Submitted to *Transportation Science*

manuscript (Please, provide the mansucript number!)

The Multi-Period Multi-Commodity Network Design Problem

Ioannis Fragkos

Rotterdam School of Management, The Netherlands, fragkos@rsm.nl

Jean-François Cordeau

HEC Montréal and CIRRELT, Canada, jean-francois.cordeau@hec.ca

Raf Jans

HEC Montréal and CIRRELT, Canada, raf.jans@hec.ca

The multi-commodity network design model is a fundamental problem that arises in a multitude of important transportation systems. We generalize the basic model to a multi-period setting, where demand for each commodity expands dynamically over a discrete time horizon. Our formulation captures time-dependent network design decisions, and is therefore able to represent the dynamic expansion of transportation and supply chain networks. Arc activation decisions are taken in every period, and once an arc is activated it can be used throughout the remaining horizon to route several different commodities. Our model captures a key timing trade-off: the earlier an arc is activated, the more periods it can be used for, but its fixed activation cost is higher, as this cost accounts not only for installation but also for maintenance and operation over the remaining horizon. We develop decomposition algorithms to solve capacitated and uncapacitated variants, and a heuristic that separates the arc selection and activation timing decisions. For capacitated variants, we develop an arc-based Lagrange relaxation, combined with a series of local improvement heuristics. For uncapacitated problems, we develop several Benders decomposition variants and show how taking advantage of the problem structure leads to enhanced algorithmic performance. Finally, we generate a new dataset of large scale instances in order to conduct computational experiments. Our results demonstrate the efficiency of our algorithms. Notably, for uncapacitated problems we are able to solve instances with 2.5 million variables to optimality in less than two hours of CPU time.

Key words: Network Design, Multi-period, Lagrange relaxation, Benders Decomposition, Heuristics

1. Introduction

Network design problems (NDP) lie at the intersection of operations research and transportation science, and represent a wide variety of transportation models in fields as diverse as road network design (Yang et al. 1998), logistics (Lee and Dong 2008), energy transport and telecommunications (Minoux 1989), and railways (Hooghiemstra et al. 1999). The single-period multi-commodity capacitated network design problem (S-CNDP) is an important model that, given a network of nodes and arcs, considers the design decisions of selecting which arcs should be operated, and how commodities should be routed from their origin to their destination nodes, while the costs of activating arcs and routing commodity flows should be jointly minimized. Although this $\mathcal{NP}$-hard problem finds numerous applications in transportation and beyond (Magnanti and Wong 1984), little attention has been given to its multi-period extensions, perhaps due to the large size of the models that arise in practical applications.

In this work, we extend the traditional S-CNDP by considering the demand of each commodity in multiple periods. Although our models allow otherwise, we focus on cases where demand is non-decreasing over time so that we capture the timing trade-off related to network expansion decisions. Concretely, activating an arc implies that this arc can be used to route commodities until the end of the planning horizon. However, the more periods the arc is under operation, the higher the fixed cost that occurs. This cost represents duration-dependent expenses, such as maintenance and operating expenses of road networks, contracting costs of third-party logistics companies, or investment costs of constructing a track in a rail network, and it leads to an important trade-off: on the one hand, activating an arc early on implies a high fixed cost, but the arc can be used to route commodities for a large number of subsequent periods. On the other hand, activating an arc later in the horizon is associated with a lower fixed cost, but the number of periods in which the arc can be used is smaller. We capture this trade-off by jointly minimizing the fixed costs of activating arcs and the variable costs of routing the demands throughout the given horizon, thereby also considering the associated activation versus routing cost trade-off in combination.

1.1. Contributions and outline

Our work extends traditional multi-commodity network design problems by introducing multiple periods, and therefore captures important capacity expansion trade-offs. Although such multi-period NDPs can provide useful input for strategic and tactical decisions, finding their optimal solution is computationally challenging. Indeed, their very large scale makes them difficult or even impossible to solve with state-of-the-art mixed-integer programming (MIP) solvers. To this end, we exploit their multi-period structure to devise specialized decomposition algorithms for both capacitated and uncapacitated variants. First, we develop an arc-based Lagrange relaxation algorithm for the capacitated problem, which attains a strong lower bound in a short amount of time, particularly for large instances. Second, we develop a stand-alone heuristic which we combine with Lagrange relaxation. Computational experiments show that our heuristic is able to attain feasible solutions within 1% of optimality an order of magnitude faster than a commercial MIP solver, while its performance scales better with problem size. Third, we develop a Benders decomposition algorithm for uncapacitated problems, that decomposes the original problem into single-period shortest path subproblems per period and per commodity. We show how Pareto-optimal Benders cuts can be generated efficiently for our application using a series of algorithmic enhancements, and compare the resulting implementations with the novel formulation of Fischetti et al. (2010). Finally, we generate new multi-period instances and perform extensive computational experiments. To test our algorithms, we generate multi-period problems with as many as 80 periods and more than 185,000 constraints and 2.5 million variables. The test problems are multi-period extensions of the Canad R instances (Gendron and Crainic 1994), that are used widely in the network design literature (Katayama et al. 2009, Yaghini et al. 2014, Ghamlouche et al. 2003, Crainic and Gendreau 2002, Chouman et al. 2009, Hewitt et al. 2010). Our computational results show that both Lagrange relaxation and Benders decomposition are efficient in finding high-quality solutions within a reasonable amount of time. For uncapacitated instances in particular, our best implementation is able to solve 52 instances to optimality that could not be solved with branch-and-cut, while it attains an average

gap of 0.42% over all instances, versus a 15.52% gap attained by branch-and-cut.

The remainder of the paper is organized as follows. Before we proceed to a formal description of the problem and methods, we review related research, and position our work within the network design research landscape in Section 2. Then, Section 3 describes the problem formulation, outlining the differences between the capacitated and the uncapacitated versions. Section 4 explains the construction of a heuristic, which is designed to perform well on large-scale problems. Section 5 provides details on the Lagrange relaxation and Benders decomposition algorithms, which are used to solve the capacitated and uncapacitated problems, respectively. Finally, Section 6 presents computational results that demonstrate the efficacy of the proposed algorithms. The paper concludes by reflecting on future research avenues, reported in Section 7.

2. Literature Review

The literature on network design problems is voluminous, ranging from linear single-commodity flow problems to non-linear capacitated multi-commodity problems and application-specific variants. One of the seminal studies in uncapacitated network design is the work of Balakrishnan et al. (1989). The authors utilize a family of dual ascent methods, with which they are able to solve problems of up to 500 integer and 1.98 million continuous variables in a few seconds. Subsequent works have focused mostly on exact approaches, such as Lagrange relaxation-based branch-and-bound (Cruz et al. 1998, Holmberg and Yuan 1998), branch-and-cut, and Benders decomposition (Randazzo and Luna 2001, Rahmaniani et al. 2016). To the best of our knowledge, the Lagrange relaxation algorithm of Holmberg and Yuan (1998) is the state-of-the-art exact approach for solving single-period uncapacitated problems, while no dedicated heuristics have been developed. Although our work focuses on exact methods, it is also related to heuristics, as we develop a custom heuristic tailored to large-scale, multi-period problems. A heuristic used by several researchers is the dynamic slope scaling approach of Kim and Pardalos (1999), which the authors utilized for single-commodity uncapacitated and capacitated network design problems. The main idea of slope scaling is to update the objective function coefficients

of the continuous variables in a dynamic way, so that the adjusted linear cost is a locally good approximation of both the linear and fixed costs. This ideas was adopted by other authors, who proposed successful implementations for problems with richer structures, such as capacitated network design (Crainic et al. 2004) or freight rail transportation (Zhu et al. 2014). Our heuristic design decisions take on a different perspective, despite the ubiquity of slope-scaling: instead of developing a stand-alone procedure, our intention is to feed a decomposition method with a good feasible solution, which can be further improved during the decomposition scheme. In this respect, we are willing to compromise solution quality for speed: since we focus on large multi-period multi-commodity instances, we decided to embed a period decomposition procedure in our heuristic and therefore avoid to solve linear programs as large as the original multi-period instances we tackle.

An interesting extension of the traditional uncapacitated problem which, to the best of our knowledge, has not received much attention, is the time-space formulation of Kennington and Nicholson (2010). Motivated by a cash management problem, the authors explain that time-space networks represent flows from origin-time pairs to destination-time pairs that are forward in time, and utilize an iterative solution strategy to solve problems of practical interest. This work has similarities to our study, as we also study a multi-period network design problem. However, our model has several important differences: (i) we incorporate multiple commodities; (ii) the fixed cost structure of our model is different, as it depends also on the period in which the arc is selected, and not only on the selected origin-destination pair; (iii) our formulation allows for paths of arbitrary length per period, instead of imposing the traversal of a unique arc per period. Related to our setting is also the work of Papadimitriou and Fortz (2015), who consider capacity installation and routing decisions in telecommunication networks. The authors propose a rolling horizon heuristic to solve practical problem instances. Although their work is close to ours, the model we develop in this paper does not consider assumptions specific to telecommunication networks, such as the convex structure of routing costs. Instead, our purpose is to provide a generic model that can be used as a foundation towards developing specific applications, such as that of Papadimitriou and Fortz (2015). In this regard, our model has

some notable differences: (i) we refer to multiple commodities instead of multiple origin-destination pairs, allowing the routing costs to vary per commodity, arc and period; (ii) we do not model link aging as a cost but instead consider a fixed activation cost that includes capacity installation and maintenance as a lump sum; and (iii), we do not model switching capacity decisions and non-linear collective routing costs, as these are specific to telecommunication environments. Rather, our model is relevant to expansion decisions of a generic infrastructure, such as urban transport and irrigation networks, and captures the timing of capacity expansion decisions.

It is well known (Balakrishnan et al. 2017) that by splitting a node into a pair of nodes with an intermediate arc, and by linking incoming arcs to the head node and outgoing arcs to the tail node we can model node-specific fixed and variable costs. Thus, a class of problems related to uncapacitated network design is that of uncapacitated facility location problems. In a multi-period setting, Jena et al. (2015, 2017) propose a formulation that unifies dynamic facility location problems in which node capacity can be expanded or reduced in a modular fashion. Their formulation has a stronger linear programming (LP) bound and outperforms several specialized formulations. Chardaire et al. (1996) study a formulation that allows for opening and closing facilities multiple times and solve it using heuristics and a Lagrange relaxation-based lower bound. Several other variants have been studied, such as the model of Shulman (1991), which takes facility type and size into account, or that of Harkness and ReVelle (2003), which considers increasing economies of scale. An overview of multi-period facility location problems can be found in Arabani and Farahani (2012).

In spite of some notable research output on the uncapacitated problem, the largest literature stream has focused on solution methods for the capacitated network design version, S-CNDP. To this end, some authors have made polyhedral studies (Atamtürk 2002, Atamtürk and Rajan 2002, Bienstock and Günlük 1996, Raack et al. 2011, Günlük 1999), recognizing that a better understanding of the S-CNDP polytope can lead to the development of better algorithms. Other authors have used decomposition approaches, such as column generation (Frangioni and Gendron 2009, 2013) and Lagrange relaxation (Frangioni and Gorgone 2014). In our work, we propose a Lagrange relaxation algorithm

for capacitated problems and Benders decomposition for uncapacitated problems, as each algorithm takes advantage of the decomposable structure of capacitated and uncapacitated problems, respectively. More specifically, our Lagrange relaxation algorithm decomposes the problem in a series of single-arc multi-period, multi-commodity problems, which we can solve in polynomial time with a custom algorithm. Furthermore, we employ the volume algorithm of Barahona and Anbil (2000) to recover an approximate primal solution that guides the search for heuristic solutions efficiently. While several other works exist that have used decomposition methods, such as Alvelos and de Carvalho (2007), Costa et al. (2009), Holmberg and Yuan (2000), Crainic et al. (2004), Frangioni and Gendron (2009), Frangioni and Gendron (2013), Frangioni and Gorgone (2014), we are the first to assess the performance of Benders decomposition and Lagrange relaxation in multi-period, multi-commodity problems.

Considering that solving the S-CNDP to optimality is computationally challenging, most authors have adopted heuristic approaches. The recent paper of Yaghini et al. (2014) proposes a tabu search algorithm that uses a neighborhood structure induced by three families of valid inequalities, namely the strong, flow cover and flow pack inequalities (Chouman et al. 2009). The neighborhood exploration uses popular heuristic techniques, such as local branching (Fischetti and Lodi 2003) and relaxation-induced neighborhoods (Danna et al. 2005). A related approach is the evolutionary algorithm of Paraskevopoulos et al. (2016), in which the authors use scatter and local search alongside new search operators that allow partial rerouting of multiple commodities in a single move. The computational experiments show that those two approaches are perhaps the most efficient heuristics at the time of this writing, as they generate better solutions when compared to an array of other methodologies, namely, the cycle-based tabu search of Ghamlouche et al. (2003), the path relinking algorithm of Ghamlouche et al. (2004), the cooperative tabu search of Crainic and Toulouse (2006), the local branching approach of Rodríguez-Martín and Salazar-González (2010), the MIP-based tabu search of Chouman et al. (2009), the MIP-based hybrid approach of Hewitt et al. (2010) and the hybrid capacity scaling approach of Katayama et al. (2009). It is worth noting that the method of Katayama et al. (2009), which is based on capacity scaling using column and row generation, seems to be the most competitive against the

heuristics of Yaghini et al. (2014) and Paraskevopoulos et al. (2016). The idea of capacity scaling comes originally from Pochet and Vyve (2004), where the authors demonstrate that it is an effective approach to solve capacitated lot sizing problems. In terms of exact methods, it is interesting to note the state-of-the-art study of Chouman et al. (2017), who develop a custom cutting plane algorithm and separation procedures for five classes of valid inequalities. Their experiments show that their separation algorithms perform better on aggregated formulations, i.e., formulations where commodities with a common origin or destination are aggregated into a single commodity, for instances with a small number of commodities, while their algorithms perform better for instances with a large number of commodities when applied to disaggregated formulations. We have decided to use disaggregated formulations based on the results of Chouman et al. (2017), as our multi-period models have a relatively large number of commodities.

