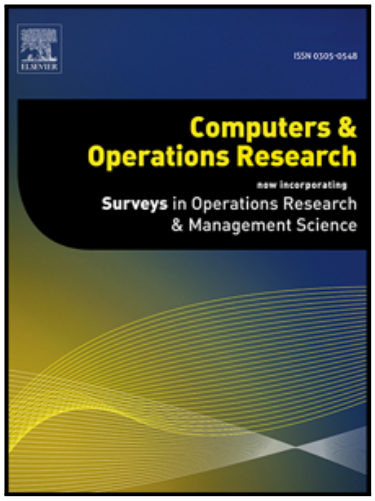
Accepted Manuscript



Exact Algorithms based on Benders Decomposition for Multicommmodity Uncapacitated Fixed-charge Network Design

Carlos Armando Zetina, Ivan Contreras, Jean-François Cordeau

PII: S0305-0548(19)30183-2
DOI: https://doi.org/10.1016/j.cor.2019.07.007
Reference: CAOR 4749

To appear in: *Computers and Operations Research*

Received date: 3 October 2018
Revised date: 11 July 2019
Accepted date: 12 July 2019

Please cite this article as: Carlos Armando Zetina, Ivan Contreras, Jean-François Cordeau, Exact Algorithms based on Benders Decomposition for Multicommmodity Uncapacitated Fixed-charge Network Design, *Computers and Operations Research* (2019), doi: https://doi.org/10.1016/j.cor.2019.07.007

Highlights

- We combine mixed integer programming tools such as Benders decomposition, branch-and-cut, lift-and-project cuts, matheuristics, cut-and-solve, and the simultaneous exploitation of two formulations to create computationally efficient exact algorithms to solve a general network design problem.
- We present a tailor-made rule for corepoint selection to obtain Pareto optimal Benders cuts that leads to significant computational time savings.
- We propose additional enhancements that improve the solution time of the two Benders decomposition-based exact algorithms presented.

Exact Algorithms based on Benders Decomposition for Multicommmodity Uncapacitated Fixed-charge Network Design

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

[a] *Canada Excellence Research Chair in Data Science for Real-time Decision-making, Polytechnique Montréal, Canada*
[b] *Concordia University and Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montréal, Canada*
[c] *HEC Montréal and Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montréal, Canada*

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, valid inequalities in the projected master problem space to close the linear programming gap, and a tailored core point selection criterion from which significant time savings are obtained. 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 speed-up of up to three orders of magnitude compared to a state-of-the-art general-purpose MIP solver's branch-and-cut and blackbox Benders decomposition algorithms.

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

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

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) 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).

The first proposed algorithm to solve MUFND 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. As the purpose of this paper is not to compete against these state-of-the-art specialized algorithms but instead to revisit Benders decomposition, we do not reimplement these algorithms for our computational experiments.

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) and hub network design (Contreras et al. 2011, de Camargo et al. 2008). It has also 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 been proven effective in solving fundamental network design problems such as the optimum communi-

cation spanning tree problem (Zetina et al. 2019) and extensions such as the multi-layer (Fortz and Poss 2009), hop-constrained (Botton et al. 2013), survivable (Gouveia et al. 2018) 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 are local access network design (Randazzo and Luna 2001), transit network design (Marín and Jaramillo 2009), cable trench problems (Calik et al. 2017), and other telecommunication problems (Gzara and Erkut 2011). A survey on the application of Benders decomposition to fixed-charge network design problems can be found in Costa (2005) while Rahmaniani et al. (2017) review its use in general optimization problems.

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.

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. In addition, we propose a tailored core point selection criterion that alone provides a significant speed-up for solving the MUFND. To the best of our knowledge, this is the first study that explicitly shows the impact different core point selection strategies have on the overall solution process, thus highlighting its importance when using Pareto-optimal Benders cuts which have become a common practice in the literature.

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 cut frequency and core point selection for Pareto-optimal cuts. In addition, we present computational evidence of the effect that parameter selection has on its performance.

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 problems with reduced feasible spaces 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's branch-and-cut algorithm and solves instances of larger size than those previously presented. The computational experiments also show that CPLEX's blackbox Benders algorithm performs significantly worse than its branch-and-cut, thus showing the importance of the refinements to the Benders decomposition algorithm proposed in the current paper. 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, the use of a tailored core point selection strategy, and the combination 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 of the arc-based formulation 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 Benders cut-and-solve 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 arc-based formulations with either *aggregated* or *disaggregated* linking constraints (Magnanti and Wong 1984,

Ahuja et al. 1993). 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 formulation with disaggregated linking constraints since its tighter linear programming (LP) relaxation is preferred when applying Benders decomposition (Magnanti and Wong 1981). The MUFND is thus formulated as follows:

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

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

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

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

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

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) contains the linking constraints that assure 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, commodities are sent between every pair of nodes, and transportation costs are commodity independent) (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). In its generic version, large formulations are split into two problems, a mixed 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 may contain 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

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

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

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

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

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

$$z \in \mathbb{R}^{|K|} \tag{12}$$

$$y \in \{0,1\}^{|A|}. \tag{13}$$

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 reduce 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). In our case, these inequalities must enforce that for a given set of arcs that form a cut between two nodes $o, d \in N$, their sum over the design variable values, y_{ij}, must be at least one, thus ensuring that the set of arcs in the cut allows at least one unit of flow to be transported from o to d. Let $\delta(S) = \{(i,j) \in A | i \in S, j \in N \backslash S\}$ and Δ_K

is the set of subsets $S \subset N$ such that $o_k \in S$ and $d_k \notin S$ for some $k \in K$. Using this notation, we use the following cutset inequalities for MUFND:

