Network Design with Service Requirements: Scaling-up the Size of Solvable Problems

Published Version:
Naga V. C. Gudapati, Enrico Malaguti, Michele Monaci (2022). Network Design with Service Requirements: Scaling-up the Size of Solvable Problems. INFORMS JOURNAL ON COMPUTING, 34(5), 2571-2582 [10.1287/ijoc.2022.1200].

Availability:
This version is available at: https://hdl.handle.net/11585/900755 since: 2022-11-10

Published:

DOI: http://doi.org/10.1287/ijoc.2022.1200

Network Design with Service Requirements: Scaling-up the Size of Solvable Problems

Naga V. C. Gudapati, Enrico Malaguti, Michele Monaci

Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione "Guglielmo Marconi",
Università di Bologna, Viale del Risorgimento 2, 40136 Bologna, Italy,
{chaithanya.gudapati, enrico.malaguti, michele.monaci}@unibo.it

The final version of this paper was published on the INFORMS Journal on Computing as:

Naga V. C. Gudapati, Enrico Malaguti, Michele Monaci (2022) Network Design with Service Requirements: Scaling-up the Size of Solvable Problems. INFORMS Journal on Computing 34(5):2571-2582.
https://doi.org/10.1287/ijoc.2022.1200

Network Design with Service Requirements: Scaling-up the Size of Solvable Problems

Naga V. C. Gudapati, Enrico Malaguti, Michele Monaci
Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione "Guglielmo Marconi",
Università di Bologna, Viale del Risorgimento 2, 40136 Bologna, Italy,
{chaithanya.gudapati, enrico.malaguti, michele.monaci}@unibo.it

Network design, a cornerstone of mathematical optimization, is about defining the main characteristics of a network satisfying requirements on connectivity, capacity, and level-of-service. It finds applications in logistics and transportation, telecommunications, data sharing, energy distribution, and distributed computing. In multi-commodity network design, one is required to design a network minimizing the installation cost of its arcs and the operational cost to serve a set of point-to-point connections. The definition of this prototypical problem was recently enriched by additional constraints imposing that each origin-destination of a connection is served by a single path satisfying one or more level-of-service requirements, thus defining the *Network Design with Service Requirements*. These constraints are crucial, e.g., in telecommunications and computer networks, in order to ensure reliable and low-latency communication. In this paper we provide a new formulation for the problem, where variables are associated with paths satisfying the end-to-end service requirements. We present a fast algorithm for enumerating all the exponentially-many feasible paths and, when this is not viable, we provide a column generation scheme that is embedded into a branch-and-cut-and-price algorithm. Extensive computational experiments on a large set of instances show that our approach is able to move a step further in the solution of the Network Design with Service Requirements, compared with the current state-of-the-art.

Key words: network design; multi-commodity flow; service requirements; branch-and-cut-and-price algorithm; budget-constrained shortest path; labeling algorithm

1. Introduction

Network design is a cornerstone of mathematical optimization, as witnessed by the large amount of literature on this topic. Indeed, historically it finds applications in logistics and transportation of goods and persons (Magnanti and Wong [1984]) and, more recently, in telecommunications, data sharing, energy distribution, and distributed computing (Gendron et al. [1999]).

Network design is about defining the main characteristics of a network satisfying requirements on connectivity, capacity, and level-of-service. Setting up the network induces some installation cost, while additional costs are incurred when operating the service. It is quite common that a larger cost in the first term yields to a reduction in the latter, and vice-versa. Thus, the problem requires to find an equilibrium in the trade-offs between the installation and the operational costs.

A prototypical network design problem is the multi-commodity network design, in which one is required to design a network minimizing the installation cost of its arcs and the operational cost to serve a set of point-to-point connections, denoted as commodities. The solutions to this problem, however, can results in networks for which some commodities experience a low-quality connection with respect to some metric, e.g., distance or number of intermediate network nodes (hops) between origin and destination. In some applications, this is a critical issue: for example in telecommunications, a common requirement consists of limiting the number of hops between origin and destination of any connection, as this has a direct effect on the latency of the communication. Similarly, in transportation networks, it is common to limit the distance traveled between origin-destination pairs, in particular when dealing with a public transport service or when transporting perishable goods.

Recently, Balakrishnan et al. [2017] filled this gap and introduced the *Network Design with Service Requirements* (NDSR), a network design problem in which additional constraints impose that each origin-destination is served by a single path satisfying one or more level-of-service requirements. More specifically, each path must satisfy a maximum length with respect to a number of specified metrics. The problem asks to select some arcs to include in the network and to define, for each commodity, a path on the selected arcs and taking into account the mentioned level-of-service requirements. The objective is to minimize a cost function consists of minimizing the total installation cost of the network arcs and of the operational cost of the selected paths. In that paper, the authors show that a model based on arc-flow variables can be hard to solve even for moderate-sized networks. Hence, through a wide polyhedral analysis they derive several families of valid inequalities, which can be exploited to strengthen the formulation. The resulting model, combined with an effective heuristic algorithm, allows to tackle larger instances of the problem.

In this manuscript, we propose a new model where variables are associated with paths satisfying the end-to-end service requirements. This way, many of the weaknesses of the arc-flow formulation are naturally overtaken without the need to recur to cut separation techniques. This desirable property comes at the cost of a formulation which is much larger, involving an exponential number of path variables. However, we show that for all the instances considered by Balakrishnan et al. [2017], we are indeed capable of quickly enumerating all the variables of the new formulation, thanks to an effective labelling algorithm, and to solve to proven optimality a much larger set of instances using a general-purpose ILP solver. In particular, our approach allows to solve a relevant fraction of the large instances introduced by Balakrishnan et al. [2017], and to compute near-optimal solutions in the remaining cases, showing that the algorithm scales efficiently to larger size of the network. In addition, we provide a new set of instances for which enumerating all the paths is not viable; for solving these large instances, we present a column generation scheme that is embedded into a full branch-and-cut-and-price algorithm.

The paper is organized as follows. In the remainder of this section, we review some literature related to the problem at hand. Section 2 formally describes the problem, reviews a mathematical formulation from the literature, introduces a novel formulation, and compares the two models. Section 3 presents a solution approach based on branch-and-cut-and-price, describing column generation and the addition of valid inequalities. Section 4 computationally compares the performances of the proposed algorithms with state-of-the-art approach on test instances from the literature. Finally, in Section 5 we present some conclusions.

