Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation

A Reduced Cost-based Restriction and Refinement Matheuristic for Stochastic Network Design

Fatemeh Sarayloo
Teodor Gabriel Crainic
Walter Rei

April 2019

CIRRELT-2019-15

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

A Reduced Cost-based Restriction and Refinement Matheuristic for Stochastic Network Design[ł]

Fatemeh Sarayloo[1,2], **Teodor Gabriel Crainic**[1,3,*], **Walter Rei**[1,3]

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

[2] Department of Computer Science and Operations Research, Université de Montréal, P.O. Box 6128, Station Centre-Ville, Montréal, Canada H3C 3J7

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

Abstract. We propose a solution approach for stochastic network design problems with uncertain demands. We investigate how to efficiently use reduced cost information as a means of guiding variable fixing to define a restriction that reduces the complexity of solving the stochastic model without sacrificing the quality of the solution obtained. We then propose a matheuristic approach that iteratively defines and explores restricted regions of the global solution space that have a high potential of containing good solutions. Extensive computational experiments show the effectiveness of the proposed approach in obtaining high-quality solutions, while reducing the computational effort to obtain them.

Keywords: Stochastic capacitated network design, uncertain multicommodity demand, two-stage formulation, reduced-cost based guidance, matheuristic.

Acknowledgements. While working on this project, T.G. Crainic was Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal. Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Discovery Grant program, and by the Fonds de recherche du Québec – Nature et technologies (FRQNT) through its strategic center infrastructure grant program.

[ł]Revised version report CIRRELT-2018-32

Results and views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect those of CIRRELT.

Les résultats et opinions contenus dans cette publication ne reflètent pas nécessairement la position du CIRRELT et n'engagent pas sa responsabilité.

---

* Corresponding author: teodorgabriel.crainic@cirrelt.net

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2019

© Sarayloo, Crainic, Rei and CIRRELT, 2019

1 Introduction

The *Multicommodity Capacitated Fixed-charge Network Design* (*MCFND*) formulation represents a generic model that can be used to formulate problems in a variety of applications such as transportation, logistics and telecommunications (Magnanti and Wong, 1984; Minoux, 1989; Crainic, 2000). In these applications, it is required to design a capacitated network to be used to route a given set of commodities in order to satisfy known demands between origin-destination pairs. In doing so, one pays not only a routing cost proportional to the number of distributed units of each commodity moved over a network arc, but also the fixed cost whenever an arc is used. The main goal of MCFND problems is to find the optimal design (i.e., selected arcs to be included in the final network) that minimizes the total cost, computed as the sum of the fixed and routing costs.

Stochastic MCFND (*SMCFND*) under demand uncertainty has received increasing attention in recent years. In this paper, we address the SMCFND as a two-stage stochastic program in which design decisions are made in the first stage before demands are observed. Once demands are observed, second-stage (routing) decisions are made to adapt the first stage solution to the observed demands. We represent the demand uncertainty using the well-known scenario-based approach where the uncertain demand is modeled via a finite number of discrete scenarios together with their associated probabilities. The SMCFND problem then becomes a mixed integer program of generally very large dimensions, that is extremely hard to solve using state-of-the-art solvers in all but trivial cases.

Stochastic network design problems are notoriously complex and difficult to address. Not surprisingly, researchers investigated how the solution to the deterministic model relates to its stochastic counterpart. It has been shown that, despite the fact that the solution to the deterministic model behaves badly in stochastic settings (Wallace, 2000; Higle and Wallace, 2003), there are situations in which the deterministic solution shares some properties with the corresponding stochastic solution (Lium et al., 2009; Thapalia et al., 2011, 2012a,b). These authors conclude that the deterministic solution carries useful information (i.e., some structural patterns) that can be leveraged to solve the stochastic case. Specifically, Crainic et al. (2017) investigated how the *reduced cost* associated with non-basic variables in deterministic solutions can be used to guide the selection of variables to exclude from the stochastic formulation. The authors did not, however, study network design formulations.

Inspired by these insights, our first goal is to investigate how to efficiently use reduced cost information extracted from the solution obtained by the deterministic (expected value) problem, as a means of guiding variable fixing, to define a good restriction that reduces the complexity of solving the SMCFND. Furthermore, we study how to improve the variable fixation performance by proposing a number of strategies in which reduced cost information are extracted from different solutions obtained by upgrading the expected value solution. Our final purpose is then to incorporate the hints derived from the analysis of the proposed variable fixation strategies, exploiting reduced cost information, into an iterative matheuristic approach, to efficiently deal with difficult stochastic instances.

The contributions of this paper is threefold. First, we propose a number of different strategies to investigate how to use the deterministic (expected value) solution and efficiently extract reduced cost information to define an appropriate restriction, without sacrifying the quality of the solution obtained. Second, we propose a new matheuristic approach which jointly makes use of a state-of-the-art commercial solver and the insights derived from the analysis of the proposed variable fixing strategies. The proposed matheuristic iteratively defines and explores restricted regions of global solution space that have a high potential of containing good (hopefully, optimal) solutions. The restricted problem, at each iteration, is defined by exploiting reduced costs information extracted from multiple solutions. Third, we carry out extensive computational experiments on large number of benchmark instances in the stochastic network design problem literature. The results show that the proposed algorithm is highly efficient in finding good-quality solutions for very difficult available instances in the literature.

The rest of the paper is organized as follows. We recall the two-stage formulation of the stochastic network design problem in Section 2, and briefly review some relevant literature in Section 3. Section 4 introduces the proposed matheuristic. Finally, we present and analyze the experimental results in Section 5 and provide concluding remarks in Section 6.

2 Problem description

The two-stage stochastic formulation, or the *a priori optimization model* (Birge and Louveaux, 2011), is a stochastic modeling approach in which decision variables are divided into two groups; namely, first stage and second stage variables. Traditionally, in the case of two-stage stochastic network design problems with uncertain demands, the first stage involves decisions on the configuration of the network (i.e., design decisions), and the second-stage consists of determining the commodity flow distribution of the observed demands in an optimal fashion based on the configuration imposed by the first stage.

Let us describe the two-stage stochastic formulation for the SMCFND problem (Crainic et al., 2011). Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a directed network with $\mathcal{N}$ representing a finite set of nodes and $\mathcal{A}$ a finite set of potential arcs. The set of commodities is represented by $\mathcal{K}$ where each is recognized by a unique pair of origin-destination nodes $(o(k), s(k))$. For each design arc $(i, j) \in \mathcal{A}$, we define the fixed cost f_{ij} incurred if the arc is included in the final design and the capacity u_{ij} limiting the total commodity flow that may use the arc (i, j). We also define the unit routing cost c_{ij} arc $(i, j) \in \mathcal{A}$.

We assume the finite scenario set $\mathcal{S}$ with the strictly positive corresponding probabilities of $p^1, \ldots, p^{|\mathcal{S}|}$. For a given scenario $s \in \mathcal{S}$, assuming that d^{ks} is the demand volume of commodity k under the scenario s, the demand of costumer i for commodity k under the scenario s, i.e., d_i^{ks}, is either set to d^{ks} if node i is the origin of commodity k, $-d^{ks}$ if node i is the destination of commodity k, or 0 otherwise.