To conclude the literature review, we note some practical applications closely related to our model. Notably, Bärmann et al. (2017) study the problem of expanding the German railway network, whose demand is anticipated to increase by 50% in the coming two decades. The distinct characteristic of their model is that capacity installation decisions may span over multiple periods, which implies that although investments have to be paid throughout the construction periods, the capacity becomes available only at the last construction period. Their work differs from ours in that they approach the problem from a revenue maximization perspective, consider cost restrictions per period, and, more importantly, address the network expansion part, starting from an existing feasible design. Stochastic network expansion has also been considered in Lai and Shih (2013), where the authors try to minimize a combination of network upgrading costs, and expected arc operations costs and unfulfilled demand. Although they consider scenario-dependent demand with 243 scenarios, their networks expand up to five periods, 15 nodes and 22 arcs, and they are an order of magnitude smaller than our instances, with respect to their number of variables. Other studies, such as Blanco et al. (2011) and Marin and Jaramillo (2008) address application-specific network expansion models using heuristics, while Petersen and Taylor (2001) use dynamic programming and develop a spreadsheet-based decision support system to investigate the economic viability of a new railway in Brazil. Although each

application has specific requirements, our model captures their predominant features and our methodologies can be easily adjusted to incorporate application-specific features, such as period-dependent budget constraints.

3. Problem Description and Formulation

We start by providing an arc-based formulation of the multi-period multi-commodity network design problem. To this end, we assume that for a given network of nodes $\mathcal{N}$ and arcs $\mathcal{A}$, there exists a set of commodities $\mathcal{K}$ that need to be routed from given origin nodes to given destination nodes during each time period $t \in T$. Each commodity should satisfy a period-dependent demand between its origin and its destination. Routing a commodity through an arc incurs a variable, commodity-specific cost. In addition, routing a commodity through an arc is possible only if this arc is activated in this period, or if it has been activated in an earlier period. Activating an arc in a specific period incurs a fixed cost, which accounts not only for activation but also for maintenance and operating costs for the remaining horizon. Under these considerations, we introduce the following notation:

Parameters

f_{ij}^t, cost of activating arc (i,j) at the start of period t

c_{ij}^k, cost of sending one unit of commodity k through arc (i,j)

u_{ij}, capacity of arc (i,j)

d^{kt}, demand of commodity k in period t

$o(k), d(k)$, origin and destination of commodity k, respectively

$$b_i^k = \begin{cases} 1, \text{if } i = o(k) \\ -1, \text{if } i = d(k) \\ 0, \text{ otherwise} \end{cases}$$

Decision Variables

x_{ij}^{kt}, fraction of demand of commodity k during period t that is directed through arc (i,j)

y_{ij}^t, $= 1$ if arc (i,j) is activated at the beginning of period t, 0 otherwise

The multi-period CNDP (M-CNDP) can then be formulated as follows.

$$v^* = \min \sum_{t\in T}\sum_{k\in\mathcal{K}}\sum_{(i,j)\in\mathcal{A}} c_{ij}^k d^{kt} x_{ij}^{kt} + \sum_{t\in T}\sum_{(i,j)\in\mathcal{A}} f_{ij}^t y_{ij}^t \qquad [M-CNDP] \qquad (1)$$

$$\text{s.t} \sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{kt} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{kt} = b_i^k, \qquad \forall i\in\mathcal{N}, \forall k\in\mathcal{K}, \forall t\in T, \qquad (2)$$

$$\sum_{k\in\mathcal{K}} d^{kt} x_{ij}^{kt} \le u_{ij} \sum_{l=1}^{t} y_{ij}^l, \qquad \forall (i,j)\in\mathcal{A}, t\in T, \qquad (3)$$

$$x_{ij}^{kt} \le \min\{1, \frac{u_{ij}}{d^{kt}}\} \sum_{l=1}^{t} y_{ij}^l, \qquad \forall (i,j)\in\mathcal{A}, \forall k\in\mathcal{K}, \forall t\in T, \qquad (4)$$

$$\sum_{t\in T} y_{ij}^t \le 1, \qquad \forall (i,j)\in\mathcal{A}, \qquad (5)$$

$$0 \le x_{ij}^{kt} \le 1, \qquad \forall (i,j)\in\mathcal{A}, k\in\mathcal{K}, t\in T, \qquad (6)$$

$$y_{ij}^t \in \{0,1\}, \qquad \forall (i,j)\in\mathcal{A}, \forall t\in T. \qquad (7)$$

The objective function (1) minimizes the joint costs of routing commodities and activating arcs throughout the horizon. The flow balance constraints (2) maintain the balance of each commodity in each node and period. Note that although we focus on the single-origin single-destination case per commodity, our formulation captures the more generic case were there are multiple demand sources ($b_i^k = 1$) and sinks ($b_i^k = -1$). Constraints (3) prevent the total amount of flow that is routed through each arc from exceeding that arc's capacity in each period. If no arcs have been activated so far, then all quantities routed through an arc are zero. Constraints (4) are redundant but potentially useful, since they strengthen the problem's LP relaxation. More precisely, they are the multi-period counterparts of the well-known "strong" constraints, often used in single-period problems to improve the LP relaxation (Gendron and Crainic 1994). Note that in multi-period problems there are $|\mathcal{K}||\mathcal{A}||T|$ such constraints, and including all of them can be prohibitive in large problems. Finally, (5) imply that each arc can be activated at most once throughout the problem horizon.

To avoid trivial solutions, we also assume that activating an arc earlier implies a higher cost, i.e., $f_{ij}^t > f_{ij}^{t+1}, \forall (i,j) \in \mathcal{A}, t \in T$. A special case of the above formulation arises when $u_{ij} \geq \sum_{k \in \mathcal{K}} d^{kt}$ for all $(i,j) \in \mathcal{A}$ and $t \in T$. Then, constraints (3) are redundant, and (4) read $x_{ij}^{kt} \leq \sum_{l=1}^{t} y_{ij}^l, \forall (i,j) \in \mathcal{A}, k \in \mathcal{K}$ and $t \in T$. We will hereafter refer to this case as the multi-period *uncapacitated* multi-commodity network design problem (M-UNDP). This variant is interesting in its own right, and we study it separately because it poses different decomposability features compared to the capacitated variant. Before we describe each decomposition method, we provide a unified heuristic approach that can be employed for both M-CNDP and M-UNDP.

4. Heuristic Search

A family of heuristics used widely in problems with an inherent multi-period structure, such as revenue management (Chand et al. 2002) and lot sizing (Beraldi et al. 2008), are rolling horizon heuristics. These assume that the problem can be solved by first considering a reduced part of the horizon and solving the associated problem, and then iteratively considering sequences of later periods until they reach the end of the horizon. The fact however that even single period problems are hard to solve implies that such methods may not be suitable to warm start a decomposition scheme. Another straightforward heuristic approach is to start by solving the problem period by period, allowing commodity flows through arcs that have been previously activated. We use this approach in our experiments as a benchmark and introduce a more elaborate method, namely a select-and-time heuristic that first selects which arcs are going to be used, and then decides when these arcs are going to be activated.

Selecting good arcs. Algorithm 1 describes the high-level steps of our select-and-time heuristic. The heuristic starts by trying to identify a set of arcs that are going to be activated in *some* period, without focusing on which period is the best to activate each arc. In order to select a set of such arcs, we solve two single-period instances. The first one, is tasked to output a good set of candidate arcs, by incorporating cost and demand information from the entire horizon, while the second one adopts a worst-case demand perspective, to make sure the proposed set of arcs leads to feasible solutions. We first start

by calculating weights that are used to average the fixed cost of each arc over periods. The weight of each period is determined in two steps. First, we set it equal to the fraction of remaining cumulative demand over the total demand, and then we normalize between zero and one (line 1). The first single-period model we solve has a fixed cost per arc (i,j) that is a weighted average of that arc's cost across the horizon, i.e., $\hat{f}_{ij} = \sum_t w^t f_{ij}^t$ (line 2). In order to capture future demand growth, we scale the first period demand of each commodity by $\max_t d^{kt} / \frac{1}{T} \sum_t d^{kt}$ (line 3). That way we obtain an instance with fixed costs that are more representative of the entire horizon, and demands that take into account fluctuations of future periods. After solving this problem, we store which arcs are selected (line 4). This selection of high-quality arcs may not be enough to guarantee feasibility throughout the horizon, and to tackle this we form an instance that has no arc activation cost and maximum demand from each commodity (line 5). This can be seen as a "worst-case" scenario from a demand perspective, because all commodity demands take their maximum values simultaneously. This instance can be solved efficiently as follows: since there are no arc activation costs, we solve an LP that allows positive flows via all arcs, and then select those arcs that have a strictly positive flow in the optimal solution. After solving this formulation (line 6), we form the set of arcs that are selected in either the first or the second model (line 7). After that initial selection of arcs, i.e., $\mathcal{A}^{pot}$, is determined, the decision of *when* to activate an arc is considered.

Arc activation timing. In order to decide when to activate arcs, we form single-period instances, whose fixed cost is a weighted average of each period's cost and the average cost of the remaining periods, and the variable cost is inflated by $\max_t d^{kt} / \frac{1}{|T|} \sum_t d^{kt}$ (lines 9 - 11). This way we take into account the variable cost of future periods, anticipating the potential increase in demand. Every time an arc is newly activated, we mark it as selected and do not consider its cost further (lines 15 - 16). It is worth noting that the procedure that updates the fixed costs is inspired by exponential smoothing forecasts. During early periods, we give more weight to the actual fixed costs rather than the anticipated future costs, acknowledging that early capacity activation decisions are more important, as they impact the routing decisions of the entire horizon. Thus, fixed costs for the first periods

Algorithm 1 The Select-and-Time heuristic procedure

Input: $d^{kt}, f^t_{ij}, c^k_{ij}, u_{ij}, b^k_i$

Output: $y^t_{ij}, x^{kt}_{ij}, v_H$

$w^t \leftarrow \frac{\sum_{l=t}^{|T|}\sum_{k\in\mathcal{K}} d^{kl}}{\sum_{l\in T}\sum_{k\in\mathcal{K}} d^{kl}}$; $w^t \leftarrow \frac{w^t}{\sum_{l\in T} w^l}$

$\hat{f}_{ij} \leftarrow \sum_{t\in T} w^t f^t_{ij}$

$\hat{d}^k \leftarrow d^{k1}\frac{\max_{t\in T} d^{kt}}{\frac{1}{|T|}\sum_{t\in T} d^{kt}}$

$\hat{y}_{ij} \leftarrow$ SolveSinglePeriod$(\hat{d}^k, \hat{f}_{ij}, c^k_{ij}, u_{ij}, b^k_i)$

$\hat{d}^k \leftarrow \max_{t\in T} d^{kt}$; $\hat{f}_{ij} \leftarrow 0$

$\bar{y}_{ij} \leftarrow$ SolveSinglePeriod$(\hat{d}^k, \hat{f}_{ij}, c^k_{ij}, u_{ij}, b^k_i)$

$\mathcal{A}^{pot} = \{(i,j) \in \mathcal{A} \mid \hat{y}_{ij} + \bar{y}_{ij} \geq 1\}$; $\mathcal{A}^0 = \mathcal{A} \backslash \mathcal{A}^{pot}$; $\mathcal{A}^1 \leftarrow \emptyset$

for $t \in T$ **do**

  $t^{max} \leftarrow \min\{t+1, |T|\}$

  $$\hat{f}_{ij} \leftarrow \begin{cases} w^t f^t_{ij} + \frac{1-w^t}{|T|-t^{max}+1}\sum_{l=t^{max}}^{|T|} f^l_{ij}, & \forall (i,j) \in \mathcal{A}^{pot} \\ 0, & \forall (i,j) \in \mathcal{A}^1 \end{cases}$$

  $\hat{c}^k_{ij} \leftarrow c^k_{ij}\frac{\max_t d^{kt}}{\frac{1}{|T|}\sum_t d^{kt}}$; $\hat{d}^k \leftarrow d^{kt}$

  $\hat{y}^t_{ij} \leftarrow$ SolveSinglePeriod$(\hat{d}^k, \hat{f}_{ij}, \hat{c}^k_{ij}, u_{ij}, b^k_i \mid y_{ij} = 1, \forall (i,j) \in \mathcal{A}^1; y_{ij} = 0, \forall (i,j) \in \mathcal{A}^0)$

  for $(i,j) \in \mathcal{A}^{pot}$ **do**

    if $\hat{y}^t_{ij} = 1$ **then**

      $\mathcal{A}^{pot} \leftarrow \mathcal{A}^{pot} \backslash \{(i,j)\}$

      $\mathcal{A}^1 \leftarrow \mathcal{A}^1 \cup \{(i,j)\}$

    end if

  end for

end for

$(y^t_{ij}, x^{kt}_{ij}, v_H) \leftarrow$ SolveMultiPeriodLP$(d^{kt}, f^t_{ij}, c^k_{ij}, u_{ij}, b^k_i \mid y^t_{ij} = \hat{y}^t_{ij}, \forall (i,j) \in \mathcal{A}, \forall t \in T)$

carry the largest weights, while for later periods the weights of each period become smaller.

Having decided when to activate each arc, the problem reduces to solving a series of linear multi-commodity flow problems for each period in the planning horizon (line 20).

The advantage of the select-and-time heuristic is that although it works with a reduced problem size, it takes into account information from multiple periods. In our computational experiments, we use a MIP solver for the single-period instances, but we do not necessarily solve these instances to optimality, as our objective is to feed the decomposition schemes with good solutions quickly. A stand-alone heuristic approach could utilize a relax-and-fix method or more advanced techniques, such as very large neighborhood search (Ahuja et al. 2002) or tabu search variants (Glover 1990). Although there is merit in these methods, they are typically more time consuming than our approach and therefore do not constitute such a good fit for our purpose.

5. Decomposition Algorithms

We take advantage of different characteristics of capacitated and uncapacitated variants: for the former we note that dualizing the flow balance constraints decomposes the problem into a series of multi-period, single-link subproblems, while for the latter, fixing the arc activation decisions leads to a series of single-period, single-commodity shortest paths. Thus, we utilize Lagrange relaxation and Benders decomposition, respectively, which are able to exploit these two decomposable structures.