Literature Review: There is a wide literature on network design problems, and many surveys have been published on these topics, see, e.g., Magnanti and Wong [1984], Crainic [2000], and Wieberneit [2007]. Depending on the specific application, different variants of these problems were considered. A notable field of research involves the design of reliable and survivable networks, that has become a major objective for telecommunication operators (see, Kerivin and Mahjoub [2005]). In this context, one is required to define a robust network preserving a given connectivity level under possible failure of certain network components. There exist several ways to express the network robustness. Under a stochastic paradigm, the network is required to remain operative either with a large probability (Song and Luedtke [2013], Barrera et al. [2015]) or after some recourse action has been implemented (Ljubić et al. [2017]). Alternatively, more conservative approaches, imposing explicit redundancy in the definition of the network, have been considered in the literature; typically, one is required to design a network having two (edge) disjoint paths for each commodity (Magnanti and Raghavan [2005], Andreas and Smith [2008], Andreas et al. [2008], and Balakrishnan et al. [2009]), while Grötschel et al. [1995] considered the case in which higher connectivity requirements are imposed.

Another class of related problems arises in applications where explicit constraints are imposed on the characteristics of each path. A common requirement to guarantee the required quality of service is to limit the number of hops of each path; this problem has been introduced by Balakrishnan and Altinkemer [1992], while Gouveia [1998] presented a strong flow formulation that has been later adopted for many hop-constrained network design problems. In some cases, the resulting network is required to have a special structure (typically, a tree), or survivability considerations have been added to the problem definition; see, e.g., Botton et al. [2013] and Gouveia et al. [2015].

Our problem is closely related to the class of multi-commodity flow problems (Kennington [1978]) in which the network is given and commodities compete for the use of the arcs, which have a limited capacity. A branch-and-cut-and-price approach using path variables has been proposed by Barnhart et al. [2000]. Another relevant special case of NDSR arises when network design has to be defined for a unique commodity, and a single metric has to be considered. The resulting budget constrained shortest path problem, introduced by Joksch [1966], is an NP-hard problem, and turns out to be a simplified version of a subproblem that we have to solve for generating columns, which takes more than one metric into consideration.

Finally, on the applications side, end-to-end service requirements have been considered by Barnhart and Schneur [1996], Kim et al. [1999], and Armacost et al. [2002], where express delivery of parcels is optimized. Though service time is a key aspect in these applications, the special structure of the networks allows to avoid to explicitly impose these constraints.

2. Problem description and formulation

We now give a formal definition of the problem addressed in this paper. We are given a directed graph $G = (\mathcal{V}, \mathcal{A})$ where $\mathcal{V}$ is the node set and $\mathcal{A}$ is the arc set, and a set $\mathcal{K}$ of commodities. Each commodity $k \in \mathcal{K}$ has associated a source node s^k and a sink node t^k. For each arc $a \in \mathcal{A}$ there is an activation cost F_a; in addition, using an arc a for a commodity k induces a flow cost c_a^k. The problem asks to send, for each commodity k, one unit of flow on a single path p^k from the source to the sink, by determining a set of arcs and the routing of the flows so that the sum of the activation and flow costs is a minimum. In addition, there is a set $\mathcal{M}$ of metrics, that determines the feasibility of the path associated with a given commodity k: for each metric $m \in \mathcal{M}$, we denote by w_a^{km} the weight of arc a with respect to the metric, and require that the sum of the weights on arcs in p^k does not exceed a given upper limit W^{km}. We denote by $\mathbf{w}_a^k$ and $\mathbf{W}^k$ the corresponding m-dimensional vectors. For example, in a transportation context, one could consider two metrics: the first one counts the number of arcs, each one associated with a certain transportation mode, whereas the second measures the total transit time associated with the path. By bounding the weight of the path used by each commodity between its origin and destination with respect to both metrics, one may limit the maximum number of transhipments and maximum transit time.

Throughout the paper, we assume that the graph includes no multiple arcs. This assumption is without loss of generality, as multiple arcs with different costs or service consumption for a given pair of nodes can be handled by the addition of dummy nodes. In addition, we assume that, for each commodity, at least one feasible path exists, since otherwise the problem is clearly infeasible.

The problem reduces to the NP-hard budget-constrained shortest path (see, Garey and Johnson [1979]) when there is a single commodity and a single metric. This shows that the problem is NP-hard.

The next section reports a descriptive formulation that has been proposed in the literature, whereas Section 2.2 introduces a novel formulation that will be used in our solution scheme.

2.1. Arc-flow formulation

The following formulation has been proposed by Balakrishnan et al. [2017] and makes use of activation variables and flow variables. All variables are binary and have the following meaning:

$$z_a = \begin{cases} 1 & \text{if arc } a \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \qquad (a \in \mathcal{A})$$

$$y_a^k = \begin{cases} 1 & \text{if commodity } k \text{ is routed on arc } a \\ 0 & \text{otherwise} \end{cases} \qquad (a \in \mathcal{A}, k \in \mathcal{K})$$

Then, the NDSR can be modelled using the following Integer Linear Programming (ILP) formulation:

$$\min \quad \sum_{a \in \mathcal{A}} F_a z_a + \sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}} c_a^k y_a^k \tag{1a}$$

$$\text{subject to} \quad \sum_{a \in \delta^+(v)} y_a^k - \sum_{a \in \delta^-(v)} y_a^k = \begin{cases} +1 & v = s^k \\ -1 & v = t^k \\ 0 & v \in \mathcal{V} \setminus \{s^k, t^k\} \end{cases} \qquad k \in \mathcal{K} \tag{1b}$$

$$\sum_{a \in \mathcal{A}} w_a^{km} y_a^k \leq W^{km} \qquad k \in \mathcal{K}, m \in \mathcal{M} \tag{1c}$$

$$y_a^k \leq z_a \qquad a \in \mathcal{A}, k \in \mathcal{K} \tag{1d}$$

$$z_a \in \{0,1\} \qquad a \in \mathcal{A} \tag{1e}$$

$$y_a^k \in \{0,1\} \qquad a \in \mathcal{A}, k \in \mathcal{K}. \tag{1f}$$

The objective function minimizes the sum of the activation and flow costs. Constraints (1b) impose flow conservation for each commodity and node, whereas (1c) concern feasibility of the paths with respect to the metrics, and inequalities (1d) force the activation of arcs that are used for routing a positive flow. Finally (1e) and (1f) define the domain of the variables. The arc-flow formulation has a polynomial size, as it includes $(|\mathcal{K}| + 1)\,|\mathcal{A}|$ variables and $|\mathcal{K}|\,(|\mathcal{V}| + |\mathcal{A}| + |\mathcal{M}|)$ constraints.