Let the design variable y_{ij} be a binary variable, which indicates if arc (i, j) is included in the network, in the first stage. Once demands are realized, in the second-stage, x_{ij}^{ks} is the amount of commodity k's demand in the resulting solution for scenario s that flows on arc (i, j). The so-called extensive form of the two-stage stochastic program may be written as follows:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in\mathcal{A}} f_{ij} y_{ij} + \sum_{s\in\mathcal{S}} p^s \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij}^{ks} && && (1)\\
\text{subject to} \quad & \sum_{j\in\mathcal{N}^+(i)} x_{ij}^{ks} - \sum_{j\in\mathcal{N}^-(i)} x_{ji}^{ks} = d_i^{ks}, && \forall i \in \mathcal{N},\ \forall k \in \mathcal{K},\ \forall s \in \mathcal{S} && (2)\\
& \sum_{k\in\mathcal{K}} x_{ij}^{ks} \le u_{ij} y_{ij}, && \forall (i,j) \in \mathcal{A},\ \forall s \in \mathcal{S} && (3)\\
& y_{ij} \in \{0, 1\}, && \forall (i,j) \in \mathcal{A} && (4)\\
& x_{ij}^{ks} \ge 0, && \forall (i,j) \in \mathcal{A},\ \forall k \in \mathcal{K},\ \forall s \in \mathcal{S} && (5)
\end{aligned}
$$

The objective function (1) minimizes the total system cost, consisting of the sum of the fixed cost for the included arcs and the expectation of routing costs taken over all the demand scenarios. Constraints (2) represent the flow conservation equations in each scenario, requiring that demand of commodity $k \in \mathcal{K}$ is routed from its origin node to its destination. Constraints (3) ensure that the same design is used in each scenario, and that arc capacity u_{ij} is never violated. Constraint (4) and (5) impose integrality and non-negativity restrictions on decision variables. We refer to this problem as the MCFND(S).

3 Literature review

The existing methodologies for stochastic network design problems are mostly based on decomposition strategies. There are two major groups of decomposition methods for stochastic integer programs: by stage and by scenario. The L-shape method is a stage-wise decomposition method, introduced by Van Slyke and Wets (1969), which has been used to develop various solution methods for stochastic problems. For completeness, detailed review on this type of decomposition approach for SMCFND may be found in Crainic et al. (2016) and Rahmaniani et al. (2017). The progressive hedging (PH) method for addressing stochastic linear programs is a scenario-wise decomposition technique that was originally proposed in Rockafellar and Wets (1991). The PH algorithm is the foundation of a number of heuristic methods for SMCFND problems (e.g., Crainic et al., 2011, 2014).

The other approach in the literature to deal with the difficulty of stochastic programs relies on considering the deterministic version and studying its solution structure to investigate its relationship with its stochastic counterpart. It is well known that solutions to deterministic formulations tend to behave badly in stochastic settings. Despite this, a number of research studies have shown that there are problems where the deterministic solution shares some properties with the corresponding stochastic solution, irrespective of their quality in terms of objective function. For example, Thapalia et al. (2011, 2012a,b) have shown that for the single-commodity network design problem, certain structural patterns from the deterministic solution reemerge in the stochastic solution, despite the fact that the value of stochastic solution (VSS) is high. (The VSS is a standard metric proposed in Birge and Louveaux (2011) which measures the expected gain from solving a stochastic model rather than its deterministic counterpart). Similar observations were made by Wang et al. (2018) for scheduled network design problems. Maggioni and Wallace (2012) analyzed the quality of the deterministic solution in terms of its structure and upgradeability to the stochastic solution in a set of stochastic programs of different types. In follow-up work to analyze the quality of the deterministic solution, Crainic et al. (2017) studied how reduced costs can be used as a measure to identify which variables should be excluded from the stochastic problem. This study concluded that reduced costs can indeed be used to efficiently identify properties from deterministic solutions that should be included in stochastic solutions. Following these insights, in the context of the SMCFND problem, we aim to exploit reduced cost information extracted from different solutions to be used as a measure to identify sets of 0 and 1 design variables to be fixed in the stochastic problem, leading to reduced-size restricted problems. This would help in algorithmic developments providing means to efficiently address large instances.

In recent years, increasing attention has been devoted to the integration, or hybridization, of metaheuristics with mathematical programming as a efficient algorithmic approach. This approach, referred to as *matheuristics*, appears very promising exploiting the synergies of mathematical programming and metaheuristics (see, Raidl (2006); Puchinger and Raidl (2005) for a survey and a taxonomy). With the expansion of general-purpose MIP solvers over the last decade, various hybridization of heuristic methods (e.g., variable fixing techniques) with commercial MIP solvers have become increasingly popular. Several matheuristic approaches to complex combinatorial problems use the idea of fixing the value for some variables as a "problem reduction" technique in order to reduce the analysis of a whole solution space to a promising region. Examples of such approaches can be found for Knapsack Problems (e.g., the core algorithm proposed by Balas and Zemel (1980) and the kernel search proposed by Angelelli et al. (2010)) and in the context of routing problems (e.g., Archetti et al. (2008) and De Franceschi et al. (2006)), where mixed integer linear programming models are solved to thoroughly explore promising regions of the solution space.

Such effective problem reduction techniques in an iterative matheuristic appear useful for stochastic problems because of their complexity and size. However, little effort has been devoted in the stochastic literature to designing such matheuristic methods. For example, Sarayloo et al. (2018) proposed an iterative matheuristic based on the problem reduction technique, in which learning techniques were used to generate a series of MIP subproblems as restricted regions. It should be noted that in Sarayloo et al. (2018), the restriction consists of fixing variables only to 1. The main question, therefore, is how to further develop the idea of fixing variables to define more restricted regions at each iteration by identifying sets of 0 and 1 design variables.

Our aim in this paper is to design a matheuristic approach by applying a problem reduction technique, fixing variables to 0 and 1 (i.e., inclusion and exclusion of variables), to further reduce the size of sub-problems and take advantage of the strong search capabilities of CPLEX as a black-box solver. We propose such a methodology in the next section.

4 The proposed matheuristic

The basic idea of our proposed method is to solve in an iterative fashion a series of restricted problems which are constructed by exploiting reduced cost information extracted from different solutions. At each iteration, we identify two distinct subset of design variables to be fixed to either 1 or 0, leading to a reduced-size model. The resulting restricted problems are then solved by a MIP solver. We believe that using a refined approach to set the fixed variables is crucial to the algorithm's success. Therefore, we study how reduced cost information extracted from the solution obtained by the LP relaxation of the EV problem can be leveraged so as to guide variable fixation within MCFND(S) formulation.

In the following section 4.1, we present a number of strategies to examine how we can identify the desired set of variables to fixed based on reduced cost information. The detailed algorithm will then be explained in Section 4.2.

