Exact Algorithms for Multicommmodity Uncapacitated Fixed-charge Network Design

Carlos Armando Zetina[a], Ivan Contreras[a], Jean-François Cordeau[b]

[a] *Concordia University and Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montréal, Canada H3G 1M8*
[b] *HEC Montréal and Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montréal, Canada H3T 2A7*

Abstract

This paper presents two exact algorithms based on Benders decomposition for solving the multicommodity uncapacitated fixed-charge network design problem. The first is a branch-and-cut algorithm based on a Benders reformulation enhanced with an in-tree matheuristic to obtain improved feasible solutions and valid inequalities in the projected master problem space to close the linear programming gap. In addition, implementation details crucial to the algorithm's efficiency such as cut and core point selection criteria are addressed. The second exact algorithm exploits the problem's structure to combine a cut-and-solve strategy with Benders decomposition. Extensive computational experiments show both exact algorithms provide a speedup of between one and two orders of magnitude compared to a state-of-the-art general-purpose MIP solver.

Keywords: Benders decomposition, lift-and-project, local branching, matheuristics

1. Introduction

Network design problems (NDPs) lie at the heart of designing and operating efficient systems in several sectors such as personnel scheduling (Bartholdi et al. 1980, Balakrishnan and Wong 1990), service network design (Crainic 2000, Andersen et al. 2009, Crainic and Rousseau 1986), logistics network design (Geoffrion and Graves 1974, Santoso et al. 2005, Cordeau et al. 2006), and transportation (Magnanti and Wong 1984, Minoux 1989). They are able to capture the system-wide interactions between strategic and operational decisions, namely arc activation and routing, to ensure cost-effective paths among a selected set of nodes. NDPs can be classified into *single* and *multicommodity* variants depending on the characteristics of the demand. In single-commodity problems, the demand at each node can be satisfied by any of the other nodes' supply since they all route the same commodity. In multicommodity problems, demand is expressed as origin-destination (OD)

Email addresses: c_zetina@encs.concordia.ca (Carlos Armando Zetina), icontrer@encs.concordia.ca (Ivan Contreras), jean-francois.cordeau@hec.ca (Jean-François Cordeau)

pairs where a destination node's demand must be satisfied by a corresponding origin node's supply.

In this paper we focus on a fundamental NDP: the *multicommodity uncapacitated fixed-charge network design problem* (MUFND). The problem is defined on a directed graph and considers a set of commodities modeled by OD pairs, each with an origin node, a destination node, and a demand quantity. The objective is to install a subset of arcs to route all commodities from their origins to their destinations at minimal cost. The MUFND is $\mathcal{NP}$-hard (Johnson et al. 1978) and generalizes a large class of well-known problems such as the *traveling salesman problem*, the *uncapacitated lot-sizing problem*, and the *Steiner network design problem* (Ortega and Wolsey 2003).

At the same time, the MUFND is generalized by the multicommodity capacitated fixed-charge network design problem. The latter has been extensively studied from different research directions. Some authors have approached this problem from a polyhedral perspective, proposing new families of valid inequalities that strengthen well-known mixed integer formulations (Bienstock and Günlük 1996, Günlük 1999, Atamtürk 2002, Atamtürk and Rajan 2002, Raack et al. 2011). Recently, Chouman et al. (2017) provided insight on the computational impact that commodity representations have on the efficiency of five families of these valid inequalities. Others have focused on the development of decomposition methods (Cruz et al. 1998, Randazzo and Luna 2001, Crainic et al. 2001, Frangioni and Gendron 2009, 2013, Frangioni and Gorgone 2014) that exploit the problem structure to decompose the model into smaller subproblems. Despite the significant contributions presented in these papers, solving the capacitated network design problem to proven optimality in reasonable time still remains an open problem.

Another line of research for solving the capacitated variant is the use of heuristic algorithms to obtain high quality solutions. Among these are the slope scaling heuristics (Kim and Pardalos 1999, 2000, Crainic et al. 2004, Katayama et al. 2009); cycle-based and other neighbourhood searches (Ghamlouche et al. 2003, 2004, Yaghini et al. 2015, Paraskevopoulos et al. 2016); and matheuristics that exploit available mathematical programming software (Hewitt et al. 2010, Rodríguez-Martín and Salazar-González 2010, Munguía et al. 2017, Gendron et al. 2018).

In the case of the MUFND, the first proposed solution algorithm is an add-drop heuristic by Billheimer and Gray (1973). Other heuristics are those of Dionne and Florian (1979), Boffey and Hinxman (1979), Los and Lardinois (1982), and Kratica et al. (2002). Lamar et al. (1990) proposed a novel form of iteratively obtaining strengthened dual bounds from a weaker formulation by adjusting artificial capacity constraints. Balakrishnan et al. (1989) presented a dual ascent algorithm and a primal heuristic to obtain solutions for large-scale instances with up to 600 arcs and 1,560 commodities. Their method obtains solutions that are between 1% and 4% away from optimality in less than 150 seconds of computing time. With respect to exact methods, Magnanti et al. (1986) developed a tailored Benders decomposition for the variant with undirected design decisions of the MUFND. They were able to solve instances with up to 130 arcs and 58 commodities to proven optimality. Holmberg and Hellstrand (1998) used a Lagrangean branch-and-bound algorithm to solve directed instances with up to 1,000 arcs and 600

commodities. Recently, Fragkos et al. (2017) used Benders decomposition to solve a multi-period extension of the MUFND. They experimented with the use of Pareto-optimal cuts and with the unified cut approach of Fischetti et al. (2010), obtaining significant computational gains with the latter on instances with up to 318 arcs, 100 commodities and 108 time periods. To the best of our knowledge, these are the current state-of-the-art exact methods for the undirected and directed variants of the MUFND.

Benders decomposition has been an effective tool for solving several classes of network design problems with various applications (Costa 2005). Some fields in which it has recently been applied are closed loop supply chains (Jeihoonian et al. 2016), hazardous material transportation (Fontaine and Minner 2018), and health services (Zarrinpoor et al. 2018). It has also recently been proven effective in solving fundamental network design problems such as the optimum communication spanning tree problem (Zetina et al. 2018) and extensions such as the multi-layer (Fortz and Poss 2009), hop-constrained (Botton et al. 2013), and multi-period (Fragkos et al. 2017) network design problems. In the last few years, it has also been applied to NDPs with parameter uncertainty as in Lee et al. (2013), Keyvanshokooh et al. (2016) and Rahmaniani et al. (2018). Other applications of Benders decomposition to fixed-charge NDPs can be found in Costa (2005) while Rahmaniani et al. (2017) provides a survey on the algorithm and its use in optimization problems.

This paper revisits the use of Benders decomposition proposed by Magnanti et al. (1986) to solve the undirected design variant of the MUFND. As in Fischetti et al. (2017), our purpose is to redesign this once discarded approach for solving the MUFND to exploit the state-of-the-art of algorithmic and computational resources. The resulting Benders algorithms use branch-and-cut (Padberg and Rinaldi 1991), local branching (Fischetti and Lodi 2003), and cut-and-solve (Climer and Zhang 2006) procedures implemented within the *cut callback* framework available in today's general purpose mixed integer programming solvers. We present, in detail, the nuances of adopting these tools and propose novel refinements to reduce the computation time required to solve the MUFND with directed arc design decisions.

We present two exact algorithms based on the Benders reformulation of a well-known mixed integer programming model of the directed MUFND. Both algorithms solve the linear relaxation of the Benders reformulation with a cutting-plane procedure to obtain Pareto-optimal cuts and cutset inequalities at each node of the enumeration tree. To accelerate the algorithms' convergence we introduce new valid inequalities referred to as *Benders lift-and-project cuts* to improve the linear programming relaxation and an in-tree matheuristic that finds better feasible solutions by using path information generated while exploring the branch-and-bound tree.

The first algorithm, referred to as a branch-and-Benders-cut algorithm, solves the Benders reformulation in one enumeration tree. We address critical implementation details that should be considered when separating cuts at fractional and integer points such as core point selection for Pareto optimal cuts and propose a tailored core point selection criterion that provides a significant speed-up for solving the MUFND.

The second method is based on the combination of a modified cut-and-solve scheme

(Climer and Zhang 2006) and our branch-and-Benders-cut algorithm. The method iteratively restricts the potential design arcs to solve smaller problems that produce a sequence of feasible solutions with non-increasing objective function value. The algorithm also allows for the recycling of Benders cuts generated in previous iterations thereby saving computational effort.

We report computational experience on several sets of benchmark instances to assess the performance of our algorithms. The proposed exact methods are up to three orders of magnitude faster than the state-of-the-art MIP solver CPLEX 12.7.1 and solve instances of larger size than those previously presented. This computational contribution is accompanied by methodological insights such as the simultaneous use of a path and arc-based formulation for the MUFND, the introduction of *Benders lift-and-project cuts* to reduce the linear programming gap, and the hybridization of two well-known mixed integer programming tools.

The remainder of the paper is organized as follows. Section 2 provides a formal definition of the MUFND and presents the arc-based formulation. Section 3 describes the Benders reformulation for the MUFND while Section 4 details the enhancements implemented in our branch-and-cut algorithm. In Section 5, we present our second algorithmic framework, a hybrid cut-and-solve Benders algorithm. Summarized results of our computational experiments are given in Section 6, while Section 7 presents concluding remarks and future lines of research.

