A Benders Decomposition Method for Two-Stage Stochastic Network Design Problems

Ragheb Rahmaniani
Teodor Gabriel Crainic
Michel Gendreau
Walter Rei

April 2017

CIRRELT-2017-22

A Benders Decomposition Method for Two-Stage Stochastic Network Design Problems

Ragheb Rahmaniani[1,2,*], **Teodor Gabriel Crainic**[1,3,], **Michel Gendreau**[1,2], **Walter Rei**[1,3]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Department of Mathematics and Industrial Engineering, Polytechnique Montréal, P.O. Box 6079, Station Centre-Ville, Montréal, Canada H3C 3A 7

[3] Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

Abstract. This paper describes a Benders decomposition algorithm capable of efficiently solving large-scale instances of the well-known multi-commodity capacitated network design problem with demand uncertainty. The problem is important because it models many real-world applications, including telecommunications, transportation, and logistics. This problem has been tackled in the literature with meta-heuristics and exact methods, but many benchmark instances, even though of moderate size, remain unsolved. To successfully apply Benders method to these instances, we propose various acceleration techniques, including the use of cutting planes, partial decomposition, heuristics, stronger cuts, reduction and warm-start strategies. Extensive computational experiments on benchmark instances were conducted to evaluate the efficiency and robustness of the algorithm as well as of the proposed strategies. The numerical results confirm the superiority of the proposed algorithm over existing ones. We dedicate this work to Professor Jacques F. Benders who left this world January 2017.

Keywords: Network design, stochastic programming, Benders decomposition, acceleration techniques.

Acknowledgements. While working on this project, the second author was Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal. Partial funding for the project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Discovery Grants program. We also gratefully acknowledge the support of the Fonds de recherche du Québec– Nature et technologie (FRQNT), the Canada Foundation for Innovation, and Ministère de l'Économie, de la Science

---

* Corresponding author: Ragheb.Rahmaniani@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2017

1 Introduction

A wide range of real-world applications in diverse settings such as logistics, transportation, production planning, and telecommunications can be modeled as *Network Design* (*ND*) problems [12, 34, 26]. In general terms, these problems revolve around the selection of a subset of (capacitated) arcs out of a potential set in order to support the flow requirements between various origin-destination (OD) pairs. The decisions are to be made such that an effective trade-off between the cost of design decisions and their impact on the performance of the resulting network is achieved, while demand and a set of problem-specific constraints are satisfied [31, 14].

Uncertainty characterizes many cases for which ND models are the methodology of choice, in particular the strategic and tactical planning processes yielding decisions to be used repetitively over a certain time horizon [19, 48, 49]. Moreover, the impact on the performance of a system of the uncertainty regarding its future state is far from trivial because mid to long-term decisions are generally difficult to reverse or costly to adjust [54]. *Stochastic programming* has become the methodology of choice to properly capture the uncertainty in such cases [28, 15].

Stochastic Network Design (*SND*) problems aim to find a single design that remains cost-effective when plausible realizations of the uncertain elements are encountered. To characterize the uncertainty set, a common practice is to use a set of discrete scenarios [2]. Once the appropriate set of scenarios is generated, the SND problem can be formulated in an extensive form as a *two-stage stochastic* program. The first-stage models the design decisions, i.e., selection of arcs. The second-stage formulates the recourse actions by measuring the cost-effectiveness of the first-stage design in servicing the demands (e.g., flow decisions) for each realization of the uncertainty.

ND constitutes a challenging class of NP-hard combinatorial optimization problems. Considering the uncertainty does not make them any easier to address as it yields much larger instances. Thus, they remain notoriously hard to solve with exact methods and off-the-shelf optimization solvers [6, 17]. To give an idea of the inherent difficulty of the generic SND problems, moderate instances with more than 20 scenarios remain still unattainable for state-of-the-art algorithms and optimization solvers; see section 5. Given the economical, political and environmental significance of many ND problems [13, 40], we need efficient and accurate methodologies to address more realistically sized instances of this important family of optimization problems, particularly in stochastic settings.

Various solution algorithms have been designed to handle stochastic programs with recourse; see [45] for a comprehensive review. Most of these solution procedures rely heavily on the premises of the *Benders Decomposition* (*BD*) method [1, 39]. The BD approach is commonly referred to as L-shaped decomposition ([53]) when applied to stochastic programs.

BD is based on a pattern of projection, outer linearization and relaxation [23, 24]. First, the model is projected onto the subspace defined by the first-stage variables. The projected term is then dualized and an equivalent model is built by enumerating all extreme points and rays of the dual polyhedron. The extreme rays indicate feasibility requirements for the first-stage variables (i.e., feasibility cuts) and the extreme points state the projected cost (i.e., optimality cuts). Enumerating all the extreme points and rays of the dual polyhedron is not computationally practical and most of the associated cuts will not be active at an optimal solution. Therefore, a relaxation of the equivalent formulation is performed such that it initially includes no cuts, these being iteratively generated by sequentially solving a *Master Problem* (*MP*) and one or several *Subproblems* (*SPs*).

The BD method enables decomposing the SND problems according to the realization of the random elements that set the values of the uncertain parameters in the model. It thus simplifies the solution of these problems since, often, a large portion of variables and constraints are associated to the large number of scenarios used to represent the uncertainty [5]. The BD algorithm is also known to be largely used as an exact method for deterministic ND problems, as it provides the means to separate the design and flow decisions [10]. Moreover, it constitutes currently the state-of-the-art exact method for SND problems [18]. We aim to further enhance this algorithm by refining existing acceleration strategies and proposing novel ones. We investigate, in particular, the idea of tightening the MP by means of various *valid inequalities* (*VI*s) appended in a cutting-plane fashion. More importantly, we propose a set of VIs that can be used to improve the quality of the classical Benders cuts. We also propose a new strategy to generate Pareto-optimal cuts, which is equivalent to the most widely used strategy in the literature, but numerically performs better. To avoid the generation of feasibility cuts, we propose to apply a relatively complete recourse property to the formulation that, when combined with the use of strengthened combinatorial cuts, can be tailored to enforce the original feasibility requirements of the problem. In essence, this strategy completely eliminates the need to generate feasibility cuts, in favor of a focused search for optimality cuts, which have a greater effect on improving the value of the lower bound generated by the BD method. Moreover, to mitigate the ineffective initial iterations of the BD method, we propose a strategy based on the idea of "properly deflecting the master solutions" so as to chose better dual values for the set of initial cuts. Last but not least, to handle large instances more effectively, size-reduction procedures and a simple heuristic are developed, and the algorithm is embedded in a branch-and-cut framework.

To test the method we propose, we address the well-known *Multi-Commodity Capacitated Fixed-charge Network Design Problem with Stochastic Demands* (*MCFNDSD*). Extensive numerical experiments on a broad range of instances show that the algorithm delivers superior performance compared to state-of-the-art solution algorithms and optimization solvers.

The contribution of this paper therefore is threefold: 1) To enhance the state-of-the-art of BD algorithms, through new acceleration strategies and the refinement of existing ones; 2) To propose an exact algorithm capable of efficiently solving standard benchmark instances of the MCFNDSD; 3) To study experimentally the algorithm and proposed strategies, and thus to characterize their behavior and suggest promising research directions.

The rest of this article is organized as follows. The problem is presented in Section 2, while Section 3 provides the literature review of the Benders decomposition algorithm and its application to the problem at hand. Section 4 details the proposed and enhanced acceleration strategies, and Section 5 displays the results of the numerical experiments and the associatd analyzes. Conclusions and future research directions are discussed in Section 6.

2 Problem Definition

In the MCFNDSD problem, a set of commodities $\mathscr{K}$ are to be simultaneously routed through a directed graph consisting a set of nodes $\mathscr{N}$ and a set of potential arcs $\mathscr{A}$. With each commodity $k \in \mathscr{K}$ is associated a stochastic demand $d^k(\omega) \geq 0$ which is to be routed from a unique origin node $O(k) \in \mathscr{N}$ to a unique destination node $D(k) \in \mathscr{N}$ for each realization ω of the uncertainty set Ω. The goal is to select a subset of the arcs to meet all demand at minimum cost, computed as the sum of the fixed design costs, charged whenever an arc is used, and the expected transportation costs. The unit transportation cost on arc $a \in \mathscr{A}$ for commodity k is denoted c_a^k, and the fixed design cost for arc $a \in \mathscr{A}$ is represented by f_a. In addition, there is a capacity limit u_a on each arc $a \in \mathscr{A}$, limiting the maximum flow traversing it.

The MCFNDSD is modeled as a two-stage stochastic mixed-integer linear program (MILP) by using binary design variables y_a indicating if arc $a \in \mathscr{A}$ is used (1) or not (0) and a recourse function $Q : \{0,1\}^{|\mathscr{A}|} \to \mathbb{R} \cup \{+\infty\}$ measuring the expected flow costs. The compact formulation of the MCFNDSD is

$$\text{MCFNDSD} = \min_{y \in Y} f^T y + Q(y). \tag{1}$$

The objective function minimizes the total fixed cost plus the expected routing costs. The feasible space of the first-stage variables is written in compact form in order to keep the generality of formulation for a broader class of the ND problems. For example, this set may contain any type of constraints such as contingency requirements or the degree restrictions on each node. The recourse function associated to the design vector y is defined by

$$Q(y) = \mathbb{E}_{\omega}\left[\Phi(y;\omega)\right], \tag{2}$$

where the function $\Phi(y;\omega)$ can be defined by using continuous flow variables $x_a^k(\omega) \geq 0$ to reflect the amount of flow on arc $a \in \mathscr{A}$ for commodity $k \in \mathscr{K}$ under realization $\omega \in \Omega$. The recourse problem $\omega \in \Omega$ is

$$\Phi(y;\omega) = \min_{x(\omega)\in\mathbb{R}_+^{|\mathscr{A}||\mathscr{K}|}} \sum_{k\in\mathscr{K}}\sum_{a\in\mathscr{A}} c_a^k x_a^k(\omega) \tag{3}$$

$$\text{s.t.} \quad \sum_{a\in\mathscr{A}(i)^+} x_a^k(\omega) - \sum_{a\in\mathscr{A}(i)^-} x_a^k(\omega) = d_i^k(\omega) \qquad \forall i \in \mathscr{N}, \forall k \in \mathscr{K} \tag{4}$$

$$\sum_{k\in\mathscr{K}} x_a^k(\omega) \leq u_a y_a \qquad \forall a \in \mathscr{A}, \tag{5}$$

where $A(i)^+$ and $A(i)^-$ indicate the set of outward and inward arcs incident to node i. The vectors $d_i^k(\omega)$ express the nodal balance and are defined as

$$d_i^k(\omega) = \begin{cases} d^k(\omega) & \text{if } i = O(k) \\ -d^k(\omega) & \text{if } i = D(k) \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The objective function (3) minimizes the total flow costs. Constraint set (4) imposes the flow conservation requirements for each commodity on each node. Constraints (5) enforce the capacity limit on each arc.

2.1 A brief review on the MCFNDSD

The outlined MCFNDSD problem is notoriously difficult to solve. Its main difficulty is rooted in the complex interrelation between the design and flow costs, weak linear programming (LP) relaxation, degeneracy, and the additional challenges stemming from the very large problem dimensions characterizing most applications [21]. Therefore, heuristics have been the most used solution algorithms to obtain good solutions within reasonable running times. Crainic *et al.* [15] devised a master-slave parallel progressive hedging based meta-heuristic. They decomposed the problem according to the scenarios where each subproblem is an NP-hard deterministic ND formulation. Then, the algorithm seeks a consensus for the design decisions among the SPs by systematically updating the fixed costs. An improvement to this algorithm, by investigating the idea of having multiple scenario subproblems, is proposed in [17]. The authors observed that despite the increased difficulty of each SP, more elegant results could be obtained. A BD-based heuristic for the problem with large number of scenarios was introduced in [5], the authors reporting encouraging results compared to the classical algorithm and state-of-the-art of commercial solvers.

Similar to the exact algorithms, it has been only quite recently that tight bounds were obtained for the deterministic version of the problem [6]. The results obtained provide a promising overture toward handling deterministic ND problems exactly. However, it is not a viable tool to address the stochastic variant of the problem, since it considers all the scenarios at the same time. An enhanced BD method and applied to the SND problems was presented in [16], the authors further improved the algorithm in [18]. To the best of our knowledge, it constitutes the state-of-the-art of the exact methods for the MCFNDSD problem. Due to the closeness of this algorithm to the method presented in this article, we shall shortly discuss it in more details.

3 The Benders Decomposition Method

Let $\bar{y}$ indicate any possible design for the considered ND problem, π and α be the dual variables associated to the constraints (4) and (5). The *dual subproblem* (DSP), which is the dual of $\Phi(\bar{y};\omega)$, can be stated as

$$\text{DSP}(\bar{y};\omega) = \max_{\pi\in\mathbb{R}^{|\mathscr{N}||\mathscr{K}|},\alpha\in\mathbb{R}_+^{|A|},s\in\mathbb{R}_+^{|\mathscr{A}||\mathscr{K}|}} \sum_{k\in\mathscr{K}} d^k(\omega)\left(\pi^k_{O(k)}-\pi^k_{D(k)}\right) - \sum_{a\in\mathscr{A}} u_a\bar{y}_a\alpha_a \tag{7}$$

$$\text{s.t.}\quad \pi^k_{a^-}-\pi^k_{a^+}-\alpha_a+s^k_a=c^k_a \qquad \forall a\in\mathscr{A},k\in\mathscr{K}, \tag{8}$$

where a^+ and a^- respectively indicate the tail and head of arc a, and s^k_a is the slack variable associated to the constraint (8) for commodity $k\in\mathscr{K}$ and arc $a\in\mathscr{A}$. The $DSP(\bar{y};\omega)$ can be either unbounded or feasible, if the dual polyhedron is not empty. The former case indicates the infeasibility of the primal subproblem for solution $\bar{y}$. Thus, there is direction of unboundedness which satisfies

$$\sum_{k\in\mathscr{K}} d^k(\omega)\left(\pi^k_{O(k)}-\pi^k_{D(k)}\right) - \sum_{a\in\mathscr{A}} u_a\bar{y}_a\alpha_a > 0, \tag{9}$$

the (π,α) is the solution of the dual cone obtained by replacing c^k_a with 0 in constraint 8. In the latter case, capturing the optimal value of $DSP(\bar{y};\omega)$ in a single variable $\theta_\omega\in\mathbb{R}^1$, we aim for θ_ω never to exceed the optimal value associated to the given design $\bar{y}$, i.e.,

$$\sum_{k\in\mathscr{K}} d^k(\omega)\left(\pi^k_{O(k)}-\pi^k_{D(k)}\right) - \sum_{a\in\mathscr{A}} u_a\bar{y}_a\alpha_a > \theta_\omega. \tag{10}$$

The BD method exploits this result. It iteratively solves an MIP problem, called MP, which involves only the y and θ variables. The MP has initially no constraints except for those imposed on the design variables. Each time the MP is solved, the optimal value of the design variables is extracted and fixed in the formulation (7-8) for each $\omega\in\Omega$. A feasibility or optimality cut for each scenario can then be generated by complementing (9) or (10) and replacing the fixed design variables with arbitrary values. These cuts are then inserted in the MP and the next iteration is performed. Accordingly, the MP at iteration t has the following form

$$MP^t = \min_{y\in Y,\theta\in\mathbb{R}^{|\Omega|}} f^Ty+\mathbb{E}_\omega[\theta_\omega] \tag{11}$$

$$\text{s.t.}\quad \sum_{k\in\mathscr{K}} d^k(\omega)\left(\bar{\pi}^k_{O(k)}-\bar{\pi}^k_{D(k)}\right) - \sum_{a\in\mathscr{A}} u_a\bar{\alpha}_a y_a \le 0 \qquad \forall\omega\in\Omega,(\bar{\pi},\bar{\alpha})\in E^t_\omega \tag{12}$$

$$\sum_{k\in\mathscr{K}} d^k(\omega)\left(\bar{\pi}^k_{O(k)}-\bar{\pi}^k_{D(k)}\right) - \sum_{a\in\mathscr{A}} u_a\bar{\alpha}_a y_a \le \theta_\omega \qquad \forall\omega\in\Omega,(\bar{\pi},\bar{\alpha})\in L^t_\omega, \tag{13}$$

where E^t_ω and L^t_ω stand for the set of feasibility and optimality cuts associated to scenario ω up to iteration t. This process continues until the upper and lower bounds coincide, for which an optimal solution to the original problem is attained. The pseudo-code of the algorithm is presented in Algorithm 1.