5.1. Capacitated Problems

5.1.1. Lagrange relaxation of M-CNDP By dualizing constraints (2) in the objective function (1), M-CNDP decomposes into a series of single-arc multi-period subproblems. We let π_i^{kt} denote the dual values associated with constraints (2). To miminize notational clutter, we remove the indices (i, j) and formulate the single-arc problems as follows:

$$v_{ij}^{\pi} = \min \sum_{t\in T}\sum_{k\in\mathcal{K}}(c^k d^{kt} + \pi_i^{kt} - \pi_j^{kt})x^{kt} + \sum_{t\in T} f^t y^t \qquad [SUB] \qquad (8)$$

$$\text{s.t} \quad \sum_{k\in\mathcal{K}} d^{kt}x^{kt} \le u\sum_{l=1}^{t} y^l, \qquad \forall t \in T \qquad (9)$$

$$x^{kt} \leq \min\{1, \frac{u}{d^{kt}}\} \sum_{l=1}^{t} y^l, \qquad \forall k \in \mathcal{K}, \forall t \in T \quad (10)$$

$$\sum_{t \in T} y^t \leq 1, \qquad (11)$$

$$0 \leq x^{kt} \leq 1, \qquad \forall k \in \mathcal{K}, t \in T \quad (12)$$

$$y^t \in \{0,1\}, \qquad \forall t \in T. \quad (13)$$

Then, the Lagrange dual optimization problem can be expressed as

$$v^*_{LR} = \max_{\pi} v(\pi) = \max_{\pi} \left\{ \sum_{(i,j) \in \mathcal{A}} v^{\pi}_{ij} - \sum_{t \in T} \sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{K}} b^k_i \pi^{kt}_i \right\}. \quad (14)$$

It is well-known that (14) is a concave optimization problem and that $v(\pi)$ is piece-wise linear (Fisher 1981). In order to calculate v^*_{LR}, we can evaluate $v(\pi)$ pointwise and apply subgradient optimization (Boyd et al. 2003). To this end, we devise an efficient approach for solving $[SUB]$, which should be invoked every time $v(\pi)$ is to be calculated. We start with an observation about the structure of optimal solutions of $[SUB]$.

REMARK 1. For any given vector $y^t \in \{0,1\},\ t \in T$ that satisfies (11), problem $[SUB]$ decomposes into a series of single-period, linear bounded knapsack problems, each of which can be solved in $\mathcal{O}(|\mathcal{K}| \log |\mathcal{K}|)$ time. Thus, $[SUB]$ can be solved in $\mathcal{O}(|\mathcal{K}||T| \log |\mathcal{K}|)$ time.

Remark 1 is useful for comparing alternative solutions that activate arcs in different periods, and have optimal routing decisions x^{kt}, given their arc activation timing y_t. More specifically, given two alternative solutions a and b for SUB with $y^{t_a} = 1$, $y^{t_b} = 1$ and $t_a < t_b$, the optimal routing of the corresponding linear programs will be related as follows:

$$\begin{cases} x^{kt}_a = x^{kt}_b = 0 \text{ for } t < t_a \\ x^{kt}_a = x^{kt}_b \text{ for } t \geq t_b. \end{cases}$$

In other words, both solutions have zero routing variables before the earliest arc is activated, and identical routing variables in periods where the arc is activated in both solutions. Thus, solution a will have a better objective than solution b if the decision to activate the arc earlier and the corresponding routing variables improve the objective,

i.e., if $\sum_{t=t_a}^{t=t_b-1}(c^k d^{kt} + \pi_i^{kt} - \pi_j^{kt})x_a^{kt} + f^{t_a} < f^{t_b}$. This can be utilized to obtain an efficient algorithm for $[SUB]$. Starting from the last period, we calculate the combined cost of activating the arc and routing the commodities. If the corresponding objective value is negative, we activate the arc ($y^{|T|} = 1$), and determine the routing by solving the corresponding bounded linear knapsack problem. Otherwise, we do not activate ($y^{|T|} = 0$). Then, in every period t we only need to calculate the marginal benefit of activating the arc at t, if it is already activated at a later period, or the full objective value and check if it is negative, if the arc is not activated already. This is a greedy procedure, in which we need to calculate the routing variables of each period and the corresponding part of the objective function, and store information about periods we have already evaluated.

We next provide information on how strong the bound obtained by the Lagrange relaxation is. To this end, we note that the linear relaxation of [SUB] has the *integrality property*, i.e., its basic feasible solutions have $y^t \in \{0, 1\}$ for all $t \in T$ without imposing the integrality restrictions explicitly (Geoffrion 1974).

PROPOSITION 1. *Problem [SUB] has the integrality property, and therefore* $v_{LP}^* = v_{LR}^*$.

Proof. See Online Supplement.

The fact that the solutions are integral means that the lower bound obtained by Lagrange relaxation is identical to that obtained by the LP relaxation of S-CNDP. However, Lagrange relaxation may be more attractive computationally, because it takes advantage of the decomposable structure by arc. Subgradient optimization, an efficient method that is employed to solve the Lagrange dual problem (14) has, however, a shortcoming: the flow balance constraints (2) that are dualized in the objective function are typically violated. The next section provides a method to circumvent this issue and utilizes a procedure that searches for improved upper bounds during subgradient optimization.

5.1.2. Lower Bounds and Approximate Primal Solutions Solving the Lagrange dual problem (14) with regular subgradient optimization has an important disadvantage, which is that the (near)-optimal Lagrange multipliers result in a primal solution that typically violates the dualized constraints (2). Moreover, devising good quality branching rules from that solution is challenging, because the arc activation decisions obtained by

subgradient optimization (y_{ij}^t) are integral. To tackle this issue, we employ the volume algorithm of Barahona and Anbil (2000), an extension of the subgradient algorithm that returns, alongside the Lagrangian lower bound, a y-solution with small violations of (2). In this way, we are able to have a solution with fractional binary variables at each subgradient iteration. This fractional solution with small violations can then be used to recover a primal feasible solution, without violations, by solving a Dantzig-Wolfe master program with dual simplex (Barahona and Anbil 2002), or guide the branching decision in a heuristic branch-and-bound scheme (de Araujo et al. 2015), or to define a neighborhood to be used in a local search heuristic, which is the approach we use here. Small violations of the recovered solutions imply it is a good approximation of the LP relaxation solution of (1)-(7), and therefore useful in defining a good search neighborhood. The idea of using a primal fractional solution and a heuristic solution to guide the search space has appeared before in diving heuristics and in the relaxation-induced neighborhood search heuristic (Danna et al. 2005), used by state-of-the-art solvers, such as CPLEX and Gurobi. To the best of our knowledge, though, neighborhoods induced by approximate primal solutions are not common in the literature. The next section describes our approach in detail.

5.1.3. Finding Upper Bounds Within the framework of the volume algorithm we find feasible solutions of the original problem, M-CNDP, by employing three local search heuristics. The first heuristic checks if moving the arc activation decisions later in time provides an improved solution. Concretely, suppose that a feasible solution $(\bar{x}_{ij}^{kt}, \bar{y}_{ij}^t)$ is known and denote $t^* < |T|$ the period such that $\bar{y}_{ij}^{t^*} = 1$, for some arc (i,j). We want to check if the change $\bar{y}_{ij}^{t^*} \leftarrow 0$, $\bar{y}_{ij}^{t^*+1} \leftarrow 1$ is beneficial. Provided that this swap leads to a feasible solution for period t^*, the fixed cost is improved by $\Delta f_{ij} = f_{ij}^{t^*+1} - f_{ij}^{t^*}$ but the flow cost of period t^* is potentially increased, because commodities cannot be routed through arc (i,j) during period t^*. Thus, one needs to check how much the flow cost of period t^* increases when arc (i,j) is not available, and compare its increase with Δf_{ij}. This procedure is described in Algorithm 2 below for $t < |T|$, but an adaption to capture the case $t = |T|$ is straightforward, at the expense of more involved notation. The LP model M_t in line 3 is constructed by considering the variables and constraints of M-CNDP indexed over period t, with fixed setup activation decisions y_{ij}^t.

Algorithm 2 Single-arc search heuristic

Input: Period t, feasible solution $(\bar{y}^t_{ij}, \bar{x}^{kt}_{ij})$, objective value $\bar{z}$

Output: Improved feasible solution $(\bar{y}^t_{ij}, \bar{x}^{kt}_{ij})$, objective value $\bar{z}$

$z^t \leftarrow \sum_{(ij)\in\bar{\mathcal{A}}} c^k_{ij} d^{kt} \bar{x}^k_{ij}$
$\bar{\mathcal{A}} \leftarrow \{(i,j) \in \mathcal{A} : \bar{y}^t_{ij} = 1\}$
$M^t \leftarrow$ LP model for period t with $x^{kt}_{ij} \leq \sum_{l=1}^{t} \bar{y}^l_{ij}, \forall (i,j) \in \mathcal{A}, \forall k \in \mathcal{K}$
for $(i,j) \in \bar{\mathcal{A}}$ **do**
    $\Delta fc \leftarrow f^t_{ij} - f^{t+1}_{ij}$
    Solve M^t with $x^{kt}_{ij} = 0, \forall k \in \mathcal{K}$
    $z' \leftarrow$ Objective(M^t) if M^t is feasible, otherwise $z' = \infty$; $\Delta z \leftarrow z' - z^t$
    if $\Delta fc - \Delta z > 0$ **then** ▷ Fixed cost reduction higher than flow cost increase
        $z \leftarrow z - (\Delta fc - \Delta z)$
        $\bar{y}^t_{ij} \leftarrow 0, \bar{y}^{t+1}_{ij} \leftarrow 1, \bar{x}^{kt}_{ij} \leftarrow$ Solution(M^t), $z^t \leftarrow \sum_{(i,j)\in\bar{\mathcal{A}}} c^k_{ij} d^{kt} \bar{x}^{kt}_{ij}$
    end if
end for
return $(\bar{y}^t_{ij}, \bar{x}^{kt}_{ij}, z)$

For large-scale models, attempting to check all arcs in set $\bar{\mathcal{A}}$ might be very expensive computationally. To this end, we restrict the search to a neighborhood defined by comparing the arc-activating variables of the best feasible solution (the incumbent) and the possibly fractional arc-activating variables, retrieved by the volume algorithm. Concretely, for a cutoff point $\delta \in (0,1)$ and for each arc (i,j), we set $y^{t^*}_{ij} = 1$ if $t^* = \arg\max\{y^t_{ij} : y^t_{ij} > \delta\}$ exists and $y^t_{ij} = 0, \forall t \in T$, otherwise. Then, we compare the resulting vector with the incumbent solution. For each period, we invoke Algorithm 2 only for the arcs that are different in the incumbent and the rounded solution. In addition, we sort the arcs in order of decreasing Δfc, in order to ensure that we first check the arcs that give the highest period-on-period reduction in fixed costs. This procedure leads to a more efficient implementation, as it utilizes a small set of relevant arcs, where there are discrepancies between the rounded approximate primal solution of the Lagrange relaxation and the incumbent solution.

Our second heuristic uses an idea similar to local branching (Fischetti and Lodi 2003), but customized to our problem. In particular, we would like to check if changing the time when an arc is activated can lead to improved solutions. To this end, given a feasible solution $\bar{y}_{ij}^t$ we define $\bar{\mathcal{A}}_t = \{(i,j) \in \mathcal{A} | \bar{y}_{ij}^t = 1\}, \forall t \in T$ and impose the constraints

$$\sum_{l=\max\{t-\tau^-,0\}}^{\min\{|T|,t+\tau^+\}} y_{ij}^l = 1, \forall t \in T, \forall (i,j) \in \bar{\mathcal{A}}_t \tag{15}$$

to the original problem M-CNDP. This explores a small neighborhood of the incumbent, in which the capacity activation decisions remain the same, but their timing can be shifted up to τ^+ periods later or τ^- periods earlier.

Our third and last heuristic applies a simple fixing procedure based on the values of the fractional y variables recovered by the volume algorithm, as follows. First, it interprets each fractional value as signifying a degree of "ambiguity", meaning that the exploration of variables closer to 0.5 takes priority. To this end, it sorts the y variables in increasing $|y_{ij}^t - 0.5|$ values and, for a given fraction f, it finds l_f and u_f in $(0,1)$ such that the first $f|\mathcal{A}||T|$ variables lie in the interval $[l_f, u_f]$. Finally, in M-CNDP, it fixes to zero those variables that are lower than l_f and to one those that are higher than u_f, respectively, and solves the remaining MIP model. The advantage of this heuristic is that it can be tuned easily by only changing f, while it searches a promising neighborhood of the fractional solution.

As our three heuristics require varying degrees of computational effort, we do not invoke all of them with the same frequency. Specifically, we invoke the single-arc heuristic every 300 subgradient iterations, while the local branching and fixing heuristics are applied after the final lower bound is computed. First, the fixing heuristic explores 10% of the fractional solution and passes on the potentially improved objective to the local-branching heuristic, for which we use $\tau^- = 4$ and $\tau^+ = 9$. Then, four rounds of the single-arc heuristic are invoked, after which the fixing heuristic explores 30% of the fractional solution and outputs the final upper bound of our algorithm.

5.2. Uncapacitated Problems

Removing constraints (3) from formulation M-CNDP gives rise to the uncapacitated version of the multi-period, multi-commodity network design problem, M-UNDP. This model can be used to represent systems with ample capacity or systems in which the cost of installing extra capacity is negligible. From a methodological point of view, the interesting characteristic of the uncapacitated model is that it exhibits a particular decomposable structure: for fixed arc opening decisions, M-FUMP decomposes into a series of independent shortest path problems, per commodity and per period. In this section, we take advantage of this property to devise an improved Benders decomposition algorithm.