2.2. Path-based formulation

The novel ILP formulation that we propose includes the same binary activation variables of model (1a)–(1f), that select the arcs to be activated, whereas flow variables are replaced by path variables that are defined as follows. Let $\mathcal{P}^k$ be the set of all feasible paths for commodity k. For each commodity k and each path $p \in \mathcal{P}^k$, let us introduce a binary path variable x_p with the following meaning:

$$x_p = \begin{cases} 1, & \text{if commodity } k \text{ is routed along path } p \\ 0 & \text{otherwise} \end{cases} \qquad (k \in \mathcal{K}, p \in \mathcal{P}^k)$$

Let c_p be the flow cost of the path p for commodity k, defined as the sum of the flow costs of all the arcs in p. The problem can thus be modelled as follows:

$$\min \quad \sum_{a \in \mathcal{A}} F_a z_a + \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}^k} c_p x_p \tag{2a}$$

$$\text{subject to} \quad z_a - \sum_{p \in \mathcal{P}^k : a \in p} x_p \geq 0 \qquad a \in \mathcal{A}, k \in \mathcal{K} \tag{2b}$$

$$\sum_{p \in \mathcal{P}^k} x_p = 1 \qquad k \in \mathcal{K} \tag{2c}$$

$$z_a \in \{0,1\} \qquad a \in \mathcal{A} \tag{2d}$$

$$x_p \in \{0,1\} \qquad p \in \mathcal{P}^k, k \in \mathcal{K}. \tag{2e}$$

The objective function minimizes activation costs and flow costs, which are here expressed in terms of path variables. Constraints (2b) are the counterpart of (1d), enforcing activation of arcs that are used by a path. Constraints (2c) ensure that, for every commodity, one feasible path is selected. Finally, (2d) and (2e) define the domain of the variables.

As it typically happens in path-based models, constraints defining the feasibility of the paths for a given commodity k are implicitly included in the definition of set $\mathcal{P}^k$, and hence the formulation does not include a direct counterpart of constraints (1c).

Observation 1 *The model obtained by relaxing integrality requirement* (2e) *admits an optimal integer solution.*

Proof: Assume that an optimal solution for the relaxation is given. For a given choice of the z variables, the x variables associated with a commodity do not interact with those of a different commodity. Thus, we concentrate on a single commodity, say k, and assume that more than one path is selected for that commodity, the sum of the values of the associated path variables being 1. By optimality of the initial solution, all the selected paths must have the same cost. Hence, by increasing the value of one path variable to 1 and setting to 0 all the remaining ones, we obtain a solution that has the same cost as the original one. □

Variable enumeration: We first observe that the path-based formulation has $O(2^{|\mathcal{V}|})$ variables and $(|\mathcal{A}|+1)\,|\mathcal{K}|$ constraints, i.e., its size can be exponential in the size of the instance. We now introduce an algorithm for enumerating all path variables; however, for large graphs, enumerating all paths can be challenging, and one may have to resort to column generation techniques, that will be discussed in Section 3.

There is a vast body of literature addressing the problem of enumerating all paths (and the closely related problem of enumerating all cycles) in a given graph. According to Mateti and Deo [1976], existing approaches can be classified into search space algorithms, backtrack algorithms, and algorithms based on the power of the adjacency matrix. Our enumeration Algorithm 1 belongs to the second category, and considers one commodity k at a time and defines all simple paths from s^k to t^k that satisfy resource constraints under all metrics. The algorithm is inspired by the labelling method proposed by Dumitrescu and Boland [2003] for the budget-constrained shortest path problem. In our algorithm, each label $\ell = \{u, c, \mathbf{w}\}$ represents a path from s^k to u having cost c and using w_m units of resources under each metric m. Each label is generated as unmarked, meaning that it has to be expanded, and then it is marked when considered for expansion. Expansion of a label ℓ associated with a node u consists in appending an arc $a = (u, v)$ to the current path. To this aim, we consider all the outgoing arcs from u and, for each neighbor node v not yet belonging to path ℓ, we check whether using the current label for reaching v preserves feasibility with respect to the metrics. In this case, we define a new label $\ell' = \{v, c + c_a^k, \mathbf{w} + \mathbf{w}_a^k\}$, i.e., we update the path cost and resource usage when using the current label for reaching v. Eventually, node v is inserted in set T, that includes all nodes associated with unmarked labels. The algorithm terminates when $T = \emptyset$, meaning that no label can be further expanded, and returns all labels associated with node t^k. Although a node can be inserted in and removed from T more than once, the convergence of the algorithm is ensured by requiring simple paths, which is checked in line 9.

The above algorithm can be improved by pre-computing, for each metric $m \in \mathcal{M}$, the shortest path from each node to t^k when the cost of each arc a is given by w_a^{km}. This figure can be used when checking feasibility of the new label in line 9: by adding this term to the left-hand-side of the inequality, we avoid generating labels that could not be feasibly expanded to node t^k.

2.3. Models comparison

In this section we compare the two formulations in terms of their linear relaxations.

Observation 2 *Any feasible solution for the linear relaxation of the path-based formulation can be mapped to a feasible solution of the same cost of the linear relaxation of the arc-based formulation, whereas the opposite does not hold.*

Algorithm 1: Compute all feasible paths for a fixed commodity

Input : k

$s := s^k, t := t^k, T := \{s\}, c := 0, \mathbf{w} := \mathbf{0}$
Define an unmarked label $\ell := \{s, c, \mathbf{w}\}$ for node s
while $T \neq \emptyset$ **do**
    pick any $u \in T$
    $T := T \setminus \{u\}$
    foreach unmarked label $\ell = \{u, c, \mathbf{w}\}$ associated with node u
        mark label ℓ
        foreach $a = (u, v) \in \delta^+(u)$
            if *(*$v \notin$ *path* ℓ*)* and $(\mathbf{w} + \mathbf{w}_a^k \leq \mathbf{W}^k)$
                define an unmarked label $\ell' = \{v, c + c_a^k, \mathbf{w} + \mathbf{w}_a^k\}$
                if *(*$v \neq t$*)*
                    $T := T \cup \{v\}$
return all labels associated with node t

Proof: Let z^*, x^* be a feasible solution of the linear relaxation of the path-based formulation. We now define a solution $\widetilde{z}, \widetilde{y}$ that is feasible for the linear relaxation of the arc-based formulation and has the same cost. First, we set $\widetilde{z} = z^*$. Then, for each arc $a \in \mathcal{A}$ and commodity $k \in \mathcal{K}$, we set

$$\widetilde{y}_a^k = \sum_{p \in \mathcal{P}^k : a \in p} x_p^*.$$

It is straightforward to check that flow conservation constraints (1b) and feasibility requirements (1c) with respect to the metrics are satisfied as y variables are obtained as combination of feasible paths, whereas constraints (1d) are implied by (2c) and by the definition of $\widetilde{y}_a^k$. The equivalence of the costs follows from the definition of the cost of each path.

Figure 1 gives a small numerical example showing that the counterpart does not hold. The instance has no flow costs, a single commodity, and a single metric, for which the capacity is $W = 2$. For each arc we report the activation cost and the weight with respect to the metric. While there is a unique feasible path $p = \{(s, t)\}$ having cost 1, an optimal solution to the linear relaxation of the arc-based formulation is given by $y_{s1} = y_{1t} = y_{st} = 1/2$ having cost $1/2$. □

The observation shows that the path-based formulation dominates the arc-based one in terms of tightness of the associated linear relaxations.