In Algorithm 1, ε_{opt} indicates the desired optimality tolerance and $\overline{DSP}(\bar{y};\omega)$ is the optimal cost of the dual subproblem ω for the given solution $\bar{y}$. To verify the optimality gap, the MP gives a valid lower bound (LB) on the optimal cost because it is a relaxation of the Benders equivalent formulation. On the other hand, the cost of the $\bar{y}$ solution plus the expected cost of the SPs gives a valid upper bound (UB) on the optimal cost, because it is equivalent to fixing the current solution $\bar{y}$ in the original formulation.

3.1 Literature review on acceleration strategies

The BD method has proven successful for a wide range of difficult optimization problems, including ND problems [10]. A naive implementation of the classical algorithm may however preform disappointingly. Researchers have

Algorithm 1 : The multi-cut Benders decomposition algorithm

```
$L^0_\omega \leftarrow \emptyset$, $E^0_\omega \leftarrow \emptyset, \forall \omega \in \Omega$; $UB \leftarrow \infty$; $LB \leftarrow -\infty$; $t \leftarrow 0$
while $(UB - LB)/UB > \varepsilon_{opt}$ do
  Solve the $MP^t$ to obtain $\bar{y}$
  if infeasible then
    return infeasible
  end if
  $t \leftarrow t+1$
  $LB \leftarrow f^T\bar{y} + \mathbb{E}_\omega\{\bar{\theta}_\omega\}$
  for $\omega \in \Omega$ do
    Solve $DSP(\bar{y};\omega)$
    if unbounded then
      Find the extreme ray $(\bar{\pi},\bar{\alpha})$
      $E^t_\omega \leftarrow E^{t-1}_\omega \cup \{(\bar{\pi},\bar{\alpha})\}$
      Break the cut generation loop
    end if
    $L^t_\omega \leftarrow L^{t-1}_\omega \cup \{(\bar{\pi},\bar{\alpha})\}$
  end for
  $UB \leftarrow \min\{UB, f^T\bar{y} + \mathbb{E}_\omega\{\overline{DSP}(\bar{y};\omega)\}\}$
end while
```

thus explored ways to overcome the drawbacks of the algorithm when applied to various optimization problems. We briefly review the acceleration strategies that are closest to what we will present in this article and refer the interested readers to [39] for a detailed review on this subject. These strategies can be classified into four categories.

Probably the most studied category revolves around the generation of cuts. Magnanti and Wong [30] were the first to propose the generation of non-dominated optimality cuts when the dual SP has multiple optima for a given feasible solution. This strategy, which entails the solution of an additional SP, is instantiated using a core point of the MP to identify a non-dominated cut. Papadakos [37] emphasized that the previous method can be inefficient due to the dependency on the auxiliary SP and the difficulty of extracting the core point, and showed how alternative core points can be used in each iteration to generate Pareto-optimal cuts by means of an independent SP. Sherali and Lunday [50] viewed the Pareto-optimal cut generation strategy as a multi-objective optimization problem. The authors showed that the cuts could be generated by simply perturbing the right-hand side values of the constraints in the primal SP. Alternative feasibility cuts for binary problems where the SP is a feasibility checking program and involves big-M constraints, where proposed in [8]. The authors showed that the combinatorial cuts can efficiently be separated by searching for minimal infeasible subsystems in the solutions of the relaxed MP. Bodur *et al.* [4] employed Gomory mixed-integer cuts to strengthen the classical Benders cuts, by adding them to the SP each time it is solved.

The numerical burden of iteratively solving a mixed-integer MP and a series of SPs is one of the major drawbacks of the BD method. Thus, the second category of acceleration strategies involves techniques to efficiently solve the MP and SPs. Geoffrion and Graves [25] proposed to solve the MP each time to ε-optimality and steadily decrease the ε value as the algorithm proceeds in order to ensure the global convergence. Similar idea at the SP level has been applied in [55] to extract sub-optimal solutions of the dual polyhedron to generate cuts. Mcdaneil and Devine [33] showed that valid optimality and feasibility cuts can be produced when the LP relaxation of the

MP is solved. The authors proposed to apply the algorithm in two phases. In the first phase, all integrality requirements are ignored to quickly generate cuts and tighten the master formulation. In the second phase, the integrality constraints are reintroduced and the solution process continues until the global optimum is found. Solving the MP heuristically has been proposed by Côté and Laughton [11]. The authors applied Lagrangian relaxation on the optimality and feasibility cuts to maintain the nice structure of their MP. Related to this general idea, different (meta-) heuristic algorithms have been employed to faster optimize the MP and avoid many costly iterations of the MP (see, e.g., [38]). Similarly, heuristics have been employed by several authors to extract approximate optimality (e.g., [41]) and feasibility cuts [29]. To avoid solving a MILP program at each iteration, modern implementations of the BD method work by building a single branch-and-bound tree (e.g., [22]). The algorithm generates the Benders cuts for the integer (and possibly fractional) solutions encountered inside the tree, and guarantees the convergence to the optimal solution. Finally, managing the size of the MP by periodically removing ineffective cuts has appeared as a promising technique to reduce the computational burden, especially when multiple cuts per iteration are added to the master formulation (e.g., [36]).

The third category revolves around the MP to enhance the quality of the generated solutions for the set of complicating variables, particularly at the initial iterations. The instability of the MP in respect to the generated solutions is one of the main drawbacks of the BD which can cause overly slow convergence of the algorithm due to initial large steps and excessive oscillations when it approaches a local optimum [44]. A constraint to restrict the Hamming distance of the generated solutions from a stabilizing point was proposed to address the issue in [49], while van Ackooji *et al.* [52] observed significant speedups when the MP was stabilized by a trust region method in L_1 norm. Heuristics are also often used as a warm-start strategy to generate a set of high-quality initial solutions and their associated cuts to tighten the relaxed MP, e.g., [27]. It is worth mentioning that heuristics have also proven very useful in mitigating the erratic progression of the primal bounds through the exploration of the neighborhoods of the current solution, e.g., [42]. Strengthening the relaxed MP with VIs is also a widely-used strategy to tighten the initial master formulation, e.g., [46]. Birge and Louveaux [3] demonstrated that it is preferable to add a single cut per SP when the decomposition yields several independent SPs. This strategy, referred to as multi-cut reformulation, prevents the loss of information in the aggregation step and strengthens the MP faster.

The last category is specifically tailored for two-stage stochastic programs, although it is equally applicable to any problem where the decomposition yields multiple SPs. It addresses the decomposition scheme of the BD method through partially projecting the second stage variables. This idea was first tested in [35] where the authors proposed to retain in the master formulation a scenario SP associated to the maximum demand. *Crainic et al.* [16] proposed various strategies to properly chose and retain a subset of the SPs in the master formulation, and then significantly improvements to this strategy through simultaneously selecting and creating SPs to be retained in the MP [18] . The authors referred to this technique as partial-decomposition strategy (PDS).

In conclusion, not a single strategy is a silver bullet and various acceleration strategies need to be incorporated into the classical BD framework. This is particularly true for realistic instances of SND problems, for which multi-cut reformulation, Pareto-optimal cut generation scheme, a two phase approach, single search tree strategy and partial decomposition strategy are considered. As the results of this paper indicate, however, these strategies are not sufficient to reach a high-performance BD method. Our goal is thus to contribute to the enhancement of the categories mentioned above, as outlined in the following section.

4 Enhancing Benders Decomposition Method

We present in this section various enhancement strategies aimed to address several drawbacks of the algorithm, including: (1) weak relaxation of the MP, (2) generation of low-quality cuts, (3) ineffective initial iterations, (4) slow progression of the primal bound, and (5) time consuming iterations. In section 4.1, VIs are proposed to alleviate the weak relaxation of the MP. The cut-generation scheme is revisited in section 4.2 to efficiently generate stronger/better cuts. A warm-start strategy is presented in section 4.3 to improve the quality of the initial cuts. Finally, reduction strategies, upper-bounding heuristic, branching and node selection rules are discussed in section 4.4 in order to effectively handle the integer phase of the algorithm.

4.1 Valid inequalities

After the relaxation step of the BD method, an important part of the formulation is projected out. Thus, the algorithm exhibits a very poor performance due to the absence of sufficient leading information. Particularly at the initial iterations of the algorithm, weak and low quality lower bounds and solutions are generated. These iterations correspond to a large portion of the computational time, while they yield a limited contribution to the overall convergence. Although the PDS can considerably dampen these undesirable behaviors, more significant accelerations can be achieved by making use of strong VIs. We shall numerically show that the best results can be attained when both strategies are applied. Note that the benefit of VIs is well established in the literature. To the best of our knowledge, they are merely used as a warm-start to the MP. In this section, we propose a cutting plane that extends the use of the VIs throughout the whole solution process. Moreover, we observed numerically that better performance can be achieved when these VIs are prioritized over the classical Benders cuts.

4.1.1 Lower bounding function

The first class of VIs contains useful information regarding the projected terms of the objective function. We observe that, for any OD pair, a minimum flow cost can be calculated since the corresponding demand must be satisfied. Let $P_{O(k)D(k)}$ indicate the shortest path connecting the origin of commodity k to its destination. Inequality (14) provides a lower estimation on the recourse cost of the scenario ω. This inequality can be justified based on the simple argument that the routing cost for each commodity is at least as large as that of the cheapest route satisfying it.

$$\theta_\omega \geq \sum_{k\in\mathscr{K}} \sum_{a\in P_{O(k)D(k)}} d_k(\omega)c_a^k \qquad \forall\omega\in\Omega. \tag{14}$$

Inequality (14) provides a weak estimation of the recourse cost, however. This is because the considered problem is capacitated. More importantly, it gives no leading information on the design (first-stage) variables. To provide a stronger bound, referred to as the *lower bounding function* (*LBF*), on scenario ω, we make use of a deterministic *Multi-Commodity Capacitated Network Flow* (*MCNF*) problem with minimum demands, i.e., $d_k = min_{\omega\in\Omega}\{d_k(\omega)\}, \forall k\in\mathscr{K}$; see Appendix B for the formulation. This problem is equivalent to the LP relaxation of the deterministic version of the considered ND problem with the demand $d_k, \forall k\in\mathscr{K}$. Its solution provides a valid lower bound on the cost of each scenario, since the demands are replaced by the minimum volume for each commodity.

Theorem 4.1. *Let $\bar{x}_a^k$ and $\bar{y}$ be the optimal solution of the MCNF problem, $\bar{\pi}$ be the dual values associated to the flow conservation constraints, then*

$$\theta_\omega \geq \sum_{k\in\mathscr{K}}\sum_{a\in\mathscr{A}} c_a^k\bar{x}_a^k + \sum_{k\in\mathscr{K}} (d_k(\omega)-d_k)\,(\bar{\pi}_{O(k)}^k - \bar{\pi}_{D(k)}^k) + \sum_{a\in\mathscr{A}} f_a\,(\bar{y}_a - y_a) \qquad \forall\omega\in\Omega, \tag{15}$$

is a VI for the MP.

Proof. See Appendix B. □

4.1.2 Cutset inequalities

Based on the feasibility requirements of the problem at hand, there should be sufficient capacity installed across any partition of the network to support the movement of the commodity flows. Let $\bar{N} \subset \mathscr{N}$ be any nonempty subset of the node set $\mathscr{N}$ and $\underline{N}$ its complement, i.e., $\underline{N} = N \backslash \bar{N}$. The corresponding cutset $(\bar{N}, \underline{N}) = \{a \in \mathscr{A} | a^+ \in \bar{N}, a^- \in \underline{N}\}$, where a^+ and a^- stand for the tail and head of arc a, respectively. Let $K(\bar{N}, \underline{N}) = \{k \in \mathscr{K} | O(k) \in \bar{N}, D(k) \in \underline{N}\}$ be the associated commodity subset. We define the maximum flow over this cutset as $d^{max}_{(\bar{N},\underline{N})} = max_{\omega \in \Omega} \left\{ \sum_{k \in \mathscr{K}(\bar{N},\underline{N})} d_k(\omega) \right\}$. $(\bar{N}, \underline{N})$ is a valid cutset if $d^{max}_{(\bar{N},\underline{N})} > 0$. According to this notation, one can derive *cover inequalities* (CI) according to definitions 4.1 and 4.2.

Definition 4.1. *$C \subseteq (\bar{N}, \underline{N})$ is a cover if the total capacity of the arcs in $(\bar{N}, \underline{N}) \backslash C$ does not support (cover) the flow of demand, i.e., $\sum_{a \in (\bar{N},\underline{N}) \backslash C} u_a < d^{max}_{(\bar{N},\underline{N})}$.*

Definition 4.2. *A cover set C is* minimal *if opening any arc in C is sufficient to cover the demand, i.e., $\sum_{a \in (\bar{N},\underline{N}) \backslash C} u_a + u_q \geq d^{max}_{(\bar{N},\underline{N})}, \forall q \in C$.*

For any cover set $C \subseteq (\bar{N}, \underline{N})$, the CI takes the form $\sum_{a \in C} y_a \geq 1$, imposing the necessity of opening at least one of the arcs in the set C to meet the flow requirements. To separate and lift this set of inequalities, we follow the procedure proposed by Chouman *et al.* [7] for the deterministic ND problem (summarized in Appendix C). Let's C_1 indicate a subset of arcs in $(\bar{N}, \underline{N})$ that take value larger than $1 - \varepsilon_{ci}$ in the current solution and, similarly, $C_0 \subseteq (\bar{N}, \underline{N})$ with value smaller than ε_{ci}, where ε_{ci} is a sufficiently small positive number. The procedure first finds a minimal cover set of C over $(\bar{N}, \underline{N}) \backslash (C_0 \cup C_1)$. Then, it applies a lifting procedure on arcs in $(\bar{N}, \underline{N}) \backslash C$ to find lifting coefficients λ_a for each variable. The violated and *lifted cover inequality* (*LCI*) of the form (16) will be found, if there is any.

$$\sum_{a \in (\bar{N},\underline{N}) \backslash C} \lambda_a y_a + \sum_{a \in C} y_a \geq 1 + \sum_{a \in (\bar{N},\underline{N}) \backslash (C \cup C_0)} \lambda_a. \tag{16}$$

The so-called *minimum cardinality inequalities* (*MCI*) also belong to the family of cutset inequalities and have frequently been used to address ND problems. MCIs are derived for the cutset $(\bar{N}, \underline{N})$, indicating the least number of arcs, denoted $l_{(\bar{N},\underline{N})}$), that must be opened in any feasible solution. Let the arc capacities in $(\bar{N}, \underline{N})$ be sorted and numbered in a non-decreasing order, i.e., $u_a \geq u_{a+1}, a \in (\bar{N}, \underline{N}), a = 1, ..., |(\bar{N}, \underline{N})|$. We define $l_{(\bar{N},\underline{N})} = max \left\{ h | \sum_{t=1,...,h} u_t \geq d^{max}_{(\bar{N},\underline{N})} \right\} + 1$. Subsequently, $\sum_{a \in (\bar{N},\underline{N})} y_a \geq l_{(\bar{N},\underline{N})}$ is a valid MCI cut. A lifting procedure similar to that for the CIs is applied to strengthen the quality of MCI cuts (see [7] and Appendix D). Define $l_{(\bar{N},\underline{N}) \backslash (C_1 \cup C_0)}$ as the least number of arcs that must be opened in set $(\bar{N}, \underline{N}) \backslash (C_0 \cup C_1)$. After applying the lifting procedure, a *lifted minimum cardinality inequality* (LMCI) of the following form can be extracted:

$$\sum_{a \in C_1 \cup C_0} \lambda_a y_a + \sum_{a \in (\bar{N},\underline{N}) \backslash (C_1 \cup C_0)} y_a \geq l_{(\bar{N},\underline{N}) \backslash (C_1 \cup C_0)} + \sum_{a \in C_1} \lambda_a. \tag{17}$$

Generating good cutsets quickly is of critical importance for the effectiveness of this family of inequalities. We have implemented the same cutset generation scheme as [7] with cardinality of one and two because in our preliminary results we observed that they are responsible for most of the lower bound improvement in our algorithm.