5.2.1. Regular Benders Decomposition Benders decomposition hinges on the observation that a mixed-integer linear programming (MIP) model can be conceptualized as a combinatorial family of linear programs. Concretely, if we let Y denote the set of binary vectors y that satisfy $\sum_{t\in T} y_{ij}^t \leq 1, \forall (i,j) \in \mathcal{A}$, then for a given $\bar{y} \in Y$, the uncapacitated model reduces to the following linear program:

$$z(\bar{y}) = \min \sum_{t\in T}\sum_{k\in\mathcal{K}}\sum_{(i,j)\in\mathcal{A}} c_{ij}^k d^{kt} x_{ij}^{kt} \tag{16}$$

$$\text{s.t} \sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{kt} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{kt} = b_i^k, \qquad [\pi_i^{kt}],\ \forall i \in \mathcal{N}, \forall k \in \mathcal{K}, \forall t \in T, \tag{17}$$

$$x_{ij}^{kt} \leq \sum_{l=1}^{t} \bar{y}_{ij}^l, \qquad [\lambda_{ij}^{kt}],\ \forall (i,j) \in \mathcal{A}, \forall k \in \mathcal{K}, \forall t \in T, \tag{18}$$

$$x_{ij}^{kt} \geq 0, \qquad \forall (i,j) \in \mathcal{A}, k \in \mathcal{K}, t \in T. \tag{19}$$

Then, the original problem can be expressed as $z = \min_{y\in Y}\{z(y) + \sum_{t\in T}\sum_{(i,j)\in\mathcal{A}} f_{ij}^t y_{ij}^t\}$. It is worth noting that (16)-(19) decomposes into a series of shortest path problems, each one corresponding to a commodity–period pair. Using O and D to denote the origin and destination of each commodity respectively, the dual of (16)-(19) can be written as

$$z(\bar{y}) = \max \sum_{t\in T}\sum_{k\in\mathcal{K}} (\pi_O^{kt} - \pi_D^{kt}) - \sum_{(i,j)\in\mathcal{A}}\sum_{k\in\mathcal{K}}\sum_{t\in T}(\sum_{l=1}^{t} \bar{y}_{ij}^l)\lambda_{ij}^{kt} \tag{20}$$

$$\text{s.t. } \pi_i^{kt} - \pi_j^{kt} - \lambda_{ij}^{kt} \leq c_{ij}^k d^{kt}, \qquad \forall (i,j) \in \mathcal{A}, k \in \mathcal{K}, t \in T \quad (21)$$

$$\lambda_{ij}^{kt} \geq 0, \qquad \forall (i,j) \in \mathcal{A}, k \in \mathcal{K}, t \in T. \quad (22)$$

The dual polyhedron $\Delta = \{(\boldsymbol{\pi}, \boldsymbol{\lambda}) | (21) - (22)\}$ is non-empty since the null vector is feasible for positive demands and routing costs. This means that Δ can be represented by a finite set of extreme rays, denoted by R_Δ, and a finite set of extreme points, denoted by P_Δ (Schrijver 1998). Using this representation, the Benders reformulation of M-FUMP is as follows:

$$\min \sum_{t \in T} \sum_{(i,j) \in \mathcal{A}} f_{ij}^t y_{ij}^t + z \quad (23)$$

$$\text{s.t.} \sum_{t \in T} \sum_{k \in \mathcal{K}} \left(\bar{\pi}_O^{kt} - \bar{\pi}_D^{kt}\right) - \sum_{(i,j) \in \mathcal{A}} \sum_{k \in \mathcal{K}} \sum_{t \in T} \left(\sum_{l=t}^{|T|} \bar{\lambda}_{ij}^{kl} \right) y_{ij}^t \leq 0, \qquad \forall (\bar{\pi}_i^{kt}, \bar{\lambda}_{ij}^{kt}) \in R_\Delta \quad (24)$$

$$\sum_{t \in T} \sum_{k \in \mathcal{K}} \left(\bar{\pi}_O^{kt} - \bar{\pi}_D^{kt}\right) - \sum_{(i,j) \in \mathcal{A}} \sum_{k \in \mathcal{K}} \sum_{t \in T} \left(\sum_{l=t}^{|T|} \bar{\lambda}_{ij}^{kl} \right) y_{ij}^t \leq z \qquad \forall (\bar{\pi}_i^{kt}, \bar{\lambda}_{ij}^{kt}) \in P_\Delta \quad (25)$$

$$\sum_{t \in T} y_{ij}^t \leq 1, \qquad \forall (i,j) \in \mathcal{A} \quad (26)$$

$$y_{ij}^{kt} \in \{0,1\}, \qquad \forall (i,j) \in \mathcal{A}, t \in T. \quad (27)$$

Constraints (24), the *feasibility cuts*, prevent the objective function of the dual problem (20)-(22) from being unbounded, and therefore the corresponding primal problem (16)-(19) from becoming infeasible. Constraints (25) are the *optimality cuts*, which impose that

$$z = \max_{P_\Delta} \left\{ \sum_{t \in T} \sum_{k \in \mathcal{K}} (\pi_O^{kt} - \pi_D^{kt}) - \sum_{(i,j) \in \mathcal{A}} \sum_{k \in \mathcal{K}} \sum_{t \in T} \left(\sum_{l=1}^{t} y_{ij}^l \right) \lambda_{ij}^{kt} \right\}$$

and therefore ensure that the z corresponds to the optimal objective value of the dual subproblem (20)-(22). Although this formulation is useful conceptually, the often large cardinalities of P_Δ and R_Δ make the a priori inclusion of the corresponding constraints inefficient from a computational point of view. Benders himself noted that formulation (23)-(27) can be solved for a limited number of feasibility and optimality cuts, in which

case it delivers a lower bound on the optimal objective function value, and new cuts can be added dynamically. Specifically, in each iteration the optimal y_{ij}^t values can be used to solve the pair of primal-dual subproblems (16)-(19) and (20)-(22) respectively. If the primal subproblem is infeasible, then the dual subproblem returns a feasibility cut, (24), whereas when the primal subproblem is feasible, the dual subproblem returns an optimality cut, (25), which is added to the master problem and the algorithm proceeds to the next iteration. If no optimality or feasibility cut is violated, then the algorithm has converged to an optimal solution.

Modern implementations of Benders decomposition rely on this framework, but often employ additional computational enhancements. First, the decomposition of the subproblem into a series of $|\mathcal{K}||T|$ subproblems allows the generation of individual cuts from each commodity–period pair. To this end, z is replaced by $\sum_{k\in\mathcal{K}}\sum_{t\in T} z^{kt}$ in (23), the summations over periods and commodities are dropped in (24) and (25), and the sets of extreme rays and points are defined over the polyhedra $\Delta^{kt} = \{(\boldsymbol{\pi},\boldsymbol{\lambda}) \mid \pi_i^{kt} - \pi_j^{kt} - \lambda_{ij}^{kt} \leq c_{ij}^k d^{kt}; \lambda_{ij}^{kt} \geq 0, \forall (i,j) \in \mathcal{A}\}$. The corresponding cuts are denser and tend to be more effective than a single cut generated from the aggregated subproblem (23)-(27) (Cordeau et al. 2001). Second, instead of solving the master program (23)-(27) to optimality in each iteration, we take advantage of the callback capabilities of modern solvers to solve it only once and add the cuts dynamically in the branch-and-bound tree. A callback allows the user to retrieve information and change parts of a model after predefined events. In this context, we start by solving the master program with a limited number of cuts, use a callback to invoke the cut-generating procedure every time a feasible solution is found and to add the generated cuts, and then return control to the solver. Bai and Rubin (2009) and Adulyasak et al. (2015) show that this technique gives very good results for the minimum tollbooth problem and for the stochastic production routing problem, respectively. All our Benders implementations employ this scheme, in addition to adding cuts from individual subproblems, and to adding two classes of valid inequalities which warm-start the master problem, as detailed in Section 5.2.5.

5.2.2. FSZ Benders Decomposition In order to unify optimality and feasibility cuts, Benders (1962) suggested a soft penalization mechanism of infeasibility, namely the introduction of a dummy variable in the primal subproblem with a large objective coefficient. This corresponds to bounding the rays of the dual polyhedron, thereby eliminating the need to generate feasibility cuts. Building upon this idea, Fischetti et al. (2010) suggest an alternative normalization approach that uses the best known flow cost values $\bar{z}^{kt}$ in the subproblem formulation. Let $(\bar{y}^t_{ij}; \bar{z}^{kt})$ denote a feasible solution of the master problem (23)-(27). For notational brevity, we suppress the commodity and period notation when we refer to a single subproblem, i.e., for a specific commodity-period pair, if there is no ambiguity. Using this notation, Fischetti et al.'s subproblem (FSZ) can be formulated as follows.

$$\max \quad \pi_O - \pi_D - \sum_{(i,j)\in\mathcal{A}} (\sum_{l=1}^{t} \bar{y}^l_{ij})\lambda_{ij} - \bar{z}\eta \qquad [FSZ] \tag{28}$$

$$\text{s.t.} \quad \pi_i - \pi_j - \lambda_{ij} \leq c_{ij} d\eta, \qquad \forall (i,j) \in \mathcal{A} \tag{29}$$

$$\sum_{(i,j)\in\mathcal{A}} w_{ij}\lambda_{ij} + w_0\eta = 1 \tag{30}$$

$$\eta \geq 0; \ \lambda_{ij} \geq 0, \qquad \forall (i,j) \in \mathcal{A}. \tag{31}$$

In this formulation, the user is able to select non-negative weights w_{ij} and w_0 to better configure the normalization hyperplane (30). The regular Benders subproblem arises as the special case of $w_0 = 1$ and $w_{ij} = 0$. Our implementation sets the convexity condition $w_0 = w_{ij} = 1$ for each arc $(i,j) \in \mathcal{A}$, and therefore can be considered as an intermediate choice between Benders decomposition with and without subproblem bounding. As long as non-negative weights are selected, FSZ has always a feasible solution and is bounded. Therefore, a cut $\pi_O - \pi_D - \sum_{(i,j)\in\mathcal{A}} (\sum_{l=1}^{t} \bar{y}^l_{ij})\lambda_{ij} - \bar{z}\eta \leq 0$ can always be generated. Note also that if no open arc has been activated until period t, i.e., $\sum_{l=1}^{t} \bar{y}^l_{ij} = 0$ for each $(i,j) \in \mathcal{A}$, an optimal solution will generate a cut with $\eta = 0$ and $\sum_{(i,j)\in\mathcal{A}} \lambda_{ij} = 1$. Thus, this subproblem is able to generate not only optimality but also feasibility cuts. Another subtle point in which this method differs from the regular normalization is that the bounding hyperplane

(30) is defined over the subspace of the λ_{ij} variables. In the absence of negative cycles, FSZ has unbounded rays in the λ_{ij} subspace, and therefore to cut off infeasible solutions from the master one should at minimum impose the normalization on the dual space that generates the cut coefficients of the y^t_{ij}. As in Fischetti et al. (2010), we were able to verify that this selection results in important computational improvements compared to normalizing over the entire dual space.

5.2.3. Pareto-Optimal Cuts Magnanti and Wong (1981) devised yet another influential cut selection framework for Benders decomposition. The cut selection problem becomes relevant when the dual subproblem (20)-(22) has multiple optimal solutions, and therefore one has to select the best among alternative cuts. A criterion that partially quantifies cut quality is *cut dominance* (Magnanti and Wong 1981): a cut generated from the extreme point $(\boldsymbol{\pi}^1, \boldsymbol{\lambda}^1)$ is said to dominate another cut generated from $(\boldsymbol{\pi}^2, \boldsymbol{\lambda}^2)$ iff

$$\pi^1_O - \pi^1_D - \sum_{(i,j)\in\mathcal{A}} \left(\sum_{l=1}^{t} y^l_{ij} \right) \lambda^1_{ij} \geq \pi^2_O - \pi^2_D - \sum_{(i,j)\in\mathcal{A}} \left(\sum_{l=1}^{t} y^l_{ij} \right) \lambda^2_{ij}$$

holds for all $y^t_{ij} \in Y = \{y^t_{ij} \in \{0,1\} \mid \sum_{t\in T} y^t_{ij} \leq 1, \forall (i,j) \in \mathcal{A}\}$ with strict inequality for at least one point. A cut is non-dominated, or *Pareto optimal* (PO) if there is no other cut that dominates it. Magnanti and Wong (1981) devised a mechanism that generates PO cuts by solving an additional linear program, formulated as follows. First, let $\boldsymbol{y}^r = (y^{t,r}_{ij})^{t\in T}_{(i,j)\in\mathcal{A}}$ denote a *core point*, i.e., a point that lies in the relative interior of $conv(Y)$, with $Y = \{y^t_{ij} \in \{0,1\} \mid \sum_{t\in T} y^t_{ij} \leq 1, \forall (i,j) \in \mathcal{A}\}$. Then, denoting by $\bar{y}^t_{ij}$ the master problem solution and by $z^* = \max\{\pi_O - \pi_D - \sum_{(i,j\in\mathcal{A}} \sum_{l=1}^{t} \bar{y}^l_{ij} \lambda^t_{ij} : (\boldsymbol{\pi}, \boldsymbol{\lambda}) \in \Delta^{kt}\}$ the subproblem solution for the pair (k,t), one can identify a PO cut by solving the following subproblem:

$$\max \quad \pi_O - \pi_D - \sum_{(i,j)\in\mathcal{A}} \left(\sum_{l=1}^{t} y^{l,r}_{ij} \right) \lambda_{ij} \qquad [PO - SUB] \tag{32}$$

$$\text{s.t.} \quad \pi_i - \pi_j - \lambda_{ij} \leq c_{ij} d, \qquad \forall (i,j) \in \mathcal{A} \tag{33}$$

$$\pi_O - \pi_D - \sum_{(i,j)\in\mathcal{A}} \left(\sum_{l=1}^{t} \bar{y}^l_{ij} \right) \lambda_{ij} = z^* \tag{34}$$

$$\lambda_{ij} \geq 0 \qquad \forall (i,j) \in \mathcal{A}. \tag{35}$$

Constraint (34) ensures that the new point is an optimal solution to the original subproblem, while the objective function ensures that it is a PO cut. Although finding a core point is $\mathcal{NP}$-hard in general (Papadakos 2008), the simple structure of Y makes it possible to characterize a family of core points, as detailed in the following remark.