4.1 Reduced cost-based variable fixing strategies

In this section, we propose several strategies to study how to efficiently exploit reduced cost information extracted from the solution obtained for the deterministic (expected value) problem as a means of guiding variable fixing in the context of stochastic network design. We consider two main factors within each strategy, namely the choice of solution for which we extract the reduced cost, and the choice of variables (i.e. design variables or flow variables). By considering these factors, we design and examine different strategies to efficiently determine the desirable set of fixed variables. In the following, we describe our proposed strategies.

Strategy 1. We first follow the variable fixing method proposed in Crainic et al. (2017). Let $\bar{\mathbf{s}} = (\bar{\mathbf{y}}, \bar{\mathbf{x}})$ be the optimal solution of the LP relaxation of the expected value (EV) problem. We recall that the EV solution is obtained by considering the expected values of the random demand variables (i.e., $d_i^k = \bar{d}_i^k$) and solving the following deterministic (single-scenario) program (DSSP):

$$\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} f_{ij}y_{ij} + \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij}^k \tag{6}$$

$$\text{subject to} \quad \sum_{j\in\mathcal{N}^+(i)} x_{ij}^k - \sum_{j\in\mathcal{N}^-(i)} x_{ji}^k = d_i^k, \qquad \forall i \in \mathcal{N},\ \forall k \in \mathcal{K} \tag{7}$$

$$\sum_{k\in\mathcal{K}} x_{ij}^k \leq u_{ij}y_{ij}, \qquad \forall (i,j) \in \mathcal{A} \tag{8}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in \mathcal{A} \tag{9}$$

$$x_{ij}^k \geq 0 \qquad \forall (i,j) \in \mathcal{A},\ \forall k \in \mathcal{K} \tag{10}$$

Thus, in this strategy, the considered solution is $\bar{\mathbf{s}}$ which is derived from DSSP (6)-(10), where the integrality constraints are relaxed, and the choice of variable is the design variables. Let $J_0^{\bar{s}} = \{1, 2, \ldots, a, \ldots, a_{max}\}$ represent the index set of zero design variables (i.e., $\bar{y}_a = 0, a \in J_0^{\bar{s}}$ in the solution $\bar{\mathbf{s}}$) and $R_y^{\bar{s},0} = \{r_1, \ldots, r_a, \ldots, r_{a_{max}}\}$ be the set of *reduced costs* with respect to the components $\bar{y}_a, a \in J_0^s$. The set $R_y^{\bar{s},0}$ is then sorted in non-decreasing order. Let $r^{max} = \max_{a\in J_0^{\bar{s}}}\{r_a : r_a \in R_y^{\bar{s},0}\}$ and $r^{min} = \min_{a\in J_0^{\bar{s}}}\{r_a : r_a \in R_y^{\bar{s},0}\}$ be respectively the maximum and the minimum of the reduced costs of the set $R_y^{\bar{s},0}$. Following this strategy, to determine the groups of variables to be fixed, the difference $r^{max} - r^{min}$ is divided into N_0 classes of constant size of $\frac{r^{max} - r^{min}}{N_0}$. We then solve the model (1)-(5) by fixing to 0 the variables belonging to the classes from p_0 to N_0 where, $1 \leq p_0 \leq N_0$.

Strategy 2. To evaluate the effect of using an improved solution in producing a good set of fixed variables, we try to upgrade the solution of the EV problem (in which the integrality constraints are imposed), thus obtaining $\mathbf{s}^{EV} = (\mathbf{y}^{EV}, \mathbf{x}^{EV})$. To do so, we use the expected value solution as an input to the MCFND(S) model (1)-(5) by adding the constraints $y \geq y^{EV}$ and then solve its LP relaxation, yielding the solution $\mathbf{s}' = (\mathbf{y}', \mathbf{x}'^{s_1}, \ldots, \mathbf{x}'^{s_{|S|}})$. Thus, in this strategy, the considered solution is $\mathbf{s}'$ and the choice of variable is the design variables $\mathbf{y}'$. Let $J_0^{s'}$ represents the index set of zero design variables, i.e., $y'_a = 0$, in the solution $\mathbf{s}'$, and $R_y^{s',0}$ be the set of reduced costs with respect to the components $\mathbf{y}'_a$ $a \in J_0^{s'}$. The same approach, as in Strategy 1, is then applied here to obtain the restricted model.

Strategy 3. In this strategy, we try to upgrade the solution of EV problem, $\mathbf{s}^{EV} = (\mathbf{y}^{EV}, \mathbf{x}^{EV})$, to improve it even further than Strategy 2. To evaluate the effect of improving the solution obtained by the EV problem on producing a good set of fixed variables, we produce a feasible solution to the MCFND(S) model (1)-(5). To do so, we use the solution obtained by the EV problem as a input to the model (1)-(5) by adding the constraints $y \geq y^{EV}$ and then solve the problem to obtain the upgraded solution $\mathbf{s}'' = (\mathbf{y}'', \mathbf{x}''^{s_1}, \ldots, \mathbf{x}''^{s_{|S|}})$. Thus, in this strategy, the considered solution is $\mathbf{s}''$ and the choice of variables is the design variables $\mathbf{y}''$. Let $J_0^{s''}$ represent the index set of zero design variables, i.e., $y''_a = 0$, in the solution $\mathbf{s}''$, and $R_y^{s'',0}$ be the set of reduced costs with respect to the components y''_a, $a \in J_0^{s''}$. It should be noted that, given the fact that we are solving the MCFND(S) model (1)-(5) with the integrality requirements, we need to perform one additional step to obtain the reduced cost values. Once the problem (1)-(5) is solved and its optimal (integer) solution, $\mathbf{s}''$, is obtained, we will then need to solve the LP relaxation of the problem (1)-(5) while the design variables are fixed to the values of the obtained optimal solution. In this way, one can obtain the set of reduced cost values associated to the design variables. Using the set of reduced costs $R_y^{s'',0}$, the same approach, as in Strategy 1, is applied to obtain the restricted model.

The potential to exclude (or include) a specific arc from the desired network can also be assessed through the reduced cost associated with the flow variables that report the amount of each commodity transported through the arc. By evaluating the opportunity cost of having excluded (or included) an arc using the specific reduced costs associated with all flow variables associated to that arc, one may hopefully obtain a good measure to determine the variables to fix. We thus propose two more strategies, as follows.

Strategy 4. In this strategy, as in Strategy 3, the considered solution is $\mathbf{s}'' = (\mathbf{y}'', \mathbf{x}''^{s_1}, \ldots, \mathbf{x}''^{s_{|S|}})$. However, we investigate the benefit of using reduced costs associated with the flow variables to identify the set of fixing variables. Let $J_0^{s''}$ represents the index set of zero design variables in the solution $\mathbf{s}''$ and r_a^{ks} be the reduced costs with respect to the flow variables of commodity k in scenario s on arc index a (i.e., $\mathbf{x}''^{ks}_a$). We define the value $\bar{r}_a = \sum_{s \in S} p^s \sum_{k \in \mathcal{K}} (1/|\mathcal{K}|) r_a^{ks}$ to aggregate all reduced costs associated with the flow variables assigned to arc index a. Let $R_x^{s'',0}$ represents the set of aggregated reduced costs corresponding to index set $J_0^{\bar{s}''}$ using the flow variables. The same approach, as in Strategy 1, is then applied here to obtain the restricted model.

