



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna Archivio istituzionale della ricerca

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

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

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>

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>).
When citing, please refer to the published version.

(Article begins on next page)

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]).

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

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

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

34 In this manuscript, we propose a new model where variables are associated with paths sat-
35 isfying the end-to-end service requirements. This way, many of the weaknesses of the arc-flow
36 formulation are naturally overtaken without the need to recur to cut separation techniques. This
37 desirable property comes at the cost of a formulation which is much larger, involving an expo-
38 nential number of path variables. However, we show that for all the instances considered by
39 Balakrishnan et al. [2017], we are indeed capable of quickly enumerating all the variables of the
40 new formulation, thanks to an effective labelling algorithm, and to solve to proven optimality

41 a much larger set of instances using a general-purpose ILP solver. In particular, our approach
42 allows to solve a relevant fraction of the large instances introduced by [Balakrishnan et al. \[2017\]](#),
43 and to compute near-optimal solutions in the remaining cases, showing that the algorithm scales
44 efficiently to larger size of the network. In addition, we provide a new set of instances for which
45 enumerating all the paths is not viable; for solving these large instances, we present a column
46 generation scheme that is embedded into a full branch-and-cut-and-price algorithm.

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

54 *Literature Review:* There is a wide literature on network design problems, and many surveys
55 have been published on these topics, see, e.g., [Magnanti and Wong \[1984\]](#), [Crainic \[2000\]](#), and
56 [Wieberneit \[2007\]](#). Depending on the specific application, different variants of these problems
57 were considered. A notable field of research involves the design of reliable and survivable net-
58 works, that has become a major objective for telecommunication operators (see, [Kerivin and](#)
59 [Mahjoub \[2005\]](#)). In this context, one is required to define a robust network preserving a given con-
60 nectivity level under possible failure of certain network components. There exist several ways to
61 express the network robustness. Under a stochastic paradigm, the network is required to remain
62 operative either with a large probability ([Song and Luedtke \[2013\]](#), [Barrera et al. \[2015\]](#)) or after
63 some recourse action has been implemented ([Ljubić et al. \[2017\]](#)). Alternatively, more conservative
64 approaches, imposing explicit redundancy in the definition of the network, have been considered
65 in the literature; typically, one is required to design a network having two (edge) disjoint paths
66 for each commodity ([Magnanti and Raghavan \[2005\]](#), [Andreas and Smith \[2008\]](#), [Andreas et al.](#)
67 [\[2008\]](#), and [Balakrishnan et al. \[2009\]](#)), while [Grötschel et al. \[1995\]](#) considered the case in which
68 higher connectivity requirements are imposed.

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

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

114 2.1. Arc-flow formulation

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

$$117 \quad z_a = \begin{cases} 1 & \text{if arc } a \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \quad (a \in \mathcal{A})$$

$$118 \quad y_a^k = \begin{cases} 1 & \text{if commodity } k \text{ is routed on arc } a \\ 0 & \text{otherwise} \end{cases} \quad (a \in \mathcal{A}, k \in \mathcal{K})$$

120 Then, the NDSR can be modelled using the following Integer Linear Programming (ILP) for-
 121 mulation:

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

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

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

$$y_a^k \leq z_a \quad a \in \mathcal{A}, k \in \mathcal{K} \quad (1d)$$

$$z_a \in \{0, 1\} \quad a \in \mathcal{A} \quad (1e)$$

$$y_a^k \in \{0, 1\} \quad a \in \mathcal{A}, k \in \mathcal{K}. \quad (1f)$$

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

128 2.2. Path-based formulation

129 The novel ILP formulation that we propose includes the same binary activation variables of model
 130 (1a)–(1f), that select the arcs to be activated, whereas flow variables are replaced by path variables
 131 that are defined as follows. Let \mathcal{P}^k be the set of all feasible paths for commodity k . For each

132 commodity k and each path $p \in \mathcal{P}^k$, let us introduce a binary path variable x_p with the following
133 meaning:

$$134 \quad x_p = \begin{cases} 1, & \text{if commodity } k \text{ is routed along path } p \\ 0 & \text{otherwise} \end{cases} \quad (k \in \mathcal{K}, p \in \mathcal{P}^k)$$