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

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 solving these problems and thus for separating cutset inequalities for fractional and integer solutions, respectively. With this in mind, we substitute the use of Benders feasibility cuts (11) with cutset inequalities (14) yielding the following final Benders reformulation:

$$\text{(MP)} \quad \text{minimize} \sum_{(i,j)\in A} f_{ij} y_{ij} + \sum_{k\in K} z_k \tag{15}$$

$$\text{subject to} \quad (10), (12)(13), (14). \tag{16}$$

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

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. A recent common practice 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 (Bodur and Luedtke 2017, Bodur et al. 2017, Fischetti et al. 2017). General-purpose solvers use this information to tighten variable 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 immediately obtain the LP optimal value at the root node of the branch-and-cut tree but pass on only the essential information to the general-purpose solver and avoid declaring an excessively large problem. In addition, these cuts allow general-purpose solvers to generate valid inequalities such as mixed-integer cuts, Gomory cuts, etc. to improve the linear programming relaxation of the formulation. This step, while simple, 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) which has been shown to significantly accelerate the convergence of the algorithm by requiring fewer Benders cuts (Magnanti et al. 1986, Zetina et al. 2019). In the case of the MUFND, Magnanti et al. (1986) 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*:

$$(\text{MCF}_k) \quad \text{minimize} \sum_{(i,j)\in A} W^k c_{ij}^k x_{ij}^k - DSP_k(\bar{y})x_0 \tag{17}$$

$$\text{subject to} \quad \sum_{j\in N} x_{ji}^k - \sum_{j\in N} x_{ij}^k = \begin{cases} -(1+x_0) & \text{if } i = o_k \\ 1+x_0 & \text{if } i = d_k \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in N \tag{18}$$

$$x_{ij}^k \leq y_{ij}^0 + x_0 \bar{y}_{ij} \qquad \forall (i,j) \in A \tag{19}$$

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

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

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 (18) and (19), respectively (Magnanti et al. 1986). The authors show that any fixed value $x_0 \geq \sum_{(i,j)\in A} y^0_{ij}$ 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 6.3.

4.3. Benders lift-and-project cuts

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). Further details and background on the evolution of lift-and-project cuts can be found in Balas and Perregaard (2002) and the references therein. The underlying idea of lift-and-project cuts is to find inequalities that are valid for the union of two polyhedra defined by the two possible values a binary variable can take. We adopt these inequalities to strengthen the master problem LP relaxation of the Benders reformulation.

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 of the form $\alpha x \geq \beta$ can be obtained by solving the following linear program (Balas 1979) for a selected $\hat{\jmath} \in \{1...p\}$ such that $\bar{x}_{\hat{\jmath}}$ is fractional:

$$
\begin{aligned}
(\text{CGLP}_{\hat{\jmath}}) \quad & \text{minimize } \alpha\bar{x} - \beta \\
\text{subject to} \quad & \alpha - u\tilde{A} + u^0 e_{\hat{\jmath}} \geq 0 \\
& \alpha - v\tilde{A} - v^0 e_{\hat{\jmath}} \geq 0 \\
& -\beta + ub = 0 \\
& -\beta + vb + v_0 = 0 \\
& u, v \geq 0,
\end{aligned}
$$

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. We use a normalization constraint proposed after the Balas et al. (1996) paper since, as pointed out in Bonami (2012), it has become the most commonly used due to its favourable performance. The normalization constraint used has the following form:

$$\sum_{i=1}^{m'} u_i + u^0 + \sum_{i=1}^{m'} v_i + v^0 = 1.$$

where m' is the total number of constraints in $\tilde{A}$. The resulting cut is then strengthened as in (Balas and Perregaard 2002) by replacing the coefficients $\alpha'_k = \min\{ua_k + u^0\lceil m_k \rceil, va_k - v^0\lfloor m_k \rfloor\}$, where $m_k = \frac{va_k - ua_k}{u^0 + v^0}$.

While solving the Benders reformulation, we do not have the complete polyhedral description of P since there would be exponentially many constraints. We thus define $\tilde{A}$ as the feasibility and optimality cuts that are binding at the LP relaxation and the $1 \geq x \geq 0$ constraints thus obtaining weakened level-1 lift-and-project cuts of MP. 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 variable $x_{\hat{\jmath}}$. In our procedure, the integer variables y_{ij} of an LP optimal solution $(\bar{y}, \bar{z})$ of MP are ordered based on the following weight: $LW_{ij} = (1 - |\bar{y}_{ij} - 0.5|)f_{ij}$. The weight LW considers both the variable's fractionality and its contribution to the objective function and thus provides a reasonable measure to rank candidates for disjunction where larger values of LW are more favourable candidates.