Strategy 5. In this strategy, the considered solution is again $\mathbf{s}'' = (\mathbf{y}'', \mathbf{x}''^{s_1}, \ldots, \bar{\mathbf{x}}''^{s_{|S|}})$. However, we consider the reduced costs corresponding to both the design and the flow variables used in the previous strategies. We thus define the *composite reduced cost*; $r_a^+ = r_a + \bar{r}_a$. Let $J_0^{s''}$ represents the index set of zero design variables, i.e., $y_a'' = 0, a \in J_0^{s''}$, in the solution $\mathbf{s}''$, and $R_{x,y}^{s'',0}$ represents the set of composite reduced costs corresponding to index set $J_0^{s''}$ using both the design and flow variables. The same approach, as in Strategy 1, is then applied here to obtain the restricted model.

To determine the desirable variables to be fixed to 1 (i.e., open arcs), one may use the same strategies described above; however, we need to consider the reduced cost associated with the variables at their upper bound (i.e., the design variables that are 1 in the solutions considered in Strategies 1-5). Thus, instead of fixing the last classes ranging from p_0 to N_0 (with the largest values of reduced cost), we fix the variables belonging to the classes ranging from 1 to p_1, where $1 \leq p_1 \leq N_1$, which have the smallest values of reduced costs.

4.2 Description of the algorithm

As described previously, the proposed matheuristic solves a sequence of restricted problems. That is, at each iteration, two distinct subsets of design variables defined and guided by reduced cost information are used to construct the restricted problem. The constructed restricted problem, defined by fixing the identified design variables to 0 or 1, is then solved by an MIP solver, at each iteration. Algorithm 1 sums up the entire procedure. We refer to the P problem as the MCFND(S) problem (1)-(5) including all binary design variables. While, the restricted problem $\tilde{P}$ represents the MCFND(S) problem restricted to the subsets of the design variables that are fixed to 0 or 1. In the following subsections, each component of the Algorithm 1 is described in details.

Algorithm 1 Reduced cost-based restriction and refinement matheuristic

```
1: *Initialization:*                                                    ▷ Section 4.2.1
2: $k := 0$; construct initial solution $y^{Ini}$; set $y^{best} := y^{Ini}$; let $J_{\mathbf{1}}^{best}$ be the index set of design variables which are 1 in $y^{best}$;
3: $k := 1$;
4: **repeat**
5:     **Constructing the restricted problem:**                          ▷ Section 4.2.2
6:     *phase 1:*
7:     Construct $AP^k$ by fixing $J_{\mathbf{1}}^{best}$ in the problem $P$ as $AP^k := P|_{J_{\mathbf{1}}^{best}}$;
8:     Generate *solution pool* $\mathcal{P}^k = \{\mathbf{s}^1, \mathbf{s}^2, \ldots, \mathbf{s}^N\}$ for $AP^k$ considering parameter $\alpha$;
9:     *phase 2:*
10:    Perform Algorithm 2 to establish two subsets $J_{\mathbf{0}}^{\mathcal{P}^k}$ and $J_{\mathbf{1}}^{\mathcal{P}^k}$ using *reduced cost information* associated with $\mathcal{P}^k$;
11:    **Solving the restricted problem:**                                ▷ Section 4.2.2
12:    Solve $\tilde{P}^k$ which is constructed by fixing $J_{\mathbf{0}}^{\mathcal{P}^k} \cup J_{\mathbf{1}}^{\mathcal{P}^k}$ in the problem $P$ as $\tilde{P}^k := P|_{J_{\mathbf{0}}^{\mathcal{P}^k} \cup J_{\mathbf{1}}^{\mathcal{P}^k}}$ yielding to the solution $\mathbf{y}^*_{\tilde{P}^k}$ with objective value $z^*_{\tilde{P}^k}$;
13:    **Improvement check and Diversification:**                         ▷ Section 4.2.3
14:    **if** $z^*_{\tilde{P}^k} < z_{best}$ **then**
15:        $\mathbf{y}^{best} := \mathbf{y}^*_{\tilde{P}^k}$ and update $J_{\mathbf{1}}^{best}$;
16:        $z_{best} := z^*_{\tilde{P}^k}$;
17:        Go to line 26;
18:    **end if**
19:    **if** time limit is not exceeded **then**
20:        **if** $\mathbf{y}^{best}$ has not been improved in the last successive $q$ attempts **then**
21:            Let $\alpha \leftarrow \alpha + \Delta(\alpha)$ and go to line 8;
22:        **else**
23:            Enlarge the search space of $\tilde{P}^k$ by reducing the size of $J_{\mathbf{0}}^{\mathcal{P}^k}$ and $J_{\mathbf{1}}^{\mathcal{P}^k}$ and go to line 12;
24:        **end if**
25:    **end if**
26:    $k := k + 1$;
27: **until** stopping criteria
28: **return** $\mathbf{y}^{best}$.
```

4.2.1 Initialization

At the beginning of the Algorithm 1, we construct an initial solution y^{Ini} using the procedure described in Strategy 3. To do so, we use the expected value solution as a input to the model (1)-(5) and then solve the problem to obtain the solution y^{Ini}. Let y^{Ini} be the current best solution (i.e., $y^{best} := y^{Ini}$), and $J_{\mathbf{1}}^{best}$ be the index set of design variables which are 1 in the solution y^{best}.

4.2.2 Constructing and solving the restricted problem - based on primal-dual information

At each iteration of Algorithm 1, a restricted problem is constructed by determining two distinct sets of fixed variables. The restricted problems are defined by exploring attributes originating from multiple solutions. The exploration is performed by examining the information obtained through a two-phase procedure. In the first phase, the primal information (i.e., solutions) are generated and, in the second phase, a learning procedure is applied on their dual information. In the following, we describe the proposed two phases leading to the restricted problem $\tilde{P}^k$ at each iteration k.

Phase 1: Generating the pool of solutions - primal information The first step to construct the restricted problem involves creating multiple solutions. We believe the solutions obtained by MCFND(S) model would provide better information as compared to solution obtained by DSSP (6)-(10). Therefore, to generate multiple good quality solutions, we aim to create solutions obtained by MCFND(S) model and store them as a pool of solutions at each iteration of Algorithm 1. To generate these solutions, we first construct a reduced size auxiliary problem at each iteration k, denoted by AP^k. To construct AP^k, we use the current best solution (i.e., $\mathbf{y}^{best}$) and fix design variables associated with indexes $j \in J_{\mathbf{1}}^{best}$ in the problem P (i.e., $AP^k := \mathrm{P}|_{J_{\mathbf{1}}^{best}}$) in order to reduce the size of problem. We note that feasible solutions for AP^k are feasible for MCFND(S) problem as well.