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

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

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

$$\sum_{p \in \mathcal{P}^k} x_p = 1 \quad k \in \mathcal{K} \quad (2c)$$

$$z_a \in \{0, 1\} \quad a \in \mathcal{A} \quad (2d)$$

$$x_p \in \{0, 1\} \quad p \in \mathcal{P}^k, k \in \mathcal{K}. \quad (2e)$$

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

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

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

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

153 *Variable enumeration:* We first observe that the path-based formulation has $O(2^{|\mathcal{V}|})$ variables and
154 $(|\mathcal{A}| + 1) |\mathcal{K}|$ constraints, i.e., its size can be exponential in the size of the instance. We now intro-
155 duce an algorithm for enumerating all path variables; however, for large graphs, enumerating all
156 paths can be challenging, and one may have to resort to column generation techniques, that will
157 be discussed in Section 3.

158 There is a vast body of literature addressing the problem of enumerating all paths (and the
159 closely related problem of enumerating all cycles) in a given graph. According to Mateti and
160 Deo [1976], existing approaches can be classified into search space algorithms, backtrack algo-
161 rithms, and algorithms based on the power of the adjacency matrix. Our enumeration Algorithm
162 1 belongs to the second category, and considers one commodity k at a time and defines all simple
163 paths from s^k to t^k that satisfy resource constraints under all metrics. The algorithm is inspired
164 by the labelling method proposed by Dumitrescu and Boland [2003] for the budget-constrained
165 shortest path problem. In our algorithm, each label $\ell = \{u, c, \mathbf{w}\}$ represents a path from s^k to u
166 having cost c and using w_m units of resources under each metric m . Each label is generated as
167 unmarked, meaning that it has to be expanded, and then it is marked when considered for expan-
168 sion. Expansion of a label ℓ associated with a node u consists in appending an arc $a = (u, v)$ to the
169 current path. To this aim, we consider all the outgoing arcs from u and, for each neighbor node
170 v not yet belonging to path ℓ , we check whether using the current label for reaching v preserves
171 feasibility with respect to the metrics. In this case, we define a new label $\ell' = \{v, c + c_a^k, \mathbf{w} + \mathbf{w}_a^k\}$,
172 i.e., we update the path cost and resource usage when using the current label for reaching v . Even-
173 tually, node v is inserted in set T , that includes all nodes associated with unmarked labels. The
174 algorithm terminates when $T = \emptyset$, meaning that no label can be further expanded, and returns all
175 labels associated with node t^k . Although a node can be inserted in and removed from T more than
176 once, the convergence of the algorithm is ensured by requiring simple paths, which is checked in
177 line 9.

178 The above algorithm can be improved by pre-computing, for each metric $m \in \mathcal{M}$, the shortest
179 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
180 when checking feasibility of the new label in line 9: by adding this term to the left-hand-side of
181 the inequality, we avoid generating labels that could not be feasibly expanded to node t^k .