2. Problem definition

The MUFND is defined on a directed graph $G = (N, A)$ where N is a set of nodes, A is a set of arcs and K is a set of commodities each defined by the tuple (o_k, d_k, W_k) representing the origin, destination, and demand quantity of a commodity $k \in K$, respectively. The key feature of this problem is its use in evaluating the trade-off between infrastructure investment and operational costs. The former is modeled by the fixed cost paid for using an arc f_{ij} joining node i to node j. The latter is modeled by a linear transportation cost c_{ij}^k paid per unit of commodity k routed on arc (i, j). The goal is to route all commodities from origins to destinations at minimal cost.

Two well-known mixed integer models for this problem are the *aggregated* and *disaggregated* arc-based formulations. Both use a set of binary variables y_{ij} to model whether arc (i, j) is installed or not and a set of continuous variables x_{ij}^k to represent the fraction of commodity k's demand routed on arc (i, j). In this study, we use the disaggregated formulation since its tighter linear programming (LP) relaxation is preferred when applying Benders decomposition (Magnanti and Wong 1981). The MUFND can be formulated as follows:

$$\text{(P)} \quad \text{minimize} \sum_{(i,j)\in A} f_{ij}y_{ij} + \sum_{k\in K}\sum_{(i,j)\in A} W^k c_{ij}^k x_{ij}^k \tag{1}$$

$$\text{subject to} \quad \sum_{j\in N} x_{ji}^k - \sum_{j\in N} x_{ij}^k = \begin{cases} -1 & \text{if } i = o_k \\ 1 & \text{if } i = d_k \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in N, k \in K \tag{2}$$

$$x_{ij}^k \leq y_{ij} \quad \forall (i,j) \in A, k \in K \tag{3}$$

$$x_{ij}^k \geq 0 \qquad \forall (i,j) \in A, k \in K \tag{4}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A. \tag{5}$$

The objective function (1) is the total cost of the network including both the installation and routing costs for all arcs and commodities. Flow conservation constraints (2) ensure that the demand for all commodities is routed from origin to destination. Constraint set (3) assures that no flow is sent through an arc that has not been installed, while (4) and (5) are the non-negativity and integrality conditions on x and y, respectively.

Note that depending on the instance data, P is a valid formulation for the Steiner tree problem (all commodities share the same origin and no transportation costs exist) or the travelling salesman problem (all arcs have the same fixed cost, the underlying graph is complete, and commodities are sent between every pair of nodes) (Holmberg and Hellstrand 1998). This shows the wide range of special cases generalized by the MUFND and as such the inherent difficulty in developing an efficient exact algorithm for this general model.

3. Benders decomposition for the MUFND

Benders decomposition is a well-known solution method for mixed integer programming problems (Benders 1962). It splits large formulations into two problems, an integer master problem and a linear subproblem. The principle behind Benders decomposition is the projection of a large problem into a smaller subspace, namely the space of the integer constrained variables. As a consequence, the projected model contains an exponential number of constraints known as Benders cuts, indexed by the extreme points and extreme rays of a special linear programming problem known as the dual subproblem (DSP) or slave problem. Noting that not all Benders cuts are necessary to obtain the optimal solution, Benders (1962) proposed to relax these and iteratively solve the integer master problem to obtain a lower bound on the integral optimal solution value and then substitute the solution into the dual subproblem thereby obtaining an upper bound and a Benders cut to be added to the master problem. This is to be repeated until the upper and lower bounds are within a given optimality tolerance ϵ. In this section, we present the derivation of the Benders reformulation of P and the use of cutset inequalities to replace the classic Benders feasibility cuts.

3.1. Benders reformulation

The following steps describe the process of applying Benders decomposition to formulation P of the MUFND. Note that by fixing $y = \bar{y}$, where $\bar{y} \in Y$ and $Y = \mathbb{B}^{|A|}$ denotes the set of binary vectors associated with the y_{ij} variables, we obtain a linear program in x that is easily solved. This new linear program will be denoted as the primal subproblem (PSP) and has the following form:

$$\text{(PSP)} \quad \text{minimize} \sum_{k \in K} \sum_{(i,j) \in A} W^k c_{ij}^k x_{ij}^k$$

$$\text{subject to} \quad \sum_{j \in N} x_{ji}^k - \sum_{j \in N} x_{ij}^k = \begin{cases} -1 & \text{if } i = o_k \\ 1 & \text{if } i = d_k \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in N, k \in K \tag{6}$$

$$x_{ij}^k \leq \bar{y}_{ij} \qquad \forall (i,j) \in A, k \in K \tag{7}$$

$$x_{ij}^k \geq 0 \qquad \forall (i,j) \in A, k \in K.$$

Note that PSP can be split into $|K|$ subproblems PSP_k, one for each commodity. Let λ and μ denote the dual variables of constraints (6) and (7), respectively. From strong duality, each PSP_k can be substituted by its linear programming dual, denoted as DSP_k, of the form:

$$\text{(DSP}_k\text{)} \quad \text{maximize } \lambda_{d_k}^k - \lambda_{o_k}^k - \sum_{(i,j) \in A} \mu_{ij}^k \bar{y}_{ij} \tag{8}$$

$$\text{subject to} \quad \lambda_j^k - \lambda_i^k - \mu_{ij}^k \leq W^k c_{ij}^k \qquad \forall (i,j) \in A$$

$$\mu_{ij}^k \geq 0 \qquad \forall (i,j) \in A$$

$$\lambda_i^k \in \mathbb{R} \qquad \forall i \in N.$$

From Farkas' Lemma, we know that for a given $k \in K$, PSP_k is feasible if and only if

$$\bar{y} \in R_k = \left\{ y \in Y |\ 0 \geq \lambda_{d_k}^k - \lambda_{o_k}^k - \sum_{(i,j) \in A} \mu_{ij}^k \bar{y}_{ij},\ \forall (\lambda^k, \mu^k) \in \Theta \right\},$$

where Θ is the recession cone of DSP_k. The inequalities that define R_k are known as Benders feasibility cuts and, although by Farkas' Lemma there exists an infinite number of them, only those associated with the (finite) set of extreme rays are necessary. Therefore, we use the representation of each polyhedron associated with each DSP_k in terms of its extreme points and extreme rays to determine whether PSP is infeasible or feasible and bounded.

Let $\text{Ext}(\text{DSP}_k)$ and $\text{Opt}(\text{DSP}_k)$ denote the sets of extreme rays and extreme points of DSP_k, respectively. If, for a given $y \in Y$, there exists at least one $k \in K$ and one extreme ray $(\lambda, \mu) \in \text{Ext}(\text{DSP}_k)$ for which

$$0 < \lambda_{d_k}^k - \lambda_{o_k}^k - \sum_{(i,j) \in A} \mu_{ij}^k \bar{y}_{ij},$$

then DSP_k is unbounded and PSP is infeasible. However, if

$$0 \geq \lambda^k_{d_k} - \lambda^k_{o_k} - \sum_{(i,j)\in A} \mu^k_{ij}\bar{y}_{ij},$$

for each $k \in K$ and each extreme ray $(\lambda, \mu) \in \text{Ext}(\text{DSP}_k)$, then all DSP_k are bounded and the PSP is feasible. The optimal value of each DSP_k is then equal to

$$\max_{(\lambda,\mu)\in \text{Opt}(\text{DSP}_k)} \lambda^k_{d_k} - \lambda^k_{o_k} - \sum_{(i,j)\in A} \mu^k_{ij}\bar{y}_{ij}.$$

Using continuous variables z_k for the transportation cost of each commodity $k \in K$, the *Benders reformulation* of P is

$$(\text{MP}_0) \quad \text{minimize} \sum_{(i,j)\in A} f_{ij}y_{ij} + \sum_{k\in K} z_k \tag{9}$$

$$\text{subject to} \quad z_k \geq \lambda^k_{d_k} - \lambda^k_{o_k} - \sum_{(i,j)\in A} \mu^k_{ij}y_{ij} \qquad \forall k \in K, (\lambda,\mu)_k \in \text{Opt}(\text{DSP}_k) \tag{10}$$

$$0 \geq \bar{\lambda}^k_{d_k} - \bar{\lambda}^k_{o_k} - \sum_{(i,j)\in A} \bar{\mu}^k_{ij}y_{ij} \qquad \forall k \in K, (\bar{\lambda},\bar{\mu})_k \in \text{Ext}(\text{DSP}_k) \tag{11}$$

$$z \in \mathbb{R}^{|K|}$$

$$y \in \{0,1\}^{|A|}.$$

MP_0, also known as the Benders master problem, exploits the decomposability of the subproblems by disaggregating the feasibility and optimality cuts per commodity. This type of *multi-cut* reformulation leads to a better approximation of the transportation costs at each iteration, which has been empirically shown to improve solution times (Magnanti et al. 1986, Contreras et al. 2011).