As stated in Algorithm 1, line 8, we generate multiple solutions for AP^k and store them in the solution pool $\mathcal{P}^k$. The solution pool $\mathcal{P}^k$, for AP^k, contains N different solutions $\mathbf{s}^1, \mathbf{s}^2, \ldots, \mathbf{s}^i, \ldots, \mathbf{s}^N$ whose objective functions $z(\mathbf{s}^i)$ are within $\alpha\%$ of the optimum, i.e., such that $z(\mathbf{s}^i) \leq z(\mathbf{s}^{k,best}) + \alpha z(\mathbf{s}^{k,best})/100$, $\forall i = 1, \ldots, N$ where $\mathbf{s}^{k,best}$ and $z(\mathbf{s}^{k,best})$ are the optimal solution to the AP^k and its objective function value, respectively .

We note that the approach we use to generate $\mathcal{P}^k$ is to use the solution pool functionality of the CPLEX solver. These solutions are generated during the global MIP tree exploration performed by CPLEX, where the generated solutions in pool $\mathcal{P}^k$ are distinguishable by the values of their (binary) design variables only.

Phase 2: Reduce cost based learning - dual information The main purpose of this phase is to identify two index sets of desirable arcs to be fixed to closed $J_{\mathbf{0}}^{\mathcal{P}^k}$ or opened $J_{\mathbf{1}}^{\mathcal{P}^k}$ in $\tilde{P}^k$ according to information learned from the reduced costs associated with the solutions in pool $\mathcal{P}^k = \{\mathbf{s}^1, \mathbf{s}^2, \ldots, \mathbf{s}^i, \ldots, \mathbf{s}^N\}$.

The steps of this phase are stated in Algorithm 2. For each generated solution $\mathbf{s}^i \in \mathcal{P}^k$, we represent the index set of design variables, which are 0 by $J_{\mathbf{0}}^{\mathbf{s}^i}$; the index set of design variables, which are 1 by $J_{\mathbf{1}}^{\mathbf{s}^i}$; the value of objective function by $z(\mathbf{s}^i)$; and the weight by $w(\mathbf{s}^i) = \frac{1}{z(\mathbf{s}^i) - min_{\mathbf{s}^i \in \mathcal{P}} z(\mathbf{s}^i)}$, indicating a relative quality of $\mathbf{s}^i$. The index sets of desirable variables $J_{\mathbf{0}}^{\mathcal{P}^k}$ and $J_{\mathbf{1}}^{\mathcal{P}^k}$ are created according to the desirability factor l_a associated to each arc a measured using all of the solutions in $\mathcal{P}^k$. We first define the desirability factor l_a^i associate with each arc j obtained by the solution i. To do so, we consider three alternative variants (in lines 3-5) to extract the reduced cost information according to different choices of variables: 1) if the choice of variable is y, it means we consider the reduced cost values associate with design variables, i.e., r_a, as the desirability factor, $l_a^i := r_a$. 2) if the choice of variable is x, it means we consider $\bar{r}_a$ as the desirability factor, $l_a^i := \bar{r}_a$. Recall that $\bar{r}_a = \sum_{s\in S} p^s \sum_{k\in K}(1/|\mathcal{K}|)r_a^{ks}$, where r_a^{ks} is the reduced costs with respect to the flow variables of commodity k in scenario s on arc index a (i.e., $x_a^{\prime\prime ks}$). 3) if the choice of variable is both of x and y, it means we consider the composite reduced cost, $r^+ = r_a + \bar{r}_a$, as the desirability factor, i.e., $l_a^i := r_a^+$. Once, the values of l_a^i associated with each solution i are computed, we aggregate the desirability factors over all solutions (in line 7) as follows: $l_{a,0} = \sum\limits_{\mathbf{s}^i\in\mathcal{P}} w(\mathbf{s}^i) * l_a^i$ for $a \in \bigcap_{\mathbf{s}^i\in\mathcal{P}^k} J_{\mathbf{0}}^{\mathbf{s}^i}$ and $l_{a,1} = \sum\limits_{\mathbf{s}^i\in\mathcal{P}} w(\mathbf{s}^i) * l_a^i$ for $a \in \bigcap_{\mathbf{s}^i\in\mathcal{P}^k} J_{\mathbf{1}}^{\mathbf{s}^i}$. Let $L_0^k = \{(a, l_{a,0}) | a \in \bigcap_{\mathbf{s}^i\in\mathcal{P}^k} J_{\mathbf{0}}^{\mathbf{s}^i}\}$ and $L_1^k = \{(a, l_{a,1}) | a \in \bigcap_{\mathbf{s}^i\in\mathcal{P}^k} J_{\mathbf{1}}^{\mathbf{s}^i}\}$. We then sort L_0^k according to $l_{a,0}$ in non-decreasing order. Let l_0^{max} and l_0^{max} the maximum and minimum values in L_0^k. To determine the cluster of desirable variable to be fixed to zero, we divide the difference $l_0^{max} - l_0^{min}$ in N_0 classes of constant size of $\frac{l_0^{max}-l_0^{min}}{N_0}$ and store the index of variables belonging to the classes p_0 to N_0 $(1 \le p_0 \le N_0)$ in $J_{\mathbf{0}}^{\mathcal{P}^k}$. We perform the same sorting procedure for L_1^k according to $l_{a,1}$. Let l_1^{max} and l_1^{min} be the maximum and minimum values in L_1^k, respectively. We then divide the difference $l_1^{max} - l_1^{min}$ in N_1 classes of constant size of $\frac{l_1^{max}-l_1^{min}}{N_1}$ and store the index of variables belonging to the classes 1 to p_1 ($1 \le p_1 \le N_1$) in $J_{\mathbf{1}}^{\mathcal{P}^k}$. The two sets $J_{\mathbf{0}}^{\mathcal{P}^k}$ and $J_{\mathbf{1}}^{\mathcal{P}^k}$ are returned as the index sets of the most desirable arcs, at iteration k, to be fixed to be closed and opened, respectively.

Solving the restricted problem Once the index sets of desirable closed and open arcs (i.e., $J_{\mathbf{0}}^{\mathcal{P}^k}$ and $J_{\mathbf{1}}^{\mathcal{P}^k}$) are established, we then construct the restricted problem $\tilde{P}^k$ by fixing the design variables belonging to the two sets $J_{\mathbf{0}}^{\mathcal{P}^k}$ and $J_{\mathbf{1}}^{\mathcal{P}^k}$ to 0 and 1, in the problem P, respectively (i.e., $\tilde{P}^k :=$ P$|_{J_{\mathbf{0}}^{\mathcal{P}^k}\cup J_{\mathbf{1}}^{\mathcal{P}^k}}$). We then solve $\tilde{P}^k$ to obtain solution $\mathbf{y}^*_{\tilde{P}^k}$ with objective value $z^*_{\tilde{P}^k}$ (line 12).