The structure of feasible solutions for the linear relaxation of the arc-based formulations was analyzed by Balakrishnan et al. [2017], showing that fractional solutions may arise for two main reasons:

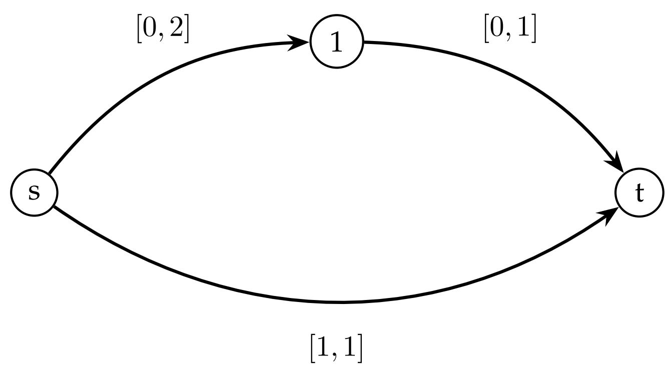

Figure 1 Simple example for which the path-based formulation dominates the arc-flow formulation.

- for a given commodity, the model may route part of the flow on a path that is less expensive but infeasible with respect to the metric requirements (see again Figure 1);
- arc activation variables can be set at a fractional value to allow sharing the activation cost of some arcs among different paths associated with different commodities.

Accordingly, Balakrishnan et al. [2017] introduced different families of valid inequalities to cut some of these solutions. The first type of fractionalities do not appear in the path-based formulation, in which feasibility of the paths is enforced when defining the variables; thus, adding similar inequalities would be useless. On the other hand, the second type of fractionality may affect the path-based formulation as well, as shown in Figure 2. In this example, there are three commodities, no flow costs and activation costs equal to one for arcs $(3,6),(4,7),(5,8)$ and zero for the remaining arcs. The figure shows an optimal solution of the linear relaxation of the path-based formulation, where the flow of each commodity is split into two paths, the costly arcs are activated at value 0.5 and the resulting cost is $3/2$. On the other hand, any integer feasible solution has a cost at least equal to 2. For this reason, in our approach we consider the possibility to add some classes of valid inequalities of the second type.

3. Branch-and-cut-and-price approach

In this section, we introduce an exact algorithm based on the path-formulation that can be used when enumerating all paths is unpractical. The algorithm adopts a branch-and-bound strategy and solves, at each node, the linear relaxation of the model by means of column generation techniques. The basic scheme is possibly enriched by the addition of valid inequalities, that do not change the structure of the method, thus resulting in a robust branch-and-cut-and-price algorithm.

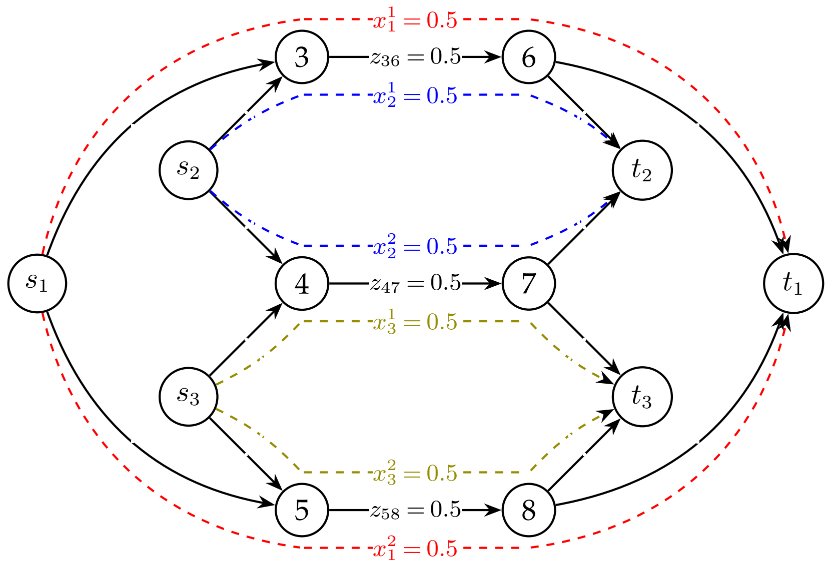

Figure 2 **Fractional solution of the linear relaxation of the path-based formulation.**

3.1. Column generation and labelling

Column generation is an iterative scheme used for solving linear models with an exponentially large number of variables. At each iteration, a *restricted master problem* including a subset of the variables is solved, and its dual solution is used to determine new variables (if any) that have a negative reduced cost and can be added to the formulation in order to converge to an optimal solution. The absence of variables with negative reduced cost is a certificate that the current restricted master problem, though having a limited number of variables, includes an optimal solution to the complete formulation. The reader is referred to Desrosiers and Lübbecke [2005] for a comprehensive discussion about this technique.

In our setting we assume without loss of generality that constraints (2c) are rewritten as inequalities. At each iteration, the restricted master includes all the z variables, and a non-empty subset $\widetilde{\mathcal{P}^k} \subseteq \mathcal{P}^k$ of path variables for each commodity k (notice that by construction the restricted master always includes a feasible solution). Assume that the restricted master has been solved to optimality, and let γ_a^k and ρ^k be optimal non-negative dual variables associated with constraints (2b) and (2c), respectively. The *reduced cost* for a path variable x_p for a commodity k is given by

$$\bar{c}_p = c_p + \sum_{a \in p} \gamma_a^k - \rho^k = \sum_{a \in p} \left(c_a^k + \gamma_a^k\right) - \rho^k = \sum_{a \in p} \widetilde{c}_a^k - \rho^k,$$

where the arc costs are $\widetilde{c}_a^k = c_a^k + \gamma_a^k$. Thus, the *pricing problem* for a given commodity k is to find a feasible path whose reduced cost is negative, and can be formulated as a *budget-constrained shortest path problem* under costs $\widetilde{c}_a^k$ and resources defined by the metrics. If the cost of this shortest path is strictly smaller than ρ^k, the corresponding path variable is added to the restricted master, and the process is iterated; if no path variable is generated for any commodity, the optimal solution of the current restricted master is an optimal solution for the linear relaxation of the problem.

Solution of the budget-constrained shortest path problem: Enumeration Algorithm 1 can be modified to compute the shortest path under resource constraints for a given commodity k, a problem which is NP-hard even if the graph is acyclic and $|\mathcal{M}| = 1$ (see, Garey and Johnson [1979]). The resulting Algorithm 2 differs from the enumeration one starting from line 11, where a dominance check aimed at avoiding expansion of suboptimal paths is introduced. More precisely, label ℓ' is dominated by another label ℓ'' associated with the same node if its cost and its resource usage are larger then or equal to the cost and usage of ℓ''. In this case ℓ' is marked. Vice-versa, it may also happen that ℓ' dominates ℓ'', in which case we mark ℓ''. Node v is inserted in set T only if label ℓ' remains unmarked. The algorithm returns a unique path, corresponding to the label with minimum cost among all those associated with node t^k.