REMARK 2. The point $y_{ij}^{t,r} = \epsilon$ for each $(i,j) \in \mathcal{A}, t \in T$ with $0 < \epsilon < \frac{1}{|T|}$ is in the relative interior of $conv(Y)$.

This family of core points is easy to characterize but lacks two important features, that make it less attractive for a practical implementation. First, some points may become infeasible as new feasibility cuts are generated. For example, suppose a small $\epsilon > 0$ is selected, representing a core point very close to the null vector but with positive coordinates. The generation of cutset inequalities (Barahona 1996), such as $\sum_{l=1}^{t} \sum_j y_{Oj}^l \geq 1$, which impose that at least one arc emanating from the origin should be activated, will render this point infeasible. This is a relevant concern, because cutset inequalities are generated within the realm of the Benders algorithm as feasibility cuts (Costa et al. 2009). Second, a desirable property of a core point is to cut off as much space from the current point $\bar{y}_{ij}^t$ as possible. To this end, we note that we can generate a core point with an appropriate perturbation of $\bar{y}_{ij}^t \in Y$.

REMARK 3. If $\bar{y}_{ij}^t \in Y$, then

$$y_{ij}^{t,r} = \begin{cases} 1-\epsilon, & \text{if } \bar{y}_{ij}^t = 1 \\ \epsilon, & \text{if } \bar{y}_{ij}^t = 0 \end{cases}, \forall (i,j) \in \mathcal{A}, t \in T$$

is a core point for $\epsilon \in (0,1)$.

Selecting a small ϵ guarantees that we generate a core point which will never be infeasible and has the additional benefit that it lies in the neighborhood of $\bar{y}_{ij}^t$. Thus, we make use of this selection policy in our implementation. We also note that, from a computational perspective, an optimal solution to the regular Benders subproblem (20)-(22) is feasible to the PO subproblem. When the two problems are solved back-to-back, this feasible solution is used by modern solvers as a starting point to optimize PO-SUB efficiently. As suggested by our computational experiments, this typically requires a small number of simplex pivots, not only because the initial point is feasible, but also because the new objective function is a perturbed version of the previous one.

5.2.4. Efficient Generation of Pareto-Optimal Cuts Even if solving two consecutive linear programs to obtain PO cuts may come at a low computational cost, it may have a noticeable impact on performance, especially when solving large-scale instances. For single-period uncapacitated network design problems, Magnanti et al. (1986) have shown that a PO cut can be generated by solving a single minimum cost flow problem instead of two generally structured linear programs. Interestingly, their main argument can be readily extended to multi-period problems such as the one we consider here. To see this, note that the dual of (32)-(35) can be written as follows:

$$\max \quad d \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} - z^* x_0 \tag{36}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{A}_i^+} x_{ij} - \sum_{j\in\mathcal{A}_i^-} x_{ji} = \begin{cases} 1+x_0, & \text{if} \quad i=o \\ -1-x_0, & \text{if} \quad i=d \\ 0, & \text{otherwise} \end{cases}, \qquad \forall i\in\mathcal{N} \qquad [\pi_i] \tag{37}$$

$$0 \le x_{ij} \le x_0 \sum_{l=1}^{t} \bar{y}_{ij}^{l} + \sum_{l=1}^{t} y_{ij}^{l,r}, \qquad \forall (i,j)\in\mathcal{A} \qquad [\lambda_{ij}\ge 0] \tag{38}$$

where x_0 represents the dual price of constraint (34). Magnanti et al. (1986) observe that x_0 can be fixed to any value greater than or equal to $\sum_{(i,j)\in\mathcal{A}}\sum_{l=1}^{t} y_{ij}^{l,r}$ at an optimal solution. In what follows, and also in our computational experiments, we set $x_0 = \sum_{(i,j)\in\mathcal{A}}\sum_{l=1}^{t} y_{ij}^{l,r}$. Therefore, the objective term z^*x_0 becomes a constant and (36)-(38) is recast as a minimum cost flow problem, while its dual solution corresponds to a PO cut.

An added benefit of this formulation is that we can exploit its structure in a multi-period setting. In particular, under certain circumstances, the optimal solution of (36)-(38) for a given commodity-period pair may yield information about the optimal solution of later periods of the same commodity, as stated formally in Proposition 2 below.

PROPOSITION 2. *Let* $(\tilde{x}_{ij}^t; \tilde{\pi}_i^t, \tilde{\lambda}_{ij}^t)$ *be an optimal primal-dual pair of (36)-(38) when solved for* $t < T$. *Then,* $(\tilde{x}_{ij}^t; \kappa\tilde{\pi}_i^t, \kappa\tilde{\lambda}_{ij}^t)$ *with* $\kappa = \frac{d^{t+1}}{d^t}$ *is an optimal primal-dual pair for (36)-(38) when solved for* $t+1$ *if any of the following cases holds true:*

1. $\bar{y}_{ij}^{t+1} = 0$ *for all* $(i,j) \in \mathcal{A}$
2. $\tilde{\lambda}_{ij}^{t} = 0$ *for all* $(i,j) : \bar{y}_{ij}^{t+1} > 0$.

Proof. For case 1, note that the network is identical in periods t and $t+1$, and therefore $\tilde{x}_{ij}^{t}$ is also optimal for $t+1$. Since $(\tilde{\pi}_i^t, \tilde{\lambda}_{ij}^t)$ is optimal for t, its feasibility implies that $\tilde{\pi}_i^t - \tilde{\pi}_j^t - \tilde{\lambda}_{ij}^t \leq d^t c_{ij} \Rightarrow \kappa\tilde{\pi}_i^t - \kappa\tilde{\pi}_j^t - \kappa\tilde{\lambda}_{ij}^t \leq d^{t+1} c_{ij} \Rightarrow (\kappa\tilde{\pi}_i^t, \kappa\tilde{\lambda}_{ij}^t)$ is feasible for $t+1$. Also, from strong duality for $(\tilde{x}_{ij}^t; \tilde{\pi}_i^t, \tilde{\lambda}_{ij}^t)$ we get $d^t \sum_{(i,j)\in\mathcal{A}} c_{ij}\tilde{x}_{ij}^t = \tilde{\pi}_O^t - \tilde{\pi}_D^t - \sum_{(i,j)\in\mathcal{A}} \sum_{l=1}^{t} \bar{y}_{ij}^l \tilde{\lambda}_{ij}^t \Rightarrow d^{t+1} \sum_{(i,j)\in\mathcal{A}} c_{ij}\tilde{x}_{ij}^t = \kappa\tilde{\pi}_O^t - \kappa\tilde{\pi}_D^t - \kappa \sum_{(i,j)\in\mathcal{A}} \sum_{l=1}^{t} \bar{y}_{ij}^l \tilde{\lambda}_{ij}^t$, which proves that $(\kappa\tilde{\pi}_i^t, \kappa\tilde{\lambda}_{ij}^t)$ is optimal for period $t+1$. For case 2, there are new arcs that are activated in the network, but their corresponding reduced cost is zero, and therefore $\tilde{x}_{ij}^t$ is also optimal for $t+1$. The remainder of the proof proceeds as in case 1. □

Proposition 2 allows us to generate PO cuts from some subproblems without solving additional linear programs. This is true for all commodities in a particular period if no new arcs are activated. Case 2 implies that even when new arcs are activated, no linear programs need to be solved for commodities that do not benefit from new arc activation. Although theoretically these adjustments enable us to compute PO cuts faster, it is not clear that they lead to an overall more efficient Benders algorithm, because the cuts generated when solving additional LPs may differ. Therefore, we provide a thorough assessment of the relative performance of these enhancements in our computational experiments.

5.2.5. Strengthening the Master Problem We use two families of cutting planes that strengthen the formulation of the master problem and improve the convergence of the Benders algorithms. First, we employ origin-destination cuts, which are a subset of the cutset inequalities (Chouman et al. 2017) and impose that at least one arc from each origin and to each destination should be activated, respectively. To explain our second set of inequalities, let p^* denote the shortest path cost for a certain commodity-period pair and p_{ij}^* denote the shortest path cost when arc (i,j) is removed from the graph. Then, the cut $z \geq p_{ij}^* + (p^* - p_{ij}^*) \sum_{l=1}^{t} y_{ij}^l$ is a valid cut (Magnanti et al. 1986), as it expresses that the flow cost for a commodity is either higher than the shortest path cost when the complete graph is considered ($\sum_{l=1}^{t} y_{ij}^l = 1$) or higher than p_{ij}^* when arc (i,j) is not activated ($\sum_{l=1}^{t} y_{ij}^l = 0$). In our implementation, instead of adding $|\mathcal{T}||\mathcal{K}||\mathcal{A}|$ such

inequalities, we find for each commodity the arc whose removal increases the shortest path cost the most and add inequalities only for this arc.

6. Computational Experiments

This section reports on computational experiments that aim to illustrate the efficiency of the developed solution methods. All experiments were carried out using two threads on an Intel Xeon X5675 3.07GHz processor. All algorithms are coded in Python 2.7.9 using the Numpy library for numeric manipulations (van der Walt et al. 2011) and Gurobi v7.5.0 to solve the mixed-integer linear programs. When it comes to the inclusion of the strong inequalities (4) in the LP relaxation of (1)-(7), we implemented and tested four alternative strategies: (i) not including strong inequalities; (ii) adding all of them a priori in the model; (iii) adding them dynamically (via a callback) when they are violated, and (iv) adding them in a lazy cut pool and resorting to Gurobi's pool management mechanism. Strategy (iv) gave the best results overall, when considering CPU time and optimality gap quality, and therefore we adopt this strategy throughout our computational experiments. We set a time limit of 7,200 seconds and 24GB of RAM for all methods. The source code of our implementations can be found at the public Github repository https://github.com/IoannisFragkos/Multi-period-network-design.

6.1. Data Construction Methodology

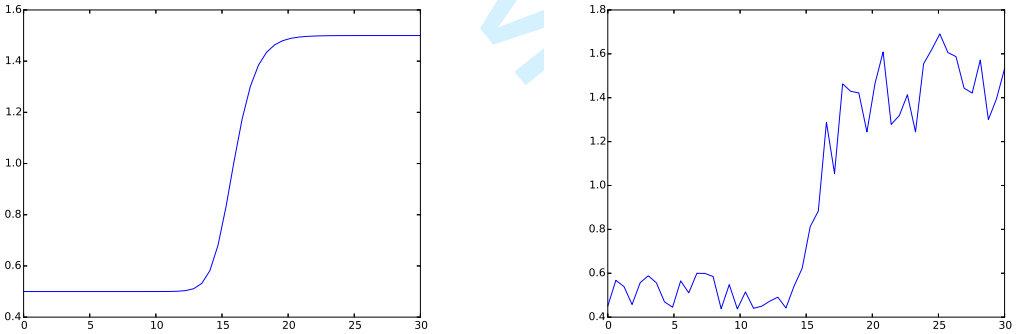
In order to test the performance of our algorithms, we utilize a large number of instances. In particular, we create multi-period extensions of a subset of the Canad R dataset, which has been used extensively by several authors such as Crainic et al. (2000, 2001), Crainic and Gendreau (2002), Ghamlouche et al. (2003), Crainic et al. (2004), Costa et al. (2009). It is also used in Katayama et al. (2009) and Yaghini et al. (2014), which, to the best of our knowledge, seem to be the best-performing algorithms at the time of this writing. More information about the single-period data generators can be found in Gendron and Crainic (1994, 1995).

We extend existing instances by introducing multiple periods in which the demand and fixed cost of each commodity vary. For commodity demands we utilize a generalized logistic

function, which signifies a simple model of demand growth (Richards 1959), with additional random perturbation. Note that although our methods are applicable to decreasing demand patterns, in such cases optimizing in the first period guarantees optimality for the entire horizon. We instead consider the case of network expansion, where the timing of arc activation becomes pertinent. Concretely, we use

$$d^{kt} = (1+r)\left(\lambda + \frac{\mu-\lambda}{(1+e^{-B(t-\frac{T}{2})})^{\frac{1}{\nu}}}\right) d^{k1}, \forall k \in \mathcal{K}, 1 < t \leq T,$$

where r denotes demand variability, λ and μ represent the minimum and maximum asymptotic demand respectively, B denotes the growth rate, which we set to 1, and ν is a locational parameter that determines near which asymptote the maximum growth occurs. We introduce two levels of variability, with r taking vales in $U(-0.1, 0.1)$ for small and in $U(-0.5, 0.5)$ for large variability, respectively. Figure 1 shows an example of a logistic growth function with no perturbation to the left, and one with a high level of perturbation to the right.



(a) Logistic growth function for $\lambda = 0.5, \mu = 1.5$. (b) Logistic growth with random perturbations.

Figure 1 Demand is modeled using logistic growth functions and varying levels of perturbations.

From each original single-period instance we generate multi-period instances by varying (i) the number of periods and (ii) the demand variability of each instance. In particular, we generate instances with 20, 40, 60 and 80 periods and with low and high demand variability. Under a full factorial design, 8 multi-period instances are generated

from each single-period instance, accounting for 408 instances in total. Table 1 gives an overview of their characteristics. The instances and detailed results can be downloaded from https://www.dropbox.com/sh/2f3pnubszqgwysb/AABggvjgmVU6v7XR9aIeN_7oa?dl=0.

It is worth noticing that the instances span across a wide variety of periods, nodes, arcs and commodities, resulting in a number of variables between 35,700 and 2,569,440 and a number of constraints between 10,720 and 185,520. Also important is that for capacitated instances the average optimality gap, i.e., the gap between the lower and upper bound, for the entire dataset is at 19%, between a minimum of 0% and a maximum of 100%, calculated by using the best known upper and lower bounds obtained by Gurobi for each instance. This comes as no surprise as even single-period capacitated problems are hard to solve to optimality, and most authors have designed heuristic approaches to tackle them. Uncapacitated instances on the other hand are more amenable to exact approaches, although the sheer scale of the large cases makes them challenging. Because of this important distinction, we report computational results for capacitated and uncapaciated instances in separate sections.