4.2.3 Improvement check and Diversification

In this part of algorithm, we check the improvement and, if needed, perform the diversification step (line 14 to 25). Once the restricted problem is solved (line 12), the following steps depend on the solution found by the solver. If a better solution is found, it becomes a new incumbent ($y^{best} := y^*_{\tilde{P}^k}$), and the search continues from its solution in the next iteration (lines 14 to 18). However, if the new found solution is not better than the current best solution and the time limit is not exceeded, we attempt to improve the solution by

Algorithm 2 Reduced cost-based learning procedure

```
1: Initialization State the sets $J_{\mathbf{0}}^{\mathbf{s}^i}$ and $J_{\mathbf{1}}^{\mathbf{s}^i}$ for $\mathbf{s}^i \in \mathcal{P}^k$, let $w(\mathbf{s}^i)$ be the weight of solution $\mathbf{s}^i$;
2: for all $\mathbf{s}^i \in \mathcal{P}^k$ do
3:     if the choice of variable is $y$, then let $l_a^i := r_a$ where $a \in J_{\mathbf{0}}^{\mathbf{s}^i}$ or $a \in J_{\mathbf{1}}^{\mathbf{s}^i}$;
4:     if the choice of variable is $x$, then let $l_a^i := \bar{r}_a$ where $a \in J_{\mathbf{0}}^{\mathbf{s}^i}$ or $a \in J_{\mathbf{1}}^{\mathbf{s}^i}$ ;
5:     if the choice of variable is both of $x$ and $y$, then let $l_a^i := r_a^+$ where $a \in J_{\mathbf{0}}^{\mathbf{s}^i}$ or $a \in J_{\mathbf{1}}^{\mathbf{s}^i}$;
6: end for
7: Aggregate the desirability factor $l_a^i$ over all solutions as follows:
       $l_{a,0} = \sum_{\mathbf{s}^i \in \mathcal{P}} w(\mathbf{s}^i) * l_a^i$ where $a \in \bigcap_{\mathbf{s}^i \in \mathcal{P}^k} J_{\mathbf{0}}^{\mathbf{s}^i}$;
       $l_{a,1} = \sum_{\mathbf{s}^i \in \mathcal{P}} w(\mathbf{s}^i) * l_a^i$ where $a \in \bigcap_{\mathbf{s}^i \in \mathcal{P}^k} J_{\mathbf{1}}^{\mathbf{s}^i}$;
8: Let $L_0^k = \{(a, l_{a,0}) | a \in \bigcap_{\mathbf{s}^i \in \mathcal{P}^k} J_{\mathbf{0}}^{\mathbf{s}^i}\}$. Sort $L_0^k$ in non-decreasing order according to $l_{a,0}$ and then create the set $J_{\mathbf{0}}^{\mathcal{P}^k}$ according to Section 4.2.2;
9: Let $L_1^k = \{(a, l_{a,1}) | a \in \bigcap_{\mathbf{s}^i \in \mathcal{P}^k} J_{\mathbf{1}}^{\mathbf{s}^i}\}$. Sort $L_1^k$ in non-decreasing order according to $l_{a,0}$ and then create the set $J_{\mathbf{1}}^{\mathcal{P}^k}$ according to Section 4.2.2;
10: return $J_{\mathbf{0}}^{\mathcal{P}^k}$ and $J_{\mathbf{1}}^{\mathcal{P}^k}$.
```

performing the diversification step (line 19-25) as follows. If y^{best} has not been improved in the last q attempts, we go to line 8 and generate a different solution pool by increasing parameter α (line 21). Otherwise, we attempt to improve the solution by enlarging the search space with freeing more variables in the current restricted problem $\tilde{P}^k$. To do so, we remove ν_0 (ν_1) percent of variables with the largest (smallest) values of $l_{a,0}$ ($l_{a,1}$) from $J_{\mathbf{0}}^{\mathcal{P}^k}$ ($J_{\mathbf{1}}^{\mathcal{P}^k}$) to reduce the number of variables that are fixed in $\tilde{P}^k$ and then go to line 12 to find a better solution. The stopping criteria is the maximum computational time denoted as t_{max}.

5 Experimental results

This section presents the results of extensive computational experiments performed to assess the performance of the proposed matheuristic. We first describe the test instances and experimental settings in Section 5.1 and then provide a comparative analysis of the different proposed strategies in Section 5.2. We then detail the numerical results of the proposed matheuristic (denoted by RCHeur) by analyzing 1) in Section 5.3.1, the impact of the various internal features of the proposed RCHeur, and 2) in Section 5.3.2, the power of the proposed RCHeur in dealing with difficult instances through a comparative analysis of its performance versus the following alternative algorithms: IBM-ILOG

CPLEX 12.7 set to its default settings (CPLEX in the following), the Benders algorithm implemented in CPLEX 12.7 (Benders in the following) and the Learn&Optimize (denoted by L&Opt) procedure proposed in Sarayloo et al. (2018). By proceeding in this way, RCHeur is compared to both 1) exact solution methods: one that solves the problem directly (i.e., CPLEX) and one that solves the problem via the application of a specialized decomposition strategy for stochastic models (Benders); and 2) a matheuristic method that currently defines the state-of-the-art heuristic algorithm for the considered problem (i.e., L&Opt).

5.1 Data and experimental settings

We used 11 problem classes (R5-R15) from the set of R instances of the stochastic FCMND problem introduced in Crainic et al. (2011). Each class is characterized by a number of nodes $|\mathcal{N}|$, number of arcs $|\mathcal{A}|$, and number of commodities $|\mathcal{K}|$, specified in Table 1. Each of these classes contains five networks with different "ratio" index valued 1, 3, 5, 7, and 9, which indicate continuously increasing ratios of fixed to variable costs and total demand to total network capacity Crainic et al. (2011). For each of these networks, there are instances with 16, 32, and 64 scenarios. Demands were assumed to be linearly correlated, and three different levels of correlations (0, 0.2, and 0.8) were considered to create different instances.

Table 1: Characteristics of instances

Problem	$\|\mathcal{N}\|$	$\|\mathcal{A}\|$	$\|\mathcal{K}\|$	Problem	$\|\mathcal{N}\|$	$\|\mathcal{A}\|$	$\|\mathcal{K}\|$
R04	10	60	10	R10	20	120	40
R05	10	60	25	R11	20	120	100
R06	10	60	50	R12	20	120	200
R07	10	82	10	R13	20	220	40
R08	10	83	25	R14	20	220	100
R09	10	83	50	R15	20	220	200

Algorithms were implemented in C++. The numerical experiments were performed on a Sun Fire X4100 cluster of 16 computers, each has two 2.6 GHz Dual-Core AMD Opteron processors and 8192 Megabytes of RAM, operating under Solaris 2.10. We used CPLEX 12.7 to solve the MIP problems. For the *Benders* method, we used automated Benders' feature implemented in CPLEX 12.7, where CPLEX includes all integer variables in the master and the continuous variables into subproblems to solve the overall problem using the decomposition strategy. The time limit is set to 500 minutes, when calling CPLEX in the following experiments.