Exploiting the structure of the MUFND, we replace the Benders feasibility cuts (11) with cutset inequalities which are sufficient to guarantee the feasibility of PSP (Costa et al. 2009). The advantage of using cutset inequalities is that they can be efficiently separated by solving a minimum cut problem over an auxiliary network. Algorithms such as the Edmonds-Karp algorithm (Edmonds and Karp 1972) and breadth first search are efficient in separating cutset inequalities for fractional and integer solutions, respectively. With this in mind, we substitute the use of Benders feasibility cuts with cutset inequalities yielding the following final Benders reformulation:

$$\text{(MP)} \quad \text{minimize} \sum_{(i,j)\in A} f_{ij}y_{ij} + \sum_{k\in K} z_k \tag{12}$$

$$\text{subject to} \quad z_k \geq \lambda^k_{d_k} - \lambda^k_{o_k} - \sum_{(i,j)\in A} \mu^k_{ij} y_{ij} \qquad \forall k \in K, (\lambda,\mu)_k \in \text{Opt}(\text{DSP}_k) \tag{13}$$

$$\sum_{(i,j)\in\delta(S)} y_{ij} \geq 1 \qquad \forall S \in \Delta_K \tag{14}$$

$$z \in \mathbb{R}^{|K|}$$

$$y \in \{0,1\}^{|A|},$$

where $\delta(S) = \{(i,j) \in A | i \in S, j \in N \backslash S\}$ and Δ_K is the set of subsets $S \subset N$ such that $o_k \in S$ and $d_k \notin S$ for some $k \in K$.

4. A branch-and-Benders-cut algorithm for the MUFND

Since its introduction, Benders decomposition has been successfully used to solve several difficult problems such as airline scheduling (Cordeau et al. 2001, Papadakos 2009), facility location (Geoffrion and Graves 1974, Fischetti et al. 2017, Ortiz-Astorquiza et al. 2017), hub network design (Contreras et al. 2011), and fixed-charge network design (Costa 2005). Although the initially proposed algorithm suffered from slow convergence, through the years researchers have devised enhancements to significantly increase its speed. Recent implementations of Benders decomposition incorporate additional strategies such as the generation of strong cuts, cut selection, stabilization, lower bound reinforcing, and solving one enumeration tree (Botton et al. 2013, Naoum-Sawaya and Elhedhli 2013, Adulyasak et al. 2015, van Ackooij et al. 2016, Fischetti et al. 2017, Rahmaniani et al. 2017, Bodur and Luedtke 2017, Ortiz-Astorquiza et al. 2017). Choosing the best enhancements for a given problem is not a trivial task since each improves the performance in a different manner.

Our branch-and-Benders-cut algorithm employs the following algorithmic features: a preprocessing routine to solve the linear relaxation, the generation of Pareto-optimal cuts, a core point selection criterion, lower bound strengthening via lift-and-project cuts, an in-tree matheuristic, and fine-tuning of cut parameters. In the following sections we explain each of the aforementioned enhancements.

4.1. Preprocessing

Since MP is a Benders reformulation of the original formulation P, by relaxing the integrality constraints and adding all Benders cuts to MP, we would obtain the LP relaxation solution of P. This is particularly important to note when implementing Benders decomposition in a single enumeration tree. One of the recent common practices is to solve MP as a linear program with a cutting plane algorithm and use the Benders cuts generated as part of the problem definition declared to the MIP solver. General-purpose

solvers use this information to increase lower bounds, infer good branching rules, and fix variables in their preprocessing routine to reduce the underlying linear program's size.

In our algorithm, we solve MP as a linear program before declaring it within the MIP framework. However, instead of defining the problem with all Benders cuts generated so far, we only include the Benders cuts that are binding at the optimal LP solution as in Fischetti et al. (2017) and in Bodur and Luedtke (2017). This guarantees that we obtain the LP optimal value before attempting to separate Benders cuts but pass on only the essential information to the general-purpose solver and avoid declaring an excessively large problem. This step, while obvious, significantly helps the solution process.

4.2. Pareto-optimal cut separation

Since the seminal paper on cut selection for Benders decomposition by Magnanti and Wong (1981), Pareto-optimal cuts have become a standard practice. The approach applies to problems for which there is an infinite number of alternative optimal solutions to DSP_k and therefore Benders optimality cuts. This is particularly the case in network design problems known for their primal degeneracy. For a minimization problem, the authors define cut dominance as follows. Given two cuts defined by dual solutions u and u^1 of the form $z \geq f(u) + yg(u)$ and $z \geq f(u^1) + yg(u^1)$, respectively, the cut defined by u dominates that defined by u^1 if and only if $f(u) + yg(u) \geq f(u^1) + yg(u^1)$ with strict inequality holding for some feasible y of MP. If a cut defined by u is not dominated by any other optimality cut, then this cut is said to be a Pareto-optimal Benders cut.

In general, to obtain Pareto-optimal Benders cuts an additional linear program must be solved at each iteration. This additional linear program is the same as the dual subproblem with two exceptions. The first is that a point y^0 in the relative interior of the master problem space, known as a core point, replaces the master problem solution $\bar{y}$ in the objective function (8). The second is that an equality constraint is added to ensure that the obtained solution belongs to the set of alternative optimal solutions of DSP_k for the current master problem solution $\bar{y}$. In most cases, these modifications break the structure of the dual subproblem exploitable by an efficient combinatorial algorithm. This leads to having to solve an additional linear program. Papadakos (2008) addresses this issue and presents a modified procedure that does not require solving an additional linear program. The modified dual subproblem uses a point that must satisfy characteristics that are more relaxed than Magnanti and Wong's conditions.

In our algorithm, we use the "tailored" subproblem for the MUFND as presented in Magnanti et al. (1986). The authors point out that the additional linear program for each commodity $k \in K$ in the classic Pareto-optimal approach is equivalent to solving the following *parametric minimum cost flow problem*:

$$(\text{MCF}_k) \quad \text{minimize} \sum_{(i,j)\in A} W^k c_{ij}^k x_{ij}^k - DSP_k(\bar{y})x_0 \tag{15}$$

$$\text{subject to} \quad \sum_{j\in N} x_{ji}^k - \sum_{j\in N} x_{ij}^k = \begin{cases} -(1+x_0) & \text{if } i = o_k \\ 1+x_0 & \text{if } i = d_k \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in N \tag{16}$$

$$x_{ij}^k \leq y_{ij}^0 + x_0 \bar{y}_{ij} \qquad \forall (i,j) \in A \tag{17}$$

$$x_{ij}^k \geq 0 \qquad \forall (i,j) \in A$$

$$x_0 \in \mathbb{R}.$$

The problem can be interpreted as that in which a rebate of $\text{DSP}_k(\bar{y})$ is given for each additional unit of the commodity routed on the network with demand and capacities defined by (16) and (17), respectively (Magnanti et al. 1986). The authors show that any fixed value $x_0 \geq \sum_{(i,j)\in A} y_{ij}^0$ is optimal for MCF_k, leaving a minimum cost flow problem to be solved for each commodity $k \in K$.

As a result of fixing x_0, it is no longer necessary to solve $\text{DSP}_k(\bar{y})$ since it is now multiplied by a constant in MCF_k. This observation allows us to save computational time by solving MCF_k directly as the separation problem rather than as a complementary problem for Pareto-optimal Benders cuts.

Magnanti and Wong (1981) note that the selection of different core points y^0 leads to varied Pareto-optimal cuts. To the best of our knowledge, the question of selecting an adequate y^0 has not been addressed before in the literature. We provide some guidelines and computationally test different strategies for this in Section 6.2.

4.3. Benders lift-and-project cuts

With the current trend of implementing Benders decomposition within a branch-and-cut framework, the need for problem formulations to have a strong LP relaxation has become more important. Unfortunately, several problems do not satisfy this property. Recently, Bodur and Luedtke (2017) and Bodur et al. (2017) proposed the use of general mixed integer cuts such as mixed integer rounding and split cuts, respectively, within a branch-and-Benders-cut algorithm to improve the LP relaxation. Their results show a significant decrease in the LP gap leading to faster solution times.

Although Hellstrand et al. (1992) show that the polytope defined by P is quasi-integral, the use of modified pivots for integral basic solutions is impractical due to P's degeneracy (Balas and Padberg 1972). In addition, modified integral pivots require the complete formulation whereas our Benders reformulation is a projection into the smaller subspace of the integer variables. As a result, we must find a way to close the LP gap within the Benders decomposition framework. To do this, we adopt the lift-and-project cuts proposed by Balas et al. (1993) to strengthen the master problem LP relaxation.

Lift-and-project cuts, a result of disjunctive programming theory (Balas 1979), were initially proposed as a cutting plane algorithm by Balas et al. (1993) but were later extended to the branch-and-cut framework (Balas et al. 1996) by proving the ability to

find globally valid cuts at nodes within the enumeration tree by means of a closed form lifting procedure. The framework is as follows.

Given $P = \{x \in \mathbb{R}^n | \tilde{A}x \geq \tilde{b}\}$ with inequalities of the form $1 \geq x \geq 0$ included in $\tilde{A}x \geq \tilde{b}$ and $P_D = conv\{x \in P | x_j \in \{0,1\}, \forall j = 1...p\}$ where $p < n$, lift-and-project cuts can be obtained by:

1 Selecting an index $\hat{\jmath} \in \{1...p\}$. Multiplying $\tilde{A}x \geq \tilde{b}$ by $(1 - x_{\hat{\jmath}})$ and $x_{\hat{\jmath}}$.

2 Linearizing the obtained system by substituting $y_i = x_{\hat{\jmath}} x_i$ and $x_i = x_i x_i$.

3 Projecting the system back into the original space by means of a cone projection.

Balas (1979) shows that it is possible to obtain the "deepest" lift-and-project cut of the form $\sum_{i \in I} \alpha_i x_i \geq \beta$ that cuts off the LP optimum $\bar{x}$ for a given $\hat{\jmath} \in \{1...p\}$ by solving the following linear program:

$$\begin{aligned} (\text{CGLP}_{\hat{\jmath}}) \quad & \text{minimize} \sum_{i \in I} \alpha_i \bar{x}_i - \beta \\ \text{subject to} \quad & \alpha - uA + u^0 e_{\hat{\jmath}} \geq 0 \\ & \alpha - vA - v^0 e_{\hat{\jmath}} \geq 0 \\ & -\beta + ub = 0 \\ & -\beta + vb + v_0 = 0 \\ & u, v \geq 0, \end{aligned}$$

where $e_{\hat{\jmath}}$ is the vector of all 0s except for a 1 in the $\hat{\jmath}$-th component.