4.1.3 Network connectivity cuts

The feasibility requirements of the ND problem requires the existence of at least one connecting path with sufficient capacity between each OD pair. This implies that a sufficient number of arcs exiting (entering) each origin (destination) node should be opened to provide the capacity required to move the demand out of (into) the respective node. Additionally, at least one arc should enter and one should exit any node than might be a transshipment node for a commodity. These conditions are generally enforced through the flow conservation constraints (4). Yet, after the decomposition, the enforcing constraints at each node are eliminated and the network is disconnected, particularly for initial iterations of the algorithm. To alleviate this issue, we propose the following *network connectivity inequalities* (*NCI*), to be added in priority to the relaxed MP.

The NCIs aim to force the feasibility requirements described above. Considering first origin (destination) nodes, consider node $i \in \mathcal{N}$ with $d_i^{max} = max_{\omega\in\Omega}\{\sum_{k\in\mathcal{K}|O(k)=i} d_k(\omega) > 0\}$ (the inner sum computed on $D(k) = i$ for destination nodes), and set $\mathcal{A}^+(i)$ ($\mathcal{A}^-(i)$) of exiting (entering) arcs. Sort (and number) the arcs in $\mathcal{A}^+(i)$ in non-decreasing order, i.e., $u_a \geq u_{a+1}, a \in \mathcal{A}^+(i), a = 1, ..., |\mathcal{A}^+(i)|$ (same for $\mathcal{A}^-(i)$). Then, the least number of exiting (entering) arcs that must be opened in any feasible solution may be defined as $l_i^+ = max\left\{h|\sum_{t=1,...,h} u_t \geq d_i^{max}\right\} + 1$, yielding the following valid cardinality NCI inequality

$$\sum_{a\in\mathcal{A}^+(i)} y_a \geq l_i^+. \tag{18}$$

Let $\mathcal{I} = \{i \in \mathcal{N} | i \neq O(k) \text{ and } i \neq D(k), \forall k \in \mathcal{K}\}$ be the set of purely intermediary nodes (i.e., no commodity originates or terminates at such a node). The following proposition defines NCI inequalities addressing the connectivity issue.

Proposition 4.1. *Given the set of intermediary nodes $\mathcal{I}$, the following set of inequalities are valid for the MCFNDSD and thus for the MP.*

$$y_a \leq \sum_{b\in\mathcal{A}^+(i)} y_b \quad \forall i \in \mathcal{I}, a \in \mathcal{A}^-(i), \tag{19}$$

$$y_b \leq \sum_{a\in\mathcal{A}^-(i)} y_a \quad \forall i \in \mathcal{I}, b \in \mathcal{A}^+(i). \tag{20}$$

Proof. For an intermediary node, exiting arcs can exist if there is at least one entering arc, and vice versa. This follows from the positive fixed costs in the objective function and the flow conservation requirements. □

The inequalities (19) and (20) can be further strengthened for the first phase of the algorithm according to the following lemma.

Lemma 4.1. *If the integrality requirement is relaxed, i.e., $y_a \in [0,1], \forall a \in \mathcal{A}^+(i) \cup A^-(i)$ and $i \in \mathcal{I}$, then inequalities (19) and (20) can rewritten as*

$$\sum_{a\in\mathcal{A}^+(i)} u_a y_a - \sum_{a\in\mathcal{A}^-(i)} u_a y_a = 0 \qquad \forall i \in \mathcal{I}. \tag{21}$$

Proof. The inward flow to node $i \in \mathcal{I}$ is equal to $max_{\omega\in\Omega}\{\sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}^+(i)} x_a^k(\omega)\}$ which entails $\sum_{a\in\mathcal{A}^+(i)} u_a y_a \geq max_{\omega\in\Omega}\{\sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}^+(i)} x_a^k(\omega)\}$ at any feasible solution. Since node i is an intermediary node, the same amount of flow should exit from it, i.e., $max_{\omega\in\Omega}\{\sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}^+(i)} x_a^k(\omega)\} = max_{\omega\in\Omega}\{\sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}^-(i)} x_a^k(\omega)\}$, which gives $\sum_{a\in\mathcal{A}^-(i)} u_a y_a \geq max_{\omega\in\Omega}\{\sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}^+(i)} x_a^k(\omega)\}$. When the binary requirement is relaxed, the flow approximates the exact amount of the opened capacity. This completes the proof since $\sum_{a\in\mathcal{A}^+(i)} u_a y_a = max_{\omega\in\Omega}\{\sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}^+(i)} x_a^k(\omega)\} = max_{\omega\in\Omega}\{\sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}^-(i)} x_a^k(\omega)\} = \sum_{a\in\mathcal{A}^-(i)} u_a y_a$. □

4.1.4 Flow cuts

The PDS provides the possibility of generating a wide range of VIs involving both flow and design variables. *Strong inequalities* (SI) are the well-known examples of such inequalities. SIs are very effective when incorporated in a cutting-plane method. They are derived from the observation that any commodity flow on an arc cannot be larger than its corresponding demand or the arc capacity. Let $\bar{\Omega}$ indicate the set of retained and created scenarios and $b_a^k(\omega) = min\{d_k(\omega), u_a\}$, then

$$x_a^k(\omega) \leq b_a^k(\omega) y_a \qquad \forall \omega \in \bar{\Omega}, a \in \mathscr{A}, k \in \mathscr{K}, \tag{22}$$

is valid for the MCFNDSD and hence for the Benders MP. These inequalities entail an easy separation and are added to the MP if they violate the current solution by at least η units. We also examined flow cover and flow pack inequalities (see [7]). We decided, however, to exclude them from our algorithm because their contribution on the lower bound improvement tended to be limited when the above inequalities were already appended.

4.2 Strengthening the Benders Cuts

After focusing on improving the quality of the master solutions in the previous subsections, we now turn to the cut-generation scheme. In particular, we reinterpret the cut-generation strategy of Magnanti and Wong [30] in order to generate Pareto-optimal cuts more efficiently, discus a strategy to handle cuts with big-M coefficients, and further improve the quality of the optimality cuts by making use of VIs at the SP level.

4.2.1 Generating Pareto-optimal cuts

The primal SP is degenerated, thus, multiple optimal solutions can be extracted from its dual formulation. Each alternative dual solution produces an optimality cut of particular strength [51]. The dual values thus need to be judiciously selected in order to append the best possible cuts to the MP. Magnanti and Wong [30] employed the notion of *dominance* to generate the strongest cut possible.

Definition 4.3. *An optimality cut (13) corresponding to $(\hat{\pi}, \hat{\alpha}) \in U(DSP(\bar{y}; \omega))$ dominates or is stronger than that corresponding to $(\bar{\pi}, \bar{\alpha}) \in U(DSP(\bar{y}; \omega))$ if*

$$\sum_{k\in\mathscr{K}} d^k(\omega)\left(\hat{\pi}^k_{O(k)} - \hat{\pi}^k_{D(k)}\right) - \sum_{a\in\mathscr{A}} u_a \hat{\alpha}_a y_a \geq \sum_{k\in\mathscr{K}} d^k(\omega)\left(\bar{\pi}^k_{O(k)} - \bar{\pi}^k_{D(k)}\right) - \sum_{a\in\mathscr{A}} u_a \bar{\alpha}_a y_a \quad y \in Y, \tag{23}$$

with strict inequality for at least one point $y \in Y$.

Definition 4.4. *An optimality cut $(\hat{\pi}; \hat{\alpha}) \in U(DSP(\bar{y}, \omega))$ is called Pareto-optimal if it is not dominated by any other cut. Similarly, the dual point $(\hat{\pi}, \hat{\alpha})$ is called Pareto-optimal.*

Note that $U(DSP(\bar{y}; \omega))$ indicates the dual polyhedron of $DSP(\bar{y}; \omega)$. The authors used the *core point* notion to formulate an auxiliary problem to extract the Pareto-optimal dual solution. A point y^0 is called core point of Y if it belongs to the relative interior of the convex hull of the Y set.

Definition 4.5. *Let y^0 and $\overline{DSP}(\bar{y}; \omega)$ respectively indicate the core point and objective function of the dual SP (7–8). Then, the optimal solution $(\hat{\pi}, \hat{\alpha})$ of*

$$DSP(y^0, \omega) = \max_{\pi \in \mathbb{R}^{|\mathscr{N}||\mathscr{K}|}, \alpha \in \mathbb{R}_+^{|\mathscr{A}|}, s \in \mathbb{R}_+^{|\mathscr{A}||\mathscr{K}|}} \sum_{k\in\mathscr{K}} d^k(\omega)\left(\pi^k_{O(k)} - \pi^k_{D(k)}\right) - \sum_{a\in\mathscr{A}} u_a y_a^0 \alpha_a \tag{24}$$

$$s.t. \qquad \pi^k_{a^-} - \pi^k_{a^+} - \alpha_a + s^k_a = c^k_a \qquad \forall a \in \mathscr{A}, k \in \mathscr{K} \tag{25}$$

$$\sum_{k\in\mathscr{K}} d^k(\omega)\left(\pi^k_{O(k)} - \pi^k_{D(k)}\right) - \sum_{a\in\mathscr{A}} u_a \bar{y}_a \alpha_a = \overline{DSP}(\bar{y}; \omega), \tag{26}$$

is Pareto-optimal.

The auxiliary problem (24–26) is solved after solving the regular SP (7–8) in order to derive the corresponding Pareto-optimal cut. The dependency of the Magnanti-Wong approach on this auxiliary problem can, however, overshadow the performance of the overall algorithm. It doubles the number of the SPs and the secondary problem is relatively more difficult to solve. We found that the latter issue is to a very large extend due to the presence of the equality constraint (26), which may entail numerical instabilities as well.

The purpose of the equality constraint (26) is to restrict the SP (7-8) to the optimal face of the dual polyhedron where all the alternate optimal solutions exist. The objective function (24) attempts to pick a solution, among the available alternatives on the optimal face, which gives the tightest cut as measured from an interior point of the MP. This equality constraint is not required, however, to extract the best dual solution. To remove it from formulation (24-26), we first recall the following definitions from linear programming theory.

Definition 4.6. *An alternate optimal solution exists if at least one nonbasic variable possesses a reduced cost of zero.*

Thus, when we identify that at least one alternate dual optima exists, we search the best one to generate the cut by restricting the SP to the optimal face; Corollary 4.1 of linear programming theory states how to fix the SP to the optimal face.

Corollary 4.1. *Variables with nonzero reduced cost maintain their current value at any alternate optimal solution.*

Proof. If a variable with nonzero reduced cost changes its value, the optimal objective value should change as well, which contradicts the definition of the alternate optima. □

From the duality theory, an active constraint on the optimal face has a nonzero dual value. Thus, each inequality constraint with nonzero dual value should be converted into an equality. This is equivalent to fixing the slack variables with nonzero reduced cost to zero. Using these definitions, Lemma 4.2 formally introduces an equivalent SP to extract Pareto-optimal cuts.

Lemma 4.2. *Let $(\vec{\alpha}, \vec{\pi}, \vec{s})$ indicate the vector of variables with nonzero reduced cost, as obtained from solving the DSP$(\bar{y};\omega)$ problem. Then, the solution of*

$$DSP(y^0;\omega) = \max_{\pi\in\mathbb{R}^{|\mathcal{N}||\mathcal{K}|},\alpha\in\mathbb{R}_+^{|\mathcal{A}|},s\in\mathbb{R}_+^{|\mathcal{A}||\mathcal{K}|}} \sum_{k\in\mathcal{K}} d^k(\omega)\left(\pi^k_{O(k)} - \pi^k_{D(k)}\right) - \sum_{a\in\mathcal{A}} u_a y^0_a \alpha_a \tag{27}$$

$$\text{s.t.} \quad \pi^k_{a^-} - \pi^k_{a^+} - \alpha_a + s^k_a = c^k_a \qquad \forall a\in\mathcal{A}, k\in\mathcal{K} \tag{28}$$

$$(\vec{\alpha}, \vec{\pi}, \vec{s}) = \mathbf{0}, \tag{29}$$

is equivalent to that obtained from the Magnanti-Wong problem.

Proof. The equivalence is evident because only variables with zero reduced cost can change their value, which does not effect the objective value of (7-8). Note that, the variables are unbounded and if they have a nonzero reduced cost, their value at the optimal solution is necessarily zero. □

The equality (29) fixes a set of variables to their current value (zero), which can efficiently be handled by the optimization solvers, e.g., CPLEX. This variable fixation also helps the solver to further reduce the scale of the auxiliary SP (27-29) through inspection and removal of the unnecessary constraints and variables.

4.2.2 Improving the optimality cuts

We make use of VIs to improve the quality of the optimality cuts as well as the LP relaxation. The underlying idea is to apply a cutting-plane method on each SP to tighten its formulation before generating the Benders cut. To do so, once the SP is solved, we can extract the value of the flow variables, denoted as $\bar{x}$. Given the current $(\bar{x}, \bar{y})$ solution, violated VIs can be extracted and added to the SP. Note that this is equivalent to adding columns to the dual SP. We show that the tightened SP yields stronger cuts, which also improves the LP relaxation bound.

Various VIs can be used to realize the aforementioned goal. We have experimented with two well-known families of inequalities, namely SIs and *flow pack* (FP) inequalities [7]. In the rest of this section, we present the theoretical results only for the case when the SIs are used. These results can be easily generalized for other VIs.

To demonstrate the impact of the SIs on the classical optimality cuts, assume that they are added to the primal formulation of the SP (3–5) and β indicates the vector of the associated dual variables. The dual formulation, referred to as *Strong Dual Subproblem* (SDSP), is

$$SDSP(\bar{y};\omega) = \max_{\pi\in\mathbb{R}^{|\mathscr{N}||\mathscr{K}|}, \alpha\in\mathbb{R}_+^{|\mathscr{A}|}, s,\beta\in\mathbb{R}_+^{|\mathscr{A}||\mathscr{K}|}} \sum_{k\in\mathscr{K}} d^k(\omega)\left(\pi^k_{O(k)} - \pi^k_{D(k)}\right) - \sum_{a\in\mathscr{A}} u_a\bar{y}_a\alpha_a - \sum_{k\in\mathscr{K}}\sum_{a\in\mathscr{A}} b^k_a(\omega)\bar{y}_a\beta^k_a \tag{30}$$

$$\text{s.t.} \quad \pi^k_{a^-} - \pi^k_{a^+} - \alpha_a - \beta^k_a + s^k_a = c^k_a \quad \forall a\in\mathscr{A}, \quad k\in\mathscr{K}. \tag{31}$$

In comparison to problem (7–8), the above formulation is larger in scale and may entail a higher degree of degeneracy. This is, however, well compensated through significant reductions in the number of iterations. Proposition 4.2 establishes the relation among the two dual formulations $SDSP(\bar{y};\omega)$ and $DSP(\bar{y};\omega)$.

Proposition 4.2. *For a given first stage solution $\bar{y}$, the relation $SDSP(\bar{y};\omega) \geq DSP(\bar{y};\omega)$ among the two formulations of the SPs always holds true.*

Proof. Consider the primal formulation of the two dual problems. The objective functions are equal but the feasible regions of the SP with the SIs, denoted $\bar{\Phi}(\bar{y};\omega)$, is a subset of the one without the SIs, denoted $\Phi(\bar{y};\omega)$. This implies that the latter is a relaxation of the former: $\bar{\Phi}(\bar{y};\omega) \geq \Phi(\bar{y};\omega)$. According to the strong duality theory, we have $SDSP(\bar{y};\omega) = \bar{\Phi}(\bar{y};\omega) \geq \Phi(\bar{y};\omega) = DSP(\bar{y};\omega)$. □

Recalling the dominance rule and the properties of the core point, we can now formally state the effect of the SIs on the optimality cuts.