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:

$$(P_{Heur}) \quad \text{minimize} \sum_{(i,j)\in A} f_{ij}y_{ij} + \sum_{k\in K}\sum_{\mu\in\Omega_k}[W^k \sum_{(i,j)\in A} c_{ij}^k v_k^\mu(i,j)]\Theta_k^\mu \tag{20}$$

$$\text{subject to} \quad \sum_{\mu\in\Omega_k} \Theta_k^\mu = 1 \qquad \forall k \in K \tag{21}$$

$$\sum_{\mu\in\Omega_k} v_k^\mu(i,j)\Theta_k^\mu \leq y_{ij} \qquad \forall (i,j) \in A, k \in K \tag{22}$$

$$\Theta_k^\mu \in \{0,1\} \qquad \forall k \in K, \mu \in \Omega_k. \tag{23}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A. \tag{24}$$

The objective function (20) represents the total cost of routing the commodities on the network, including design and transportation costs. Constraints (21) ensure that each commodity is routed by exactly one path while (22) 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 (22) can be disaggregated. However, preliminary tests showed it to have a negative effect on the computation time.

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.

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.

In our implementation, we begin by first solving the LP relaxation of MP as described in Section 4.1. The cut-and-solve algorithm then begins by considering the union of shortest paths of each commodity, denoted as $\cup_{k \in K} P_k$, obtained over the network with an arc cost $\tilde{c}^k_{ij}$ that considers the transportation cost, fixed-charge, demand quantity, and the LP relaxation solution. We define this arc cost as $\tilde{c}^k_{ij} = W^k c^k_{ij} + (1 - \hat{y}_{ij}) f_{ij}$, where $\hat{y}_{ij}$ represents the value of y_{ij} at the optimal LP solution. The reason that these modified costs work well is because they use the LP solution as a predictor of which arcs are likely to be part of the optimal solution and thus consider a lower additional fixed cost for arcs with an LP solution value closer to one so as to promote its use when finding the shortest paths of each commodity.

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, $\chi((\bar{y}, \bar{z})_0)$ refers to the objective function value of activating and routing according to the initial solution previously described.

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

$$(\text{MP}_{sparse}(t)) \quad \text{minimize} \sum_{(i,j)\in A} f_{ij}y_{ij} + \sum_{k\in K} z_k \tag{25}$$

$$\text{subject to} \quad z_k \geq \lambda^k_{d_k} - \lambda^k_{o_k} - \sum_{(i,j)\in A} \mu^k_{ij} y_{ij} \qquad \forall (\lambda,\mu)_k \in Opt(DSP_k), k \in K \tag{26}$$

$$\sum_{(i,j)\in\delta(S)} y_{ij} \geq 1 \qquad \forall S \in \Delta_K \tag{27}$$

$$z \in \mathbb{R}^{|K|} \tag{28}$$

$$y \in \{0,1\}^{|A|} \tag{29}$$

$$\sum_{(i,j)\notin I(t-1)} y_{ij} \leq t \tag{30}$$

$$\sum_{(i,j)\notin I(s)} y_{ij} \geq s+2 \qquad \forall s = 0,...,t-1 \tag{31}$$

$$\sum_{(i,j)\in A} f_{ij}y_{ij} + \sum_{k\in K} z_k \leq \chi((\bar{y},\bar{z})_{t-1}), \tag{32}$$

where $\Delta_k = \{S \subset N | \exists k \in K \text{ where } o_k \in S, d_k \notin S\}$. The constraints (25)-(29) are the Benders master problem reformulation of the MUFND. Constraint (30) is the piercing cut that allows at most t variables not in the previous solution to take the value of 1 while constraints (31) are the negations of (30) from previous iterations. The latter ensure that previously searched areas of the feasible space are not considered in the new sparse problem. Finally, constraint (32) 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 $\text{MP}_{sparse}(t)$ is feasible, we define $\text{MP}_{sparse}(t+1)$ 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 $\text{MP}_{sparse}(t)$ are the same or until $\text{MP}_{sparse}(t)$ is infeasible, due to (32). If either occurs, the dense problem, MP_{dense}, is defined. The dense problem is similar to $\text{MP}_{sparse}(t)$ with the exception that (30) is replaced by $\sum_{(i,j)\notin I(t-1)} y_{ij} \geq t+1$ and $(\bar{y},\bar{z})_{t-1}$ in (32) 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. The testbed can be divided into three classes. The first are the 31 "C" capacitated instances with many commodities compared to nodes. Each capacitated instance in this class represents a unique combination of commodity flows, fixed costs, and per unit routing costs. Thus, all 31 instances remain in our testbed after ignoring information on arc capacities. The second are the 12 "C+" capacitated instances with few commodities compared to the number of nodes. Out of these 12, only eight represented unique combinations of commodity flows, fixed costs, and per unit routing costs. Finally, Class III is divided into two subgroups: Class III-A and III-B are each comprised of 81 "R" capacitated instances on small and medium sized graphs, respectively. These instances consist of three levels of capacity tightness for each unique combination of commodity flows, fixed costs, and per unit routing costs, thus leading to 27 uncapacitated instances from each subgroup in our MUFND testbed.

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 whose integer optimal solution is not obtained by solving the linear programming (LP) relaxation, i.e., the instance has an LP gap strictly greater than 0 where the LP gap is defined as $LPgap = 100 \times \frac{(Opt-LP)}{Opt}$ where Opt is the objective function value of an integer optimal solution and LP is the objective function value at an optimal solution of the LP relaxation. 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 presents the number of instances in our testbed whose LP gap (%) falls within the ranges indicated as column headings. For example, 6 instances of Class I have an LP gap between 4 and 7.2%.