The feasible space of $\text{CGLP}_{\hat{\jmath}}$ is a convex cone. Therefore, a normalization constraint must be added to ensure a finite optimal solution. It has been shown (Balas and Perregaard 2002) that varied normalizations lead to significantly different cuts. In our implementation, we use the following normalization constraint:

$$\sum_i u_i + u^0 + \sum_i v_i + v^0 = 1.$$

The resulting cut can be strengthened by using a closed formula derived by imposing integrality constraints of other $\{0,1\}$ variables (Balas and Perregaard 2002). We adopt this strengthening procedure in our algorithm. For a given variable x_k, its coefficient α_k in the lift-and-project cut can be replaced by $\alpha'_k = \min\{ua_k + u^0 \lceil m_k \rceil, va_k - v^0 \lfloor m_k \rfloor\}$, where

$$m_k = \frac{va_k - ua_k}{u^0 + v^0}.$$

While solving our Benders reformulation we do not have the complete polyhedral description of P since there would be exponentially many constraints. We resort to defining $\tilde{A}$ as the feasibility and optimality cuts that are binding at the LP relaxation and the

$1 \geq x \geq 0$ constraints. Although this is known to give a weaker lift-and-project cut, we also highlight its role in significantly reducing the size of $CGLP_{\hat{j}}$ leading to times of less than 0.02 seconds to obtain a cut. A similar strategy is used by Balas and Perregaard (2002) outside the context of Benders decomposition. Another important factor in the lift-and-project process is selecting the index $\hat{j}$. In our implementation we choose the fractional variable of the LP solution with highest fixed cost.

During our experiments we noted that these lift-and-project cuts are very dense and often with small, numerically unstable coefficients. This led to numerical issues when too many were added to the master problem at the same time. To circumvent this, we ensure that at most seven lift-and-project cuts are added. We use an additional stopping criterion of increase in the LP optimal value. In other words, if the effect of adding a Benders lift-and-project cut is negligible, we then stop generating them. Note that the effect of the lift-and-project cut is highly dependent on the variable $\hat{j}$ chosen. However, we found this criterion to be an effective rule to prevent numerical instabilities in our algorithm.

4.4. An in-tree matheuristic

An important factor in solving difficult optimization problems, in particular when using branch-and-bound methods, is obtaining high quality feasible solutions. Finding these early in the enumeration process often leads to smaller search trees since they provide better bounds for pruning and a guide for selecting variables to branch on. If found in a preprocessing stage, they can be used to perform variable elimination tests as in Contreras et al. (2009, 2011). Preliminary tests showed that the latter approach eliminated few variables from the problem even if the optimal solution value was used for these variable elimination tests. We therefore propose an in-tree matheuristic that exploits the information generated during the enumeration process. Our algorithm uses the paths obtained while solving the Benders subproblems as variables in a path-based formulation of the MUFND.

Let Θ_k^μ denote a binary variable whose value is 1 if path μ is used for commodity k and 0 otherwise, while y_{ij} denotes the network design variables as in P. Define parameter $v_k^\mu(i,j) = 1$ if arc (i,j) belongs to path μ for commodity k, and 0 otherwise. Finally, let Ω_k denote the set of paths from $o(k)$ to $d(k)$ and Ω represent the union of these sets over K. With this notation we have the following path-based formulation for the MUFND:

$$(P_{Heur}) \quad \text{minimize} \sum_{(i,j)\in A} f_{ij} y_{ij} + \sum_{k\in K} \sum_{\mu\in\Omega_k} [W^k \sum_{(i,j)\in A} c_{ij}^k v_k^\mu(i,j)]\Theta_k^\mu \tag{18}$$

$$\text{subject to} \quad \sum_{\mu\in\Omega_k} \Theta_k^\mu = 1 \qquad \forall k \in K \tag{19}$$

$$\sum_{\mu\in\Omega_k} v_k^\mu(i,j)\Theta_k^\mu \leq y_{ij} \qquad \forall (i,j) \in A, k \in K \tag{20}$$

$$\Theta_k^\mu \in \{0,1\} \qquad \forall k \in K, \mu \in \Omega_k. \tag{21}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A. \tag{22}$$

The objective function (18) represents the total cost of routing the commodities on the network, including design and transportation costs. Constraints (19) ensure that each commodity is routed by exactly one path while (20) force the design variables of the arcs used in a path to take value 1 if the path is used to route a commodity. We note that (20) can be disaggregated to provide a tighter formulation. However, preliminary tests showed it to have a negative effect on the computation time due to the increase in problem size.

Given the exponential number of paths for a given commodity, P_{Heur} is usually solved using column generation and branch-and-price when used to represent the MUFND. In our algorithm, however, we will only use this formulation to solve for improved MUFND solutions obtained from paths generated while solving the Benders subproblems. This avoids having to solve several rounds of pricing problems at each branch-and-price node. The only added computational effort comes from the fact that primal solutions to MCF_k, the Benders subproblem to obtain Pareto optimal cuts, sends different amounts of flow through several paths from origin to destination. To obtain single paths to be used in P_{Heur}, we solve a shortest path problem over two networks derived from the primal solution of MCF_k. The first network contains the arcs that send any flow greater than 0.1 in the solution while the second contains arcs that send more than 1 unit of flow. This provides us with the potential to generate two different paths at a low computational cost every time the Benders subproblem is solved.

Finally, note that the branching over design variables during the enumeration process of our Benders algorithm forces the generation of a varied set of paths for the Benders subproblems and hence variables for P_{Heur}. The integration of our in-tree matheuristic into our branch-and-Benders-cut algorithm provides a means of exploiting two formulations of the same problem simultaneously as in Hewitt et al. (2010) for capacitated multicommodity network design with the difference that our algorithm solves the problem to optimality whereas theirs finds feasible solutions with a quality certificate.

4.5. Implementation details

We begin our solution process by solving the LP relaxation of MP using our cutset separating routines and MCF_k as our separation oracle to obtain Pareto-optimal Benders cuts with core points defined as will be described in Section 6.2. Upon confirming that no more violated Benders cuts exist, we use those that are binding at the optimal solution to obtain a violated Benders lift-and-project cut. This cut is added to the MP relaxation and we resume separating violated Benders cuts. We repeat this process until our stopping criteria are satisfied.

We then define the MIP problem in CPLEX with the active Benders and lift-and-project cuts as lazy and user constraints. This prevents defining an excessively large initial problem. Another of the important aspects to consider when implementing this method is the separation and cut adding frequency. Adding too few cuts leads to an underestimation of the lower bounds of nodes in the enumeration tree, while adding too many cuts leads to large LPs that require a longer computation time to solve.

Several cutting frequencies were tested in preliminary experiments. The best of the tested strategies was separating Benders cuts at all nodes in the first five levels of the

enumeration tree and then separating at every 100th node. For all these, only one round of violated Benders cuts are added. For fractional solutions, a minimum violation of 0.01 was required to add the cut to the constraint pool of the node's linear problem.

Lastly, to prevent executing our in-tree matheuristic too frequently, we limit its use to only one node at each depth of the enumeration tree greater than ten that is divisible by five. In addition, to ensure it has the potential to find an improved solution, it is only called if at least N new paths have been generated since its last execution. Finally, to avoid spending excessive time in this heuristic process, a time limit of thirty seconds was set for each execution.

5. A cut-and-solve algorithm for the MUFND

Introduced by Climer and Zhang (2006) in the artificial intelligence community, cut-and-solve has been used to solve well-known combinatorial optimization problems such as the *travelling salesman problem* and the *single-source capacitated facility location problem* (Yang et al. 2012, Gadegaard et al. 2018). The cut-and-solve framework is closely related to local branching (Fischetti and Lodi 2003) in the sense that at each level of the enumeration tree only two child nodes exist, one corresponding to a smaller "sparse" problem and the other as its complement known as the "dense" problem. However, while in local branching one begins with a feasible solution and defines the subproblems based on the Hamming distance, cut-and-solve allows for more generic problem definitions and does not require an initial feasible solution. Since our proposed framework is more closely aligned with the latter, we adopt the cut-and-solve terminology and notation for the rest of the paper. We next provide a brief description of the cut-and-solve procedure as presented by Climer and Zhang (2006).

The "sparse" and "dense" problems are defined by constraints over a set of variables. These constraints, known as "piercing" cuts, are of the form $\sum_{i \in I} x_i \leq \sigma$ and $\sum_{i \in I} x_i \geq \sigma + 1$ where $I \subset N$ is a subset of the problem's binary variables and $\sigma \in \mathbb{Z}$.

Upon branching, the "sparse" problem is solved to optimality by means of branch-and-bound or any exact method to obtain a primal bound (UB_{sparse}) on the original problem. This highlights the need to define sparse problems that are easily solved. Next, the linear relaxation of the dense problem is solved to obtain a lower bound (LB_{dense}) on the remaining solution space of the original problem. If $LB_{dense} \geq UB_{sparse}$ then UB_{sparse} is optimal for the complete problem. Otherwise, another piercing cut is defined over the dense problem and the procedure is repeated.