Proposition 4.3. *The cut generated from $SDSP(\bar{y};\omega)$ is not dominated by the one extracted from $DSP(\bar{y};\omega)$.*

Proof. To prove the theorem by contradiction, assume that $(\bar{\pi}_1, \bar{\alpha}_1, \bar{\beta}_1) \in U(DSP(\bar{y};\omega))$ dominates $(\bar{\pi}_2, \bar{\alpha}_2, \bar{\beta}_2) \in U(SDSP(\bar{y};\omega))$, where $\beta_1 = 0$. (Note that, to lighten the presentation, we present the equations using a vector representation.) Based on Definition 4.3, we have

$$d^T\bar{\pi}_1 - \bar{\alpha}_1^T uy - b^T\bar{\beta}_1 y \geq d^T\bar{\pi}_2 - \bar{\alpha}_2^T uy - b^T\bar{\beta}_2 y \qquad \forall y\in Y, \tag{32}$$

such that there is at least one $\bar{y}\in Y$ that

$$d^T\bar{\pi}_1 - \bar{\alpha}_1^T u\bar{y} - b^T\bar{\beta}_1\bar{y} > d^T\bar{\pi}_2 - \bar{\alpha}_2^T u\bar{y} - b^T\bar{\beta}_2\bar{y}. \tag{33}$$

On the other hand, based on inequality (32) we have

$$d^T\bar{\pi}_1 - \bar{\alpha}_1^T uy^0 - b^T\bar{\beta}_1 y^0 \geq d^T\bar{\pi}_2 - \bar{\alpha}_2^T uy^0 - b^T\bar{\beta}_2 y^0 \qquad \forall y^0\in Y^c, \tag{34}$$

where Y^c is convex hull of Y. The last inequality is true because any $y^0 \in Y^c$ can be expressed as $y^0 = \sum_{t \in T} \lambda_t y_t$, $\sum_{t \in T} \lambda_t = 1$, $\lambda_t \geq 0$ for some finite number of $y_1, ..., y_{|T|} \in Y$. On the other hand, we know that for y^0 the following inequality holds according to Proportion 4.2

$$d^T \bar{\pi}_2 - \bar{\alpha}_2^T u y^0 - b^T \bar{\beta}_2 y^0 \geq d^T \bar{\pi}_1 - \bar{\alpha}_1^T u y^0 - b^T \bar{\beta}_1 y^0. \tag{35}$$

Note, $y^0 \in ri(Y^c)$ because y^0 is a Magnanti-Wong point. Based on the inequalities (34) and (35), we can write the following equality

$$d^T \bar{\pi}_1 - \bar{\alpha}_1^T u y^0 - b^T \bar{\beta}_1 y^0 = d^T \bar{\pi}_2 - \bar{\alpha}_2^T u y^0 - b^T \bar{\beta}_2 y^0. \tag{36}$$

Based on Theorem 6.4 of [43], it exists a scalar $\theta > 1$ such that $\hat{y} = \theta y^0 + (1 - \theta)\bar{y}$ belongs to Y^c. Multiplying equality (36) by θ and the strict inequality (33) by $1 - \theta$ (which reverses the inequality), and adding these two, we get the following strict inequality

$$d^T \bar{\pi}_1 - \bar{\alpha}_1^T u \hat{y} = d^T \bar{\pi}_1 - \bar{\alpha}_1^T u \hat{y} - b^T \bar{\beta}_1 \hat{y} < d^T \bar{\pi}_2 - \bar{\alpha}_2^T u \hat{y} - b^T \bar{\beta}_2 \hat{y}, \tag{37}$$

which contradicts inequality (32) and hence our assumption. □

We will numerically verify in section 5.2 that applying a cutting plane on each SP not only yields larger step-increases in the lower bound but also results in tighter LP relaxation bounds.

4.2.3 Feasibility cuts

As defined in constraints (12), the cuts associated with infeasible solutions do not improve the values of the θ variables. They also make the solution of the MP more complicated. More importantly, we observed that, in most iterations, the infeasibility of the $\bar{y}$ solution is rather limited, meaning that merely a small portion of the demands is left unsatisfied. In such cases, optimality cuts can still be generated instead of feasibility cuts which, considering our preliminary results, is a more numerically efficient strategy. To do so, one can employ the strategy proposed by [47] solving an auxiliary MILP problem in order to relax a minimum number of constraints of the subproblem so that an optimality cut can be generated. We propose a different and simpler strategy, however, which avoids solving an auxiliary MILP problem. We propose to apply a relatively complete recourse property to model (3-5), consisting of adding a dummy arc between the nodes of each OD pair to provide extra capacity at a high premium unit price. The inclusion of these dummy arcs in all recourse problems $\omega \in \Omega$ defines an outsourcing strategy enabling any arbitrary quantity of a commodity to be flowed directly between its associated origin and destination nodes. As a result, $\Phi(y; \omega)$ is always feasible.

These optimality cuts using dummy arcs may involve big-M coefficients. To avoid the insertion of such weak cuts and maintaining the global convergence, we make use of combinatorial cuts. We first identify the infeasibility of the current solution using Property 4.1.

Property 4.1. *For a given design solution $\bar{y}$, if there exists at least one SP such that a positive amount of flow is routed on a dummy arc, then $\bar{y}$ is an infeasible design to the MCFNDSD.*

When the current solution is identified as infeasible according to Property 4.1, the optimality cuts for those subproblems that involve big coefficients are not added to the MP. Instead, the following combinatorial cut is added:

$$\sum_{a \in \mathscr{A} | \bar{y}_a = 1} (1 - y_a) + \sum_{a \in \mathscr{A} | \bar{y}_a = 0} y_a \geq 1. \tag{38}$$

By adding an inequality (38) each time the MP solution is infeasible, one ensures that the original feasibility requirements defined in the problem are enforced. These combinatorial cuts can be further strengthened according to the following proposition.

Proposition 4.4. *For a given infeasible integer solution $\bar{y}$, the following inequality is globally valid.*

$$\sum_{a\in\mathscr{A}|\bar{y}_a=0} y_a \geq 1. \tag{39}$$

Proof. Let $\mathscr{A}_0$ and $\mathscr{A}_{01}$ indicate the set of binary variables fixed to 0 and the set of free variables at the current integer node of the branch-and-bound tree. We can easily conclude that the infeasibility is due to not opening enough capacity, since otherwise the dummy arcs would not have been used due to their large flow costs. This corresponds to opening at least one arc in the set of closed arcs to break the infeasibility, i.e., $a \in \mathscr{A}_0 \cup (\mathscr{A}_{01}|\bar{y}_a = 0)$. This is equivalent to $\sum_{a\in\mathscr{A}_0} y_a + \sum_{a\in\mathscr{A}_{01}|\bar{y}_a=0} y_a = \sum_{a\in\mathscr{A}|\bar{y}_a=0} y_a \geq 1$ because $\bar{y}_a = 0, \forall a \in \mathscr{A}_0$ and $\mathscr{A}_0 \cap \mathscr{A}_{01} = \emptyset$. □

4.3 Warm-Start Strategy

The BD method is very well-known for suffering from ineffective initial iterations due to the generation of low-quality solutions and zig-zagging behavior. To effectively overcome these drawbacks, we propose a *warm-start* (*WS*) strategy to generate an initial set of tight cuts. Unlike the heuristic-based warm-start strategies, the underlying idea of our approach is to deflect the current master solution. Given an initial starting point y^{ws}, the current master solution $\bar{y}$, and a convex combination weight $0 < \lambda < 1$, we deflect the current master solution according to $y^{ws} = \lambda\bar{y} + (1-\lambda)y^{ws}$. Then, y^{ws} is used to generate the cuts instead of $\bar{y}$. If the starting solution also satisfies the core point properties, we do not need to solve the auxiliary SP (24-26) to generate pareto-optimal cuts, since the deflected point guarantees the generation of a non-dominated cut (see [37], Theorem 6). The upper bound generated using the deflected solution is also valid for the LP relaxation of the problem. Therefore, as long as we are applying this strategy, no auxiliary SP is required to be solved. This yields appreciable savings in computational time. The current master solution might be infeasible and thus render the new y^{ws} infeasible as well. In this case, we solve an auxiliary problem to reset y^{ws} at a feasible solution at the vicinity of $\bar{y}$. This auxiliary problem is presented in Appendix A.

This strategy is also capable of considerably alleviating the instability issue of the MP. It dampens the initial large steps of the algorithm through taking an average with a centered solution. Thus, the y^{ws} and the whole procedure can also be interpreted as a stabilizing point and a stabilization strategy [20].

The strategy is sensitive to the initial value of the y^{ws}. We thus propose to solve a MCNF problem with maximum demand, i.e., $d_k = max_{\omega\in\Omega}\{d_k(\omega)\}, \forall k \in \mathscr{K}$. Given the solution $\tilde{y}$ of this problem, we set $y^{ws}_a = y^{max}_a := max\{0.5, \tilde{y}_a\}, \forall a \in \mathscr{A}$.

4.4 Managing the Branch-and-Bound Tree

The search tree is one of the most important parts of the algorithm, since its size directly determines its efficiency. To the best of our knowledge, there is no in-depth study in the literature to address the branch-and-bound tree in the cutting plane implementation of the BD method. We examined various cut generation strategies. The preliminary experiments indicated that generating as many cuts as possible at the root node (i.e., first phase of the algorithm) and then generating cuts merely for the potential incumbent solutions yields the best performance. Generating cuts for every or the first 100 fractional solutions inside the tree caused overly slow performance of the algorithm. The reason was due to the low impact of the cuts, generating too many of them and spending too much time on their extraction and handling. In the rest of this section, we present branching and node selection strategies, a primal heuristic, and reduction procedures.

4.4.1 Branching and node selection rules

Node and variable selection decisions are of crucial importance in our algorithm. With respect to the branching, We perform an irregular branching at the root node while, for the rest of the algorithm, we follow the default settings of the optimization solver (CPLEX). To create the first two child nodes at the top of the tree, we first identify two set of variables, $\tilde{\mathscr{A}_0}$ and $\tilde{\mathscr{A}_1}$, which include the variables that very likely take value 0 or 1 in an optimal solution. Identifying the variables in each set relies on the information gathered during the first phase of the algorithm. Variables which consistently took values smaller than ε_{branch} or larger than $1-\varepsilon_{branch}$ are respectively added to $\tilde{\mathscr{A}_0}$ and $\tilde{\mathscr{A}_1}$, such that, for the current master solution $\bar{y}$ they satisfy $\sum_{a\in\tilde{A}_1}(1-\bar{y}_a)+\sum_{a\in\tilde{A}_0}\bar{y}_a \leq 1-\varepsilon_{branch}$, with ε_{branch} a small positive number. The left branch is then created by adding two constraints of the form $\sum_{a\in\tilde{\mathscr{A}_0}} y_a = 0$ and $\sum_{a\in\tilde{\mathscr{A}_1}} y_a = |\tilde{\mathscr{A}_1}|$, which is equivalent to fixing the variables. The complement to this node indicates that at least one variable in the joint set $\tilde{\mathscr{A}_0}\cup\tilde{\mathscr{A}_1}$ should change its value. Therefore, the right branch is created by adding a combinatorial cut of the form $\sum_{a\in\tilde{\mathscr{A}_1}}(1-y_a)+\sum_{a\in\tilde{\mathscr{A}_0}} y_a \geq 1$.

The left branch has a considerably small sub-domain, which helps to quickly find good feasible solutions. The right branch has also an improved bound since it violates the root solution. As for the node selection strategy, we tested several different strategies. At the end, we decided to follow the default of the optimization solver which implements the state-of-the-art strategies.

4.4.2 Upper bounding heuristic

Our branch-and-cut algorithm is dependent on the quality of the incumbent solution to avoid searching inferior regions of the solution space. We thus propose a simple heuristic to generate tight upper bounds on the optimal integer solution at the end of first phase. The proposed heuristic, unlike the ones in the literature, depends on the BD method itself rather than the problem. Algorithm 2 presents the pseudo-code of this heuristic.

Algorithm 2 : The upper bounding heuristic

```
Extract the sets $\hat{\mathscr{A}_0}$, $\hat{\mathscr{A}_1}$, and $\hat{\mathscr{A}_{01}}$ based on the LP solution of the first phase
$\imath \leftarrow$ maximum number of iterations; $T_h \leftarrow$ time limit per iteration
RMP $\leftarrow$ derive the restricted MP through imposing $\sum_{a\in\hat{\mathscr{A}_0}} y_a = 0$ and $\sum_{a\in\hat{\mathscr{A}_1}} y_a = |\hat{\mathscr{A}_1}|$
for all $t = 0 : \imath$ do
  Solve RMP for the given time limit $T_h$
  if RMP infeasible then
    Add strong combinatorial cut $\sum_{a\in\hat{\mathscr{A}_0}} y_a \geq 1$
    Break the loop
  end if
  Evaluate the current solution, generate cuts, and add the cuts to the formulation
  Update the current best upper bound (if possible)
  if the solution was infeasible for at least one SP based on Propoerty 4.1 then
    Add strong combinatorial cut $\sum_{a\in\hat{\mathscr{A}_0}} y_a + \sum_{a\in\hat{\mathscr{A}_{01}}|\bar{y}_a=0} y_a \geq 1$
  else
    Add combinatorial cut $\sum_{a\in\mathscr{A}|\bar{y}_a=1}(1-y_a)+\sum_{a\in\mathscr{A}|\bar{y}_a=0} y_a \geq 1$
  end if
end for
Remove the fixing constraints $\sum_{a\in\hat{\mathscr{A}_0}} y_a = 0$ and $\sum_{a\in\hat{\mathscr{A}_1}} y_a = |\hat{c}A_1|$ and go to the second phase.
```

We observed that the LP solution is fairly tight and close to an optimal integer solution. Thus, a simple *fix-and-optimize* procedure can yield high quality solutions. The variables that take values smaller than 0.1 or greater than 0.9 are respectively added to the sets $\hat{\mathscr{A}}_0$ and $\hat{\mathscr{A}}_1$. The rest of the variables are added to the set of free variables $\hat{\mathscr{A}}_{01}$. To derive the *Restricted Master Problem* (RMP), variables in $\hat{\mathscr{A}}_0$ and $\hat{\mathscr{A}}_1$ are fixed to 0 and 1, respectively. The RMP is then solved. If feasible, the produced solution is evaluated to get an upper bound and a set of optimality cuts. In order to avoid revisiting that solution, a valid combinatorial inequality is also added to the RMP. If the RMP became infeasible, the heuristic procedure terminates and a strong combinatorial cut is added to the formulation. The algorithm iterates for a maximum ι iterations. A time limit of T_h is imposed on each iteration of the RMP. At the end, all the constraints imposing the bounds on the variables are removed from the model. But, all the generated optimality and combinatorial cuts will remain in the formulation, since they are both useful and valid.

4.4.3 Reduction procedures

The efficiency of the algorithm can be improved by reducing the size of the formulation. The SPs can be solved more efficiently and the search tree becomes more manageable. The reduction ideas are mainly based on the well-known reduced-cost methodology [22, 9]. One flow conservation constraint in formulation (3–5) is redundant for each commodity k. Thus, we can fix the dual variable associated to its origin (or destination) to zero, i.e., $\pi^k_{O(k)} = 0, \forall k \in \mathscr{K}$.

Many of the proposed VIs provide no further convergence advantage after a few iterations. Likewise, a small subset of the Benders cuts only are required for global convergence. Thus, a *cleanup* strategy is required to keep the scale of the MP manageable by removing the unnecessary cuts and VIs. The proposed cleanup strategy keeps a record of the slack values for each cut and inequality. If a cut or a VI exhibits a large slack value over some predefined number of iterations, it will be considered as a candidate for removal from the master formulation. In order to avoid frequent cut removal, which profoundly disturbs the optimization solver, the candidate cuts will be removed just before the second phase of the algorithm. Note that this strategy is not applied during the second phase because the cuts are appended to the master formulation as lazy constraints and the solver eliminates them if they become slack. Last but not least, the cleanup is also applied to the SPs as soon the LP phase is terminated in order to remove the additional VIs which will be inactive during the second phase, where cuts are generated only for integer y solutions.

5 Numerical Results and Analysis

In this section, we present the numerical assessment of the proposed algorithm. We first experiment different versions of the BD algorithm to evaluate the impact of the proposed algorithmic enhancements on convergence. The second part of the analysis is dedicated to a comparison between our exact algorithm and state-of-the-art algorithms and optimization solvers. We test the robustness and limitations of our method on larger instances in the third part of the experiments.