6.2. Capacitated Instances

For capacitated instances, we first assess the Lagrange relaxation lower bound quality, and then the quality of the upper bound obtained by our heuristics. Then, we proceed to further computational experiments, demonstrating the usefulness of our approach. In all variants of our Lagrange relaxation algorithm performed 1,000 subgradient iterations and it was initialized using the select-and-time heuristic.

6.2.1. Lower Bound Quality In theory, the Lagrange dual provides the same lower bound as the LP relaxation of problem (1)-(6) when the tight inequalities (4) are included. In practice, when LR is solved by subgradient optimization an approximate solution is obtained, which may or may not be close to the LP bound, depending on implementation-specific details (Fisher 1981). In order to assess the quality of the lower bound, we solved the LP relaxation of (1)-(6) with Gurobi, adding the strong inequalities (4) as lazy constraints. Since our intention is to obtain the lower bound using strong inequalities only, we deactivated all default Gurobi cuts and solved the problem at the root node only,

# Instances	$\|\mathcal{N}\|$	$\|\mathcal{A}\|$	$\|\mathcal{K}\|$	$\|T\|$	Bin Vars	Vars	Constraints
12	10	35	50	20	700	35,700	10,720
18	10	60	50	20	1,200	61,200	11,220
18	10	83	50	20	1,660	84,660	11,680
18	20	120	100	20	2,400	242,400	42,420
18	20	220	100	20	4,400	444,400	44,420
18	20	318	100	20	6,360	642,360	46,380
12	10	35	50	40	1,400	71,400	21,440
18	10	60	50	40	2,400	12,2400	22,440
18	10	83	50	40	3,320	169,320	23,360
18	20	120	100	40	4,800	484,800	84,840
18	20	220	100	40	8,800	888,800	88,840
18	20	318	100	40	12,720	1,284,720	92,760
12	10	35	50	60	2,100	107,100	32,160
18	10	60	50	60	3,600	183,600	33,660
18	10	83	50	60	4,980	253,980	35,040
18	20	120	100	60	7,200	727,200	127,260
18	20	220	100	60	13,200	1,333,200	133,260
18	20	318	100	60	19,080	1,927,080	139,140
12	10	35	50	80	2,800	142,800	42,880
18	10	60	50	80	4,800	244,800	44,880
18	10	83	50	80	6,640	338,640	46,720
18	20	120	100	80	9,600	969,600	169,680
18	20	220	100	80	17,600	1,777,600	177,680
18	20	318	100	80	25,440	2,569,440	185,520

Table 1 **Instance characteristics. Constraint count excludes the strong inequalities. The number of instances per category may vary, since some instances were infeasible.**

thereby allowing only strong inequalities to be generated. We then excluded 120 out of 408 instances in which Gurobi failed to solve the LP relaxation with the strong inequalities to

optimality within the time limit of 7,200 seconds, as in those instances it returned a lower bound weaker than that obtained by Lagrange relaxation. To this end, Table 2 reports the lower bound gap, defined as $(LB_{GRB} - LB_{LR})/LB_{GRB}$ and CPU time obtained by Lagrange relaxation for the instances that Gurobi could calculate an exact lower bound.

	# Instances		LR LB Gap	CPU Time (s)	
$\|\mathcal{N}\|$	All	GRB Solved	(%)	GRB	LR
10	192	192	0.85	439	145
20	216	96	1.11	3,135	593
$\|\mathcal{A}\|$					
35	48	48	0.89	73	95
60	72	72	0.68	436	141
83	72	72	1.00	688	183
120	72	47	1.83	3,320	406
220	72	29	0.51	2,962	598
318	72	20	0.25	2,949	816
$\|\mathcal{C}\|$					
50	192	192	0.85	439	145
100	216	96	1.11	3,135	593
$\|T\|$					
20	102	102	0.82	2102	181
40	102	102	1.02	2847	340
60	102	96	1.02	3114	462
80	102	84	0.92	3113	527
Total	408	384	0.94	2773	369

Notes. Column 4 is calculated as $(LB_{GRB} - LB_{LR})/LB_{GRB}$.

Table 2 Average Lagrange relaxation gaps and CPU Time for instances where the exact lower bound is found.

We first observe that, overall, the lower bound obtained by Lagrange relaxation deviates less than 1% from the exact lower bound, obtained by including all inequalities (4), as the 0.94% gap indicates. When broken down by each input size, no large variability is apparent, while the gaps appear to be relatively insensitive to problem size. With respect to CPU time performance, Lagrange relaxation is seven times faster than Gurobi overall, but the difference can be as large as 11 times.

The results of Table 2 demonstrate that Lagrange relaxation generates near-optimal lower bounds in a time-efficient and scalable fashion for instances whose LP relaxation was successfully solved to optimality by Gurobi. When it comes to the harder 120 instances for which the LP relaxation could not be solved to optimality, a distinction can be made between those in which Gurobi returned a zero lower bound (24 instances), and those in which Gurobi returned a lower bound weaker than that of Lagrange relaxation (96 instances). Table 3 reports the gap and CPU time for those latter instances, i.e., those that Gurobi did not solve to optimality but returned a positive lower bound.

Table 3 shows that Lagrange relaxation delivers significantly stronger lower bounds for these 96 instances in which Gurobi fails to solve the LP relaxation to optimality, with the average gap between the two bounds being 13.59%. In addition, Lagrange relaxation is 10 times faster for these instances, confirming its scalability.

To summarize the results of Tables 2 and 3, we notice that Lagrange relaxation delivers near-optimal lower bounds in cases where the optimal lower bounds can be verified, and significantly stronger lower bounds for instances where the LP relaxation cannot be solved to optimality within the time limit. In both cases, the CPU time consumed by Lagrange relaxation is significantly lower, suggesting that it exhibits better scalability than solving the LP relaxation. Having established that our approach pr oduces near-optimal lower bounds, we proceed to assess the quality of the upper bounds obtained by our heuristics.

6.2.2. Upper Bound Quality In order to assess the upper bound quality obtained by our heuristics, we use formulation (1)-(6), but this time we let Gurobi solve it to optimality by using its default cutting planes, also including the strong inequalities in the lazy cut pool. Gurobi was able to solve 163 out of 408 instances to proven optimality

	# Total GRB	GRB	CPU Time (s)	
$\|\mathcal{A}\|$	Unsolved (LB=0)	LB Gap (%)	GRB	LR
220	43 (11)	12.57	6,975	700
318	52 (13)	16.91	7,200	881
$\|T\|$				
20	13 (0)	6.7	7200	393
40	32 (0)	12.52	6975	608
60	35 (6)	16.26	7,201	823
80	40 (18)	15.67	7,201	970
Total	120 (24)	13.59	7,125	727

Notes. Column 3 is calculated as $(LB_{LR} - LB_{GRB})/LB_{LR}$, only for instances where GRB returned a positive lower bound. The number of instances with a zero lower bound is shown in parentheses. All instances have 20 nodes and 100 commodities.

Table 3 Average Lagrange relaxation gaps and CPU Time for instances where a non-zero, suboptimal lower bound is found by Gurobi.

within the time limit of 7,200 seconds, and we used the optimal solutions of these instances to assess the upper bounds obtained heuristically by three approaches: first, a myopic heuristic (MH) that solves each period independently, records which arcs are activated and uses them in the next periods and iterates; second, our select and time heuristic (S&T); and third, the hybrid bound obtained by S&T and Lagrange relaxation (S&T+LR). Table 4 reports how many instances were solved to optimality, the CPU times and upper bound gaps of each method, obtained as $(UB_m - OPT)/OPT$, where UB_m denotes the best upper bound found by each method $m \in \{MH, S\&T, S\&T + LR\}$.

The experiments show that the myopic heuristic is dominated by S&T, as it delivers solutions of worse quality and requires a higher amount of CPU time. On average, MH has an upper bound gap of 27.88%, and, while it requires a rather small amount of CPU time for small instances, it requires considerably high CPU times for instances with a

	# Instances		Gap (%)			CPU Time (s)			
$\|\mathcal{N}\|$	All	GRB Optimal	MH	S&T	S&T+LR	MH	S&T	S&T+LR	GRB
10	192	123	27.06	4.62	0.72	22	14	180	1,388
20	216	40	30.4	5.5	1.32	945	150	803	4,258
$\|\mathcal{A}\|$									
35	48	48	25.17	3.05	1.08	4	5	123	320
60	72	40	23.29	6.77	0.44	20	13	194	1,755
83	72	35	33.96	4.32	0.54	50	27	241	2,433
120	72	15	14.48	4.15	2.75	443	93	602	5,838
220	72	13	40.13	5.73	0.6	1,188	195	840	3,026
318	72	12	39.77	6.92	0.33	1,308	174	1,012	3,617
$\|\mathcal{C}\|$									
50	192	123	27.06	4.62	0.72	22	14	180	1,388
100	216	40	30.4	5.5	1.32	945	150	803	4,258
$\|T\|$									
20	102	51	20.26	2.23	0.43	198	30	177	1,113
40	102	42	26.32	4.44	0.49	300	54	329	2,267
60	102	37	32.86	6.32	1.05	267	56	409	2,522
80	102	33	36.07	7.71	1.82	240	57	492	2,901
Total	408	163	27.88	4.84	0.87	249	47	332	2,092

Notes. Gaps are calculated with respect to the optimal solution, as obtained by Gurobi.

Table 4 Average upper bound gaps and CPU Times for instances solved to optimality.

large amount of arcs, nodes or commodities. The S&T heuristic achieves gaps that are five times smaller, on average, while it needs a fifth of the time MH requires. Combined with Lagrange relaxation, S&T+LR achieves gaps lower than 1% on average, but this comes at the expense of a higher CPU time. Nevertheless, S&T+LR is about six times faster than solving the problem to optimality, while it can be as much as 10 times as fast. In summary, the S&T heuristic appears to strike a good balance between solution quality and CPU time, given that its goal is to initialize the Lagrange relaxation algorithm, while the

hybrid S&T+LR scheme delivers high quality upper bounds in a time-efficient manner. We next turn our attention to a larger set of instances, namely those where Gurobi was able to calculate both a lower and an upper bound, but no optimal solution.

6.2.3. Comparison with Branch-and-Cut Although 163 instances were solved to proven optimality, a non-trivial optimality gap, i.e., a gap less than 100%, was calculated by Gurobi for 357 instances. We focus on these instances to compare the efficiency of our algorithm with the branch-and-cut implementation of Gurobi. Correspondingly, Table 5 reports the obtained results, where the gaps are calculated for each method using the corresponding upper and lower bounds.

Overall, Lagrange relaxation achieves a lower gap than branch-and-cut in our dataset, requiring about eight times less CPU time. For instances with a small number of nodes, arcs or commodities Gurobi returns better average gaps, at the expense, however, of larger CPU times. For larger instances, our algorithm outperforms branch-and-cut both in terms of CPU time and optimality gaps.

In a final experiment, we attempt to tackle instances in which branch-and-cut failed to return an optimality gap. These are 51 instances, all of which have 20 nodes and 100 commodities. For these instances, we tune our heuristics so that they spend more time searching for high quality feasible solutions. Specifically, we (i) increase the time limit of the single-period MPNP solved in line 3 of Algorithm 1 from 100 seconds to 500 seconds, (ii) increase the time limit of the fixing heuristic from 1,000 seconds to 2,000 seconds and (iii) we only employ the incremental heuristics and the fixing heuristic once at the end of Lagrange relaxation instead of applying the fixing heuristic twice and the local branching heuristic once. Table 6 reports the results.

As evidenced by Table 6, these 51 instances are particularly hard to tackle, with our algorithm obtaining an average gap of less than 10%. Although it is beyond the scope of our current research, an exact branch-and-bound algorithm based on Lagrange relaxation can be developed to tackle those instances and further reduce their optimality gap. The relatively short amount of CPU time required by Lagrange relaxation, and the fact that branching on a single arc is unlikely to change significantly the neighborhood of the optimal Lagrange multipliers, make such an approach a promising direction for future research.

	# Instances		Optimality Gap (%)		CPU Time (s)	
$\|\mathcal{N}\|$	All	GRB Gap	GRB	LR	GRB	LR
10	192	192	1.05	3.30	3,477	236
20	216	165	15.03	7.60	6,487	996
$\|\mathcal{A}\|$						
35	48	48	0.00	2.08	320	123
60	72	72	1.76	3.56	4,175	241
83	72	72	1.03	3.85	4,883	307
120	72	56	6.33	6.02	6,836	679
220	72	55	15.64	8.11	6,214	1,064
318	72	54	23.43	8.71	6,404	1,255
$\|\mathcal{C}\|$						
50	192	192	1.05	3.3	3,477	236
100	216	165	15.03	7.6	6,487	996
$\|T\|$						
20	102	102	4.49	3.79	4,157	412
40	102	101	10.14	5.67	5,149	629
60	102	85	9.01	6.11	5164	664
80	102	69	6.27	5.93	5,145	690
Total	408	357	7.51	5.29	4,868	587

Notes. Optimality gap is calculated as $(UpperBound - LowerBound)/UpperBound$.

Table 5 Average optimality gap and CPU Time for instances where Gurobi reported a non-trivial optimality gap.

6.3. Uncapacited Instances

We investigate the efficiency of our Benders decomposition implementations for uncapacitated network design instances by utilizing the same set of multi-period instances but without considering their capacity. Specifically, we benchmark (i) a basic implementation (BND-R); (ii) Adding PO cuts (BND-MW1); (iii) Adding PO cuts solving only one sub-

	# Instances		Optimality Gap (%)		CPU Time (s)	
$\|\mathcal{A}\|$	All	GRB Gap	GRB	LR	GRB	LR
120	72	16	100	5.37	7,200	2,661
220	72	17	100	13.05	7,234	3,627
318	72	18	100	10.54	7,341	4,801
$\|T\|$						
40	102	1	100	4.74	7,200	572
60	102	17	100	9.9	7,245	3,878
80	102	33	100	9.83	7,271	3,762
Total	408	51	100	9.76	7,261	1,644

Table 6 Average optimality gap and CPU Time for instances where Gurobi failed to return an upper and/or lower bound.

problem, as in Magnanti et al. (1986) (BND-MW2) (iv) Same as in (iii) but skipping subproblems (BND-MW3) and finally the formulation of Fischetti et al. (2010) (BND-F). All implementations update the core point using Observation 3, and exploit modern callback technology to avoid solving the master problem multiple times (Bai and Rubin 2009) and make use of the cutting planes described in section 5.2.5. Similar to the capacitated experiments, our basic benchmark is the best Gurobi (GRB) formulation, which uses the cut pool to handle the separation of the strong inequalities (18). Tables 7 and 8 show the average optimality gaps and CPU times obtained by each method, respectively.