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. The codes and instances of the final versions of the presented algorithms are available at *https://sites.google.com/view/carloszetina/Research/Publications*.

6.1. Implementation details of the branch-and-Benders-cut algorithm

As with any ad hoc exact algorithm, implementation details play a key role in its performance. In this section, we outline the sequence of steps taken during the algorithm and present preliminary experiments that justify the choice of parameters used. Our solution procedure follows the framework of the recent branch-and-Benders cut implementations of Fischetti et al. (2016, 2017), Bodur and Luedtke (2017) and Bodur et al. (2017).

The solution process begins by solving the LP relaxation of MP using the Edmonds-Karp minimum cut algorithm as a cutset separation routine and MCF_k as our separation oracle to obtain Pareto-optimal Benders cuts with core points defined as will be described in Section 6.3. Upon confirming that no more violated Benders cuts exist, we use those that are binding at the LP optimal solution as the partial polyhedral description used to obtain level-1 Benders lift-and-project cuts. The obtained cuts are added to the MP relaxation and we resume separating Benders cuts until no more violated Benders cuts are obtained.

We then define the MIP problem in CPLEX with the binding Benders and lift-and-project cuts as lazy and user constraints. This prevents defining an excessively large initial problem. An important aspect 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.

To aid our decision of selecting a reasonable cutting and separation frequency we tested three different strategies: Infrequent, Moderate, and Aggressive. The infrequent strategy attempts to separate Benders cuts at all nodes of depth at or below 3 and every 120th node thereafter, while the moderate configuration does so for all nodes of depth at or below 5 and every 100th node thereafter, and finally, the aggressive configuration does so for all nodes of depth at or below 10 and every 80th node thereafter. For all configurations, at most one round of separating Benders cuts is done at each node and only cuts with a violation greater than 10^{-5} are added to the problem. Table 2 shows the number of nodes explored, Benders cuts added throughout the process, and finally the solution time in seconds for running the branch-and-Benders-cut algorithm with each configuration on ten instances in the 70th to 80th percentile of difficulty measured by solution time. In other words, we first solved all instances using CPLEX's branch-and-cut, ordered them with respect to time taken to solve, and took the ten instances for which the solution times were higher than 70 per cent of the testbed and lower than the 20 percent worst performing.

Table 2: Impact of cutting frequency

	Infrequent			**Moderate**			**Aggressive**		
$\|N\|, \|A\|, \|K\|$	**Nodes**	**Cuts**	**Secs.**	**Nodes**	**Cuts**	**Secs.**	**Nodes**	**Cuts**	**Secs.**
20,230,200	619	3,926	35.03	500	5,837	41.15	262	9,642	57.31
30,520,100	13,539	7,046	1,175.18	12,647	6,527	1,290.56	6,761	15,493	1,947.48
30,700,400	17,083	8,650	1,315.70	14,546	10,536	1,305.84	10,322	11,785	1,307.17
30,700,400	3,887	4,643	106.42	3,570	4,342	96.93	2,846	5,336	198.92
20,120,200	7,240	12,372	1,036.07	5,290	12,692	698.79	1,391	30,062	810.04
20,220,100	2,913	4,928	163.25	1,908	5,489	125.42	927	17,534	262.48
20,220,200	2,963	8,551	393.86	2,055	9,194	279.43	1,256	26,174	823.98
20,320,200	1,069	7,191	112.02	594	9,056	133.65	142	12,292	62.16
50,1400,800	2,204	4,592	132.61	1,938	4,720	132.53	1,407	4,679	217.73
50,1400,1200	2,632	9,273	246.59	2,162	8,944	220.44	1,590	8,607	455.74
Avg.	**5,415**	**7,117**	**471.67**	**4,521**	**7,734**	**432.47**	**2,690**	**14,160**	**614.30**

We note that there is a strong correlation between the number of cuts added and the number of nodes explored. Adding more cuts leads to exploring fewer nodes in the solution process. From these experiments, we note that the best trade-off lies in the moderate cutting frequency since it requires on average the least solution time. We thus select this strategy in our branch-and-Benders implementation.

Lastly, to prevent executing our in-tree matheuristic too frequently, we limit its use to

every 100th node. In addition, to ensure it has the potential to find an improved solution, it is only executed 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.

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

In theory, one is able to converge to the optimal IP solution by adding a finite number of lift-and-project cuts. However, in practice, this is far from reasonable as the number of cuts, while finite, may be exponential in the size of the problem. As a result, we must set a limit to the number of lift-and-project cuts that will be added to the master problem. To aid this decision, we run experiments over the ten instances previously studied.

With the cutting and separation frequency strategy fixed to moderate, we again define three strategies for adding Benders lift-and-project cuts: Mild, Moderate, and Aggressive. The mild configuration adds lift-and-project cuts by using the top 10%, according to LW, of the candidate disjunction variables $\bar{x}_{\hat{j}}$ to define $CGLP_{\hat{j}}$, while the moderate strategy uses the top 20%, and finally the aggressive strategy uses the top 50% of the candidate disjunction variables.

An important indicator of the effectiveness of cutting planes is the percentage of the LP gap closed by adding them to the linear relaxation. This percentage is calculated as $100 \times \frac{(LP_{MPLP} - LP_{MP})}{Opt - LP_{MP}}$, 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.