To carry out the numerical tests, we have used standard benchmark instances, referred to as **R** and **S** in the literature [15, 16, 17]. There are 16 instances in the **S** data set. These instances are defined on fully connected graphs of 16 or 30 nodes, with 14, 40 or 80 commodities, and 10, 20, 60 or 90 scenarios. For presentation purposes, we divided them into three classes based on the number of commodities, which determines the instance difficulty. Twelve instance classes of **R** family, r04 to r14 inclusively, along with five different cost/capacity ratios (1, 3, 5, 7, and 9) were selected. Three numbers of scenarios (16, 32, and 64) are considered for each instance of the **R** set. To generate these scenarios, demands were assumed to be linearly correlated and five levels of positive correlation (0%, 20%, 40%, 60%, and 80%) were used to create different instances. In summary, 825 different instances of the **R** set are considered. Attributes of the considered instance classes are described in Table 1.

Table 1: Attributes of the instance classes.

Name	$\|N\|$	$\|A\|$	$\|K\|$	$\|\Omega\|$	Cost/Capacity Ratio	Correlation	Number of Instances
s01	16, 30	$\|N\|^2$	14	10, 20	-	-	4
s02	16, 30	$\|N\|^2$	40	20, 60, 90	-	-	6
s03	16, 30	$\|N\|^2$	80	20, 60, 90	-	-	6
r04	10	60	10	16, 32, 64	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	75
r05	10	60	25	16, 32, 64	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	75
r06	10	60	50	16, 32, 64	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	75
r07	10	82	10	16, 32, 64	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	75
r08	10	83	25	16, 32, 64	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	75
r09	10	83	50	16, 32, 64	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	75
r10	20	120	40	16, 32, 64	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	75
r11	20	120	100	16, 32, 64	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	75
r12	20	120	200	16, 32, 64	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	75
r13	20	220	40	16, 32, 64	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	75
r14	20	220	100	16, 32, 64	1, 3, 5, 7, 9	0, 0.2, 0.4, 0.6, 0.8	75

All programs were coded in C++, using the CPLEX version 12.6.1.0 as the optimization solver. The code was compiled with g++ 4.8.1, executed on Intel Xeon E7-8837 CPUs running at 2.67GHz with 16GB RAM under a Linux operating system, in single-thread mode. The branch-and-cut algorithm was also implemented using the CPLEX callable libraries.

5.1 Implementation details

A number of algorithmic components need to be specified for a complete description of the implementations used to perform the numerical tests.

First, the VIs are added to the MP and SPs during the first phase of the algorithm only. LBF and NCI inequalities are added to the master formulation a priori. The other inequalities are added to the MP or SP formulation in a cutting plane method. At the MP level, our initial computational tests indicated that superior performance can be attained through prioritizing the VIs over the Benders cuts. This means that we keep solving the MP and generating the VIs as long as a violated one can be found. This considerably reduces number of the SPs to solve, which constitutes the major computational bottleneck of the algorithm. At the SP level, solving several SPs to generate a single cut is computationally costly. Moreover, in the presence of the SIs, few violated FP inequalities can be found. For this reason, the SIs are first added to the SP formulations and the FPs are appended in a cutting plane fashion. The cutting plane is executed on each SP only for one iteration per cut generation cycle.

The **S** instances involve arcs that start and end at the same node. These arcs can be removed from the network, when the associated node is not both origin and destination for a specific commodity $k \in \mathscr{K}$. This follows from the non-negative cost coefficients in the objective function. Moreover, when several commodities share the same OD pair, they can be aggregated into a single commodity if their flow costs are identical.

To specify the stopping criteria, a run time limit of 2 hours, maximum number of 1000 iterations, and optimality tolerance of 1% were considered. The time limit and optimality tolerance for the first phase were fixed to 1 hour and 0.3%. To set the search parameters, extensive computational tests on 5 chosen instances from the **R** family were conducted and the following parameters have been fixed throughout the numerical tests: $\varepsilon_{gap} = 10^{-2}$, $\varepsilon_{ci} = 10^{-5}$, $\eta = 10^{-2}$, $y^0 = 0.5$, $\lambda = 0.5$, $\varepsilon_{branch} = 10^{-1}$, $T_h = 5\%$"of total time", $\iota = 5$, $\varepsilon_h = 10^{-3}$, $T_I =$ "total iterations of the first phase $-$ 2”.

The following notation is used in Tables 2 to 7 and 9: "Time (Sec.)" gives the CPU time in seconds, "Gap (%)" presents the optimality gap in percentage, "Sol. (%)" indicates percentage of the instances solved to optimality and "Ave." represents the weighted average values. Given the upper bound UB and lower bound LB, the optimality gap is calculated as 100(UB - LB)/UB. For the sake of ease in the comparisons, when the algorithm fails to find any feasible solution, the reported optimality gap is set to 100%.

5.2 Analysis of the algorithmic enhancements

The goal of this analysis is to assess the effectiveness of the proposed algorithmic enhancements. We investigate to what degree our acceleration strategies are further enhancing the state-for-the-art BD method, as discussed in section 3.1 and outlined in [18]. Therefore, we activated the proposed strategies one by one and observed their cumulative additive value with respect to the convergence of the state-of-the-art Benders algorithm for the problem at hand [18]. For the PDS, we have implemented the best strategy reported in [18], i.e., retaining two scenarios based on the row covering strategy and creating a single artificial scenario. As suggested by the same authors, we have added the SIs and simple cardinality inequalities to the MP in order to expedite the convergence rate of the algorithm.

For the numerical analyses of this section, we have chosen a subset of the **R** and **S** instances: r04 to r10 with 64 scenarios and s01 and s02. This subset covers all the instances addressed in the literature (i.e., **R** set) and gives the opportunity to assess the behavior of the algorithm on very different network structures (i.e., **S** set).

5.2.1 Feasibility cuts

We first study the proposed strategy to avoid the generation of classical feasibility cuts. As the numerical results in Table 2 indicate, adding the relatively complete recourse to the model and using strong combinatorial cuts outperforms the use of classical feasibility cuts. This can partially be explained by the fact that most of the master solutions are slightly infeasible, meaning that the constructed network at the MP fails to cover only a small portion of the demand. In this case, the generated optimality cuts seem to improve the lower bound faster than the inclusion of feasibility cuts. Moreover, the current solutions to the MP that are infeasible are usually so for only a small subset of the subproblems. Thus, following the proposed strategy, optimality cuts are still being added for all feasible subproblems. In turn, this accelerates the rate at which the relaxation defined by the MP can be improved.

5.2.2 Pareto-optimal cuts

Table 3 displays the numerical comparison between our cut generation strategy denoted "Proposed approach (I)", and that of [30]. We also examined a variant of our approach in which the core point is dynamically updated according to $y^0 \leftarrow \frac{1}{2}(\bar{y} + y^0)$. This variant is referred to as "Proposed approach (II)".

Two key observations can be made from Table 3. First, the proposed cut generation scheme outperforms the original method of [30]. For 125 out of 175 **R** instances that both methods solve, the proposed approach reduced the average CPU time by 21.23%. For the other 50 instances, our approach improved the average optimality gap

Table 2: Assessing the performance of the strategy to avoid generating feasibility cuts

	Benders with feasibility cuts			Benders with complete recourse		
	Time (s)	Gap (%)	Sol. (%)	Time (s)	Gap (%)	Sol. (%)
r04	160.43	0.65	100.00	86.01	0.65	100.00
r05	224.16	0.79	100.00	190.57	0.76	100.00
r06	3135.42	1.52	76.00	2655.89	1.39	72.00
r07	993.86	0.91	92.00	618.58	0.85	100.00
r08	2877.81	1.86	64.00	2494.79	1.41	76.00
r09	5989.05	1.98	24.00	5602.56	1.70	36.00
r10	5879.67	3.72	20.00	5694.71	3.84	24.00
s01	56.03	0.17	100.00	56.86	0.15	100.00
s02	7200.00	54.59	0.00	7200.00	54.40	0.00
Ave. **R.**	2751.49	1.63	68.00	2477.59	1.51	72.57
Ave. **S.**	4342.41	32.82	40.00	4342.74	32.70	40.00

and CPU time by 2.32% and 9.53%, respectively. The superiority of our approach is particularly evident for the instances for which the algorithm executes more iterations involving the dummy arcs. In this case, the resolution of the auxiliary SP (24-26) is considerably slower due to the equality constraint (26).

Second, updating the core point yields a positive impact on the overall performance, requiring 2.55% less iterations, on average, to converge. This is probably due to the fact that the updated core point gives a better measure to emulate the optimal direction at the current iteration. Note that, the core point should be updated only when no dummy arc is used, i.e., the current solution is feasible according to Property 4.1.

The same observations can be made for the **S** instances. The three methods seem to have the same average running time. This is mainly due to the instances in the s02 class. They are larger in scale than those in s01, and consume the whole running time. The Magnanti and Wong [30] approach fails to find a feasible solution for 3 instances in s02, while this value for the proposed approaches reduces to 2.

We also tested the methods in [37] and [50] to generate Pareto-optimal cuts (results available from the first author), and observed that the former is less efficient. This is due to the generation of non-exact cuts with respect to the current master solution, particularly when the algorithm approaches a local optima. On the other hand, [50] generated stronger cuts than [30], but required higher CPU times. The method we propose alleviates all these shortcomings.

5.2.3 Warm start

The warm-start strategy was proposed to overcome the ineffective initial iterations. Thus, we first illustrate its impact for the first phase of the algorithm, as it is deactivated after 15 iterations or when this phase ends. The initial point in this experiment is set equal to the core point, i.e., $y^{ws} = y^0$. The comparative results are depicted in Figure 1. On average, the algorithm with the warm-start requires 53.32% less time to optimize the first phase for the **R** instances and 43.40% less for the **S** instances. More significant savings are obtained on the larger instances. It is worth mentioning that the infeasibility rate of the algorithm during this phase dropped by 75.08% and 83.10% for the **R** and **S** instances, respectively.

We next examine the impact of the warm-start strategy on the convergence of the algorithm. Two variants of

Table 3: Comparing cut generation schemes

	Magnanti & Wong [1981]			Proposed approach (I)			Proposed approach (II)		
	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)
r04	86.01	0.65	100.00	68.50	0.63	100.00	77.76	0.65	100.00
r05	190.57	0.76	100.00	136.90	0.72	100.00	158.21	0.69	100.00
r06	2655.89	1.39	72.00	2355.46	1.24	76.00	2391.80	1.26	76.00
r07	618.58	0.85	100.00	511.80	0.80	100.00	431.30	0.81	100.00
r08	2494.79	1.41	76.00	2278.20	1.36	76.00	2386.64	1.34	76.00
r09	5602.56	1.70	36.00	5376.49	1.69	36.00	5248.34	1.69	44.00
r10	5694.71	3.84	24.00	5643.28	3.38	32.00	5501.45	3.23	36.00
s01	56.86	0.15	100.00	53.07	0.12	100.00	10.68	0.03	100.00
s02	7200.00	54.40	0.00	7200.00	52.44	0.00	7200.00	51.17	0.00
Ave. **R**:	2477.59	1.51	72.57	2338.66	1.41	74.29	2313.64	1.38	76.00
Ave. **S**:	4342.74	32.70	40.00	4341.23	31.51	40.00	4324.27	30.72	40.00

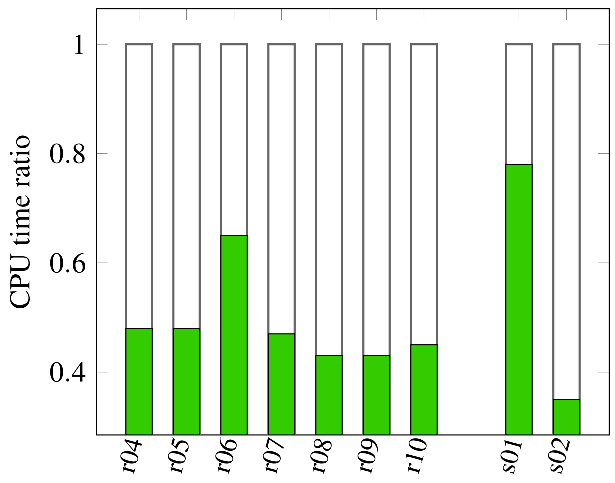

Figure 1: CPU time to solve the first phase with warm-start activated (dark bars) or not (light bars)

the algorithm with the warm-start strategy are compared: one in which the initial point is set to the core point ($y^{ws} = y^0$), and a second one in which it is set to the point proposed in section 4.3 ($y^{ws} = y^{max}$). The numerical results are summarized in Table 4.

Comparing the results to those reported in Table 3, we can confirm that the warm-start strategy has a positive impact on the performance of the BD method. The largest gap reduction was observed for the **S** instances, an average improvement of 25.74% for the s02 family. The overall time improvement may seem somehow limited in light of the results depicted in Figure 1. This can be explained by two facts. The time to solve the first phase compared to the overall time requirement is rather small, this is particularly true for the **R** instances. Moreover, many of the previously unsolved instances still remain unsolved, although with lower optimality gaps. This does not favorably affect the average time.

When the warm-start strategy is initialized with y^{max}, a larger number of the **R** instances are solved. For the **S** instances, a further optimality gap improvement of 78.63% is observed in the s02 category. The reason we observe more significant changes for the **S** instances can be explained by their fully connected network structure. For such instances, the time spent on the SPs is noticeably larger when the master solution is infeasible. Since using the warm-start significantly reduces the number of infeasible iterations, the LP phase is optimized faster: 69.97%

Table 4: The impact of the warm-start strategy on algorithm convergence given initial point setting

	WS ($y^{ws} = y^0$)			WS ($y^{ws} = y^{max}$)		
	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)
r04	63.68	0.64	100.00	72.34	0.62	100.00
r05	94.39	0.70	100.00	95.60	0.58	100.00
r06	2121.76	1.14	80.00	2124.45	1.10	80.00
r07	421.61	0.77	100.00	404.51	0.78	100.00
r08	2392.24	1.24	76.00	2395.59	1.19	76.00
r09	5195.91	1.42	44.00	4992.15	1.32	52.00
r10	5355.16	2.81	36.00	5443.17	2.85	40.00
s01	39.27	0.30	100.00	15.57	0.05	100.00
s02	7200.00	38.00	0.00	7200.00	8.12	0.00
Ave. **R**:	2234.96	1.24	76.57	2218.26	1.21	78.29
Ave. **S**:	4335.71	22.91	40.00	4326.23	4.89	40.00

on average. In addition to having a tighter root node relaxation, more time remains for the second phase of the algorithm to find better incumbent solutions. Last remark, the average time requirement for some instances, e.g., r10, has slightly increased. This indicates that although the proposed initialization performs the best on average, it could be further improved, the algorithm bing sensitive to the initial point. Thus, further research on this subject would be worthwhile.

5.2.4 Valid inequalities

Next, we examine the impact of the VIs on the performance of the algorithm. We experimented with three versions of the algorithm. Only the VIs at the SP level are activated in the first version, the VIs are only added to the MP in the second and, the VIs are applied to the both MP and SPs in the third. The numerical results are summarized in Table 5.

Table 5: Impact of the VIs on the algorithm performance

	VIs for the SPs			VIs for the MP			VIs for both SPs and MP		
	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)
r04	60.02	0.57	100.00	41.11	0.58	100.00	29.26	0.53	100.00
r05	110.34	0.39	100.00	53.34	0.57	100.00	34.40	0.38	100.00
r06	2765.27	1.29	64.00	1876.71	0.91	80.00	2687.94	0.93	80.00
r07	460.53	0.55	100.00	326.85	0.71	100.00	210.30	0.53	100.00
r08	1337.47	0.70	92.00	1220.91	0.89	96.00	551.66	0.58	100.00
r09	3174.41	1.72	64.00	4233.37	1.17	68.00	2542.66	1.49	76.00
r10	5196.56	2.40	44.00	4229.81	1.52	68.00	3967.93	1.56	64.00
s01	9.99	0.02	100.00	16.52	0.04	100.00	16.33	0.02	100.00
s02	285.93	0.11	100.00	7200.00	6.35	0.00	189.40	0.13	100.00
Ave. **R**:	1872.08	1.09	80.57	1711.73	0.91	87.43	1432.02	0.86	88.57
Ave. **S**:	175.55	0.08	100.00	4326.61	3.83	40.00	120.17	0.09	100.00

The VIs at all levels add significant value to the algorithm, as illustrated by contrasting compare these results to those reported in Table 4. In all cases, the CPU time and average optimality gaps were reduced, while the percentage of solved instances increased. The best results are attained when VIs are added at both MP and SP levels.