We propose the use of our branch-and-Benders-cut algorithm as the black box MIP solver within the cut-and-solve algorithm and a tailored rule for selecting the variables to consider in the "sparse" problems. Two important advantages of using our Benders algorithm within the cut-and-solve framework are the reduced problem size and the re-usability of the Benders cuts generated in previous sparse problems. On the other hand, some advantages to using cut-and-solve over Benders is that piercing cuts significantly reduce the solution space and the optimal values of previous sparse problems are useful for pruning branches in the enumeration tree.

Our cut-and-solve algorithm begins by considering the union of shortest paths, denoted as $\cup_{k\in K} P_k$, of each commodity using only their transportation costs. The resulting set contains on average approximately 90% of the arcs open in an optimal solution and is thus an ideal candidate to define our first "sparse" problem.

For ease of exposition we introduce the following notation. Let $(\bar{y}, \bar{z})(t)$ represent the solution of the t-th sparse problem, $I(t)$ denote the set of indices of arc variables whose value is 1 in $(\bar{y}, \bar{z})(t)$ and $\chi(\bar{y}, \bar{z})(t))$ be its objective function value. In particular, $(\bar{y}, \bar{z})(0)$ refers to the objective function value of activating and routing on the arcs of the union of shortest paths.

At a given $t \geq 1$, we define the following sparse problem:

$$(\mathrm{MP}_{sparse}(t)) \quad \text{minimize} \sum_{(i,j)\in A} f_{ij} y_{ij} + \sum_{k\in K} z_k \tag{23}$$

$$\text{subject to} \quad z_k \geq \lambda^k_{d_k} - \lambda^k_{o_k} - \sum_{(i,j)\in A} \mu^k_{ij} y_{ij} \qquad \forall (\lambda, \mu)_k \in Opt(DSP_k), k \in K \tag{24}$$

$$\sum_{(i,j)\in\delta(S)} y_{ij} \geq 1 \qquad \forall S \in \Delta_K \tag{25}$$

$$z \in \mathbb{R}^{|K|} \tag{26}$$

$$y \in \{0,1\}^{|A|} \tag{27}$$

$$\sum_{(i,j)\notin I(t-1)} y_{ij} \leq t \tag{28}$$

$$\sum_{(i,j)\notin I(s)} y_{ij} \geq s+2 \qquad \forall s = 0, ..., t-1 \tag{29}$$

$$\sum_{(i,j)\in A} f_{ij} y_{ij} + \sum_{k\in K} z_k \leq \chi((\bar{y}, \bar{z})(t-1)), \tag{30}$$

where $\Delta_k = \{S \subset N | \exists k \in K \text{ where } o_k \in S, d_k \notin S\}$. The constraints (23)-(27) are the Benders master problem reformulation of the MUFND. Constraint (28) is the piercing cut that allows at most t variables not in the previous solution to take the value of 1 while constraints (29) are the negations of (28) from previous iterations. The latter ensure that previously searched areas of the feasible space are not considered in the new sparse problem. Finally, constraint (30) imposes that the optimal solution of the current sparse problem has objective value of at most the optimal value of the previous one. This constraint ensures that the obtained solutions do not worsen after each iteration and saves computation time since its value is used as a pruning criterion for the enumeration tree.

If $\mathrm{MP}_{sparse}(t)$ is feasible, we define $\mathrm{MP}_{sparse}(t+1)$ without solving the LP relaxation of the corresponding dense problem. In this respect, our algorithm bears resemblance to local branching as proposed in Fischetti and Lodi (2003). This is done until two successive optimal solutions to $\mathrm{MP}_{sparse}(t)$ are the same or until $\mathrm{MP}_{sparse}(t)$ is infeasible, due to (30). If either occurs, the dense problem, MP_{dense}, is defined. The dense problem is similar to $\mathrm{MP}_{sparse}(t)$ with the exception that (28) is replaced by $\sum_{(i,j)\notin I(t-1)} y_{ij} \geq t+1$

and $(\bar{y}, \bar{z})(t-1)$ in (30) is replaced by the best solution found so far. We then solve $\text{MP}_{dense}(t)$ which will either be feasible and hence provide the true optimal solution or will be infeasible meaning that the best solution found so far is indeed optimal.

The presented framework provides a novel research direction, different from Rei et al. (2009), for combining cut-and-solve and Benders decomposition. Below we present the pseudocode of our cut-and-solve Benders algorithm.

Algorithm 1 Cut-and-solve Benders algorithm for the MUFND

Require: 0: Initialization
  $(\bar{y}, \bar{z})(0) = \cup_{k \in K} P_k$, $t = 1$, $best = (\bar{y}, \bar{z})(0)$
  Step 1: Define and solve $\text{MP}_{sparse}(t)$
  if ($\text{MP}_{sparse}(t)$ is feasible $\wedge$ $(\bar{y}, \bar{z})(t-1) \neq (\bar{y}, \bar{z})(t)$) **then**
    $best = (\bar{y}, \bar{z})(t)$
    $t = t + 1$;
    `Goto` **Step 1**
  else
    `Goto` **Step 2**
  end if
  Step 2: Define and solve MP_{dense}
  if MP_{dense} is feasible **then**
    Update *best*
  end if
  Return *best*.

6. Computational experiments

We perform extensive computational experiments to evaluate the efficiency of our proposed methods and the effect of the enhancements implemented. Our analyses focus on: the LP gap closed by adding Benders lift-and-project cuts to MP's linear relaxation, adequate core point selection, and the efficiency of our proposed solution methods versus the state-of-the-art general-purpose MIP solver CPLEX 12.7.1.

We use the well-known "Canad" multicommodity capacitated network design instances (Crainic et al. 2001) as our testbed. This dataset consists of 205 instances with arc capacities. Ignoring the capacity constraints leaves a total of 93 distinct instances for our experiments. The testbed can be divided into three classes. The first are the 31 "C" instances with many commodities compared to nodes while the second are eight "C+" instances with few commodities compared to nodes. Finally, Class III is divided into two subgroups. Class III-A and III-B are each comprised of 27 "R" instances on small and medium sized graphs, respectively.

We generate eight large-scale instances, denoted as Class IV, on which we test our algorithms with a 24-hour time limit. These were generated using the Mulgen generator (Crainic et al. 2001) available at *http://pages.di.unipi.it/frangio/* with sizes of up to 1,500

arcs and 1,500 commodities. To the best of our knowledge these are the largest instances of the MUNDP to be solved by an exact algorithm.

Another characteristic of our testbed is the existence of instances that have an LP gap strictly greater than 0. This is important in our analysis as we need to test our algorithm's ability to quickly explore the enumeration tree and the efficiency of our proposed lift-and-project Benders cuts. The "Canad" testbed contains several instances with this property. Table 1 details the number of instances for each LP gap range (%) in our testbed.

Table 1: Distribution of "Canad" instances' LP gaps (%)

Class	**0**	$(0, 1]$	$(1, 2]$	$(2, 3]$	$(3, 4]$	$[4, 7.2]$	**Total**
Class I	10	7	2	4	2	6	31
Class II	7	1					8
Class III-A	26		1				27
Class III-B	9	2		2	5	9	27
Class IV	0	3	3	2			8
Total	**52**	**13**	**6**	**8**	**7**	**15**	**101**

All algorithms were coded in C using the callable library for CPLEX 12.7.1. The separation and addition of cutset inequalities and Benders optimality cuts is implemented via lazy cut callbacks and user cut callbacks. For a fair comparison, all use of CPLEX was limited to one thread and the traditional MIP search strategy. Experiments were executed on an Intel Xeon E5 2687W V3 processor at 3.10 GHz under Linux environment.

6.1. Impact of lift-and-project cuts on LP gap

As shown in Table 1, Class I, III-B, and IV contain most of the instances with higher LP gap. Preliminary tests showed these to be the most difficult to solve, in particular when it came to proving optimality. It is in this spirit that we proposed using lift-and-project cuts to improve the LP bound at the root node.

Table 2 shows the percentage of the LP gap ($LP_{imp}\%$) closed by applying at most seven lift-and-project cuts to the linear relaxation of our Benders master problem. This percentage is calculated as $LP_{imp} = 100 \times \frac{(LP_{MPLP} - LP_{MP})}{Opt - LP_{MP}}$ where LP_{MPLP} is the optimal value of the linear relaxation of the master problem with the additional lift-and-project cuts, LP_{MP} is the optimal value of the linear relaxation of the master problem, and Opt is the optimal value of the problem.

The average improvement of the LP gap is of 5.26% over the 50 solved instances with an LP gap. There are many factors that contribute to this behavior. The first is that lift-and-project cuts as proposed by Balas et al. (1993) require the complete formulation of the problem. In our implementation, we use a relaxation comprised of only the Benders cuts that are binding at the LP solution. In Balas et al. (1996), this relaxation is shown to generate weaker cuts. Second, our stopping criterion for lift-and-project cuts is conservative, avoiding generating too many of them at the beginning due to their numerical instability. Finally, our simplified variable selection rule also contributes to this performance.