Table 3 shows for each strategy the number of lift-and-project cuts added (LiftP cuts), the percentage of the LP gap closed at the root node (% gap closed), and the time in seconds needed to solve each of the instances to proven optimality (Time (secs.)).

Table 3: Impact of Lift-and-Project cut strategy on solution time

	Mild			**Moderate**			**Aggressive**		
$\|N\|, \|A\|, \|K\|$	**LiftP cuts**	**% gap closed**	**Time (secs.)**	**LiftP cuts**	**% gap closed**	**Time (secs.)**	**LiftP cuts**	**% gap closed**	**Time (secs.)**
20,230,200	7	2.82	47.07	15	3.15	53.93	39	4.51	67.69
30,520,100	13	3.99	2,796.10	27	4.95	1,259.13	68	5.19	1,592.10
30,700,400	11	2.49	1,380.54	23	3.10	1,197.52	59	5.00	1,539.78
30,700,400	10	4.75	113.69	20	5.98	165.28	50	6.89	104.19
20,120,200	7	0.53	627.74	15	0.68	634.80	38	0.84	596.36
20,220,100	8	1.77	100.10	16	1.94	123.59	40	2.28	136.28
20,220,200	8	0.72	309.23	16	0.77	126.75	40	1.11	248.26
20,320,200	10	1.34	72.01	20	1.56	199.00	51	1.54	346.40
50,1400,800	18	8.04	170.76	36	12.71	380.44	90	19.12	660.29
50,1400,1200	19	10.84	416.25	39	15.09	631.00	99	23.69	971.07
Avg.	**11**	**3.73**	**603.35**	**23**	**4.99**	**477.14**	**57**	**7.02**	**626.24**

While a more aggressive Benders lift-and-project cutting strategy closes a larger percentage of the LP gap, after a certain point it has an adverse effect on the overall solution time of the branch-and-Benders-cut algorithm. As a result, we implement the moderate cutting strategy in the remainder of our computational experiments as it provides the best trade-off between the percentage of LP gap closed and overall solution time. Table 4 shows the percentage of LP gap closed over the entire testbed.

Table 4: LP gap (%) closed

Class	**No. of instances**	**% gap closed**
Class I	21	12.57
Class II	1	15.39
Class III-A	1	21.06
Class III-B	18	4.05
Class IV	8	17.13
Total	**49**	**10.42**

6.3. 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 $k \in K$. 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 when considering only transportation costs, 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 $(i,j) \notin \cup_{k\in K} P_k$.

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 $\delta(S)_k = \{(i,j) \in A | o_k \in S \text{ and } d_k \in N \backslash S\}$ 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.

Table 5 details for each strategy the number of explored nodes, number of Benders cuts added and solution time on the ten instances used in our preliminary tests. In addition, it presents the same information for a more straightforward Benders implementation that does not use Pareto-optimal cuts (Benders no POCs).

Table 5: Impact of core point selection

	Benders no POCs			**Strategy 1**			**Strategy 2**			**Strategy 3**		
$\|N\|,\|A\|,\|K\|$	**Nodes**	**Cuts**	**Secs.**	**Nodes**	**Cuts**	**Secs.**	**Nodes**	**Cuts**	**Secs.**	**Nodes**	**Cuts**	**Secs.**
20,230,200	849	14,015	223.27	423	6,728	51.31	723	6,543	69.20	585	4,814	32.35
30,520,100	16,570	12,793	5,078.79	12,217	7,434	1,306.51	16,540	10,427	3,384.50	15,398	6,928	1,093.01
30,700,400	36,597	32,030	32,781.59	17,204	10,178	1,938.39	17,153	16,548	4,746.47	11,890	10,573	833.49
30,700,400	5,310	12,393	1,283.89	3,525	5,172	156.53	4,069	7,025	316.38	3,682	4,953	105.81
20,120,200	8,149	18,985	2,159.13	3,182	12,507	440.36	5,038	15,072	1,003.64	3,755	10,297	326.92
20,220,100	4,248	10,417	672.93	1,665	6,419	108.25	2,998	7,934	287.84	1,925	6,098	109.10
20,220,200	3,657	15,482	1,135.57	1,134	10,338	177.87	3,810	11,104	702.86	1,962	9,418	270.62
20,320,200	847	10,455	239.21	757	6,627	88.11	306	12,935	84.57	618	9,690	127.87
50,1400,800	4,059	12,186	2,262.91	3,043	6,541	191.47	3,196	6,166	270.67	2,911	3,672	92.05
50,1400,1200	5,025	18,355	3,872.63	2,576	9,372	309.76	3,853	14,700	767.25	1,751	8,155	171.26
Avg.	**8,531**	**15,711**	**4,970.99**	**4,573**	**8,132**	**476.86**	**5,769**	**10,845**	**1,163.34**	**4,448**	**7,460**	**316.25**

As seen in Table 5, there is a significant computational gain of up to an order of magnitude when using Pareto-optimal Benders cuts. This comes as a result of the need to add fewer Benders cuts and exploring fewer nodes in the enumeration tree. Among the corepoint updating strategies, 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 not only using Pareto-optimal Benders cuts, but also core point selection strategies that exploit problem structure.