A careful analysis of Tables 7 and 8 suggests some important observations. First, Benders reformulations seem to be intrinsically more efficient compared to branch-and-cut (GRB). Specifically, our best implementation (BND-F), attains an average gap which is more than one order of magnitude lower, and consumes about 60% of GRB's CPU time. This performance difference has been observed for other problems as well, such as facility location (Fischetti et al. 2016), suggesting that a modern implementation of Benders decomposition can be a superior alternative to an off-the-shelf, state-of-the-art solver. Second, with respect to the basic Benders implementation, we note that although it achieves a far better gap than branch-and-cut, it falls behind the more sophisticated implementations by a large

		Optimality Gap (%)					
$\|\mathcal{N}\|$	#	GRB	BND-R	BND-MW1	BND-MW2	BND-MW3	BND-F
10	216	0	0	0	0	0	0
20	216	30.24	12.93	4.66	3.04	2.98	0.83
$\|\mathcal{A}\|$							
35	72	0	0	0	0	0	0
60	72	0	0	0	0	0	0
83	72	0	0	0	0	0	0
120	72	12.84	1.72	0.47	0.16	0.13	0
220	72	32.27	16.89	2.65	1.35	1.41	1.26
318	72	45.61	20.18	10.87	7.61	7.41	1.24
$\|\mathcal{C}\|$							
50	216	0	0	0	0	0	0
100	216	30.24	12.93	4.66	3.04	2.98	0.83
$\|T\|$							
20	108	3.18	1.92	0.59	0.19	0.22	0
40	108	12.84	3.54	1.2	0.66	0.91	0.29
60	108	22.3	7.83	1.6	0.86	0.86	0.48
80	108	22.16	12.57	5.94	4.38	3.99	0.9
Total	432	15.12	6.47	2.33	1.52	1.49	0.42

Notes. Optimality gaps are calculated as $(UB - LB)/UB$. The bottom row displays the total number of instances (column 2) and the overall average of each implementation.

Table 7 Comparison of optimality gaps obtained by each implementation.

margin, implying that the impact of adding PO cuts significantly enhances performance. In addition to the poor CPU time performance of basic Benders, it is worth noting that it ran out of memory in 36 instances. These 36 instances consumed an average of 1,787 seconds of CPU time before running out of memory and resulted in an average gap of 55%, which is still better compared to Gurobi's average gap for these instances, which is 85%. Third, when assessing the impact of PO cuts, we observe a non-trivial gap improvement between

		CPU Time (s)					
$\|\mathcal{N}\|$	#	GRB	BND-R	BND-MW1	BND-MW2	BND-MW3	BND-F
10	216	76	72	275	23	15	32
20	216	3,729	4,310	5,468	3,144	3,129	2,327
$\|\mathcal{A}\|$							
35	72	34	47	78	17	11	19
60	72	75	61	395	20	14	35
83	72	119	108	353	31	19	42
120	72	1,984	4,107	4,342	1,530	1,535	424
220	72	4,439	3,910	5,989	3,940	4,002	2,972
318	72	4,763	4,913	6,083	3,963	3,851	3,585
$\|\mathcal{C}\|$							
50	216	76	72	275	23	15	32
100	216	3,729	4,310	5,468	3,144	3,129	2,327
$\|T\|$							
20	108	1,113	2,131	2,232	956	941	630
40	108	1,902	2,473	2,905	1,599	1,592	1,074
60	108	2,208	2,251	3,367	1,971	1,998	1,336
80	108	2,387	1,909	2,953	1,807	1,756	1,678
Total	432	1,902	2,191	2,866	1,583	1,572	1,180

Notes. The bottom rows display the total number of instances (column 2) and the overall arithmetic and normalized shifted geometric average CPU Time of each implementation, respectively.

Table 8 Comparison of CPU Times obtained by each implementation.

the basic Benders BND-R and the basic PO Benders BND-MW1 implementations. Further, BND-MW1 ran out of memory in 12 instances instead of 36, but these improvements come at a high CPU time cost, because two LPs are needed in order to generate every cut, making BND-MW1 having the worst time performance across all methods. Fourth, implementation BND-MW2, which generates PO cuts using a single subproblem per itera-

tion dominates BND-MW1 in terms of both gap and CPU time performance, across every single averaged configuration, while it ran out of memory for six instances. This confirms the importance of an efficient generation of PO cuts, which is further enhanced in implementation BND-MW3, although the differences with BND-MW2 are small. This may seem counter-intuitive at first, but a closer look at the construction process of the added cuts shows that their high similarity may undermine their efficiency. To see this, suppose that a PO cut $\bar{\pi}_O^{kt} - \bar{\pi}_D^{kt} - \sum_{(i,j)\in\mathcal{A}} \sum_{l=1}^{t} \bar{\lambda}_{ij}^{kt} y_{ij}^{l} \leq z^{kt}$, generated from subproblem (k,t), is added to the master problem (23)-(27). If we add $\kappa \left(\bar{\pi}_O^{kt} - \bar{\pi}_D^{kt} - \sum_{(i,j)\in\mathcal{A}} \sum_{l=1}^{t+1} \bar{\lambda}_{ij}^{kt} y_{ij}^{l} \right) \leq z^{kt+1}$ from subproblem $(k,t+1)$, then that latter cut is very similar to the former one, in the sense that both cuts are parallel in the (y_{ij}^{l}) coordinates, for all $(i,j) \in \mathcal{A}, l \in \{1,\ldots,t\}$, which is a relatively large part of the master problem feasible space. Thus, although these cuts do improve the optimality gap, the improvement they offer is marginal. Fifth, BND-F is the best-performing implementation, having both the lowest gap and CPU time, while no instances ran out of memory. In particular, it was able to close the gap for 54 instances that GRB failed to solve to optimality, and for 39 instances that no algorithm solved optimally, nine of which have 2.5 million variables. The ability of this model to incorporate the current flow cost in the cut-generating subproblem and to generate cuts that unify feasibility and optimality leads to important improvements over the Magnanti-Wong implementations. Finally, the breakdown per instance attribute shows that problems with 10 nodes and 50 commodities can all be solved to optimality rather easily by all algorithms, while problems with 20 nodes are 100 commodities are far more challenging. Problems with more arcs or periods are also more challenging to solve, while the toughest instances tend to be the largest ones, 18 of which have 2.5 million variables.

On further analysis, it is interesting to investigate the strength of cuts each Benders implementation requires. To this end, Table 9 shows the average number of cuts each algorithm generated for instances solved to optimality. We should acknowledge that, despite that all algorithms use the same master problem, solver and overall settings, the number of cuts generated does not depend only on the efficacy of each scheme, but on a series of other factors, such as the branching paths, node heuristics, LP relaxation bases and node cuts, that may differ across implementations. However, there is no evidence to support

that these factors introduce a systematic bias in favor of certain implementations, and therefore the average number of Benders cuts generated over a large number of instances may lead to some notable insights. First, basic Benders, BND-R is outperformed by all other algorithms since it generates the largest average number of cuts. Interestingly, the implementation that generates the fewest Benders cuts is BND-MW1, which solves two LPs to generate a single PO cut. MW2 generates slightly more cuts, while BND-MW2 generates significantly more cuts, confirming our earlier observation that some of the generated cuts may not be effective due to their near-parallel hyperplane orientation. This experiment underlines that there may be a considerable spectrum when it comes to the performance variability of PO cuts. In particular, generating the cuts by solving LPs could result in strong cuts, as there exists some variability embedded in the pivoting rules that return one among multiple optimal solutions. An interesting direction for future research is to devise a mechanism that exploits this variability in a systematic way.

7. Conclusions and future research

We introduce a multi-period extension of the traditional multi-commodity network design problem and study its capacitated and uncapacitated variants. By utilizing decomposition algorithms that take advantage of the problem structure, we are able to develop two heuristic and decomposition based procedures that are superior to existing approaches. For capacitated problems, it is notable that Lagrange relaxation scales very well with problem size, and delivers good quality lower and upper bounds across a large range of problem sizes, even for problems that are intractable by state-of-the-art solvers. For uncapacitated problems, we employ a variety of Benders decomposition schemes and show that the best one is able to solve to optimality problems with as many as 2.5 million variables.

There are many avenues on which future research can be conducted. For capacitated problems, the development of a branch-and-bound scheme based on the volume algorithm appears to be an approach worth exploring. Investigating the efficiency of well-established heuristics, such as adaptive large neighborhood search (Ropke and Pisinger 2006), cycle-based evolutionary algorithms (Paraskevopoulos et al. 2016), tabu search (Glover 1990), slope scaling (Crainic et al. 2004) or column and row generation techniques (Katayama

		# Optimality + Feasibility Cuts				
$\|\mathcal{N}\|$	#	BND-R	BND-MW1	BND-MW2	BND-MW3	BND-F
10	216	5,089	3,141	3,936	4,555	2,683
20	180	29,458	22,770	24,388	29,040	23,524
$\|\mathcal{A}\|$						
35	72	4,634	3,468	3,633	4,185	2,491
60	72	4,522	2,473	3,914	4,650	2,136
83	72	6,112	3,483	4,262	4,830	3,422
120	72	27,483	19,293	19,956	26,239	16,544
220	54	29,486	22,495	25,964	27,255	24,367
318	54	32,062	27,681	28,723	34,559	31,989
$\|\mathcal{C}\|$						
50	216	5,089	3,141	3,936	4,555	2,683
100	180	29,458	22,770	24,388	29,040	23,524
$\|T\|$						
20	108	8,092	7,416	8,031	8,012	9,079
40	108	18,076	14,216	15,576	18,923	15,180
60	96	19,999	14,499	15,961	19,561	13,339
80	84	19,710	12,488	13,791	16,955	10,873
Total	396	16,166	12,063	13,233	15,685	12,156

Notes. Column 2 shows the number of instances that all algorithms solved to optimality.

Table 9 Average number of cuts for problems solved to optimality.

et al. 2009) can lead to further improved solutions. For uncapacitated problems, further improvements on the Benders cut generation mechanism can be investigated. Specifically, it is interesting to study how different core point selection strategies impact the solution time and the efficacy of generated cuts. In addition, problem extensions with demand uncertainty are interesting to study both from a theoretical and from a practical perspective. For such problems, Benders decomposition can still be employed but one may have

to tackle extra complications, such as the existence of non-anticipativity constraints. We hope that our current study inspires future work on such problems.

Acknowledgments

This research was partly funded by the Fonds de recherche du Quèbec - Nature et technologies. This support is gratefully acknowledged.

Appendix A: Proof and illustration of Proposition 1

A.1. Proof

We prove hereby that Problem [SUB] has the integrality property.

Proof. We will show that there exists no point with at least one fractional y^t that is an extreme point of [SUB]. To this end, note that the origin is a feasible point of [SUB]. Then, let a feasible point p be such that there exist s periods $\{t_1,\dots,t_s\}$ with $(y^{t_j})_p \in (0,1)$ and $(y^t)_p = 0$ for $t \notin \{t_1,\dots,t_s\}$. We construct feasible points $q_j, j=1,\dots,s$ such that the original point p can be written as their convex combination, using as weights the components of $(y^t)_p$, i.e., $(y^t)_p = \sum_{j=1}^{s}(y^{t_j})_p (y^t)_{q_j}$; $(x^{kt})_p = \sum_{j=1}^{s}(y^{t_j})_p (x^{kt})_{q_j}$ (1). To this end, for each $j \in \{1,\dots,s\}$ we set $(y^{t_j})_{q_j} = 1; (y^t)_{q_j} = 0$, for all $t \neq t_j$ and $(x^{kt})_{q_j} = \min\{1, A_j^{kt}\}$, if $t \geq t_j$; 0, otherwise, for all $k \in \mathcal{K}, t \in T$, where $A_1^{kt} = (x^{kt})_p/(y^{t_1})_p$ and $A_j^{kt} = \left[(x^{kt})_p - \sum_{l=1}^{j-1}(y^{t_l})_p (x^{kt})_{q_l}\right]/(y^{t_j})_p$ for all $j \in \{2,\dots,s\}$. Note that for each t and k there exists at least one j such that $A_j^{kt} \leq 1$, otherwise $(x^{kt})_{q_j} = 1$ for all j, and $A_s^{kt} > 1$ implies that $(x^{kt})_p > \sum_{j\in\{1,\dots s\}}(y^{t_j})_p$ which is not true because p is feasible. Thus, let j^* denote the first j such that $A_j^{kt} \leq 1$. We evaluate each $(x^{kt})_{q_j}$ for $j < j^*$, $j = j^*$ and $j > j^*$ and substitute them back in (1).
Case (i): $j < j^*$. By the definition of j^* and $(x^{kt})_{q_j}$ it holds that $(x^{kt})_{q_j} = 1$.
Case (ii): $j = j^*$. Likewise, the definitions of j^* and $(x^{kt})_{q_j}$ give $(x^{kt})_{q_{j^*}} = A_{j^*}^{kt}$ (2).
Case (iii): $j > j^*$. We first note that the quantities A_j^{kt} can be written recursively as $A_{j+1}^{kt} = \frac{(y^{t_j})_p}{(y^{t_{j+1}})_p}[A_j^{kt} - (x^{kt})_{q_j}]$ (3). This recursion written for $j = j^*$ implies that $A_{j^*+1}^{kt} = 0$ (because of (2)), which in turn gives $x_{q_{j^*+1}}^{kt} = 0$. Then, using (3) it is easy to show by induction that $A_j^{kt} = (x^{kt})_{q_j} = 0$ for all $j > j^*$.