182 2.3. Models comparison

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

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

Algorithm 1: Compute all feasible paths for a fixed commodity**Input** : k

```

1  $s := s^k, t := t^k, T := \{s\}, c := 0, \mathbf{w} := \mathbf{0}$ 
2 Define an unmarked label  $\ell := \{s, c, \mathbf{w}\}$  for node  $s$ 
3 while  $T \neq \emptyset$  do
4   pick any  $u \in T$ 
5    $T := T \setminus \{u\}$ 
6   foreach unmarked label  $\ell = \{u, c, \mathbf{w}\}$  associated with node  $u$ 
7     mark label  $\ell$ 
8     foreach  $a = (u, v) \in \delta^+(u)$ 
9       if  $(v \notin \text{path } \ell)$  and  $(\mathbf{w} + \mathbf{w}_a^k \leq \mathbf{W}^k)$ 
10        define an unmarked label  $\ell' = \{v, c + c_a^k, \mathbf{w} + \mathbf{w}_a^k\}$ 
11        if  $(v \neq t)$ 
12           $T := T \cup \{v\}$ 
13 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 \tilde{z}, \tilde{y} that is feasible for the linear relaxation of the arc-based formulation and has the same cost. First, we set $\tilde{z} = z^*$. Then, for each arc $a \in \mathcal{A}$ and commodity $k \in \mathcal{K}$, we set

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

187 It is straightforward to check that flow conservation constraints (1b) and feasibility requirements
188 (1c) with respect to the metrics are satisfied as y variables are obtained as combination of feasible
189 paths, whereas constraints (1d) are implied by (2c) and by the definition of \tilde{y}_a^k . The equivalence of
190 the costs follows from the definition of the cost of each path.

191 Figure 1 gives a small numerical example showing that the counterpart does not hold. The
192 instance has no flow costs, a single commodity, and a single metric, for which the capacity is
193 $W = 2$. For each arc we report the activation cost and the weight with respect to the metric. While
194 there is a unique feasible path $p = \{(s, t)\}$ having cost 1, an optimal solution to the linear relaxation
195 of the arc-based formulation is given by $y_{s1} = y_{1t} = y_{st} = 1/2$ having cost $1/2$. \square

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

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

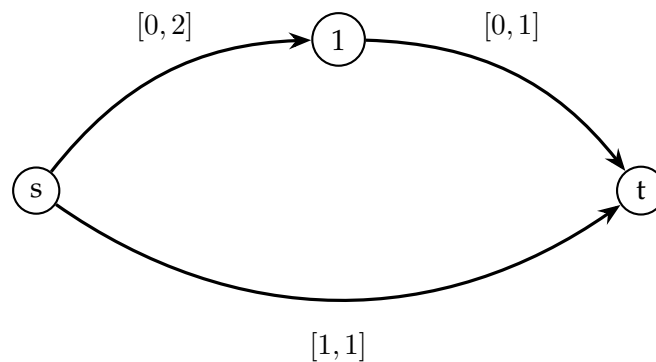


Figure 1 Simple example for which the path-based formulation dominates the arc-flow formulation.

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

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

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

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

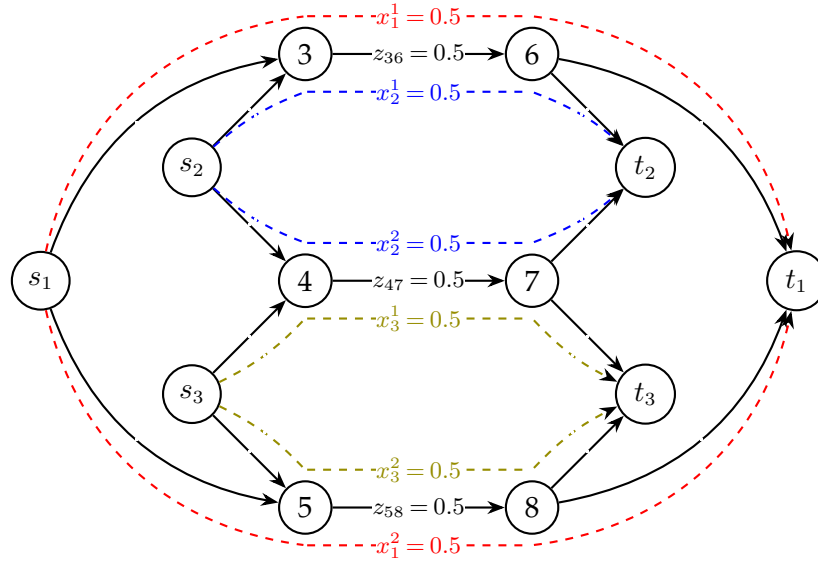


Figure 2 Fractional solution of the linear relaxation of the path-based formulation.

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

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

where the arc costs are $\tilde{c}_a^k = c_a^k + \gamma_a^k$. 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 \tilde{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

236 process is iterated; if no path variable is generated for any commodity, the optimal solution of the
 237 current restricted master is an optimal solution for the linear relaxation of the problem.

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

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

Input : k