Algorithm 2: Compute a constrained shortest path for a fixed commodity

```
   Input : k
1  s := s^k, t := t^k, T := {s}, c := 0, w := 0
2  Define an unmarked label ℓ := {s, c, w} for node s
3  while T ≠ ∅ do
4  |  pick any u ∈ T
5  |  T := T \ {u}
6  |  foreach unmarked label ℓ = {u, c, w} associated with node u
7  |  |  mark label ℓ
8  |  |  foreach a = (u, v) ∈ δ+(u)
9  |  |  |  if (v ∉ path ℓ) and (w + w_a^k ≤ W^k)
10 |  |  |  |  define an unmarked label ℓ' = {v, c + c̃_a^k, w + w_a^k}
11 |  |  |  |  if (ℓ' is dominated by a label ℓ'' associated with node v)
12 |  |  |  |  |  mark label ℓ'
13 |  |  |  |  if (ℓ' dominates a label ℓ'' associated with node v)
14 |  |  |  |  |  mark label ℓ''
15 |  |  |  |  if (v ≠ t) and (ℓ' is unmarked)
16 |  |  |  |  |  T := T ∪ {v}
17 return the unmarked label with minimum cost c associated with node t
```

3.2. Branching scheme

In our branching scheme we always select a z variable for branching. According to Observation 1, at each node where all the z variables attain integer values, there exists an optimal solution in which all the x variables are integer as well. Notice that this is the solution returned by solving the restricted master problem by means of the simplex algorithm.

A positive effect of this branching strategy is that it does not affect the structure of the pricing subproblem. This is a crucial property for designing an effective branch-and-price algorithm, as it allows to solve the column generation subproblem throughout all the branching tree by means of the same effective labelling algorithm used at the root node. Clearly, imposing $z_a = 1$ for some $a \in \mathcal{A}$ has no direct effect in the pricing. Conversely, when imposing $z_a = 0$, in the pricing subproblem we simply forbid the use of arc a when generating new path variables, which can be easily handled by setting $\mathcal{A} = \mathcal{A} \setminus \{a\}$.

3.3. Adding valid inequalities

In order to tighten the formulation and increase the dual bound at each node, we can add valid inequalities that cut fractional solutions in which arc activation variables are set at a fractional value to allow sharing the activation cost of some arcs among different paths.

To this aim, we adapt to our model some of the inequalities introduced by Balakrishnan et al. [2017] for the arc-flow formulation. These inequalities are obtained by analyzing the structure of the graph G and by deriving relationships between pairs of arcs (a,b) when routing the flow of a commodity k, namely:

- *OR* relationships, occurring when no more than one arc of pair (a,b) can be used to route flow from s^k to t^k;
- *IF* relationships, occurring when the flow through arc a must also be routed through b; and
- *CUT* relationship, occurring when at least one between a and b must be used to route the flow.

These relationships are then used to derive conditions that link the activation variable of an arc with the flow variables associated with the same arc and different commodities. By using the arc-flow variables, all these inequalities have the following general structure

$$\sum_{(a,k)\in C} z_a - \sum_{(a,k)\in C} y_a^k \geq q,$$

where C is a set of arc-commodity pairs and q is a scalar number.

By translating these conditions in terms of the path variables, we obtain

$$\sum_{(a,k)\in C} z_a - \sum_{(a,k)\in C} \sum_{p\in\mathcal{P}^k : a\in p} x_p \geq q \tag{3}$$

which can be enforced in the path-based formulation. For example, the following

$$x_{0,0} + x_{0,1} + x_{1,0} + x_{1,1} + x_{2,0} + x_{2,1} <= z_{36} + z_{47} + z_{58} + 1$$

is a valid inequality for the graph depicted in Figure 2. This constraint has been obtained by mapping an OR relationship (in particular, a 3-OR inequality, see Balakrishnan et al. [2017]) in terms of the path variables. It is easy to see that such an inequality is violated by the fractional solution depicted in the example.

As it happens for the branching conditions, the addition of the inequalities above does not affect the structure of the pricing problem at a generic node of the branching tree. Indeed, for a given commodity k, constraint (3) only affects those paths that contain an arc a such that pair $(a,k) \in C$. For each such path, the reduced cost of the associated variable is thus $\bar{c}_p = \sum_{a \in p} \left(c_a^k + \gamma_a^k\right) + \phi^C - \rho^k$ where ϕ^C is the dual variable associated with constraint (3). More in general, given a collection $\mathcal{C}$ of inequalities, the reduced cost of a path associated with commodity k is

$$\bar{c}_p = \sum_{a \in p} \Big(c_a^k + \gamma_a^k + \sum_{C \in \mathcal{C}:(a,k) \in C} \phi^C\Big) - \rho^k$$

Hence, the only effect of additional inequalities on the shortest path computation is on the definition of arc costs $\tilde{c}_a^k$, which now include the dual variables of these constraints as well.

This allows us to solve the column generation subproblem with no modification of the labelling algorithm even after the addition of valid inequalities. The resulting algorithm is then a robust branch-and-cut-and-price.

4. Computational experiments

In our computational experiments we explore three directions. First, we compare the computational performance of the path-based formulation with the arc-based formulation. Our second order of business is to determine the features of the instances for which full enumeration of all feasible paths is possible, and when instead one has to resort to column generation. In this case, the solver cannot be used as a black box, and the addition of valid inequalities may be an effective option for accelerating the solution process. Finally, we evaluate the effect of adding valid inequalities to the path-based formulation, in terms of bound given by the linear relaxation and overall performance of the algorithm.

4.1. Instances from the literature and state-of-the-art

We now describe a benchmark of instances that has recently been introduced by Balakrishnan et al. [2017], who kindly provided us the code for generating the numerical data. Each instance is characterized by the following parameters: the number of nodes $|\mathcal{V}|$, number of arcs $|\mathcal{A}|$, and number of commodities $|\mathcal{K}|$. Nodes are randomly located on a rectangular grid and are connected by a spanning arborescence; then, $|\mathcal{A}| - |\mathcal{V}| + 1$ arcs are added to the arc set, making sure that the resulting network contains one directed path for each pair of nodes. The source and terminal node for each commodity are randomly selected in $\mathcal{V}$. The activation cost of each arc depends on the euclidean distance between the two endpoints and on a random parameter. A parameter γ governs the ratio between flow costs and activation costs. Coefficients $\mathbf{w}_a^k$ for a given arc $a \in \mathcal{A}$ are negatively correlated to the activation cost F_a through a parameter β and a random term. All instances consider $|\mathcal{M}| = 2$ metrics. Weight limits for each commodity k and each metric equal the length (using arc weights as lengths) of the q-th shortest path from s^k to t^k, where q is a random parameter having uniform distribution in an interval of size ΔQ centered in Q_{avg}. A particular combination of network size ($|\mathcal{V}|$, $|\mathcal{A}|$, and $|\mathcal{K}|$), cost structure and service requirements (β, γ, Q_{avg}, and ΔQ) is referred to as a scenario. Overall, Balakrishnan et al. [2017] defined 18 scenarios: the first seven scenarios share the same default values of the parameters for cost structure and service requirements, while considering varying network sizes ranging from 30 nodes, 120 arcs, and 90 commodities to 50 nodes, 250 arcs, and 150 commodities. Scenarios 8-15 are all defined with a fixed network size ($|\mathcal{V}| = 50$, $|\mathcal{A}| = 200$, and $|\mathcal{K}| = 150$) and different cost structure and service requirements. Finally, the last 3 scenarios have the default values of the parameters defining cost structure and service requirements and are characterized by larger size of the network, up to 80 nodes, 320 arcs, and 240 commodities. For each scenario, five instances were generated, for a total of 90 instances. In the rest of the paper, we will refer to each scenario as $|\mathcal{V}|/|\mathcal{A}|/|\mathcal{K}|/\beta\,\gamma\,Q_{avg}\,\Delta Q$ where the last four parameters take values in $\{L, M, H\}$ to denote low, medium and high figures, respectively.