Substitution of the above in (1) gives $\sum_{j=1}^{s}(y^{t_j})_p (x^{kt})_{q_j} = \sum_{j=1}^{j^*-1}(y^{t_j})_p + \underbrace{(y^{t_{j^*}})_p \left[(x^{kt})_p - \sum_{j=1}^{j^*-1}(y^{t_j})_p\right]/(y^{t_{j^*}})_p}_{A_{j^*}^{kt}} + \underbrace{0+\cdots+0}_{\text{Terms } \{j^*+1,\dots,s\}} = (x^{kt})_p$. This final relationship, alongside the fact that $\sum_j (y^{t_j})_p \leq 1$ and that the origin is feasible signifies that p can be written as a convex combination of q_i feasible points and the origin, which completes the proof. □

We next provide a small example that shows how points q_j are constructed.

A.2. Illustration: construction of points q_j

We illustrate hereby the construction of points q_j for a small example with one commodity and seven periods, in Table 10. There, it can be readily verified that $(y^t)_p = \sum_{j=1}^{3}(y)_{q^l}(y^t)_{q_l}$ and $(x^t)_p = \sum_{j=1}^{3}(y)_{q^l}(x^t)_{q^l}$, for all $t = 1, \ldots, 7$.

Point	Weight	Component	t=1	t=2	t=3	t=4	t=5	t=6	t=7
p	–	y		0.2		0.1			0.6
		x		0.2	0.1	0.25	0.25	0.05	0.8
q_1	$(y)_{q_1} = 0.2$	y		1					
		x		1	0.5	1	1	0.25	1
		A		1	0.5	1.25	1.25	0.25	4
q_2	$(y)_{q_2} = 0.1$	y				1			
		x				0.5	0.5		1
		A				0.5	0.5		6
q_3	$(y)_{q_3} = 0.6$	y							1
		x							0.83
		A							0.83

Table 10 Illustration of point construction. Empty spaces imply zero coordinates.

References

Adulyasak, Y., J.-F. Cordeau, R. Jans. 2015. Benders decomposition for production routing under demand uncertainty. *Operations Research* **63**(4) 851–867.

Ahuja, R., Ö. Ergun, J.B. Orlin, A.P. Punnen. 2002. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics* **123**(1) 75–102.

Alvelos, F., J.M. de Carvalho. 2007. An extended model and a column generation algorithm for the planar multicommodity flow problem. *Networks* **50**(1) 3–16.

Arabani, A.B., R.Z. Farahani. 2012. Facility location dynamics: An overview of classifications and applications. *Computers & Industrial Engineering* **62**(1) 408–420.

Atamtürk, A. 2002. On capacitated network design cut–set polyhedra. *Mathematical Programming* **92**(3) 425–437.

Atamtürk, A., D. Rajan. 2002. On splittable and unsplittable flow capacitated network design arc–set polyhedra. *Mathematical Programming* **92**(2) 315–333.

Bai, L., P.A. Rubin. 2009. Combinatorial Benders cuts for the minimum tollbooth problem. *Operations Research* **57**(6) 1510–1522.

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

Balakrishnan, A., T.L. Magnanti, R.T. Wong. 1989. A dual-ascent procedure for large-scale uncapacitated network design. *Operations Research* **37**(5) 716–740.

Barahona, F. 1996. Network design using cut inequalities. *SIAM Journal on Optimization* **6**(3) 823–837.

Barahona, F., R. Anbil. 2000. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming* **87**(3) 385–399.

Barahona, F., R. Anbil. 2002. On some difficult linear programs coming from set partitioning. *Discrete Applied Mathematics* **118**(1) 3–11.

Bärmann, A., A. Martin, H. Schülldorf. 2017. A decomposition method for multiperiod railway network expansion—with a case study for Germany. *Transportation Science* doi:10.1287/trsc.2017.0747.

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

Beraldi, P., G. Ghiani, A. Grieco, E. Guerriero. 2008. Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs. *Computers & Operations Research* **35**(11) 3644–3656.

Bienstock, D., O. Günlük. 1996. Capacitated network design-polyhedral structure and computation. *INFORMS Journal on Computing* **8**(3) 243–259.

Blanco, V., J. Puerto, A.B. Ramos. 2011. Expanding the Spanish high-speed railway network. *Omega* **39**(2) 138–150.

Boyd, S., L. Xiao, A. Mutapcic. 2003. Subgradient methods. *Lecture notes of EE392o, Stanford University, Autumn Quarter* **2004** 2004–2005.

Chand, S., V.N. Hsu, S. Sethi. 2002. Forecast, solution, and rolling horizons in operations management problems: a classified bibliography. *Manufacturing & Service Operations Management* **4**(1) 25–43.

Chardaire, P., A. Sutter, M.-C. Costa. 1996. Solving the dynamic facility location problem. *Networks* **28**(2) 117–124.

Chouman, M., T.G. Crainic, B. Gendron. 2009. A cutting-plane algorithm for multicommodity capacitated fixed-charge network design. CIRRELT Technical Report 2009-20, Université de Montréal, Canada.

Chouman, M., T.G. Crainic, B. Gendron. 2017. Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design. *Transportation Science* **51**(2) 650–667.

Cordeau, J.-F., F. Soumis, J. Desrosiers. 2001. Simultaneous assignment of locomotives and cars to passenger trains. *Operations Research* **49**(4) 531–548.

Costa, A.M., J.-F. Cordeau, B. Gendron. 2009. Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications* **42**(3) 371–392.

Crainic, T.G., A. Frangioni, B. Gendron. 2001. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics* **112**(1) 73–99.

Crainic, T.G., M. Gendreau. 2002. Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics* **8**(6) 601–627.

Crainic, T.G., M. Gendreau, J.M. Farvolden. 2000. A simplex-based tabu search method for capacitated network design. *INFORMS Journal on Computing* **12**(3) 223–236.

Crainic, T.G., B. Gendron, G. Hernu. 2004. A slope scaling/Lagrangean perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design. *Journal of Heuristics* **10**(5) 525–545.

Crainic, T.G., Y. Liand M. Toulouse. 2006. A first multilevel cooperative algorithm for capacitated multicommodity network design. *Computers & Operations Research* **33**(9) 2602–2622.

Cruz, F.R.B., J.M. Smith, G.R. Mateus. 1998. Solving to optimality the uncapacitated fixed-charge network flow problem. *Computers & Operations Research* **25**(1) 67–81.

Danna, E., E. Rothberg, C. Le Pape. 2005. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming* **102**(1) 71–90.

de Araujo, S., B. De Reyck, Z. Degraeve, I. Fragkos, R. Jans. 2015. Period decompositions for the capacitated lot sizing problem with setup times. *INFORMS Journal on Computing* **27**(3) 431–448.

Fischetti, M., I. Ljubić, M. Sinnl. 2016. Redesigning Benders decomposition for large-scale facility location. *Management Science* **63**(7) 2146–2162.

Fischetti, M., A. Lodi. 2003. Local branching. *Mathematical Programming* **98**(1-3) 23–47.

Fischetti, M., D. Salvagnin, A. Zanette. 2010. A note on the selection of Benders' cuts. *Mathematical Programming* **124**(1-2) 175–182.

Fisher, M.L. 1981. The Lagrangian relaxation method for solving integer programming problems. *Management Science* **27**(1) 1–18.

Frangioni, A., B. Gendron. 2009. 0–1 reformulations of the multicommodity capacitated network design problem. *Discrete Applied Mathematics* **157**(6) 1229–1241.

Frangioni, A., B. Gendron. 2013. A stabilized structured Dantzig–Wolfe decomposition method. *Mathematical Programming* **140**(1) 45–76.

Frangioni, A., E. Gorgone. 2014. Bundle methods for sum-functions with "easy" components: applications to multicommodity network design. *Mathematical Programming* **145**(1-2) 133–161.

Gendron, B., T.G. Crainic. 1994. Relaxations for multicommodity capacitated network design problems. CRT Technical Report 965, Université de Montréal, Canada.

Gendron, B., T.G. Crainic. 1995. Bounding procedures for multicommodity capacitated network design problems. CRT Technical Report 95-12, Université de Montréal, Canada.

Geoffrion, A.M. 1974. Lagrangean relaxation for integer programming. *Approaches to Integer Programming*. Springer, 82–114.

Ghamlouche, I., T.G. Crainic, M. Gendreau. 2003. Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. *Operations Research* **51**(4) 655–667.

Ghamlouche, I., T.G. Crainic, M. Gendreau. 2004. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. *Annals of Operations Research* **131**(1-4) 109–133.

Glover, F. 1990. Tabu search: A tutorial. *Interfaces* **20**(4) 74–94.

Günlük, O. 1999. A branch-and-cut algorithm for capacitated network design problems. *Mathematical Programming* **86**(1) 17–39.

Harkness, J., C. ReVelle. 2003. Facility location with increasing production costs. *European Journal of Operational Research* **145**(1) 1–13.

Hewitt, M., G.L. Nemhauser, M.W.P. Savelsbergh. 2010. Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS Journal on Computing* **22**(2) 314–325.

Holmberg, K., D. Yuan. 1998. A Lagrangean approach to network design problems. *International Transactions in Operational Research* **5**(6) 529–539.

Holmberg, K., Di. Yuan. 2000. A Lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research* **48**(3) 461–481.

Hooghiemstra, J.S., L.G. Kroon, M.A. Odijk, M. Salomon, P.J. Zwaneveld. 1999. Decision support systems support the search for win-win solutions in railway network design. *Interfaces* **29**(2) 15–32.

Jena, S.D., J.-F. Cordeau, B. Gendron. 2015. Dynamic facility location with generalized modular capacities. *Transportation Science* **49**(3) 484–499.

Jena, S.D., J.-F. Cordeau, B. Gendron. 2017. Lagrangian heuristics for large-scale dynamic facility location with generalized modular capacities. *INFORMS Journal on Computing* **29**(2) 388–404.

Katayama, N., M. Chen, M. Kubo. 2009. A capacity scaling heuristic for the multicommodity capacitated network design problem. *Journal of Computational and Applied Mathematics* **232**(1) 90–101.

Kennington, J.L., C.D. Nicholson. 2010. The uncapacitated time-space fixed-charge network flow problem: an empirical investigation of procedures for arc capacity assignment. *INFORMS Journal on Computing* **22**(2) 326–337.

Kim, D., P.M. Pardalos. 1999. A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. *Operations Research Letters* **24**(4) 195–203.

Lai, Y.C., M.C. Shih. 2013. A stochastic multi-period investment selection model to optimize strategic railway capacity planning. *Journal of Advanced Transportation* **47**(3) 281–296.

Lee, D.H., M. Dong. 2008. A heuristic approach to logistics network design for end-of-lease computer products recovery. *Transportation Research Part E: Logistics and Transportation Review* **44**(3) 455–474.

Magnanti, Thomas L, Paul Mireault, Richard T Wong. 1986. Tailoring Benders decomposition for uncapacitated network design. *Netflow at Pisa* 112–154.

Magnanti, T.L., R.T. Wong. 1981. Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research* **29**(3) 464–484.

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

Marin, A., P. Jaramillo. 2008. Urban rapid transit network capacity expansion. *European Journal of Operational Research* **191**(1) 45–60.

Minoux, M. 1989. Networks synthesis and optimum network design problems: Models, solution methods and applications. *Networks* **19**(3) 313–360.

Papadakos, N. 2008. Practical enhancements to the Magnanti–Wong method. *Operations Research Letters* **36**(4) 444–449.

Papadimitriou, D., B. Fortz. 2015. A rolling horizon heuristic for the multiperiod network design and routing problem. *Networks* **66**(4) 364–379.

Paraskevopoulos, D.C., T. Bektaş, T.G. Crainic, C.N. Potts. 2016. A cycle-based evolutionary algorithm for the fixed-charge capacitated multi-commodity network design problem. *European Journal of Operational Research* **253**(2) 265–279.

Petersen, E.R., A.J. Taylor. 2001. An investment planning model for a new north-central railway in Brazil. *Transportation Research Part A: Policy and Practice* **35**(9) 847–862.

Pochet, Y., M. Van Vyve. 2004. A general heuristic for production planning problems. *INFORMS Journal on Computing* **16**(3) 316–327.

Raack, C., A.M.C.A Koster, S. Orlowski, R. Wessäly. 2011. On cut-based inequalities for capacitated network design polyhedra. *Networks* **57**(2) 141–156.

Rahmaniani, R., T.G. Crainic, M. Gendreau, W. Rei. 2016. The Benders decomposition algorithm: A literature review. *European Journal of Operational Research* **259**(3) 801–817.

Randazzo, C.D., H.P.L. Luna. 2001. A comparison of optimal methods for local access uncapacitated network design. *Annals of Operations Research* **106**(1-4) 263–286.

Richards, F.J. 1959. A flexible growth function for empirical use. *Journal of Experimental Botany* **10**(2) 290–301.

Rodríguez-Martín, I., J.J. Salazar-González. 2010. A local branching heuristic for the capacitated fixed-charge network design problem. *Computers & Operations Research* **37**(3) 575–581.

Ropke, S., D. Pisinger. 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science* **40**(4) 455–472.

Schrijver, A. 1998. *Theory of Linear and Integer Programming*. John Wiley & Sons, New York, NY.

Shulman, A. 1991. An algorithm for solving dynamic capacitated plant location problems with discrete expansion sizes. *Operations Research* **39**(3) 423–436.

van der Walt, S., S.C. Colbert, G. Varoquaux. 2011. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering* **13**(2) 22–30.

Yaghini, M., M. Karimi, M. Rahbar, M.H. Sharifitabar. 2014. A cutting-plane neighborhood structure for fixed-charge capacitated multicommodity network design problem. *INFORMS Journal on Computing* **27**(1) 48–58.

Yang, H., H. Bell, G. Michael. 1998. Models and algorithms for road network design: a review and some new developments. *Transport Reviews* **18**(3) 257–278.

Zhu, E., T.G. Crainic, M. Gendreau. 2014. Scheduled service network design for freight rail transportation. *Operations Research* **62**(2) 383–400.