Table 2: LP gap (%) closed

Class	**No. of instances**	LP_{imp}
Class I	21	6.61
Class II	1	9.07
Class III-A	1	20.45
Class III-B	18	3.65
Class IV	8	2.99
Total	**49**	**5.26**

6.2. Impact of core point selection

Despite the use of Pareto optimal Benders cuts being now common practice, little computational experimentation with core point selection strategies has been done. We next show how core point selection influences the solution time and that a tailored core point selection strategy can lead to significant time savings. We test three strategies for core point selection. The first is the most common practice in the literature while the second is a novel strategy that can be applied to any Benders reformulation of a mixed binary program. Finally, the third is a strategy tailor-made for the MUFND and is based on the union of shortest paths of the commodities $k \in K$, as in the definition of our initial "sparse" problem. Let y^0 and $\bar{y}$ denote the current core point and master problem solution, respectively. The details of the three core point selection strategies are as follows:

1 Initialize y^0=$\{1\}^{|A|}$ and dynamically update the core point as $y^0 = 0.5y^0 + 0.5\bar{y}$ as in Papadakos (2008) and similar to Fischetti et al. (2017).

2 Initialize a stabilizer point $\hat{y}$ as $\hat{y} = \{1\}^{|A|}$ which will then be updated as better incumbent solutions are found during the enumeration process. Dynamically update the core point as $y^0 = 0.5\hat{y} + 0.5\bar{y}$.

3 Fix the core point throughout the entire process based on the arcs that are present in at least one of the commodities' shortest paths, denoted as $\cup_{k\in K} P_k$. The fixed core point is defined as $y^0_{ij} = 0.7$ if $(i,j) \in \cup_{k\in K} P_k$ and $y^0_{ij} = 0.2$ if $(i,j) \notin \cup_{k\in K} P_k$. These values were chosen after running preliminary experiments with the values of $\{0.5, 0.6, 0.7, 0.8, 0.9\}$ for arcs in the routing solution and $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ for arcs not in the routing solution.

Note that in all three cases, the proposed core point is in the interior of the $\{0,1\}^{|A|}$ hypercube. However, to solve MCF_k, y^0 must not only lie in the interior of the $\{0,1\}^{|A|}$ hypercube but must also define a network through which one unit of demand can be sent from o_k to d_k, $\forall k \in K$. Failure to do so could lead to MCF_k being infeasible despite $\bar{y}$ being a feasible solution for the MUFND. This was observed empirically to have a particularly pernicious effect on the overall computation time.

To remedy this we solve a minimum cut for each $k \in K$ to check for feasibility when defining the fixed core point of strategy 3. If there exists a minimum cut $\delta(S)_k = \{(i,j) \in$

$A|o_k \in S$ and $d_k \in N\backslash S\}$ for a commodity $k \in K$ with $\sum_{(i,j)\in\delta(S)_k} y^0_{ij} < 1$, we then increase the value of each arc in $\delta(S)_k$ by $[1/|\delta(S)_k|]+0.01$ and check again for a violated cutset. This is repeated until no such cutset exists. We place a cap on the value of y^0_{ij} to be at most 0.9999 to ensure y^0 remains an interior point. Given that the core point is fixed throughout the solution process, this verification is only done once at the beginning. Since the other strategies constantly update the core point, running this procedure every time proved to be time consuming. To circumvent this we run this procedure only when MCF_k becomes infeasible. The solution times, in seconds, of these strategies implemented in our branch-and-Benders-cut algorithm are shown in Table 3.

Table 3: Impact of core point selection- time in seconds

Class	**Nb**	**Strategy 1**	**Strategy 2**	**Strategy 3**
Class I	31	119.46	286.42	73.74
Class II	8	12.96	6.50	6.91
Class III-A	27	0.03	0.04	0.04
Class III-B	27	227.52	451.77	187.15
Class IV	6	112.17	204.02	67.25
Total	**99**	**107.31**	**225.80**	**78.78**

The best performing is our tailored core point selection strategy (strategy 3) which saves over a quarter of the average computation time of the second best performing strategy, the well-known dynamic mid-point update (strategy 1). The worst is the incumbent stabilizer update (strategy 2). These results show the added value of using core point selection strategies that exploit problem structure.

6.3. Computation time

We now compare the computation time of each of our proposed algorithms. We begin by focusing on our branch-and-Benders-cut algorithm since we use the best performing as the black box solver in our cut-and-solve/local branching algorithm. To show the impact of each enhancement, we present four versions of our branch-and-Benders-cut algorithm. The first is without using our in-tree matheuristic nor our lift-and-project cuts (B_0). The second is the same, with the addition of the in-tree heuristic (B_1). B_2 is the branch-and-Benders-cut algorithm with lift-and-project cuts added at the root node and the final version (B_3) combines them all. A time limit of 24 hours is set for all algorithms.

The results are presented in Table 4 with the exception of the instances of classes II and III-A which were all solved in less than a second by our four algorithms and CPLEX. The first three columns describe the problem class, instance sizes ($|N|, |A|, |K|$), and number of instances in each instance group respectively. For each version of the algorithm, two columns are displayed,"Seconds" which denotes the average solution time in seconds and "Nodes" which refers to the average number of nodes explored. Finally, we point out that the averages of class IV and the total test bed are taken only over the instances with comparable solution times to avoid the averages being skewed by large numbers, i.e. instance groups 50,1500,1000 and 50,1500,1500 are omitted.

Table 4: Computational performance of branch-and-Benders-cut algorithm

			B_0		B_1		B_2		B_3	
Class	$(\|N\|,\|A\|,\|K\|)$	**Nb**	**Seconds**	**Nodes**	**Seconds**	**Nodes**	**Seconds**	**Nodes**	**Seconds**	**Nodes**
I	20,230,40	3	0.22	0	0.21	0	0.21	0	0.21	0
	20,230,200	4	15.17	274.50	15.65	271.00	21.58	395.00	26.15	496.00
	20,300,40	4	0.24	0.25	0.24	0.25	0.28	0.75	0.28	0.75
	20,300,200	4	12.11	200.50	12.80	159.00	25.94	371.50	34.67	384.25
	30,520,100	4	305.08	3,335.25	157.50	2,456.50	286.05	2,694.75	233.35	4,439.25
	30,520,400	4	9.14	35.25	9.06	35.25	12.21	36.75	12.34	36.25
	30,700,100	4	8.86	188.25	7.92	162.00	9.42	209.25	8.55	256.25
	30,700,400	4	332.35	4,613.25	293.22	5,322.25	254.17	3,800.50	276.19	3,716.75
	Sub-Total	**31**	**88.14**	**1,115.77**	**64.07**	**1,084.68**	**78.68**	**968.84**	**76.35**	**1,203.81**
III-B	20,120,40	3	0.11	0	0.10	0	0.10	0	0.11	0
	20,120,100	3	1.97	70.67	1.98	70.67	3.42	88.67	3.50	93.00
	20,120,200	3	214.59	1,808.00	255.64	1,921.00	166.83	1,062.33	178.16	1,521.67
	20,220,40	3	2.00	100.67	1.95	99.00	2.08	111.67	2.34	99.33
	20,220,100	3	39.81	688.67	49.54	839.33	177.31	2,548.33	124.55	2,232.33
	20,220,200	3	775.17	3,644.67	929.44	3,285.33	1,320.94	3,583.33	1,460.67	3,738.00
	20,320,40	3	11.44	876.67	15.14	849.67	12.47	872.00	12.69	832.67
	20,320,100	3	6.87	104.33	7.17	104.33	12.50	223.67	11.81	199.33
	20,320,200	3	625.82	1,723.00	669.79	2,273.67	1,012.85	2,476.67	1,087.30	3,176.00
	Sub-Total	**27**	**186.42**	**1,001.85**	**214.53**	**1,049.22**	**300.94**	**1,218.52**	**320.13**	**1,321.37**
IV	40,1200,400	1	9.19	6.00	8.97	6.00	15.56	7.00	15.68	7.00
	40,1200,800	1	53.00	537.00	55.98	669.00	68.69	715.00	63.87	711.00
	40,1200,1200	1	57.07	61.00	57.34	61.00	82.95	74.00	86.69	74.00
	50,1400,400	1	29.94	649.00	31.70	580.00	34.82	601.00	36.54	597.00
	50,1400,800	1	117.31	1,938.00	127.06	1,886.00	177.01	3,860.00	196.47	3,215.00
	50,1400,1200	1	201.33	2,162.00	198.49	2,257.00	183.17	1,655.00	208.59	1,929.00
	50,1500,1000	1	54,074.52	419,563.00	54,680.74	420,298.00	44,367.11	389,246.00	57,487.39	555,136.00
	50,1500,1500	1	69,637.47	262,007.00	69,055.11	262,007.00	time	483,149.00	time	438,179.00
	Sub-Total	**6**	**77.97**	**892.17**	**79.92**	**909.83**	**93.70**	**1,152.00**	**101.31**	**1,088.83**
	Total	**64**	**128.65**	**1,046.75**	**129.03**	**1,053.33**	**173.86**	**1,091.34**	**181.53**	**1,242.63**

We note that with respect to computation time, implementing Benders without including Benders lift-and-project cuts performs on average the fastest. Between the two versions that exclude it, B_0 is on average marginally better than B_1 over the instances of class III-B and IV but 30% slower on average over the instances of class I. This shows that although in most cases using the in-tree heuristic will increase the solution time, there are instances for which it pays off significantly, for example instance groups 30,520,100 and 30,700,400 of class I.

