are discouraged due to the resulting high costs and optimal solution contains mostly path segments with very small number of arcs. We think that, since the number of path segments with fewer arcs are much less than the number of path segments with larger number of arcs, MCF can find the optimal solution much faster for type-II problems.

6. Conclusion

This study revisits the network design problem with the relay placement perspective and introduces efficient solution algorithms to solve this relatively new and challenging network design problem. The literature on network design problems is quite extensive but even though relays are common in many transportation and telecommunication applications, there are very few studies that consider relays as part of the network design. Our study attempts to fill this gap in the literature.

Network design problems are challenging problems and incorporating relay placement decisions complicates them further by allowing paths with cycles to appear in the optimal solutions. In order to address this challenging problem, we propose a multi-commodity flow based formulation and a branch-and-price algorithm to solve it by using the path-segment notion of Yıldız and Karaşan (2017) and Yıldız et al. (2016). Our path-segment formulations enable quite efficient representation of the reach constraints in the model and provide opportunities to device graph transformations and novel algorithmic approaches to improve the performance of our branch-and-price procedure.

With both practical and theoretical motivations, we pay a special attention to NDR-S in this study. As one of the main contributions of this study, we show special properties satisfied by a subset of optimal solutions for this interesting problem. Then we propose a novel tree formulation and a branch-and-price-and-cut algorithm that exploits this special structure. Considering the special properties of NDR-S exclusively and tailoring our algorithm respectively, we achieve to solve large problem instances. As a future research direction, we are interested in adapting this approach to address other location and routing problems with reach constraints.

Acknowledgments

This work was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under the grant number 2211A. We are also thankful to an anonymous reviewer for indicating a bug in the implementation.