6.4. 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 (Ben). The second is the same, with the addition of the in-tree heuristic (Ben+H). Ben+LP is the branch-and-Benders-cut algorithm with lift-and-project cuts added at the root node and the final version (Ben+H+LP) combines them all. A time limit of 24 hours is set for all algorithms.

The results are presented in Table 6 except 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,"Secs." 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 testbed 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 6: Computational performance of branch-and-Benders-cut algorithm

			Ben		**Ben+H**		**Ben+LP**		**Ben+H+LP**	
Class	$\|N\|, \|A\|, \|K\|$	**Nb**	**Secs.**	**Nodes**	**Secs.**	**Nodes**	**Secs.**	**Nodes**	**Secs.**	**Nodes**
	20,230,40	3	0.20	0	0.21	0	0.22	0	0.22	0
	20,230,200	4	23.83	807	16.10	274	26.02	345	28.15	358
	20,300,40	4	0.24	0	0.24	0	0.27	1	0.28	1
I	20,300,200	4	22.36	677	13.74	201	26.41	328	28.33	330
	30,520,100	4	302.26	4,321	385.64	3,207	346.18	3,361	392.00	3,380
	30,520,400	4	9.57	117	9.29	35	16.24	37	16.60	37
	30,700,100	4	9.36	379	7.99	179	14.06	312	12.84	319
	30,700,400	4	259.88	3,709	235.86	4,045	373.24	4,429	377.12	4,452
	Class Avg.	**31**	**80.99**	**1,292**	**86.32**	**1,025**	**103.56**	**1,137**	**110.38**	**1,145**
	20,120,40	3	0.10	0	0.10	0	0.10	0	0.10	0
	20,120,100	3	3.58	427	1.97	71	2.79	77	3.11	80
	20,120,200	3	131.70	2,768	213.92	1,780	233.09	1,563	229.71	1,610
	20,220,40	3	2.57	718	1.74	91	2.52	140	2.66	140
III-B	20,220,100	3	46.51	1,443	45.76	765	49.62	825	45.47	924
	20,220,200	3	1,476.54	5,546	793.64	3,492	1,577.11	4,630	2,358.90	4,203
	20,320,40	3	25.84	3,033	12.45	883	24.19	1,261	20.34	1,594
	20,320,100	3	9.69	556	7.13	104	17.35	198	17.34	201
	20,320,200	3	720.30	2,863	705.18	2,001	803.75	2,519	823.07	2,694
	Class Avg.	**27**	**268.54**	**1,928**	**197.99**	**1,021**	**301.17**	**1,246**	**388.97**	**1,272**
	40,1200,400	1	11.95	3	9.58	6	53.98	5	51.28	5
	40,1200,800	1	42.36	687	63.92	556	279.73	685	289.85	665
	40,1200,1200	1	51.29	119	62.85	61	649.83	47	556.54	47
IV	50,1400,400	1	25.56	998	31.29	643	203.49	968	298.94	982
	50,1400,800	1	103.24	2,377	124.27	1,985	300.42	2,370	430.26	2,276
	50,1400,1200	1	212.81	3,611	210.15	2,233	522.34	3,596	497.52	3,520
	50,1500,1000	1	53,946.52	419,563	48,522.85	381,225	43,764.95	301,027	52,531.55	357,322
	50,1500,1500	1	69,283.07	262,007	68,200.91	255,307	time	332,866	time	390,397
	Class Avg.	**6**	**74.54**	**1,299**	**83.68**	**914**	**334.97**	**1,279**	**354.07**	**1,249**
	Testbed Avg.	**64**	**159.50**	**1561**	**133.18**	**1013**	**208.62**	**1196**	**250.76**	**1208**

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, Ben+H is on average slightly faster than Ben over the testbed. These time savings are more pronounced in instance groups that require that Ben explore a larger number of nodes as in instance groups 20,220,200; 20,320,40; and 50,1500,1000.

While incorporating Benders lift-and-project cuts has a better solution time when compared to Ben for the instance of size 50,1500,1000, it produces on average over a thirty percent increase in solution time over the complete testbed. Despite closing on average ten percent of the LP gap, implementations of branch-and-Benders-cut with lift-and-project cuts explore more nodes than the other variants for over ninety percent of the instances. In addition, 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. From Table 6 we conclude that the addition of these cuts negatively influences the branching within the enumeration tree and the overall solution process.

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 computation time 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 requires to solve P_{Heur}.

Finally, we note that both Ben and Ben+H are able to solve the two largest instances within the 24-hour time limit. In particular, incorporating our in-tree matheuristic proved beneficial for these instances. On the other hand, including Ben+LP rendered a time saving of one-fifth of the computation time required by Ben 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 Ben as the black box solver for our cut-and-solve algorithm since it is more consistently the fastest solver over all instances in the testbed.

Performance profiles have become the standard for benchmarking solvers because of their ability to consider both efficiency and robustness simultaneously by plotting a performance function over a parameter representing relative time t. This performance function is defined as the proportion of the testbed that can be solved by an algorithm a if given up to t times the time taken by the fastest algorithm. The performance function of an algorithm a at $t = 1$ shows the percentage of a given testbed for which it was the fastest solver. On the other hand, performance function values where t is larger show an algorithm a's robustness in being able to solve instances of the testbed given enough time. Further details of performance profiles can be found in Dolan and Moré (2002).