While incorporating Benders lift-and-project cuts has a better solution time when compared to B_0 for the instances in class I, it produces on average over a 30% increase in solution time over the complete testbed. B_2 provides the fastest solution time for instance groups 30,700,400 of class I; 20,120,200 of class III-B; and 50,1500,1000 of class IV; however, for other instance groups, it can double the solution time despite exploring fewer nodes (see instance group 20,220,200 of class III-B). An explanation for this is the density and numerical instability of the Benders lift-and-project cuts. As mentioned before, these cuts have several non-zero coefficients close to zero. This leads to more time required to solve the underlying linear programs and in some instances numerical instability that prevents CPLEX from constructing an advanced basis for nodes in the tree. We also note that their use rarely leads to a reduction in the size of the enumeration

tree as both B_3 and B_4 show a larger average number of nodes explored than B_0 and B_1. This indicates that the addition of these cuts negatively influences the branching within the enumeration tree.

Incorporating both Benders lift-and-project cuts and our in-tree heuristic simultaneously is the version that requires the most computation time on average over the entire test bed. In fact, its solution time is worse than the versions that incorporate them individually. One of the main factors contributing to this is the increase that comes from incorporating Benders lift-and-project cuts which as we have seen, also negatively influences the branching within the enumeration tree. The rest of the solution time increase can be explained by the additional time the in-tree heuristic required to solve P_{Heur}.

Finally, we note that both B_0 and B_1 are able to solve the two largest instances that required significantly more time. In particular, incorporating our in-tree matheuristic proved marginally beneficial for the largest instance. On the other hand, including Benders lift-and-project cuts rendered a time saving of one-fifth of the computation time required by B_0 to solve the second largest instance. This again shows the unpredictable effect of lift-and-project cuts in our branch-and-Benders-cut algorithm.

Considering these results, we choose B_0 as the black box solver for our cut-and-solve algorithm. Figure 1 is the performance profile of our branch-and-Benders-cut algorithm ($BB\&C$), our cut-and-solve (CS/LB) and solving P with CPLEX 12.7.1's branch-and-cut algorithm CPX. We do not compare with CPLEX's black box Benders implementation since preliminary results showed it to perform significantly worse than CPLEX's branch-and-cut. Figure 1 plots the number of instances solved by each algorithm within a given number of seconds.

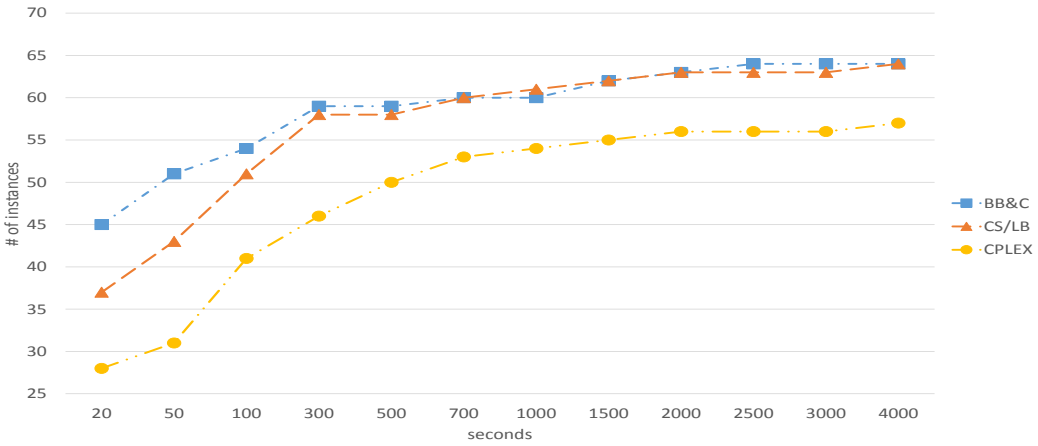

Figure 1: Number of instances solved in a given time limit

We note that both our proposed methodologies are able to solve more instances in less time than CPLEX, having solved close to 83% of the instances in less than 100 seconds and solving all except the two largest instances within 45 minutes. CPLEX on the other hand manages to solve only 41 instances within 100 seconds; less than the number solved by our branch-and-Benders-cut algorithm in 20 seconds. CPLEX runs into trouble proving

optimality for 10 instances requiring over an hour for the least troublesome and over half a day for the most burdensome. In addition, it is unable to solve the two largest instances within the one-day time limit while our branch-and-Benders-cut algorithm solves all instances within twenty hours of CPU time.

When comparing the performance between (CS/LB) and ($BB\&C$), we see from Figure 1 that the latter is faster at solving instances. However, when given additional time, both algorithms have similar behavior. To make a more precise comparison, Table 5 contains the average times, in seconds, required for each class and description of our testbed.

Our results show that on average both $BB\&C$ and CS/LB are an order of magnitude faster than CPLEX for instances all three are able to solve. This speedup is even more significant when limiting our analysis to the large-scale instances. For these, our branch-and-Benders-cut algorithm is up to three orders of magnitude faster than CPLEX. The instances of Class III-B also show a significant saving in computation time in favor of our Benders decomposition-based algorithms. The savings obtained with $BB\&C$ can be largely attributed to solving smaller underlying linear programs in the enumeration tree and exploring the nodes in significantly less time.

Table 5: Comparison of computation times in seconds

Class	$(\|N\|,\|A\|,\|K\|)$	**Nb**	***CPX***	***CS/LB***	***BB&C***
I	20,230,40	3	0.07	0.21	0.22
	20,230,200	4	252.95	34.56	15.17
	20,300,40	4	0.17	0.33	0.24
	20,300,200	4	303.24	30.35	12.11
	30,520,100	4	3,181.33	172.15	305.08
	30,520,400	4	95.46	20.61	9.14
	30,700,100	4	71.61	19.56	8.86
	30,700,400	4	10,479.58	550.03	332.35
	Sub-Total	**31**	**1,856.05**	**106.80**	**88.14**
III-B	20,120,40	3	0.05	0.10	0.11
	20,120,100	3	13.42	5.56	1.97
	20,120,200	3	361.23	245.47	214.59
	20,220,40	3	6.91	6.81	2.00
	20,220,100	3	153.86	54.32	39.81
	20,220,200	3	1,615.31	1,396.17	775.17
	20,320,40	3	27.79	36.06	11.44
	20,320,100	3	69.25	29.36	6.87
	20,320,200	3	2,592.58	435.30	625.82
	Sub-Total	**27**	**537.82**	**245.46**	**186.42**
IV	40,1200,400	1	59.82	19.95	9.19
	40,1200,800	1	4,483.75	141.31	53.00
	40,1200,1200	1	1,664.10	110.68	57.07
	50,1400,400	1	575.91	68.06	29.94
	50,1400,800	1	39,051.12	296.61	117.31
	50,1400,1200	1	57071.73	500.18	201.33
	50,1500,1000	1	time	time	54,074.52
	50,1500,1500	1	time	time	69,637.47
	Sub-Total	**6**	**17,151.07**	**189.47**	**77.97**
	Total	**64**	**2,733.83**	**173.05**	**128.65**

This is surprising because unlike solving P with CPLEX, our branch-and-Benders-cut algorithm does not explicitly have a complete description of the problem's polytope

but must instead estimate it on the fly by generating Benders cuts. This leads to the possibility of underestimating the solution of the underlying linear programs at each node of the enumeration tree, leading to weak dual bounds. This is more likely to occur in $BB\&C$ since we only allow for one round of Benders cuts to be added at non-root nodes of our enumeration tree. However, due to the enhancements proposed, we leave the root node with a linear program that captures most of the important characteristics in a smaller problem.

On the other hand, our modified cut-and-solve algorithm's performance is also two orders of magnitude faster than CPLEX for large-scale instances while for Class III-B, it saves over 50% of the solution time. On average, CS/LB solves five sparse problems before proving optimality of its obtained solution. Each of these sparse problems are solved up to three orders of magnitude faster than solving the complete problem with CPLEX and sometimes in half the time than if solved with branch-and-Benders-cut algorithm. It is because of these time savings that it outperforms CPLEX in all instances and our branch-and-Benders-cut algorithm in instance group 20,320,200 of class III-B. The advantage of this method is that it finds the optimal solution early on and spends the rest of the time proving optimality by solving another sparse problem followed by the remaining dense problem.

Finally, we point out that while dimensionality does play a role in the computation time required to solve these instances, there exist other factors that contribute to the difficulty of these problems. This can be seen in the difference in solution time between the instance group 30,520,100 and 30,520,400 of class I where the group with four times more commodities is solved in significantly less computing time. The same is seen when comparing differences in number of arcs. Instance group 30,700,100 (class I) requires significantly less computing time than the instance group 30,520,100 (class I) which has less arcs. Identifying the other factors that make some network design instances particularly difficult for mixed integer programs would allow researchers to devise algorithms with an improved, more stable performance.

7. Conclusion

We have presented two exact solution algorithms for the multicommodity uncapacitated fixed-charge network design problem that significantly outperform the state-of-the-art general-purpose MIP solver CPLEX. The first exact algorithm is based on implementing Benders decomposition within a branch-and-cut framework using Pareto-optimal cuts, appropriate core point selection, and an in-tree matheuristic. These additional refinements also serve as general guidelines for implementing this algorithm for other mixed integer problems.

We present a novel strategy for improving the LP bound of our Benders reformulation by means of Benders lift-and-project cuts applied to the master problem's feasibility and optimality cuts. These are obtained using a modified cut generating linear program that takes less than 0.02 seconds to solve. This procedure extends beyond the MUFND and can be applied to all problems that allow a mixed integer programming formulation and

corresponding Benders reformulation.