VIs at the MP and SP level display different behaviors for the **R** and **S** instances. In the former case, the VIs at the SP level have less impact than those at the MP level, while the opposite appears true for the **S** instances. To explain these observations, Figure 2 compares the versions 1 (VIs added to the SPs) and 2 (added to the MP) displaying on the left the relative improvement (in %) of the LP relaxation at the root node, and on the right the relative increase (in %) CPU times.

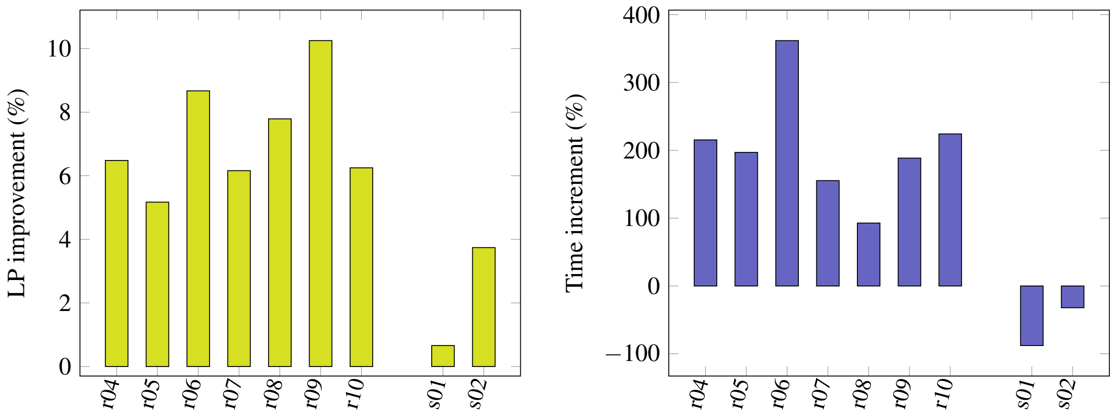

Figure 2: Comparing LP relaxations (left) and CPU times (right) when VIs are added to the SP and to the MP only

Figure 2 illustrates that adding VIs to the SPs is more effective, by more than 7.25% (on average), in tightening the LP relaxation, than adding to the MP. This, however, increases the CPU time to optimize the LP phase for the **R** instances by 204.97%. The LP improvement compared to the optimal solution is small, while the increased time requirement is very large. For the **S** family, both time and LP relaxation have improved, because the addition of the stronger optimality cuts reduces the number of iterations to optimize the LP phase by an average of 72.69%. Worth mentioning, the cutting plane at the SP level has been so effective that the lower bound at the root node is less than 1.92% and 0% from the final lower bounds for the **R** and **S** instances, respectively. The same values for the cutting plane at the MP level are 8.75% and 0.75%. When both cutting plane methods are activated, the LP relaxation is further tightened and the CPU time remains reasonably low. This has thus given the best results.

In summary, examining the **S** and **R** instances, we conclude that the VIs are more effective in dense networks. The NCI inequalities are effective only when the number of nodes is larger than the number of commodities. Moreover, the decomposition provides the capability to efficiently handle an exponential number of the well-known family of VIs, such as SIs. The LBF cuts can significantly improve the initial lower bound. Figure 3 depicts the quality of the lower bound during the initial iterations of the algorithm when these cuts are appended to the master formulation versus the case when they are ignored. To conduct this experiment, all proposed enhancements are turned off and the algorithm is ran for one iteration only. On average, the initial lower bound is 92.44% higher than the case where the LBF inequalities are not added to the MP. The average time spend on the generation of these inequalities is also less than 6.10 seconds.

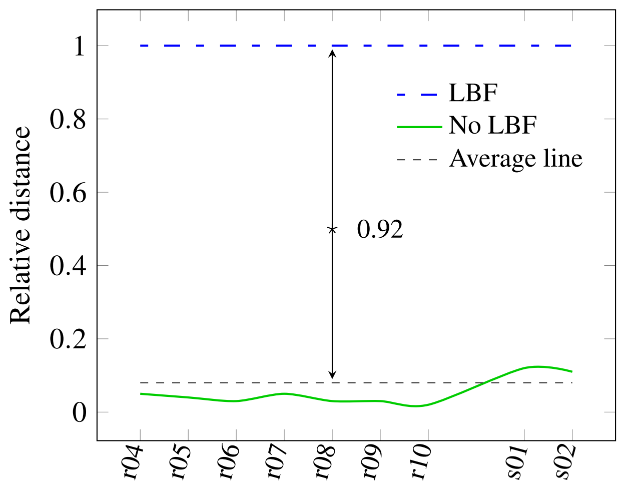

Figure 3: Improvement of the initial lower bound when the LBFs are appended

5.2.5 Heuristic

Multiple integer solutions can be extracted at each iteration of the heuristic. All these solutions are valid to generate both optimality cuts and upper bounds. We have thus examined two versions of the heuristic, using only the best solution in Heuristic (I), while Heuristic (II) uses all the extracted solutions. Figure 4 depicts the relative (in %) distance of the upper bounds attained by each version compared to the best known solution.

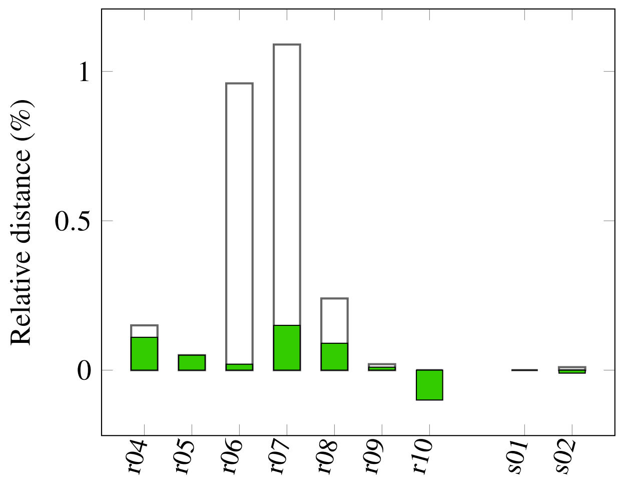

Figure 4: Relative distance of the upper bound attained by Heuristic (I) (white bars) and (II) (dark bars) from the best known value

Heuristics (I) and (II) are able to find solutions that are respectively 0.36% and 0.05% from the best known upper bound. Heuristic (I) finds a better or equal upper bound for 110 instances out of 175. This value goes up to 124 for Heuristic (II), which is, however, more time consuming by 36.29% (on average) than Heuristic (I). For the **S** instances, both heuristics consistently find the best known incumbents, and Heuristic (II) for two instances finds even better bounds.

We next examine the impact of the heuristics on the performance of the algorithm. The results for both versions are summarized in Table 6.

The algorithm with Heuristic (I) improves the average run time, while it solves a larger portion of the instances with Heuristic (II). The major advantage of the heuristics is that they remove many candidate solutions as well as a massive part of the branch-and-bound tree. If we compare the number of explored nodes with and without

Table 6: The impact of the heuristic on algorithm performance

	Heuristic (I)			Heuristic (II)		
	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)
r04	28.77	0.59	100.00	37.12	0.59	100.00
r05	34.85	0.46	100.00	46.79	0.47	100.00
r06	2474.42	1.02	80.00	2633.02	0.98	80.00
r07	174.29	0.64	100.00	218.41	0.63	100.00
r08	490.69	0.64	100.00	791.41	0.58	100.00
r09	2390.35	1.41	76.00	2373.02	1.40	80.00
r10	3990.64	1.42	64.00	4064.08	1.41	64.00
s01	41.47	0.02	100.00	17.22	0.02	100.00
s02	201.00	0.20	100.00	242.76	0.18	100.00
Ave. **R**:	1369.14	0.88	88.57	1451.98	0.87	89.14
Ave. **S**:	137.18	0.13	100.00	152.54	0.12	100.00

the heuristic, we observe an average reduction of 23.12% when the heuristic is activated. However, the heuristic does not always yield a net computational advantage, mainly because of its non-trivial running time. For example, consider **r10** for which using Heuristic (II) yields a better upper bound than the best solution previously found without the heuristic. This did not positively affect the number of solved instances, although the average optimality gap is reduced. The time spent on the heuristic accounts for 20.47% of the total running time, on average. For some instances this value goes up to 50%. Adding this to the time spent on the first phase, very limited time remains for the second phase. Thus, this time is not enough to close the optimality gap of challenging instances. When we increase the time limit to 5 hours, we observe that the algorithm with Heuristic (II) not only solves a wider range of instances but also entails lower CPU times (see Table 10 in Appendix E).

5.2.6 Value of the PDS

The initial motivation for the PDS was alleviating the weak relaxation of the MP. In this paper, this drawback has also been tackled by means of VIs and a warm-start strategy. In order to verify whether or not it is still beneficial to use the PDS in our algorithm, we have turned off the PDS feature and ran the best version of our algorithm once again. The results are presented in Table 7.

We observe that for small and medium instances, the PDS tends to increase the cost of the iterations. This is clearly due to the additional complexity it inserts into the MP. This observation relates to the instances with very tight or very loose capacity ratio. However, if the cut generation cycle is much more time consuming than the MP, the PDS usually improves the performance. This is certainly true for the **S** instances since the cut generation cycle is the most time consuming part of the algorithm. Nonetheless, the algorithm with the PDS solves a wider range of instances. The PDS thus appears a valid and efficient acceleration technique.

5.3 Comparison with other approaches

We assess the performance of our algorithm versus alternate exact approaches, i.e., CPLEX 12.6 and the algorithm of Crainic *et al.* [18]. The best strategy for each enhancement, as introduced in the previous section, is included in our algorithm. In order to examine the efficiency and robustness of the algorithms, a wider range of instances

Table 7: Impact of the PDS on convergence

	Without PDS			With PDS		
	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)
r04	25.95	0.59	100.00	37.12	0.59	100.00
r05	48.43	0.51	100.00	46.79	0.47	100.00
r06	2451.07	1.01	76.00	2633.02	0.98	80.00
r07	218.84	0.68	100.00	218.41	0.63	100.00
r08	393.53	0.64	100.00	791.41	0.58	100.00
r09	2197.37	1.71	80.00	2373.02	1.40	80.00
r10	4326.32	1.84	56.00	4064.08	1.41	64.00
s01	18.63	0.14	100.00	17.22	0.02	100.00
s02	1759.21	0.20	100.00	242.76	0.18	100.00
Ave. **R**:	1380.21	0.99	87.43	1451.98	0.87	89.14
Ave. **S**:	1062.98	0.17	100.00	152.54	0.12	100.00

was considered: 525 **R** (r04–r10) and 12 **S** (s01-s02) instances. The results are summarized in Table 8, where "TR" and "GR" indicate the time and gap ratios calculated by dividing the results of each approach by those of our algorithm. The column labeled "SD (%)" gives the number of solved instances by our approach minus the alternate approaches in %. Consequently, the bigger the value for these measures, the higher the efficiency of our algorithm.

Table 8: Comparing the proposed BD algorithm to alternate exact methods

	CPLEX 12.6			Crainic et al. [2016]		
	TR	GR	SD (%)	TR	GR	SD (%)
r04	6.17	1.10	0.00	3.78	1.61	0.00
r05	18.74	1.01	0.00	4.17	6.01	0.00
r06	10.56	3.68	0.00	6.00	2.89	18.67
r07	3.94	1.43	0.00	4.98	2.99	2.67
r08	21.80	2.19	20.00	18.88	9.22	13.33
r09	8.11	3.98	10.67	20.11	1.43	26.67
r10	2.92	8.48	25.33	2.55	5.39	38.67
s01	8.03	3.96	0.00	7.11	8.51	0.00
s02	10.30	6.15	100.00	88.52	298.46	50.00
Ave. **R**:	10.32	3.12	8.00	8.64	4.22	14.29
Ave. **S**:	9.39	5.28	60.00	55.95	182.48	30.00

The figures displayed in Table 8 illustrate that the proposed algorithm outperforms other methods for all criteria. Compared to the [18], our algorithm is 2.55 to 88.52 times faster. Compared to the CPLEX, these values range from 2.92 to 21.80. The average optimality gaps of our approach are also 1.01 to 8.48 and 1.43 to 298.46 times lower than those of CPLEX and [18], respectively. For **R** (**S**) instances, the maximum optimality gap for CPLEX, [18] and our algorithm are respectively 57.56% (11.87%), 8.88% (100%) and 4.51% (0.49%). Our algorithm is then able to solve more instances, e.g., 48 and 78 challenging instances in comparison to CPLEX and [18] (which cannot find feasible solutions for half the s02 instances), respectively.

It is interesting to observe the convergence behavior of these three exact algorithm over time as illustrated in Figure 5 for instance 49 from class r10 (stopping criteria was fixed at 10 hours run time and optimality gap of 0.03%). We see that, while CPLEX starts off with tighter bounds than [18], it terminates with the largest optimality gap. Our algorithm quickly finds tight bounds. The initial bounds are much tighter than alternate approaches and it quickly converges to an optimal solution. It is interesting to observe that the alternate methods also find the optimal solution, although at much slower peace. However, they cannot prove its optimality due to slow progression of the lower bound.

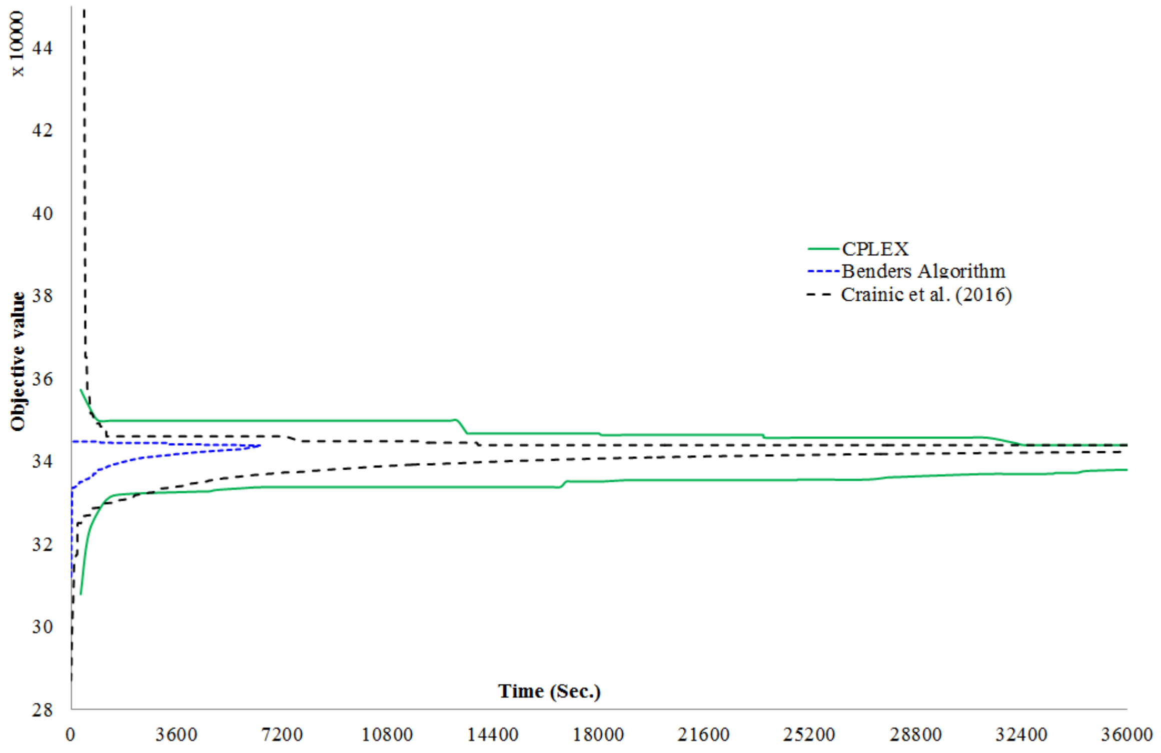

Figure 5: Convergence behavior of the exact algorithms over time

5.3.1 Large instances

We conclude the experimental study by focusing on the performance of our algorithm on larger instances, when the available computing time is also larger. We present the results for a time limit of 10 CPU hours, for the largest instance classes in our sets, r11 - r14 and s03. For a more comprehensive view of the topic, we also included the instances that remained unsolved in previous sections, i.e., r06, r09 and r10. The numerical results for the BD algorithm we propose, CPLEX and [18] are summarized in Table 9.