5.2 Analyzing different strategies in using reduced cost information

In this section, we analyze and compare different strategies proposed in Section 4.1. In this part of experiment, we focus on relatively easy instances (R5-R10 with ratios 1, 3, 5, 7, and 9 and correlations 0 and 0.8). By doing so, we aim to be able to qualify the quality of solutions obtained by different strategies, as the optimal solution of the majority of these instances can be obtained by CPLEX.

5.2.1 Comparing the strategies for fixing variables to 0

In this section, we focus on investigating the reduced cost of the non-basic variables which are at their lower bound (i.e., 0). We first present the results obtained by applying Strategy 1 where the optimal solution of the LP relaxation of the EV problem is used (i.e., $\bar{\mathbf{s}}$). Strategy 1 is denoted by $Str1(p_0, N_0)$ where the set of reduced cost values $R_y^{\bar{s},0}$ is divided into N_0 equivalent sized classes, and then the variables belonging to the classes p_0 to N_0 are fixed to 0.

We perform the experimental analysis exploring the behaviour of Strategy 1 while varying the values p_0 and N_0 to determine the compromised values of ${p_0}^\star$ and ${N_0}^\star$ for different instances. We present the comparative results according to the following measures: feasibility, solution quality, and computational efforts. We note that the compromised values $({p_0}^\star, {N_0}^\star)$ are then used for the rest of strategies to compare their performance. Given the fact that fixing design variables to 0 may result in infeasibility issues, we report in Table 2 the number of instances which are infeasible by performing Strategy 1 with the following (p_0, N_0) values: $Str1(p_0, 3), p_0 = 2, 3$ and $Str1(p_0, 9), p_0 = 3, 4, 5, 6, 7, 8, 9$. Moreover, to qualify the results obtained by performing Strategy 1 in terms of solution quality and computational time, we provide the comparative analysis versus CPLEX in Table 3. The Gap and Time values reported for CPLEX refer, respectively, to the optimal gap and the computational time in seconds. As for the "$Str1$", Gap and Time represent the corresponding optimality gap relative to the lower bound of CPLEX and the total computational time, respectively.

Table 2 shows that the total number of infeasible instances is increased from 3 instances (in the case of $Str(9, 9)$) to 19 instances (in the case of $Str(4, 9)$). However, the sharp increase in the the number of infeasible instances happens in the case of $Str(3, 9)$). It means that fixing the variables belonging to the first three classes of variables (i.e., $Str(p_0, N_0)$ p_0 =1,2, and 3) results in many infeasibility issues.

As shown in Table 3, the results in the case of $Str1(3, 3)$, i.e., fixing one out of 3 classes of variables ($(p_0, N_0) = (3, 3)$), are as follows. The number of infeasible instances is 9,

Table 2: The number of infeasible instances (INF)

		$Str1(p_0, 3)$		$Str1(p_0, 9)$						
`Ratio`	`Ins`	2	3	3	4	5	6	7	8	9
1	36	0	0	4	0	0	0	0	0	0
3	36	0	0	4	0	0	0	0	0	0
5	36	7	0	15	7	7	7	0	0	0
7	36	12	9	21	12	12	9	9	9	3
9	36	0	0	10	0	0	0	0	0	0
Total	180	19	9	54	19	19	16	9	9	3

the average optimality gap is 2.12% which is better than CPLEX with the average gap of 2.57% and the average computational time is reduced almost 10% compared to CPLEX. Considering that $Str(3, 3)$ provides a little reduction in time (almost 10%), fixing fewer number of variables occured in $Str(p_0, 9), p_0 = 8, 9$ don't seem reasonable since they can't provide much fixed variables. However, in the case of $Str1(2, 3)$, i.e., fixing two out of 3 classes of variables ($(p_0, N_0) = (2, 3)$), the number of infeasible instances is 19, the average optimality gap is 1.59% which is better than CPLEX with the average of 2.49%, and the computational time is reduced almost 50% relative to CPLEX. We note that fixing higher numbers of variables occurred in $Str(p_0, 9), p_0 = 1, 2, 3$ results in a significant increase in the number of infeasible instances (more than 54 out of 180 instances) as shown in Table 2. Therefore, it seems that $Str1(2, 3)$ is able to provide a good performance in terms of improvement in solution quality and reduction in computational time, both compared to CPLEX, and is a good compromise between the considered instances. In the following, our goal is to examine if it is possible to improve the obtained results of Strategy 1, i.e., $Str1(2, 3)$, by upgrading the expected value solution and using a different choice of variables, as explained earlier in Strategies 2 to 5. To do so, we present the results of other strategies proposed in Section 4.1 and compare them with the values obtained by $Str1(2, 3)$.

Table 4 displays the comparative results of performing Strategies 1 to 5 considering $(p_0^\star, N_0^\star) = (2, 3)$, i.e, fixing to 0 about 66% of non-basic variables with the highest reduced costs relative to $R_y^{\bar{s},0}$,$R_y^{s',0}$, $R_y^{s'',0}$ $R_x^{s'',0}$, and $R_{x,y}^{s'',0}$ in Strategies 1 to 5, respectively. As previously described, the Gap and Time values reported for CPLEX refer, respectively, to the optimal gap and the total computational time in seconds represented in parenthesis. As for the different strategies "$Str1$" to "$Str5$", Gap and Time represent the corresponding optimality gaps relative to the lower bound of CPLEX and the total computational time expressed in seconds, respectively. Column "INF" indicates the number of infeasible instances. It should be noted that we consider a gap of 100% for infeasible instances to make the results comparable over all strategies. The results clearly show that there are no more infeasibility issues in Strategies 2 to 5, indicating the noticeable effect of upgrading the EV solution. In terms of solution quality, the performance of using reduced cost is enhanced by providing an improvement of at least 10.35% in optimality gap, when we

Table 3: The performance comparisons of $Str1(p_0, N_0)$ vs. CPLEX for fixing to 0

Ratio	Ins	CPLEX	$Str1(2,3)$		CPLEX	$Str1(3,3)$	
		Gap(%) (Time)	Gap(%) (Time)	INF	Gap(%) (Time)	Gap(%) (Time)	INF
1	36	0.00 (154)	0.00 (52)	0	0.00 (154)	0.00 (204)	0
3	36	6.7 (14081)	2.75 (8604)	0	6.7 (14081)	5.09 (12241)	0
5	36	2.46 (14081)	2.10 (10203)	7	2.82 (15453)	2.36 (12445)	0
7	36	0.24 (7010)	0.4 (1670)	12	0.32 (7390)	0.45 (4182)	9
9	36	3.05 (11622)	2.70 (6229)	0	3.05 (14461)	2.74 (11880)	0
Avg	180	2.49 (10388)	1.59 (5351)		2.57 (9017)	2.12 (8192)	

upgrade the solutions in Strategies 2 to 5 (with the average optimality gap of at most 1.7%), compared to Strategy 1 (with the average optimality gap of 12.05%) which uses the solution of the LP relaxation of the EV problem. Furthermore, using the reduced costs associated with flow variables (i.e., $R_x^{s'',0}$), as defined in Strategy 4, provides the least computational time compared to the other strategies.