Figure 1 is the performance profile of our branch-and-Benders-cut algorithm (Ben), our Benders cut-and-solve (CS/LB), and solving P with both CPLEX 12.7.1's branch-and-cut algorithm (CPX) using default settings and CPLEX's black box Benders implementation

(CPXBen) in which we provide annotations that force CPLEX to decompose the problem in the same manner that we did.

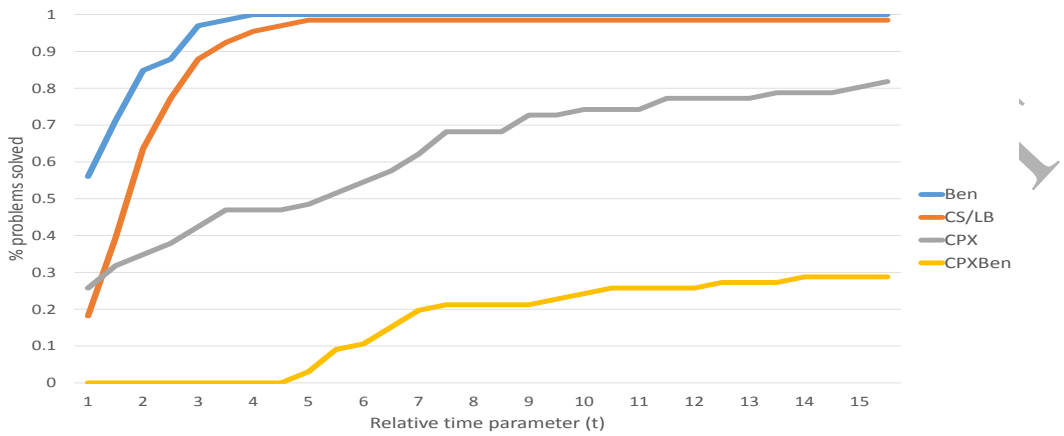


Figure 1: Performance profile of all exact algorithms

We note that both of our proposed methodologies are more robust than using either CPLEX's branch-and-cut or black box Benders implementation with the latter being the worst performing by a significant margin. The branch-and-Benders-cut implementation is the fastest algorithm for 56% of the instances in our testbed while CPLEX's branch-and-cut is the fastest for 25% and finally the Benders cut-and-solve for 19%. As more time is given, we note that the Benders cut-and-solve algorithm rapidly surpasses the performance of CPLEX's branch-and-cut to become the second most robust algorithm to solve the instances of the testbed with our branch-and-Benders-cut algorithm being the most robust.

The branch-and-Benders-cut algorithm is clearly both the most efficient and robust of all solutions algorithms requiring only 2.8 times more time than the corresponding fastest solution algorithm of each instance to solve the entire testbed. On the other hand, given the same time extension, our Benders cut-and-solve is able to solve 94% of the instances, CPLEX's branch-and-cut 46% and CPLEX's black box Benders implementation 0%.

To make a more precise comparison, we next present the average solution times of each algorithm, excluding CPLEX's black box Benders implementation which timed out for over 70% of the instances in the testbed. Table 7 contains the average times in seconds required both to solve instances to proven optimality and to find a solution that is one percent away from the optimum. As in the previous table, the averages of Class IV and the total testbed omit instance groups 50, 1500, 1000 and 50, 1500, 1500 which were not solved to optimality by all solvers.

Table 7 shows that on average Ben and CS/LB are an order of magnitude faster than CPLEX at both solving to proven optimality and obtaining a solution that is one percent away from the optimal value. This speed-up is even more significant when limiting our analysis to the large-scale instances. For these, our branch-and-Benders-cut algorithm is three orders of magnitude faster than CPLEX at solving these instances to proven

optimality. The instances in other classes also show a significant time saving in favour of our Benders decomposition-based algorithms. Finally, we note that our branch-and-Benders-cut algorithm is capable of solving the two largest instances that CPLEX is unable to solve in 24 hours of computing time.

The savings obtained with the branch-and-Benders-cut algorithm can be largely attributed to solving smaller underlying linear programs in the enumeration tree, exploring the nodes in significantly less time, and leaving the root node of the enumeration tree with the linear programming relaxation of the complete formulation. We believe the latter makes a significant difference since when comparing to CPLEX's black box Benders implementation we note it does not leave the root node with the LP bound, thus leading to larger enumeration trees.