Table 1 reports the state-of-the-art results for what concerns the optimal solution of NDSR, obtained by the best-performing algorithm proposed by Balakrishnan et al. [2017], denoted as `composite` in what follows. This algorithm implements a branch-and-cut scheme built on top of the general-purpose ILP solver CPLEX 12.5.1 for solving the arc-flow formulation. These results refer to the 90 instances derived from the 18 scenarios described above; instances are grouped by scenario, i.e., every row reports aggregate results for five instances and provides the number of instances solved to proven optimality, the average percentage gap, and the average computing time (in seconds, with respect to instances that are solved to optimality only). For a given instance of the problem, denoting by L and U be the best lower and upper bound, respectively, the resulting percentage gap is computed as $100\frac{U-L}{U}$. All these figures are taken from Balakrishnan et al. [2017], and correspond to experiments executed on a Intel core i5 using an integrality gap for early termination equal to 0.1%. As a consequence of this tolerance, for some scenarios the algorithm solves all the associated 5 instances to optimality though returning a strictly positive percentage gap. Detailed computational results are available for the instances of the first 7 scenarios, whereas only aggregated results are reported for the remaining scenarios.

	scenario	composite # opt	% gap	time
1	30/120/90/MMMM	5	0.02	175
2	40/160/120/MMMM	4	0.18	133
3	50/150/150/MMMM	5	0.08	230
4	50/200/100/MMMM	4	0.43	1055
5	50/200/150/MMMM	2	1.33	350
6	50/200/200/MMMM	3	0.45	735
7	50/250/150/MMMM	1	2.61	2631
8	50/200/150/LMMM		0.50	
9	50/200/150/HMMM		2.00	
10	50/200/150/MLMM		0.90	
11	50/200/150/MHMM		0.10	
12	50/200/150/MMLM		0.10	
13	50/200/150/MMHM		1.90	
14	50/200/150/MMML		0.70	
15	50/200/150/MMMH		0.70	
16	60/240/180/MMMM		0.70	
17	70/280/210/MMMM		2.50	
18	80/320/240/MMMM		2.40	
	summary	45*	0.98	

Table 1 State-of-the-art for the exact solution of NDSR.

The results in Table 1 show that `composite` is able to solve 24 instances out of 35 in the first 7 scenarios, with average percentage gap equal to 0.73. For only two scenarios, this approach is able to solve all the corresponding 5 instances to optimality. Overall, the algorithm solves 45 instances out of 90, with an average percentage gap around 1%, showing that this benchmark is quite challenging for the state-of-the-art computational approach.

4.2. Results of the path-based formulation

Although the original benchmark is not available, we generated 90 instances (5 for each scenario) having the same parameters used by Balakrishnan et al. [2017] by running their instance generator. All our algorithms were implemented in C++ and were run on an AMD Ryzen Threadripper 3960X running at 3.8 GHz in single-thread mode, with a time limit of 1 hour per instance. Both the arc-flow and the path-based formulations were solved using Gurobi version 9.1.1 as ILP solver, whereas the branch-and-cut-and-price was implemented on top of the SCIP optimization suite (version 7.0.1 with its default SoPlex solver), which allows to embed a column generation scheme within the enumeration process (see Gamrath et al. [2020]). Our source code and instances are publicy available at `https://github.com/CGudapati/NDSR-Code`.

Table 2 gives the results of our experiments and compares the following approaches:

- `base-model` corresponds to the direct application of general-purpose ILP solver Gurobi to the arc-flow formulation;
- `all-path` denotes the algorithm obtained by enumerating all feasible paths through Algorithm 1 and solving the resulting path-based formulation using the Gurobi ILP solver. This

approach does not include cut separation nor column generation, allowing us to use the solver as a black box, so as to exploit its full capabilities.

Table 2 gives the same information as Table 1. In addition, for algorithm `all-path` we report the number of path variables enumerated by the labelling algorithm. The enumeration time is always very small (at most 0.5 seconds) and it is included in the computing time of the algorithm.

		base-model			all-path			
	scenario	# opt	% gap	time	# opt	% gap	time	# path
1	30/120/90/MMMM	5	0.00	446.75	5	0.00	1.41	895
2	40/160/120/MMMM	4	0.21	1352.68	5	0.00	7.66	1098
3	50/150/150/MMMM	5	0.00	776.90	5	0.00	3.24	1409
4	50/200/100/MMMM	2	1.36	2953.33	5	0.00	41.71	956
5	50/200/150/MMMM	1	5.93	2817.12	5	0.00	478.80	1455
6	50/200/200/MMMM	0	3.81	–	5	0.00	324.22	1898
7	50/250/150/MMMM	0	9.20	–	4	0.38	171.85	1388
8	50/200/150/LMMM	2	3.09	2455.09	5	0.00	32.90	1249
9	50/200/150/HMMM	0	12.02	–	3	1.64	365.97	1795
10	50/200/150/MLMM	0	9.02	–	5	0.00	639.56	1455
11	50/200/150/MHMM	1	6.60	3557.67	5	0.00	782.21	1455
12	50/200/150/MMLM	5	0.00	1009.59	5	0.00	1.51	804
13	50/200/150/MMHM	0	10.68	–	4	0.64	345.37	2085
14	50/200/150/MMML	0	6.26	–	5	0.00	305.79	1421
15	50/200/150/MMMH	1	2.79	1902.54	5	0.00	100.74	1153
16	60/240/180/MMMM	0	10.68	–	5	0.00	603.16	1711
17	70/280/210/MMMM	0	11.91	–	2	0.54	686.22	1961
18	80/320/240/MMMM	0	15.86	–	2	1.25	951.94	2307
	summary	26	6.08	1371.97	80	0.25	288.22	1472

Table 2 Results on generated benchmark.