We observe that our algorithm performs better than other exact methods. The average optimality gap for the **R** instances for the three methods are 2.35%, 11.55% and 3.66%, respectively, while the average time requirements are 14721.59, 22017.86 and 17422.54 CPU seconds. Our algorithm is capable of solving 65.90% of the instances, more than the 50.48% of CPLEX and 59.43% by the method of [18]. It is noticeable that our proposed method is the only one able to solve all the s03 instances, and to do it in short times, which emphasizes its robustness in handling different instances.

Looking closely to the convergence behavior of the proposed algorithm on large instances, we observed some

Table 9: Computational experiments on larger instances

	Proposed BD algorithm			Cplex 12.6			Crainic et al. [2016]		
	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)
r06	2366.40	0.67	100.00	9436.76	0.57	85.33	3494.55	0.80	98.67
r09	4888.16	0.86	94.67	13983.34	1.64	78.67	6238.04	0.98	97.33
r10	10074.77	0.92	88.00	24408.12	4.65	45.33	16021.18	1.35	70.67
r11	14155.99	1.96	62.67	20576.50	11.24	60.00	17436.35	2.86	54.67
r12	13844.30	2.31	74.67	21792.24	11.95	52.00	21124.33	3.80	46.67
r13	28814.03	4.09	20.00	29440.17	19.29	20.00	28155.89	5.38	28.00
r14	28907.46	5.63	21.33	34487.90	31.54	12.00	29487.41	10.45	20.00
s03	4958.44	0.69	100.00	29682.71	5.51	33.33	36000	54.211915	0.00

difficulties for some instances, mainly due to the a less-well-performing second phase inducing long plateaus without any improvement in the bounds. We examined, in particular, the 187 **R** instances (out of 525) for which the algorithm failed to reach optimality. We observed, on average for these unsolved instances, a lower bound value at the root node of 1108493.01 at time 2043.91, and a final lower bound of 1116109.30 at time 36099.04. The very low improvement rate corresponding to these figures indicates the need for more research into improving the second phase of the BD algorithm, in order to broaden its application as an exact method to these challenging instances of the SND problem.

6 Conclusions and Remarks

We presented various acceleration strategies to boost the convergence of the Benders decomposition method. The strategies include those already certified in the literature for their positive impact on the algorithm as well as new techniques. We added two cutting-plane methods to the Benders framework to overcome three main drawbacks which make its application less effective than alternative approaches, i.e., weak linear relaxation of the problem at hand, weak optimality cuts, and weak relaxation of the master problem. To overcome the ineffective initial iterations, a warm-start strategy was used, which allows generating a set of tight cuts quickly. We also proposed to generate Pareto-optimal cuts through fixing variables in the auxiliary problem of [30] by exploiting the dual information from the solution of the primary subproblem. A simple heuristic was also developed to find high-quality incumbent solutions. To avoid extraction and inclusion of feasibility cuts, we developed relatively complete recourse property for the problem, and proposed strengthened alternate cuts to the classical feasibility ones to avoid generating optimality cuts with big coefficients.

The proposed algorithm was successfully tested on a wide range of stochastic network design instances. We observed that the proposed algorithm is at least 10 times faster than the state-of-the-art algorithms and commercial solvers. We also realized that valid inequalities at both master and subproblem level are very important. The results indicated that VIs for the SP are even more important when tight inequalities are present. Moreover, we noticed that carefully updating the core point at each iteration can improve the performance of the BD algorithm through the generation of better cuts.

There are several directions to be explored to further enhance the proposed algorithm. We observed that the algorithm is sensitive to the core point in terms of both initialization and updating. Despite the importance of this subject, there is still no theoretical guiding on how to initiate and update this point. The second phase of the algorithm takes the largest portion of the running time. Although it starts with a fairly tight upper bound, it hardly improves this bound. The lower bound also progresses very slowly. Thus, developing advanced acceleration strategies specialized for the second phase of the algorithm appears an important subject for future research. Another line of research revolves around improving the proposed heuristic so as to make it more rapid while keeping the same level of accuracy. In the same line of research, studies on selection criteria for carefully choosing solutions from a pool appear promising, in order to avoid using many useless solutions to generate bounds and cuts. Last but no least, we found great opportunities in improving the proposed algorithm by using parallelization techniques. We plan to report on this topic in the near future.

Acknowledgments

While working on this project, the second author was Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal,. Partial funding for the project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Discovery Grants program. We also gratefully acknowledge the support of the Fonds de recherche du Québec, the Canada Foundation for Innovation, and Ministère de l'Économie, de la Science et de l'Innovation -Québec through their infrastructure grants.

References

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

[2] J. R. Birge and F. Louveaux, *Introduction to stochastic programming*, Springer, New York, 1997.

[3] J. R. Birge and F. V. Louveaux, *A multicut algorithm for two-stage stochastic linear programs*, European Journal of Operational Research, 34 (1988), pp. 384–392.

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

[5] N. Boland, M. Fischetti, M. Monaci, and M. Savelsbergh, *Proximity Benders: a decomposition heuristic for stochastic programs*, Journal of Heuristics, 22 (2016), pp. 181–198.

[6] M. Chouman, T. G. Crainic, and B. Gendron, *The impact of filtering in a Branch-and-cut algorithm for multicommodity capacitated fixed charge network design*, Publication CIRRELT-2014-35, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada, 2014.

[7] M. Chouman, T. G. Crainic, and B. Gendron, *Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design*, Transportation Science, (2016), `https://doi.org/10.1287/trsc.2015.0665`.

[8] G. Codato and M. Fischetti, *Combinatorial Benders' cuts for mixed-integer linear programming*, Operations Research, 54 (2006), pp. 756–766.

[9] I. Contreras, J.-F. Cordeau, and G. Laporte, *Benders decomposition for large-scale uncapacitated hub location*, Operations research, 59 (2011), pp. 1477–1490.

[10] A. M. COSTA, *A survey on Benders decomposition applied to fixed-charge network design problems*, Computers & operations research, 32 (2005), pp. 1429–1450.

[11] G. CÔTÉ AND M. A. LAUGHTON, *Large-scale mixed integer programming: Benders-type heuristics*, European Journal of Operational Research, 16 (1984), pp. 327–333.

[12] T. G. CRAINIC, *Service network design in freight transportation*, European Journal of Operational Research, 122 (2000), pp. 272 – 288.

[13] T. G. CRAINIC AND M. FLORIAN, *National planning models and instruments*, INFOR: Information Systems and Operational Research, 46 (2008), pp. 299–308.

[14] T. G. CRAINIC, A. FRANGIONI, AND B. GENDRON, *Bundle-based relaxation methods for multicommodity capacitated fixed charge network design*, Discrete Applied Mathematics, 112 (2001), pp. 73–99.

[15] T. G. CRAINIC, X. FU, M. GENDREAU, W. REI, AND S. W. WALLACE, *Progressive hedging-based metaheuristics for stochastic network design*, Networks, 58 (2011), pp. 114–124.

[16] T. G. CRAINIC, M. HEWITT, AND W. REI, *Partial decomposition strategies for two-stage stochastic integer programs*, Publication CIRRELT-2014-13, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada, 2014.

[17] T. G. CRAINIC, M. HEWITT, AND W. REI, *Scenario grouping in a progressive hedging-based metaheuristic for stochastic network design*, Computers & Operations Research, 43 (2014), pp. 90–99.

[18] T. G. CRAINIC, M. HEWITT, AND W. REI, *Partial Benders decomposition strategies for two-stage stochastic integer programs*, Publication CIRRELT-2016-37, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada, 2016.

[19] T. G. CRAINIC, K. H. KIM, ET AL., *Intermodal transportation*, Transportation, 14 (2006), pp. 467–537.

[20] M. FISCHETTI, I. LJUBIC, AND M. SINNL, *Redesigning Benders decomposition for large-scale facility location*, Management Science, (2016), `https://doi.org/10.1287/mnsc.2016.2461`.

[21] B. GENDRON, T. G. CRAINIC, AND A. FRANGIONI, *Multicommodity capacitated network design*, Springer, 1999.

[22] B. GENDRON, M. G. SCUTELLÀ, R. G. GARROPPO, G. NENCIONI, AND L. TAVANTI, *A Branch-and-Benders-cut method for nonlinear power design in green wireless local area networks*, European Journal of Operational Research, 255 (2016), pp. 151–162.

[23] A. M. GEOFFRION, *Elements of large-scale mathematical programming: Part I: Concepts*, Management Science, 16 (1970), pp. 652–675.

[24] A. M. GEOFFRION, *Elements of large scale mathematical programming: Part II: Synthesis of algorithms and bibliography*, Management Science, 16 (1970), pp. 676–691.

[25] A. M. GEOFFRION AND G. W. GRAVES, *Multicommodity distribution system design by Benders decomposition*, Management science, 20 (1974), pp. 822–844.

[26] W. KLIBI, A. MARTEL, AND A. GUITOUNI, *The design of robust value-creating supply chain networks: A critical review*, European Journal of Operational Research, 203 (2010), pp. 283 – 293.

[27] H. LIN AND H. ÜSTER, *Exact and heuristic algorithms for data-gathering cluster-based wireless sensor network design problem*, IEEE/ACM Transactions on Networking, 22 (2014), pp. 903–916.

[28] A.-G. LIUM, T. G. CRAINIC, AND S. W. WALLACE, *A study of demand stochasticity in service network design*, Transportation Science, 43 (2009), pp. 144–157.

[29] C. LUONG, *An examination of Benders decomposition approaches in large-scale healthcare optimization problems*, master's thesis, University of Toronto, 2015.

[30] T. L. MAGNANTI AND R. T. WONG, *Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria*, Operations research, 29 (1981), pp. 464–484.

[31] T. L. MAGNANTI AND R. T. WONG, *Network design and transportation planning: Models and algorithms*, Transportation science, 18 (1984), pp. 1–55.

[32] S. MARTELLO AND P. TOTH, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, Inc., New York, NY, USA, 1990.

[33] D. MCDANIEL AND M. DEVINE, *A modified Benders' partitioning algorithm for mixed integer programming*, Management Science, 24 (1977), pp. 312–319.

[34] M. MINOUX, *Discrete cost multicommodity network optimization problems and exact solution methods*, Annals of Operations Research, 106 (2001), pp. 19–46.

[35] S. MIRHASSANI, C. LUCAS, G. MITRA, E. MESSINA, AND C. POOJARI, *Computational solution of capacity planning models under uncertainty*, Parallel Computing, 26 (2000), pp. 511–538.

[36] R. PACQUEAU, S. FRANCOIS, AND H. LE NGUYEN, *A fast and accurate algorithm for stochastic integer programming, appllied to stochastic shift scheduling*, Publication G-2012-29, Groupe d'études et de recherche en analyse des décisions (GERAD), Université de Montréal, Montréal, QC, Canada, 2012.

[37] N. PAPADAKOS, *Practical enhancements to the Magnanti–Wong method*, Operations Research Letters, 36 (2008), pp. 444–449.

[38] C. A. POOJARI AND J. E. BEASLEY, *Improving Benders decomposition using a genetic algorithm*, European Journal of Operational Research, 199 (2009), pp. 89–97.

[39] R. RAHMANIANI, T. G. CRAINIC, M. GENDREAU, AND W. REI, *The Benders decomposition algorithm: A literature review*, European Journal of Operational Research, (2016), `https://doi.org/http://dx.doi.org/10.1016/j.ejor.2016.12.005`.

[40] R. RAHMANIANI, G. RAHMANIANI, AND A. JABBARZADEH, *Variable neighborhood search based evolutionary algorithm and several approximations for balanced location–allocation design problem*, The International Journal of Advanced Manufacturing Technology, 72 (2014), pp. 145–159.

[41] G. R. RAIDL, T. BAUMHAUER, AND B. HU, *Speeding up logic-based Benders decomposition by a metaheuristic for a bi-level capacitated vehicle routing problem*, in International Workshop on Hybrid Metaheuristics, Springer, 2014, pp. 183–197.

[42] W. REI, J.-F. CORDEAU, M. GENDREAU, AND P. SORIANO, *Accelerating Benders decomposition by local branching*, INFORMS Journal on Computing, 21 (2009), pp. 333–345.

[43] R. T. ROCKAFELLAR, *Convex Analysis*, vol. 28 of Princeton Math. Series, Princeton University Press, 1970, `http://www.math.washington.edu/~rtr/papers.html`.

[44] A. RUBIALES, P. LOTITO, AND L. PARENTE, *Stabilization of the generalized Benders decomposition applied to short-term hydrothermal coordination problem*, Latin America Transactions, IEEE (Revista IEEE America Latina), 11 (2013), pp. 1212–1224.

[45] A. RUSZCZYŃSKI, *Some advances in decomposition methods for stochastic linear programming*, Annals of Operations Research, 85 (1999), pp. 153–172.

[46] G. K. SAHARIDIS, M. BOILE, AND S. THEOFANIS, *Initialization of the Benders master problem using valid inequalities applied to fixed-charge network problems*, Expert Systems with Applications, 38 (2011), pp. 6627–6636.

[47] G. K. SAHARIDIS AND M. G. IERAPETRITOU, *Improving benders decomposition using maximum feasible subsystem (mfs) cut generation strategy*, Computers & chemical engineering, 34 (2010), pp. 1237–1245.

[48] B. SANSÒ AND P. SORIANO, *Telecommunications network planning*, Springer Science & Business Media, 2012.

[49] T. SANTOSO, S. AHMED, M. GOETSCHALCKX, AND A. SHAPIRO, *A stochastic programming approach for supply chain network design under uncertainty*, European Journal of Operational Research, 167 (2005), pp. 96–115.

[50] H. D. SHERALI AND B. J. LUNDAY, *On generating maximal nondominated Benders cuts*, Annals of Operations Research, 210 (2013), pp. 57–72.

[51] G. SIERKSMA, *Linear and integer programming: theory and practice*, CRC Press, 2001.

[52] W. VAN ACKOOIJ, A. FRANGIONI, AND W. DE OLIVEIRA, *Inexact stabilized Benders' decomposition approaches to chance-constrained problems with finite support*, Applied Mathematics and Computation, 270 (2015), pp. 193–215.

[53] R. M. VAN SLYKE AND R. WETS, *L-shaped linear programs with applications to optimal control and stochastic programming*, SIAM Journal on Applied Mathematics, 17 (1969), pp. 638–663.

[54] X. WANG, T. G. CRAINIC, AND S. WALLACE, *Stochastic scheduled service network design: The value of deterministic solutions*, Publication CIRRELT-2016-14, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada, 2016.

[55] G. ZAKERI, A. B. PHILPOTT, AND D. M. RYAN, *Inexact cuts in benders decomposition*, SIAM Journal on Optimization, 10 (2000), pp. 643–657.

A Feasibility-restoration strategy

In the proposed warm start strategy, the convex combination $y^{ws} = \lambda \bar{y} + (1-\lambda) y^{ws}$ may yield an infeasible y^{ws} solution. To restore y^{ws} to a feasible solution, we solve following linear program:

$$\begin{aligned}
\min_{\hat{y} \in R_+^{|A|}} \quad & \sum_{a \in \mathscr{A}} f_a \hat{y}_a \\
\text{s.t.} \quad & \sum_{a \in i^+} x_a^k(\omega) - \sum_{a \in i^-} x_a^k(\omega) = d_i^k(\omega) && \forall i \in N, \forall k \in \mathscr{K} \\
& \sum_{k \in \mathscr{K}} x_a^k(\omega) \le u_a (\bar{y}_a + \hat{y}_a) && \forall a \in \mathscr{A} \\
& \hat{y}_a \le 1 - \bar{y}_a && \forall a \in \mathscr{A},
\end{aligned}$$

where ω is the subproblem for which y^{ws} has been infeasible. After solving this problem, y^{ws} is set equal to $\bar{y}_a + \hat{y}_a$.