Finally, we present a strategy that combines ideas from cut-and-solve/local branching and our proposed branch-and-Benders-cut algorithm. The advantages of this method are: breaking down the problem into a few sparse MIPs which make it easier to obtain high quality feasible solutions, the non-increasing optimal values obtained from the sparse problems, the reduced size of the sparse problem solution space, and the re-usability of Benders cuts generated in previous iterations. The results of our implementation show this fusion to be a promising method for solving large-scale MIPs.

Acknowledgments: This research was partially funded by the Canadian Natural Sciences and Engineering Research Council [Grants 418609-2012 and 04959-2014]. This support is gratefully acknowledged.

References

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

Andersen J, Crainic TG, Christiansen M (2009) Service network design with management and coordination of multiple fleets. *European Journal of Operational Research* 193(2):377–389.

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

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

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

Balakrishnan N, Wong RT (1990) A network model for the rotating workforce scheduling problem. *Networks* 20(1):25–42.

Balas E (1979) Disjunctive programming. *Annals of Discrete Mathematics* 5:3–51.

Balas E, Ceria S, Cornuéjols G (1993) A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical Programming* 58(1):295–324.

Balas E, Ceria S, Cornuéjols G (1996) Mixed 0–1 programming by lift-and-project in a branch-and-cut framework. *Management Science* 42(9):1229–1246.

Balas E, Padberg MW (1972) On the set-covering problem. *Operations Research* 20(6):1152–1161.

Balas E, Perregaard M (2002) Lift-and-project for mixed 0–1 programming: recent progress. *Discrete Applied Mathematics* 123(1):129–154.

Bartholdi JJ, Orlin JB, Ratliff HD (1980) Cyclic scheduling via integer programs with circular ones. *Operations Research* 28(5):1074–1085.

Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4:238–252.

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

Billheimer J, Gray P (1973) Network design with fixed and variable cost elements. *Transportation Science* 7(1):49–74.

Bodur M, Dash S, Günlük O, Luedtke J (2017) Strengthened Benders cuts for stochastic integer programs with continuous recourse. *INFORMS Journal on Computing* 29(1):77–91.

Bodur M, Luedtke JR (2017) Mixed-integer rounding enhanced Benders decomposition for multiclass service-system staffing and scheduling with arrival rate uncertainty. *Management Science* 63(7):2073–2091.

Boffey T, Hinxman A (1979) Solving the optimal network problem. *European Journal of Operational Research* 3(5):386–393.

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

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

Climer S, Zhang W (2006) Cut-and-solve: An iterative search strategy for combinatorial optimization problems. *Artificial Intelligence* 170(8):714–738.

Contreras I, Cordeau JF, Laporte G (2011) Benders decomposition for large-scale uncapacitated hub location. *Operations research* 59(6):1477–1490.

Contreras I, Díaz JA, Fernández E (2009) Lagrangean relaxation for the capacitated hub location problem with single assignment. *OR Spectrum* 31(3):483–505.

Cordeau JF, Pasin F, Solomon MM (2006) An integrated model for logistics network design. *Annals of Operations Research* 144(1):59–82.

Cordeau JF, Stojković G, Soumis F, Desrosiers J (2001) Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science* 35(4):375–388.

Costa AM (2005) A survey on Benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research* 32(6):1429–1450.

Costa AM, Cordeau JF, Gendron B (2009) Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications* 42(3):371–392.

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

Crainic TG, Frangioni A, Gendron B (2001) Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics* 112(13):73–99.

Crainic TG, Gendron B, Hernu G (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 TG, Rousseau JM (1986) Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem. *Transportation Research Part B: Methodological* 20(3):225–242.

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

Dionne R, Florian M (1979) Exact and approximate algorithms for optimal network design. *Networks* 9(1):37–59.

Edmonds J, Karp RM (1972) Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the Association for Computing Machinery* 19(2):248–264.

Fischetti M, Ljubić I, Sinnl M (2017) Redesigning Benders decomposition for large-scale facility location. *Management Science* 63(7):2146–2162.

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

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

Fontaine P, Minner S (2018) Benders decomposition for the hazmat transport network design problem. *European Journal of Operational Research* 267(3):996–1002.

Fortz B, Poss M (2009) An improved Benders decomposition applied to a multi-layer network design problem. *Operations Research Letters* 37(5):359–364.

Fragkos I, Cordeau JF, Jans R (2017) The multi-period multi-commodity network design problem. Technical Report CIRRELT 2017-63, Université de Montréal.

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

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

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

Gadegaard SL, Klose A, Nielsen LR (2018) An improved cut-and-solve algorithm for the single-source capacitated facility location problem. *EURO Journal on Computational Optimization* 6(1):1–27.

Gendron B, Hanafi S, Todosijevi R (2018) Matheuristics based on iterative linear programming and slope scaling for multicommodity capacitated fixed charge network design. *European Journal of Operational Research* 268(1):70–81.

Geoffrion AM, Graves GW (1974) Multicommodity distribution system design by Benders decomposition. *Management Science* 26(8):855–856.

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

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

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

Hellstrand J, Larsson T, Migdalas A (1992) A characterization of the uncapacitated network design polytope. *Operations Research Letters* 12(3):159–163.

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

Holmberg K, Hellstrand J (1998) Solving the uncapacitated network design problem by a Lagrangean heuristic and branch-and-bound. *Operations Research* 46(2):247–259.

Jeihoonian M, Zanjani MK, Gendreau M (2016) Accelerating Benders decomposition for closed-loop supply chain network design: Case of used durable products with different quality levels. *European Journal of Operational Research* 251(3):830–845.

Johnson DS, Lenstra JK, Kan AHGR (1978) The complexity of the network design problem. *Networks* 8(4):279–285.

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

Keyvanshokooh E, Ryan SM, Kabir E (2016) Hybrid robust and stochastic optimization for closed-loop supply chain network design using accelerated Benders decomposition. *European Journal of Operational Research* 249(1):76–92.

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

Kim D, Pardalos PM (2000) Dynamic slope scaling and trust interval techniques for solving concave piecewise linear network flow problems. *Networks* 35(3):216–222.

Kratica J, Tošić D, Filipović V, Ljubić I (2002) A genetic algorithm for the uncapacitated network design problem. Roy R, Köppen M, Ovaska S, Furuhashi T, Hoffmann F, eds., *Soft Computing and Industry: Recent Applications*, 329–336.

Lamar BW, Sheffi Y, Powell WB (1990) A capacity improvement lower bound for fixed charge network design problems. *Operations Research* 38(4):704–710.

Lee C, Lee K, Park S (2013) Benders decomposition approach for the robust network design problem with flow bifurcations. *Networks* 62(1):1–16.

Los M, Lardinois C (1982) Combinatorial programming, statistical optimization and the optimal transportation network problem. *Transportation Research Part B: Methodological* 16(2):89–124.

Magnanti T, Mireault P, Wong R (1986) Tailoring Benders decomposition for uncapacitated network design. Gallo G, Sandi C, eds., *Network Flow at Pisa*, 112–154, Mathematical Programming Studies, volume 26.

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

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

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

Munguía LM, Ahmed S, Bader DA, Nemhauser GL, Goel V, Shao Y (2017) A parallel local search framework for the fixed-charge multicommodity network flow problem. *Computers & Operations Research* 77:44–57.

Naoum-Sawaya J, Elhedhli S (2013) An interior-point Benders based branch-and-cut algorithm for mixed integer programs. *Annals of Operations Research* 210(1):33–55.

Ortega F, Wolsey L (2003) A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks* 41(3):143–158.

Ortiz-Astorquiza C, Contreras I, Laporte G (2017) An exact algorithm for multi-level uncapacitated facility location. Forthcoming in Transportation Science.

Padberg M, Rinaldi G (1991) A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review* 33(1):60–100.

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

Papadakos N (2009) Integrated airline scheduling. *Computers & Operations Research* 36(1):176–195.

Paraskevopoulos DC, Bekta T, Crainic TG, Potts CN (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.

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

Rahmaniani R, Crainic TG, Gendreau M, Rei W (2017) The Benders decomposition algorithm: A literature review. *European Journal of Operational Research* 259(3):801–817.

Rahmaniani R, Crainic TG, Gendreau M, Rei W (2018) Accelerating the Benders decomposition method: Application to stochastic network design problems. *SIAM Journal on Optimization* 28(1):875–903.

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

Rei W, Cordeau JF, Gendreau M, Soriano P (2009) Accelerating benders decomposition by local branching. *INFORMS Journal on Computing* 21(2):333–345.

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

Santoso T, Ahmed S, Goetschalckx M, Shapiro A (2005) A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research* 167(1):96–115.

van Ackooij W, Frangioni A, de Oliveira W (2016) Inexact stabilized Benders' decomposition approaches with application to chance-constrained problems with finite support. *Computational Optimization and Applications* 65(3):637–669.

Yaghini M, Karimi M, Rahbar M, Sharifitabar MH (2015) A cutting-plane neighborhood structure for fixed-charge capacitated multicommodity network design problem. *INFORMS Journal on Computing* 27(1):48–58.

Yang Z, Chu F, Chen H (2012) A cut-and-solve based algorithm for the single-source capacitated facility location problem. *European Journal of Operational Research* 221(3):521–532.

Zarrinpoor N, Fallahnezhad MS, Pishvaee MS (2018) The design of a reliable and robust hierarchical health service network using an accelerated Benders decomposition algorithm. *European Journal of Operational Research* 265(3):1013–1032.

Zetina CA, Contreras I, Fernández E, Luna-Mota C (2018) Solving the optimum communication spanning tree problem. Forthcoming in European Journal of Operational Research.