```

1   $s := s^k, t := t^k, T := \{s\}, c := 0, \mathbf{w} := \mathbf{0}$ 
2  Define an unmarked label  $\ell := \{s, c, \mathbf{w}\}$  for node  $s$ 
3  while  $T \neq \emptyset$  do
4      pick any  $u \in T$ 
5       $T := T \setminus \{u\}$ 
6      foreach unmarked label  $\ell = \{u, c, \mathbf{w}\}$  associated with node  $u$ 
7          mark label  $\ell$ 
8          foreach  $a = (u, v) \in \delta^+(u)$ 
9              if  $(v \notin \text{path } \ell)$  and  $(\mathbf{w} + \mathbf{w}_a^k \leq \mathbf{W}^k)$ 
10                 define an unmarked label  $\ell' = \{v, c + \tilde{c}_a^k, \mathbf{w} + \mathbf{w}_a^k\}$ 
11                     if ( $\ell'$  is dominated by a label  $\ell''$  associated with node  $v$ )
12                         mark label  $\ell'$ 
13                     if ( $\ell'$  dominates a label  $\ell''$  associated with node  $v$ )
14                         mark label  $\ell''$ 
15                     if  $(v \neq t)$  and ( $\ell'$  is unmarked)
16                          $T := T \cup \{v\}$ 
17 return the unmarked label with minimum cost  $c$  associated with node  $t$ 
    