Although a one-to-one comparison with Table 1 is not possible, these results confirm the outcome of the computational experiments reported by Balakrishnan et al. [2017] for the first seven scenarios, i.e., that `composite` outperforms the `base-model`, which can solve only small instances and has large percentage gaps for most unsolved scenarios. On the other hand, algorithm `all-path` solves all but one instance in the first seven scenarios, and has a percentage gap equal to 0.38 for the remaining instance. This is due to the fact that the formulation is tight and that, for these instances, the number of path variables does not grow up: this number is always smaller than 2000, which makes the model solvable with a limited computational effort. The performances of algorithm `all-path` remain satisfactory for instances in scenarios 8-15: the algorithm solves 37 of the 40 associated instances, and has an average gap equal to 0.28%. Finally, for very large instances (scenarios 16 to 18), the algorithm solves 9 instances out of 15 and has an average gap equal to 0.60%. Overall, our algorithm solves to proven optimality almost 90% of the instances with an average gap of 0.25%, thus considerably improving over the state-of-the-art.

4.3. Results on additional instances

The results in Table 2 show that, for those instances, the number of feasible paths is quite small. Thus, not surprisingly, the `all-path` approach is always the most performing approach. Our second set of experiments is aimed at evaluating the limits of applicability of explicit enumeration of all path variables, and the alternative use of the branch-and-price algorithm described in Section 3 when enumeration is unpractical. Hence, we generated additional instances derived from the instances in scenarios 1–7, in which the number of feasible paths is increasing. To minimize the number of parameters for defining the additional instances, we simply introduce a parameter $\alpha \geq 1$ that is used to scale each upper limit W^{km} for a commodity k and metric m. This has the effect to make less binding the constraints defining the feasibility of a path with respect to the metrics.

Table 3 reports aggregated results, summarizing 35 instances per line, obtained with different values of α ranging from 1.00 to 3.00. We compare the `base-model`, the `all-path` approach, and the branch-and-price algorithm and report, for each solution method, the number of optimal solutions, the average percentage gap and the average computing time (with respect to instances solved to optimality only). For `all-path` we also report the total number of feasible paths; this figure is averaged over all the 35 instances of a line, provided that enumeration of all paths was completed within the time limit for all the instances. Finally, for branch-and-price we give the average number of path variables that have been generated during the execution of the algorithm (with respect to instances solved to optimality only).

The results in Table 3 show that, for values of $\alpha < 2$, the total number of paths is still manageable (below 200,000) and `all-path` remains the best option. Conversely, for larger values of α, in many cases enumerating all path variables within the time limit is not possible or the path-based formulation has too many variables, and hence a method based on column generation is advisable. Indeed, for $\alpha = 2$, branch-and-price solves 21 instances compared to the 17 solved by

	`base-model`			`all-path`				branch-and-price			
α	# opt	% gap	time	# opt	% gap	time	# path	# opt	% gap	time	# path
1.00	17	2.93	1191.34	34	0.05	146.25	1300	31	0.38	401.20	1116
1.25	3	8.45	1680.47	21	1.45	410.74	6428	16	2.65	816.47	5896
1.50	6	6.01	976.83	20	1.72	514.40	33,178	14	2.64	667.09	10,816
1.75	9	3.97	903.57	19	1.71	480.65	169,286	17	2.15	539.52	11,209
2.00	14	2.40	798.33	17	1.97	449.87	855,441	21	1.55	830.85	10,110
2.25	18	1.53	613.12	14	–	673.01	–	23	1.19	522.94	9115
2.50	22	0.91	654.06	9	–	957.18	–	26	0.80	475.83	7743
2.75	23	0.70	554.46	5	–	998.54	–	27	0.68	577.48	7479
3.00	25	0.61	470.26	3	–	1627.80	–	28	0.58	628.28	7048

Table 3 Results on additional instances.

`all-path`, and this gap increases for larger values of α. Finally, we observe that the performances of the `base-model` as well improve for increasing α, which suggests that the problem is easier when feasibility constraints are not too demanding. This confirms the outcome of some observations by Balakrishnan et al. [2017] about the structure of optimal solutions of the linear relaxation of this formulation, as these solutions are allowed to use infeasible paths at a fractional level.

4.4. Strengthening the model

As already mentioned, the `composite` approach is based on a branch-and-cut algorithm in which the arc-flow formulation is iteratively strengthened by means of valid inequalities, designed to cut off infeasible solutions of the linear relaxation. Balakrishnan et al. [2017] showed that adding these inequalities is beneficial to the algorithm, in terms of value of the dual bound at the root node and number of instances that can be solved to optimality.

Our third set of experiments is thus aimed at evaluating the impact of adding valid inequalities to the path-based formulation. Table 4 gives the outcome of our experiments on instances in scenarios 1–7 for the branch-and-price approach without and with the addition of valid inequalities (branch-and-cut-and-price).

The table is organized in two parts. In the first one, we report the average percentage gap of the linear relaxation in the two configurations, and the associated computing time reported by SCIP. For the version of the algorithm with cuts, we borrowed from Balakrishnan et al. [2017] the following families of inequalities: 3OR, 1CUT-IF and 1OR-IF, obtained by combining three OR conditions, one CUT with one or more IF conditions, and one OR with one or more IF conditions, respectively. The reader is referred to Balakrishnan et al. [2017] for the definition of these inequalities as well as to their separation; additional inequalities from this paper showed to have a very marginal effect in our preliminary computational experiments. Separation is carried out at the root node until no violated cut is found, according to SCIP tolerance. The results in Table 4 confirm that the addition of valid inequalities produces a tighter formulation for which the dual

	linear relaxation				exact solution					
	without cuts		with cuts		branch-and-price			branch-and-cut-and-price		
scenario	% gap	time	% gap	time	# opt	% gap	time	# opt	% gap	time
1	5.38	0.26	3.22	31.39	5	0.00	17.90	5	0.00	52.00
2	4.77	0.48	2.43	102.27	5	0.00	63.27	5	0.00	164.91
3	2.79	0.49	1.24	102.82	5	0.00	46.46	5	0.00	146.63
4	6.11	0.70	3.79	186.62	5	0.00	380.83	5	0.00	454.94
5	6.58	1.52	4.00	286.55	3	1.11	220.35	3	0.98	419.83
6	5.02	1.60	2.64	357.34	4	0.58	250.27	4	0.48	549.92
7	7.31	2.25	4.37	600.73	4	0.99	2058.17	4	0.78	1702.62
summary	5.42	1.04	3.10	238.25	31	0.38	401.20	31	0.32	463.29

Table 4 Results on the addition of valid inequalities.

gap with respect to the optimum value is quite small, and reduced by 42% with respect to the formulation without cuts (from 5.42% to 3.10%). However, separating these inequalities is time consuming in practice, which prevents the exhaustive separation of the cuts in an enumerative approach.