References

Ali, T.H., Radhakrishnan, S., Pulat, S., Gaddipati, N.C., 2002. Relay network design in freight transportation systems. Transp. Res. Part E 38 (6), 405–422.
Bapna, R., Thakur, L.S., Nair, S.K., 2002. Infrastructure development for conversion to environmentally friendly fuel. Eur. J. Oper. Res. 142 (3), 480–496.
Barnhart, C., Hane, C.A., Vance, P.H., 2000. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. Oper. Res. 48 (2), 318–326.
Beasley, J.E., 1990. Or-library: distributing test problems by electronic mail. J. Oper. Res. Soc. 1069–1072.
Cabral, E.A., Erkut, E., Laporte, G., Patterson, R.A., 2007. The network design problem with relays. Eur. J. Oper. Res. 180 (2), 834–844.
Campbell, J., 1994. Integer programming formulations of discrete hub location problems. Eur. J. Oper. Res. 72 (2), 387–405.
Capar, I., Kuby, M., Leon, V.J., Tsai, Y.-J., 2013. An arc coverpath-cover formulation and strategic analysis of alternative-fuel station locations. Eur. J. Oper. Res. 227 (1), 142–151.
Chen, S., Ljubić, I., Raghavan, S., 2010. The regenerator location problem. Networks 55 (3), 205–220.
Chen, S., Ljubić, I., Raghavan, S., 2015. The generalized regenerator location problem. J. Comput. 27 (2), 204–220.
Costa, A., Cordeau, J.-F., Laporte, G., 2009. Models and branch-and-cut algorithms for the steiner tree problem with revenues, budget and hop constraints. Networks.
de Azevedo, J., Silvestre Madeira, J., Vieira Martins, E., Pires, F., 1994. A computational improvement for a shortest paths ranking algorithm. Eur. J. Oper. Res. 73 (1), 188–191.
Edmonds, J., 1970. Submodular functions, matroids, and certain polyhedra. Combinat. Struct. Appl. 69–87.
Garey, M., Johnson, D., 1979. Computers and Intractability, 174. Freeman San Francisco.
Hwang, F.K., Richards, D.S., Winter, P., 1992. The Steiner Tree Problem. Elsevier.
Jinno, M., Takara, H., Kozicki, B., Tsukishima, Y., Sone, Y., Matsuoka, S., 2009. Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies. Commun. Mag., IEEE 47 (11), 66–73.
Kewcharoenwong, P., Üster, H., 2014. Benders decomposition algorithms for the fixed-charge relay network design in telecommunications. Telecommun. Syst. 56 (4), 441–453.
Kim, J.-G., Kuby, M., 2012. The deviation-flow refueling location model for optimizing a network of refueling stations. Int. J. Hydrogen Energy 37 (6), 5406–5420.
Kim, J.-G., Kuby, M., 2013. A network transformation heuristic approach for the deviation flow refueling location model. Comput. Oper. Res. 40 (4), 1122–1131.
Konak, A., 2012. Network design problem with relays: a genetic algorithm with a path-based crossover and a set covering formulation. Eur. J. Oper. Res. 218 (3), 829–837.
Kuby, M., Lim, S., 2005. The flow-refueling location problem for alternative-fuel vehicles. Socioecon. Plann. Sci. 39 (2), 125–145.
Kuby, M., Lim, S., 2007. Location of alternative-fuel stations using the flow-refueling location model and dispersion of candidate sites on arcs. Networks Spatial Econ. 7 (2), 129–152.
Kuby, M., Lines, L., Schultz, R., Xie, Z., Kim, J.-G., Lim, S., 2009. Optimization of hydrogen stations in florida using the flow-refueling location model. Int. J. Hydrogen Energy 34 (15), 6045–6064.
Kulturel-Konak, S., Konak, A., 2008. A local search hybrid genetic algorithm approach to the network design problem with relay stations. In: Telecommunications Modeling, Policy, and Technology. Springer, pp. 311–324.
Laporte, G., Pascoal, M.M., 2011. Minimum cost path problems with relays. Comput. Oper. Res. 38 (1), 165–173.
Lee, Y., Chiu, S.Y., Ryan, J., 1996. A branch and cut algorithm for a steiner tree-star problem. INFORMS J. Comput. 8 (3), 194–201.
Leitner, M., Ljubić, I., Riedler, M., AC Ruthmair, M., 2015. On optimal design of charging stations for electric vehicles. Technical Report. Institute of Computer Graphicsand Algorithms, TU Wien URL http://homepage.univie.ac.at/markus.leitner/research/pub/pdf/ac-tr-15-003.pdf.
Li, X., Aneja, Y., Huo, J., 2012. Using branch-and-price approach to solve the directed network design problem with relays. Omega 40 (5), 672–679.
Lübbecke, M.E., Desrosiers, J., 2005. Selected topics in column generation. Oper. Res. 53 (6), 1007–1023.
Melaina, M., Bremson, J., 2008. Refueling availability for alternative fuel vehicle markets: sufficient urban station coverage. Energy Policy 36 (8), 3233–3241.
Melaina, M.W., 2003. Initiating hydrogen infrastructures: preliminary analysis of a sufficient number of initial hydrogen stations in the us. Int. J. Hydrogen Energy 28 (7), 743–755.
MirHassani, S.A., Ebrazi, R., 2013. A flexible reformulation of the refueling station location problem. Transp. Sci. 47 (4), 617–628.
Romm, J., 2006. The car and fuel of the future. Energy Policy 34 (17), 2609–2614.
Segev, A., 1987. The node-weighted steiner tree problem. Networks 17 (1), 1–17.
Üster, H., Kewcharoenwong, P., 2011. Strategic design and analysis of a relay network in truckload transportation. Transp. Sci. 45 (4), 505–523.
Üster, H., Maheshwari, N., 2007. Strategic network design for multi-zone truckload shipments. IIE Trans. 39 (2), 177–189.
Vergara, H.A., Root, S., 2013. Mixed fleet dispatching in truckload relay network design optimization. Transp. Res. Part E 54, 32–49.
Voß, S., 1999. The steiner tree problem with hop constraints. Ann. Oper. Res. 86, 321–345.

Wang, Y.-W., Lin, C.-C., 2009. Locating road-vehicle refueling stations. Transp. Res. Part E 45 (5), 821–829.
Wang, Y.-W., Lin, C.-C., 2013. Locating multiple types of recharging stations for battery-powered electric vehicle transport. Transp. Res. Part E 58 (0), 76–87.
Wang, Y.-W., Wang, C.-R., 2010. Locating passenger vehicle refueling stations. Transp. Res. Part E 46 (5), 791–801.
Yang, X., Ramamurthy, B., 2005. Sparse regeneration in translucent wavelength-routed optical networks: architecture, network design and wavelength routing. Photonic Network Commun. 10 (1), 39–53.
Yetginer, E., Karaşan, E., 2003. Regenerator placement and traffic engineering with restoration in gmpls networks. Photonic Network Commun. 6 (2), 139–149.
Yıldız, B., Arslan, O., Karaşan, O.E., 2016. A branch and price approach for routing and refueling station location model. Eur. J. Oper. Res. 248 (3), 815–826.
Yıldız, B., Karaşan, O.E., 2015. Regenerator location problem and survivable extensions: a hub covering location perspective. Transp. Res. Part B 71, 32–55.
Yıldız, B., Karaşan, O.E., 2017. Regenerator location problem in flexible optical networks. Oper. Res. (to appear).