Table 7: Comparison of computation times in seconds

			Proven optimality			**1% from optimum value**		
Class	$\|N\|, \|A\|, \|K\|$	**Nb**	**Ben**	**CS/LB**	**CPX**	**Ben**	**CS/LB**	**CPX**
I	20,230,40	3	0.20	0.22	0.07	0.21	0.22	0.04
	20,230,200	4	23.83	25.55	252.95	11.13	3.70	60.05
	20,300,40	4	0.24	0.35	0.17	0.23	0.24	0.16
	20,300,200	4	22.36	26.42	303.24	3.98	11.44	345.20
	30,520,100	4	302.26	191.63	3,181.33	10.34	10.10	3.51
	30,520,400	4	9.57	18.89	95.46	5.99	6.28	18.16
	30,700,100	4	9.36	12.19	71.61	5.06	2.80	103.61
	30,700,400	4	259.88	286.60	10,479.58	33.01	9.79	49.29
	Class Avg.	**31**	**80.99**	**72.49**	**1,856.05**	**9.02**	**5.74**	**74.84**
III-B	20,120,40	3	0.10	0.10	0.05	0.09	0.10	0.04
	20,120,100	3	3.58	4.91	13.42	0.80	1.13	13.10
	20,120,200	3	131.70	110.87	361.23	7.65	4.13	403.99
	20,220,40	3	2.57	3.57	6.91	1.50	1.71	7.30
	20,220,100	3	46.51	44.96	153.86	3.25	3.31	204.40
	20,220,200	3	1,476.54	596.38	1,615.31	16.47	50.16	2,527.68
	20,320,40	3	25.84	21.38	27.79	3.21	1.17	19.18
	20,320,100	3	9.69	23.95	69.25	3.33	3.06	36.14
	20,320,200	3	720.30	429.10	2,592.58	170.61	280.73	3,590.40
	Class Avg.	**27**	**268.54**	**137.25**	**537.82**	**22.99**	**38.39**	**755.80**
IV	40,1200,400	1	11.95	17.08	59.82	8.81	7.56	12.13
	40,1200,800	1	42.36	173.01	4,483.75	28.94	31.79	4,282.02
	40,1200,1200	1	51.29	128.68	1,664.10	49.92	43.16	2,206.95
	50,1400,400	1	25.56	77.17	575.91	13.61	13.25	43.42
	50,1400,800	1	103.24	238.81	39,051.12	38.95	39.79	209.18
	50,1400,1200	1	212.81	465.44	57,071.73	72	95.61	552.87
	50,1500,1000	1	53,946.52	69,312.55	time	110.56	109.54	634.89
	50,1500,1500	1	69,283.07	time	time	134.48	238.67	1,165.94
	Class Avg.	**6**	**74.54**	**183.37**	**17,151.07**	**35.37**	**38.53**	**1,217.76**
	Testbed Avg.	**64**	**159.50**	**110.20**	**2,733.83**	**17.39**	**22.59**	**469.27**

On the other hand, our Benders cut-and-solve algorithm's performance is also on average three orders of magnitude faster than CPLEX over the testbed and is the fastest, on average, of the three algorithms when solving to proven optimality. It is two orders of magnitude faster than CPLEX when solving to optimality the large-scale instances while for Class III-B, it saves over seventy percent of the solution time when compared to CPLEX. On average, CS/LB solves three sparse problems before proving optimality of its obtained solution. Each of these sparse problems is 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 our branch-and-Benders-cut algorithm.

With respect to finding solutions within one percent of the optimal value, we note that both algorithms are on average an order of magnitude faster than CPLEX's branch-and-cut algorithm. Our branch-and-Benders-cut algorithm is on average, thirty percent faster than our Benders cut-and-solve algorithm over the entire testbed despite our Benders cut-and-solve being up to three times faster for some instance groups such as 30, 70, 400 of Class I. Based on these results we conclude that either Benders-based algorithms lead to significant time savings when either used as an exact algorithm or to obtain high quality feasible solutions.The results in Table 7 suggest that these algorithms have the potential to tackle even larger instances at the expense of allowing a small optimality gap.

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 outperform both the branch-and-cut and blackbox Benders decomposition algorithms of 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 and an in-tree matheuristic. These additional refinements also serve as general guidelines for implementing a branch-and-Benders-cut algorithm for other mixed integer problems. In addition, we presented a tailored core point selection criterion for our problem which leads to significant savings in computation time, thus highlighting its importance when using Pareto-optimal Benders cuts.

We present a 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, 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: The authors are grateful to the four anonymous referees for their insightful comments that improved the content and presentation of the paper. 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.

Ahuja RK, Magnanti TL, Orlin JB (1993) *Network Flows: Theory, Algorithms, and Applications* (Prentice-Hall, New Jersey, U.S.A.).

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.

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, 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.

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.

Bonami P (2012) On optimizing over lift-and-project closures. *Mathematical Programming Computation* 4(2):151–179.

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.

Calik H, Leitner M, Luipersbeck M (2017) A benders decomposition based framework for solving cable trench problems. *Computers & Operations Research* 81:128 – 140.

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(1–3):73–99.

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.

de Camargo RS, Miranda G, Luna HP (2008) Benders decomposition for the uncapacitated multiple allocation hub location problem. *Computers and Operations Research* 35(4):1047–1064.

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

Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. *Mathematical Programming* 91(2):201–213.

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, Ljubić I, Sinnl M (2016) Benders decomposition without separability: A computational study for capacitated facility location problems. *European Journal of Operational Research* 253(3):557 – 569.

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.

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.

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

Gouveia L, Joyce-Moniz M, Leitner M (2018) Branch-and-cut methods for the network design problem with vulnerability constraints. *Computers & Operations Research* 91:190 – 208.

Gzara F, Erkut E (2011) Telecommunications network design with multiple technologies. *Telecommunication Systems* 46(2):149–161.

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.

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.

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.

Marín ÁG, Jaramillo P (2009) Urban rapid transit network design: accelerated benders decomposition. *Annals of Operations Research* 169(1):35–53.

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

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 multilevel uncapacitated facility location. *Transportation Science* 1–23.

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.

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.

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.

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 (2019) Solving the optimum communication spanning tree problem. *European Journal of Operational Research* 273(1):108–117.