For this reason, in the rightmost part of the table we consider a branch-and-cut-and-price algorithm, in which separation is embedded into the branch-and-price in a heuristic way as follows: cuts are added at the root node only, and at most 25 rounds of separation are performed. At each separation round, we consider in order 3OR, 1CUT-IF and 1OR-IF, and we stop the separation as soon as a valid inequality is obtained. The inequality is added to the restricted master problem which is then re-optimized. This heuristic approach is justified by some preliminary experiments on each family of inequalities, where we evaluated the computational effort required for deriving a valid inequality and the relative effect of the inequality on the dual bound. Remind that a nice property of our approach is that the addition of new cuts does not affect the structure of the pricing subproblem, yielding a robust branch-and-cut-and-price approach. For both branch-and-price and branch-and-cut-and-price we report the number of optimal solutions, the average percentage gap and the average computing time.

The results on the exact methods show that branch-and-cut-and-price solves the same number of instances as branch-and-price, and produces slightly better gaps for unsolved instances. Indeed, both algorithms solve 31 instances, the average percentage gaps being 0.38 (for branch-and-price) and 0.32 (for branch-and-cut-and-price). Despite adding valid cuts seems to be very effective in closing the gap at the root node, its limited contribution within an enumerative scheme is due to the computational overhead required for separating cuts and for solving larger models at each decision node.

5. Conclusions

We considered an NP-hard network design problem with end-to-end service requirements that play a fundamental role in many contexts, including telecommunications and transportation. From a modelling viewpoint, we proposed a novel ILP formulation in which variables are associated with feasible paths, and discussed alternative ways for handling the exponential number of variables in the model. From a methodological perspective, we showed how a column generation algorithm can be embedded into a branch-and-cut-and-price scheme, that is robust in the sense that the structure of the subproblems is not altered by the branching conditions nor by the addition of valid inequalities. Finally, we gave a comprehensive computational analysis of the performances of the proposed algorithm, which is compared with a state-of-the-art approach proposed in the recent literature. Our computational experiments showed that the proposed algorithm outperforms its competitor and scales efficiently to larger size of the network.

The introduced path-based formulation is quite general, as all the nasty constraints appear in the definition of feasible paths only. For this reason, it may be worthy to use this modelling approach for other multi-commodity network design problems arising in different contexts.

6. Acknowledgments

This research was supported by "Mixed-Integer Non Linear Optimisation: Algorithms and Application" consortium, which has received funding from the European Union's EU Framework Programme for Research and Innovation Horizon 2020 under the Marie Skłodowska-Curie Actions Grant Agreement No. 764759. The authors are grateful to Prakash Mirchandani, who kindly provided us the code for generating the numerical data; to Antonio Frangioni, for stimulating discussions; and to the SCIP optimization suite mailing list, in particular to Ambros Gleixner, for his invaluable help.

References

A. K. Andreas and J. C. Smith. Mathematical programming algorithms for two-path routing problems with reliability considerations. *INFORMS J. Comput.*, 20(4):553–564, 2008.

A. K. Andreas, J. C. Smith, and S. Küçükyavuz. Branch-and-price-and-cut algorithms for solving the reliable h -paths problem. *J. Glob. Optim.*, 42(4):443–466, 2008.

A. P. Armacost, C. Barnhart, and K. A. Ware. Composite variable formulations for express shipment service network design. *Transportation Science*, 36(1):1–20, 2002.

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

A. Balakrishnan, P. Mirchandani, and H. P. Natarajan. Connectivity upgrade models for survivable network design. *Oper. Res.*, 57(1):170–186, Jan. 2009.

A. Balakrishnan, G. Li, and P. Mirchandani. Optimal network design with end-to-end service requirements. *Operations Research*, 65(3):729–750, 2017.

C. Barnhart and R. R. Schneur. Air network design for express shipment service. *Operations Research*, 44(6):852–863, 1996.

C. Barnhart, C. A. Hane, and P. H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2):318–326, 2000.

J. Barrera, H. Cancela, and E. Moreno. Topological optimization of reliable networks under dependent failures. *Operations Research Letters*, 43(2):132–136, 2015.

Q. Botton, B. Fortz, L. Gouveia, and M. Poss. Benders decomposition for the hop-constrained survivable network design problem. *INFORMS Journal on Computing*, 25(1):13–26, 2013.

T. Crainic. Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272–288, 2000.

J. Desrosiers and M. E. Lübbecke. *A Primer in Column Generation*, pages 1–32. Springer US, Boston, MA, 2005.

I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42(3):135–153, 2003.

G. Gamrath, D. Anderson, K. Bestuzheva, W.-K. Chen, L. Eifler, M. Gasse, P. Gemander, A. Gleixner, L. Gottwald, K. Halbig, G. Hendel, C. Hojny, T. Koch, P. Le Bodic, S. J. Maher, F. Matter, M. Miltenberger, E. Mühmer, B. Müller, M. E. Pfetsch, F. Schlösser, F. Serrano, Y. Shinano, C. Tawfik, S. Vigerske, F. Wegscheider, D. Weninger, and J. Witzig. The SCIP Optimization Suite 7.0. ZIB-Report 20-10, Zuse Institute Berlin, March 2020.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman, San Francisco, 1979.

B. Gendron, T. Crainic, and A. Frangioni. Multicommodity capacitated network design. In B. Sansò and P. Soriano, editors, *Telecommunications Network Planning*. Springer Science & Business Media, 1999.

L. Gouveia. Using variable redefinition for computing lower bounds for minimum spanning and steiner trees with hop constraints. *INFORMS Journal on Computing*, 10(2):180–188, 1998.

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

M. Grötschel, C. L. Monma, and M. Stoer. Polyhedral and computational investigations for designing communication networks with high survivability requirements. *Operations Research*, 43(6):1012–1024, 1995.

H. Joksch. The shortest route problem with constraints. *Journal of Mathematical Analysis and Applications*, 14:191–197, 1966.

J. Kennington. A survey of linear cost multicommodity network fows. *Operations Research*, 26(2): 209–236, 1978.

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

D. Kim, C. Barnhart, K. Ware, and G. Reinhardt. Multimodal express package delivery: A service network design application. *Transportation Science*, 33(4):391–407, 1999.

I. Ljubić, P. Mutzel, and B. Zey. Stochastic survivable network design problems: Theory and practice. *European Journal of Operational Research*, 256(2):333–348, 2017.

T. L. Magnanti and S. Raghavan. Strong formulations for network design problems with connectivity requirements. *Networks*, 45(2):61–79, 2005.

T. L. Magnanti and R. T. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, 18(1):1–55, 1984.

P. Mateti and N. Deo. On algorithms for enumeratingall circuits of a graph. *SIAM Journal on Computing*, 5(1):90–99, 1976.

Y. Song and J. R. Luedtke. Branch-and-cut approaches for chance-constrained formulations of reliable network design problems. *Mathematical Programming Computation*, 5(4):397–432, 2013.

N. Wieberneit. Service network design for freight transportation: a review. *OR Spectrum*, 30(1): 77–112, 2007.