B Lower-bound lifting inequality

We consider following formulation, denoted $MCNF(\omega)$, to bound the recourse cost of scenario $\omega \in \Omega$ in the master formulation.

$$\begin{aligned}
(MCNF(\omega)) \quad v(d(\omega)) := \min_{x(\omega) \in R_+^{|A||K|}} \quad & \sum_{k \in \mathscr{K}} \sum_{a \in \mathscr{A}} \left(c_a^k + \frac{f_a}{u_a} \right) x_a^k(\omega) \\
\text{s.t.} \quad & \sum_{a \in i^+} x_a^k(\omega) - \sum_{a \in i^-} x_a^k(\omega) = d_i^k(\omega) && \forall i \in N, \forall k \in \mathscr{K} \\
& \sum_{k \in \mathscr{K}} x_a^k(\omega) \le u_a && \forall a \in \mathscr{A}.
\end{aligned}$$

This formulation is equivalent to the linear programming (LP) relaxation of the deterministic multicommodity capacitated fixed-charged network design problem associated to scenario ω, since in the LP relaxation the equality $\sum_{k \in \mathscr{K}} x_a^k(\omega) = u_a y_a, \forall a \in \mathscr{A}$ is alway satisfied.

Proposition B.1. *Let $\bar{x}(\omega)$ and $\bar{y}$ be the optimal solution of problem MCNF(ω), then*

$$\theta(\omega) \ge \sum_{k \in \mathscr{K}} \sum_{a \in \mathscr{A}} c_a^k \bar{x}_a^k(\omega) + \sum_{a \in \mathscr{A}} f_a (\bar{y}_a - y_a), \tag{40}$$

is a valid cut for the Benders MP.

Proof. $MCNF(\omega)$ is a linear relaxation of the deterministic multicommodity capacitated fixed-charged network design problem associated to scenario ω and thus provides a lower bound on its optimal cost. Therefore, at any optimal solution according to the property of "wait and see" and "here and now" solutions in stochastic programming ([2]), following relation among the Benders equivalent reformulation of the extensive form and $MCNF(\omega)$ for scenario ω holds,

$$\sum_{a \in \mathscr{A}} f_a y_a + \theta(\omega) \ge \sum_{k \in \mathscr{K}} \sum_{a \in \mathscr{A}} (c_a^k + \frac{f_a}{u_a}) \bar{x}_a^k(\omega) \tag{41}$$

Accordingly to the variable transformation $\bar{y}_a = \frac{\sum_{k \in \mathscr{K}} x_a^k(\omega)}{u_a}, \forall a \in \mathscr{A}$, we have

$$\sum_{a \in \mathscr{A}} f_a y_a + \theta(\omega) \ge \sum_{k \in \mathscr{K}} \sum_{a \in \mathscr{A}} c_a^k \bar{x}_a^k(\omega) + \sum_{a \in \mathscr{A}} f_a \frac{\sum_{k \in \mathscr{K}} \bar{x}_a^k(\omega)}{u_a} = \sum_{k \in \mathscr{K}} \sum_{a \in \mathscr{A}} c_a^k \bar{x}_a^k(\omega) + \sum_{a \in \mathscr{A}} f_a \bar{y}_a \tag{42}$$

$$\to \theta(\omega) \ge \sum_{k \in \mathscr{K}} \sum_{a \in \mathscr{A}} c_a^k \bar{x}_a^k(\omega) + \sum_{a \in \mathscr{A}} f_a \bar{y}_a - \sum_{a \in \mathscr{A}} f_a y_a = \sum_{k \in \mathscr{K}} \sum_{a \in \mathscr{A}} c_a^k \bar{x}_a^k(\omega) + \sum_{a \in \mathscr{A}} f_a (\bar{y}_a - y_a), \tag{43}$$

which translates into the validity of (40). □

Solving a MCNF problem for each scenario may not computationally be interesting. For this reason, we propose to solve only a single network flow problem with minimum demand, i.e., $d_k^{min} = min_{\omega\in\Omega}\left\{d_k^{\omega}\right\}, \forall k \in \mathscr{K}$. The benefit of this auxiliary problem is that its solution gives a valid lower bound for all the scenario SPs. The obtained bound can be further improved for each scenario according to Theorem 4.1. To show the results of this theorem, we use the sensitivity analysis theory.

Let $\bar{\pi}$ be the dual variables associated to the flow conservation constraints and Δ indicate the set of alternative optimal dual solutions. The function $v(d)$ is piece-wise linear in d. Thus, $v(d^{min}+\tilde{d}) \geq v(d^{min}) + \max_{\pi\in\Delta} \pi^T\tilde{d} \geq v(d^{min}) + \bar{\pi}^T\tilde{d}$. If we set $\tilde{d} = d(\omega) - d^{min}$ for each scenario, we will have

$$\theta_\omega \geq \sum_{k\in\mathscr{K}}\sum_{a\in\mathscr{A}} c_a^k \bar{x}_a^k + \sum_{a\in\mathscr{A}} f_a(\bar{y}_a - y_a) + \sum_{k\in\mathscr{K}} (d_k(\omega) - d_k^{min})(\bar{\pi}_{O(k)}^k - \bar{\pi}_{D(k)}^k) \qquad \forall\omega\in\Omega. \tag{44}$$

This confirms the validity of the proposed inequality (15).

C Cover inequalities separation and lifting procedure

To generate a CI cut, two subsets of C_1 (open arcs) and C_0 (closed arcs) in $(\bar{N},\underline{N})$ are determined such that they satisfy following condition:

$$\sum_{a\in(\bar{N},\underline{N})\setminus(C_1\cup C_0)} u_a \geq d_{(\bar{N},\underline{N})}^{max} - \sum_{a\in C_1} u_a > 0. \tag{45}$$

To find the subsets of C_1 and C_0 which is of crucial importance, Algorithm 3 is used.

Algorithm 3 : OpenCloseArcs algorithm

```
Initialization: $U \leftarrow \sum_{a\in(\bar{N},\underline{N})} u_a, D \leftarrow d_{(\bar{N},\underline{N})}^{max}, \varepsilon_0 \leftarrow 10^{-5}, \varepsilon_1 \leftarrow 10^{-5}$
for all $a \in (\bar{N},\underline{N})$ (in arbitrary order) do
  if $\bar{y}_a \leq \varepsilon_0$ and $U - u_a \geq D$ then
    Add $a$ to $C_0$
    Close $a$ by setting $U \leftarrow U - u_a$
  end if
  if $\bar{y}_a \geq 1 - \varepsilon_1$ and $D - u_a > 0$ then
    Add $a$ to $C_1$
    Open $a$ by setting $D \leftarrow D - u_a$ and $U \leftarrow U - u_a$
  end if
end for
```

The algorithm uses U and D to represent the residual capacity and residual demand, respectively. Given the current fractional solution, $\bar{y}$, the procedure attempts to close an arc a with small $\bar{y}_a$ (as measured by ε_0) such that the residual capacity after closing that arc still covers the residual demand D, i.e., $U - u_a \geq D$. Similarly, the algorithm tries to open arc a with large $\bar{y}_a$ (as measured by $1-\varepsilon_1$) and such that there is still some residual demand to cover. To obtain a violated CI, if there is any, one needs to solve following optimization problem,

$$Z_{sep} := min \sum_{a\in(\bar{N},\underline{N})\setminus(C_1\cup C_0)} \bar{y}_a Z_a \tag{46}$$

$$s.t: \sum_{a\in(\bar{N},\underline{N})\setminus(C_1\cup C_0)} u_a Z_a \geq \sum_{a\in(\bar{N},\underline{N})\setminus C_0} u_a - d_{(\bar{N},\underline{N})}^{max} \tag{47}$$

$$Z_a \in \{0,1\} \quad \forall a \in (\bar{N},\underline{N})\setminus(C_1\cup C_0). \tag{48}$$

Which finds a cover set C over the restricted cutset (i.e., $(\bar{N},\underline{N})\backslash(C_1 \cup C_0)$), where Z_a is 1 if arc a is selected to be in the cover C, and 0 otherwise. Then, if $Z_{sep} < 1$ a violated CI is found. Solving this problem may collectively be quite time consuming as it has to be solved repeatedly. Thus, we use a heuristic which considers the arcs in non-decreasing order of the $\bar{y}_a$ value. Ties are broken by considering the arcs in non-increasing order of their capacity. Once a violated CI is obtained, it is easy to derive a minimal cover set from it, by removing as many as possible arcs with large $\bar{y}_a$ in order to meet the required condition, i.e., $\sum_{a\in C}\bar{y}_a < 1$.

To ensure the validity and also further strengthen the derived inequality, a lifting procedure is needed to be applied. The main idea of lifting is to include other variables that are not present in the cover set C. To do so, we associate a lifting coefficient λ_a to each $a \in (\bar{N},\underline{N})\backslash C$. Once the set of open and close arcs are determined, the lifting procedure is applied to the variables one-by-one. The procedure, first applies a lifting down on variables in $(\bar{N},\underline{N})\backslash(C \cup C_0)$, and then a lifting up on variables in C_0.

At a given step of the procedure, suppose we are lifting y_r. Then let L indicate the set of variables in $(\bar{N},\underline{N})\backslash C$ that has already been lifted; $\tilde{C} = (\bar{N},\underline{N})\backslash(C\cup C_0)$; $\lambda_a = 1, \forall a \in C$; and $\bar{d} = d^{max}_{(\bar{N},\underline{N})} - \sum_{a\in\tilde{C}\backslash L} u_a$. To *lift down* variable $y_r \in (\bar{N},\underline{N})\backslash(C \cup C_0)$, we solve following 0-1 knapsack problem

$$Z_{opt} := min \sum_{a\in C\cup L} \lambda_a y_a \tag{49}$$

$$s.t. \quad \sum_{a\in C\cup L} u_a y_a \geq \bar{d} + u_r \tag{50}$$

$$y_a \in \{0,1\} \quad \forall a \in C \cup L. \tag{51}$$

If this problem is feasible, the lifting coefficient is $\lambda_r = Z_{opt} - 1 - \sum_{a\in L\backslash C_0}\lambda_a$, otherwise, $\lambda_r = \sum_{a\in L\cup C}\lambda_a - \sum_{a\in L\backslash C_0}\lambda_a$. To *lift up* variable $y_r \in C_0$, following 0-1 knapsack has to be solved

$$Z_{opt} := min \sum_{a\in C\cup L} \lambda_a y_a \tag{52}$$

$$s.t. \quad \sum_{a\in C\cup L} u_a y_a \geq \bar{d} - u_r \tag{53}$$

$$y_a \in \{0,1\} \quad \forall a \in C \cup L. \tag{54}$$

If this problem is feasible, the lifting coefficient is given by $\lambda_r = 1 + \sum_{a\in L\backslash C_0}\lambda_a - Z_{opt}$; otherwise, we set $\lambda_r = \sum_{a\in L\backslash C_0}\lambda_a - \sum_{a\in C\cup L}\lambda_a$.

Since the lifting coefficient have small values, the 0-1 knapsack problems are solved efficiently using a dynamic programming algorithm ([32]). Last but not least, the order of lifting has a direct impact on the quality of the extracted cut. Thus, lifting down the variables in $(\bar{N},\underline{N})\backslash(C \cup C_0)$ is accomplished before lifting up for the variables in C_0. Also, when lifting down the variables in $(\bar{N},\underline{N})\backslash(C \cup C_0)$, variables with fractional value are lifted in non-decreasing order of their current value. Ties are broken by considering the arcs in non-increasing order of their capacity. When lifting up the variables in C_0, the exact opposite is done. As a result, following *lifted cover inequality* is obtained

$$\sum_{a\in(\bar{N},\underline{N})\backslash C} \lambda_a y_a + \sum_{a\in C} y_a \geq 1 + \sum_{a\in(\bar{N},\underline{N})\backslash(C\cup C_0)} \lambda_a. \tag{55}$$

D Minimum cardinality inequalities separation and lifting procedure

Let denote C_1 as the set of open arcs and C_0 the set of closed arcs as obtained by the OpenCloseArcs Algorithm 3. To find the lest number of arcs in $(\bar{N},\underline{N})\backslash(C_1 \cup C_0)$, let $l_{(\bar{N},\underline{N})\backslash(C_1\cup C_0)} = \max\left\{h : \sum_{a=1,..,h} u_a < d^{max}_{(\bar{N},\underline{N})\backslash(C_1\cup C_0)}\right\} + 1$ such that arc capacities are ordered in a non-increasing fashion $u_a \geq u_{a+1}$. We then seek lifting coefficients of λ

in the following inequality,

$$\sum_{a\in C_1\cup C_0} \lambda_a y_a + \sum_{a\in(\bar{N},\underline{N})\setminus(C_1\cup C_0)} y_a \geq l_{(\bar{N},\underline{N})\setminus(C_1\cup C_0)} + \sum_{a\in C_1} \lambda_a. \tag{56}$$

The lifting procedure is similar to that of LCI. Let L indicate the set of arcs lifted so far; $\tilde{C} = (\bar{N},\underline{N})\setminus(C_1 \cup C_0)$; $\lambda_a = 1, \forall a \in \tilde{C}$; and $\bar{d} = d^{max}_{(\bar{N},\underline{N})} - \sum_{a\in C_1\setminus L} u_a$. Then, to *lift down* variable y_r, the following 0-1 knapsack problem has to be solved,

$$Z_{opt} := min \sum_{a\in\tilde{C}\cup L} \lambda_a y_a \tag{57}$$

$$s.t. \quad \sum_{a\in\tilde{C}\cup L} u_a y_a \geq \bar{d} + u_r \tag{58}$$

$$y_a \in \{0,1\} \quad \forall a \in \tilde{C} \cup L. \tag{59}$$

If this problem is feasible, the lifting coefficient will be $\lambda_r = Z_{opt} - l_{(\bar{N},\underline{N})\setminus(C_1\cup C_0)} - \sum_{a\in L\setminus C_0} \lambda_a$; otherwise, it is $\lambda_r = \sum_{a\in\tilde{C}\cup L} \lambda_a - l_{(\bar{N},\underline{N})\setminus(C_1\cup C_0)} - \sum_{a\in L\setminus C_0} \lambda_a$. To *lift up* variable $y_r \in C_0$ following 0-1 knapsack problem has to be solved

$$Z_{opt} := min \sum_{a\in\tilde{C}\cup L} \lambda_a y_a \tag{60}$$

$$s.t. \quad \sum_{a\in\tilde{C}\cup L} u_a y_a \geq \bar{d} - u_r \tag{61}$$

$$y_a \in \{0,1\} \quad \forall a \in \tilde{C} \cup L. \tag{62}$$

If it is feasible, the coefficient will be $\lambda_r = l_{(\bar{N},\underline{N})\setminus(C_1\cup C_0)} + \sum_{a\in L\setminus C_0} \lambda_a - Z_{opt}$; otherwise, $\lambda_r = l_{(\bar{N},\underline{N})\setminus(C_1\cup C_0)} + \sum_{a\in L\setminus C_0} \lambda_a - \sum_{a\in\tilde{C}\cup L} \lambda_a - 1$. Note, lifting down the variables in A_1 must be accomplished before lifting up those in C_0. When lifting down, those with fractional value at current solution must be lifted first, in non-decreasing order of their current value. Ties are broken by considering the arcs in order of their capacity in a non-increasing fashion. To lift up variables in C_0, the exact opposite is performed.

E Additional numerical results

Table 10: Impact of Heuristic (II) on the proposed algorithm for a time limit of 5 hours

	Without heuristic			With heuristic		
	Time (Sec.)	Gap (%)	Sol. (%)	Time (Sec.)	Gap (%)	Sol. (%)
r04	41.66	0.58	100.00	35.28	0.59	100.00
r05	48.91	0.48	100.00	43.61	0.49	100.00
r06	5328.76	0.84	80.00	4664.60	0.78	92.00
r07	292.84	0.62	100.00	203.75	0.64	100.00
r08	1046.74	0.63	100.00	708.95	0.59	100.00
r09	6194.76	1.35	72.00	4784.35	1.28	80.00
r10	8227.40	1.38	64.00	7783.09	1.25	68.00
s01	18.21	0.14	100.00	16.65	0.14	100.00
s02	223.59	0.15	100.00	214.63	0.18	100.00
Ave. **R**:	3025.87	0.84	88.00	2603.38	0.80	91.43
Ave. **S**:	120.90	0.14	100.00	115.64	0.16	100.00