```

248 3.2. Branching scheme

249 In our branching scheme we always select a z variable for branching. According to Observation
 250 1 at each node where all the z variables attain integer values, there exists an optimal solution in

251 which all the x variables are integer as well. Notice that this is the solution returned by solving
252 the restricted master problem by means of the simplex algorithm.

253 A positive effect of this branching strategy is that it does not affect the structure of the pricing
254 subproblem. This is a crucial property for designing an effective branch-and-price algorithm, as
255 it allows to solve the column generation subproblem throughout all the branching tree by means
256 of the same effective labelling algorithm used at the root node. Clearly, imposing $z_a = 1$ for some
257 $a \in \mathcal{A}$ has no direct effect in the pricing. Conversely, when imposing $z_a = 0$, in the pricing sub-
258 problem we simply forbid the use of arc a when generating new path variables, which can be
259 easily handled by setting $\mathcal{A} = \mathcal{A} \setminus \{a\}$.

260 3.3. Adding valid inequalities

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

264 To this aim, we adapt to our model some of the inequalities introduced by [Balakrishnan et al.](#)
265 [\[2017\]](#) for the arc-flow formulation. These inequalities are obtained by analyzing the structure of
266 the graph G and by deriving relationships between pairs of arcs (a, b) when routing the flow of a
267 commodity k , namely:

- 268 • *OR* relationships, occurring when no more than one arc of pair (a, b) can be used to route
269 flow from s^k to t^k ;
- 270 • *IF* relationships, occurring when the flow through arc a must also be routed through b ; and
- 271 • *CUT* relationship, occurring when at least one between a and b must be used to route the
272 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

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

273 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

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

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

$$x_{0,0} + x_{0,1} + x_{1,0} + x_{1,1} + x_{2,0} + x_{2,1} \leq z_{36} + z_{47} + z_{58} + 1$$

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

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

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

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

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

289 4. Computational experiments

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

298 4.1. Instances from the literature and state-of-the-art

299 We now describe a benchmark of instances that has recently been introduced by Balakrishnan
 300 et al. [2017], who kindly provided us the code for generating the numerical data. Each instance
 301 is characterized by the following parameters: the number of nodes $|\mathcal{V}|$, number of arcs $|\mathcal{A}|$, and
 302 number of commodities $|\mathcal{K}|$. Nodes are randomly located on a rectangular grid and are connected
 303 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 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.

339 The results in Table 1 show that `composite` is able to solve 24 instances out of 35 in the first
 340 7 scenarios, with average percentage gap equal to 0.73. For only two scenarios, this approach
 341 is able to solve all the corresponding 5 instances to optimality. Overall, the algorithm solves 45
 342 instances out of 90, with an average percentage gap around 1%, showing that this benchmark is
 343 quite challenging for the state-of-the-art computational approach.

344 4.2. Results of the path-based formulation

345 Although the original benchmark is not available, we generated 90 instances (5 for each scenario)
 346 having the same parameters used by Balakrishnan et al. [2017] by running their instance genera-
 347 tor. All our algorithms were implemented in C++ and were run on an AMD Ryzen Threadripper
 348 3960X running at 3.8 GHz in single-thread mode, with a time limit of 1 hour per instance. Both the
 349 arc-flow and the path-based formulations were solved using Gurobi version 9.1.1 as ILP solver,
 350 whereas the branch-and-cut-and-price was implemented on top of the SCIP optimization suite
 351 (version 7.0.1 with its default SoPlex solver), which allows to embed a column generation scheme
 352 within the enumeration process (see Gamrath et al. [2020]). Our source code and instances are
 353 publicly available at <https://github.com/CGudapati/NDSR-Code>.

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

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

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

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

364

scenario	base-model			all-path			
	# 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.

365 Although a one-to-one comparison with Table 1 is not possible, these results confirm the
366 outcome of the computational experiments reported by Balakrishnan et al. [2017] for the first
367 seven scenarios, i.e., that `composite` outperforms the `base-model`, which can solve only small
368 instances and has large percentage gaps for most unsolved scenarios. On the other hand, algo-
369 rithm `all-path` solves all but one instance in the first seven scenarios, and has a percentage
370 gap equal to 0.38 for the remaining instance. This is due to the fact that the formulation is tight
371 and that, for these instances, the number of path variables does not grow up: this number is
372 always smaller than 2000, which makes the model solvable with a limited computational effort.
373 The performances of algorithm `all-path` remain satisfactory for instances in scenarios 8-15: the
374 algorithm solves 37 of the 40 associated instances, and has an average gap equal to 0.28%. Finally,
375 for very large instances (scenarios 16 to 18), the algorithm solves 9 instances out of 15 and has an
376 average gap equal to 0.60%. Overall, our algorithm solves to proven optimality almost 90% of the
377 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

α	base-model			all-path				branch-and-price			
	# 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

scenario	linear relaxation				exact solution					
	without cuts		with cuts		branch-and-price			branch-and-cut-and-price		
	% 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.

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

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

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

451 **5. Conclusions**

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

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

466 6. Acknowledgments

467 This research was supported by “Mixed-Integer Non Linear Optimisation: Algorithms and Appli-
468 cation” consortium, which has received funding from the European Union’s EU Framework Pro-
469 gramme for Research and Innovation Horizon 2020 under the Marie Skłodowska-Curie Actions
470 Grant Agreement No. 764759. The authors are grateful to Prakash Mirchandani, who kindly pro-
471 vided us the code for generating the numerical data; to Antonio Frangioni, for stimulating dis-
472 cussions; and to the SCIP optimization suite mailing list, in particular to Ambros Gleixner, for his
473 invaluable help.

474 References

- 475 A. K. Andreas and J. C. Smith. Mathematical programming algorithms for two-path routing
476 problems with reliability considerations. *INFORMS J. Comput.*, 20(4):553–564, 2008.
- 477 A. K. Andreas, J. C. Smith, and S. Küçükyavuz. Branch-and-price-and-cut algorithms for solving
478 the reliable h -paths problem. *J. Glob. Optim.*, 42(4):443–466, 2008.
- 479 A. P. Armacost, C. Barnhart, and K. A. Ware. Composite variable formulations for express ship-
480 ment service network design. *Transportation Science*, 36(1):1–20, 2002.
- 481 A. Balakrishnan and K. Altinkemer. Using a hop-constrained model to generate alternative com-
482 munication network design. *ORSA Journal on Computing*, 4(2):192–205, 1992.
- 483 A. Balakrishnan, P. Mirchandani, and H. P. Natarajan. Connectivity upgrade models for surviv-
484 able network design. *Oper. Res.*, 57(1):170–186, Jan. 2009.
- 485 A. Balakrishnan, G. Li, and P. Mirchandani. Optimal network design with end-to-end service
486 requirements. *Operations Research*, 65(3):729–750, 2017.
- 487 C. Barnhart and R. R. Schneur. Air network design for express shipment service. *Operations*
488 *Research*, 44(6):852–863, 1996.
- 489 C. Barnhart, C. A. Hane, and P. H. Vance. Using branch-and-price-and-cut to solve origin-
490 destination integer multicommodity flow problems. *Operations Research*, 48(2):318–326, 2000.
- 491 J. Barrera, H. Cancela, and E. Moreno. Topological optimization of reliable networks under depen-
492 dent failures. *Operations Research Letters*, 43(2):132–136, 2015.
- 493 Q. Botton, B. Fortz, L. Gouveia, and M. Poss. Benders decomposition for the hop-constrained
494 survivable network design problem. *INFORMS Journal on Computing*, 25(1):13–26, 2013.
- 495 T. Crainic. Service network design in freight transportation. *European Journal of Operational*
496 *Research*, 122(2):272–288, 2000.

- 497 J. Desrosiers and M. E. Lübbecke. *A Primer in Column Generation*, pages 1–32. Springer US, Boston,
498 MA, 2005.
- 499 I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the
500 weight-constrained shortest path problem. *Networks*, 42(3):135–153, 2003.
- 501 G. Gamrath, D. Anderson, K. Bestuzheva, W.-K. Chen, L. Eifler, M. Gasse, P. Gemander,
502 A. Gleixner, L. Gottwald, K. Halbig, G. Hendel, C. Hojny, T. Koch, P. Le Bodic, S. J. Maher,
503 F. Matter, M. Miltenberger, E. Mühmer, B. Müller, M. E. Pfetsch, F. Schlösser, F. Serrano, Y. Shi-
504 nano, C. Tawfik, S. Vigerske, F. Wegscheider, D. Weninger, and J. Witzig. The SCIP Optimization
505 Suite 7.0. ZIB-Report 20-10, Zuse Institute Berlin, March 2020.
- 506 M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*.
507 W.H. Freeman, San Francisco, 1979.
- 508 B. Gendron, T. Crainic, and A. Frangioni. Multicommodity capacitated network design. In
509 B. Sansò and P. Soriano, editors, *Telecommunications Network Planning*. Springer Science & Busi-
510 ness Media, 1999.
- 511 L. Gouveia. Using variable redefinition for computing lower bounds for minimum spanning and
512 steiner trees with hop constraints. *INFORMS Journal on Computing*, 10(2):180–188, 1998.
- 513 L. Gouveia, M. Leitner, and I. Ljubić. The two-level diameter constrained spanning tree problem.
514 *Mathematical Programming*, 150:49–78, 2015.
- 515 M. Grötschel, C. L. Monma, and M. Stoer. Polyhedral and computational investigations for
516 designing communication networks with high survivability requirements. *Operations Research*,
517 43(6):1012–1024, 1995.
- 518 H. Jochsch. The shortest route problem with constraints. *Journal of Mathematical Analysis and Appli-*
519 *cations*, 14:191–197, 1966.
- 520 J. Kennington. A survey of linear cost multicommodity network flows. *Operations Research*, 26(2):
521 209–236, 1978.
- 522 H. Kerivin and A. R. Mahjoub. Design of survivable networks: A survey. *Networks*, 46(1):1–21,
523 2005.
- 524 D. Kim, C. Barnhart, K. Ware, and G. Reinhardt. Multimodal express package delivery: A service
525 network design application. *Transportation Science*, 33(4):391–407, 1999.
- 526 I. Ljubić, P. Mutzel, and B. Zey. Stochastic survivable network design problems: Theory and
527 practice. *European Journal of Operational Research*, 256(2):333–348, 2017.
- 528 T. L. Magnanti and S. Raghavan. Strong formulations for network design problems with connec-
529 tivity requirements. *Networks*, 45(2):61–79, 2005.
- 530 T. L. Magnanti and R. T. Wong. Network design and transportation planning: Models and algo-
531 rithms. *Transportation Science*, 18(1):1–55, 1984.

- 532 P. Mateti and N. Deo. On algorithms for enumerating all circuits of a graph. *SIAM Journal on*
533 *Computing*, 5(1):90–99, 1976.
- 534 Y. Song and J. R. Luedtke. Branch-and-cut approaches for chance-constrained formulations of
535 reliable network design problems. *Mathematical Programming Computation*, 5(4):397–432, 2013.
- 536 N. Wieberneit. Service network design for freight transportation: a review. *OR Spectrum*, 30(1):
537 77–112, 2007.