Table 4: Performance comparisons Strategies 1 to 5 for fixing to 0

Pro	Ins	CPLEX	Str1		Str2		Str3		Str4		Str5	
		Gap(%) (Time)	Gap(%) (Time)	INF	Gap(%) (Time)	INF	Gap(%) (Time)	INF	Gap(%) (Time)	INF	Gap(%) (Time)	INF
R05	30	0.00 (1437)	26.66 (1408)	8	0.33 (135)	0	0.13 (92.9)	0	0.05 (91.3)	0	0.13 (83.4)	0
R06	30	1.61 (11401)	1.05 (4670)	0	1.00 (4969)	0	1.30 (3630)	0	1.26 (2274)	0	1.25 (2319)	0
R07	30	0.10 (1745)	6.68 (2037)	2	0.38 (179)	0	0.31 (219)	0	0.47 (232)	0	0.31 (192)	0
R08	30	0.98 (7217)	21.33 (6334)	6	1.99 (663)	0	1.25 (2724)	0	1.7 (1037)	0	1.25 (1402)	0
R09	30	4.51 (16353)	2.32 (11036)	0	2.03 (8243)	0	1.48 (7173)	0	1.98 (3087)	0	1.48 (4636)	0
R10	30	8.17 (23243)	14.28 (15953)	3	4.47 (13959)	0	4.43 (17994)	0	4.61 (9300)	0	4.34 (12117)	0
Avg	180	2.56 (10229)	12.05 (6906)		1.7 (4601)		1.48 (5305)		1.67 (2665)		1.46 (3458)	

5.2.2 Comparing the strategies for fixing variables to 1

To study the possibility of using reduced cost information for fixing variables to 1, we present the results of applying the same strategies presented in Section 4.1 on the non-basic variables at their upper bound (i.e., 1). We first examine the performance of Strategy 1. Following this strategy, denoted by $Str(p_1, N_1)$, we use the optimal solution of LP relaxation of the EV problem, i.e., $\bar{\mathbf{s}}$. The set of reduced cost values $R_y^{\bar{s},1}$ is then divided into N_1 classes, and the variables belonging to the classes 1 to p_1 are fixed to 1. Given the fact that there are no feasibility issues in these strategies by fixing to 1, we only present the comparison results on optimality gaps and computational times versus CPLEX in Table 5 to qualify the results obtained by Strategy 1. As shown in Table 5, in the case of fixing only one out of 3 classes (i.e., $(p_1, N_1) = (1, 3)$), Startegy 1 perform as well as CPLEX in terms of both optimality gaps and computational time. Moreover, in the case of fixing two out of 3 classes (i.e., $(p_1, N_1) = (2, 3)$), Strategy 1 performs slightly better than CPLEX by providing optimality gaps of 2.53% (within 9538 seconds) versus 2.56% (within 10229 seconds). The results show that Strategy 1 is not as effective in identifying fixed variables at their upper bound as it is at their lower bound. This means that the LP relaxation of the EV problem (i.e., $\bar{\mathbf{s}}$) does not provide many variable fixing choices, since there are too few design variables that are 1 in the solution $\bar{y}$ (there are a maximum of 3 design variables that are 1 in the $\bar{y}$ for the instances with ratios 1,3, and 5). This indicates the need to upgrade the EV solution, as explained in the proposed Strategies 2 to 5, in order to provide a good set of fixed variables. Nevertheless, we observed that fixing variables to 1 in Strategy 1, with the values $(p_1, N_1) = (2, 3)$ (i.e., 2 out of 3 classes), is again an acceptable compromise to produce relatively good performance over all instances. We then examined whether we could improve the performance of Strategy 1 by upgrading the solution and using different choices of variables in Strategies 2 to 5.

Table 5: The performance comparisons of $Str1(p_1, N_1)$ vs. CPLEX for fixing to 1

Ratio	Ins	CPLEX	$Str1(1,3)$	$Str1(2,3)$
		Gap(%) (Time)	Gap(%) (Time)	Gap(%) (Time)
1	36	0.00 (353)	0.00 (315)	0.00 (292)
3	36	6.70 (14081)	6.75 (13790)	6.70 (13270)
5	36	2.82 (15453)	2.83 (14972)	2.80 (14372)
7	36	0.24 (6823)	0.26 (6465)	0.27 (6185)
9	36	3.05 (14461)	2.96 (14215)	2.99 (13572)
Avg	180	2.56 (10229)	2.56 (9951)	2.53 (9538)

Table 6 shows the comparative results of performing Strategies 1 to 5 considering $(p_1, N_1) = (2, 3)$, i.e, fixing to 1 about 66% of non-basic variables with the smallest reduced costs relative to $R_y^{\bar{s},1}$,$R_y^{s',1}$, $R_y^{s'',1}$ $R_x^{s'',1}$, and $R_{x,y}^{s'',1}$ in Strategies 1 to 5, respectively. The table reports the same information as Table 4. The results show that, in terms of solution quality and computational time, the performance of using reduced cost is enhanced when we upgrade the solution in Strategies 2 to 5 comparing to strategy 1 which uses the solution of LP relaxation of the EV problem. Furthermore, using the reduced cost associated with flow variables (i.e., $R_x^{s'',1}$) in strategy 4 provides the least computational time compared to the other strategies. However, when assessing the optimality gaps obtained, we observed that all strategies 2 to 5 seem to be equivalent (i.e., the difference is at most 0.07%).

Table 6: Performance comparisons of Strategies 1 to 5 for fixing to 1

Pro	Ins	CPLEX	Str1		Str2		Str3		Str4		Str5	
		Gap (Time)	Gap(%) (Time)	INF	Gap(%) (Time)	INF	Gap(%) (Time)	INF	Gap(%) (Time)	INF	Gap(%) (Time)	INF
R05	30	0.00 (1437)	0.00 (1468)	0	0.09 (328)	0	0.23 (490)	0	0.28 (81.11)	0	0.23 (313)	0
R06	30	1.61 (11401)	1.55 (11248)	0	1.09 (7688)	0	1.09 (7815)	0	1.44 (2499)	0	1.08 (6668)	0
R07	30	0.10 (1745)	0.08 (1603)	0	0.47 (749)	0	0.46 (450)	0	0.60 (266)	0	0.46 (322)	0
R08	30	0.98 (7217)	1.01 (6354)	0	1.59 (5874)	0	1.55 (5970)	0	1.93 (1886)	0	1.55 (4324)	0
R09	30	4.51 (16353)	4.39 (15121)	0	1.96 (9442)	0	1.94 (10372)	0	1.98 (4401)	0	1.90 (8957)	0
R10	30	8.17 (23243)	8.14 (21430)	0	6.12 (26270)	0	5.65 (25396)	0	4.81 (10751)	0	5.65 (23444)	0
Avg	180	2.56 (10229)	2.53 (9538)	0	1.88 (8391)	0	1.82 (8333)	0	1.84 (3314)	0	1.81 (7338)	0

5.3 Numerical results of proposed matheuristic

In this section we present the results of the proposed matheuristic by evaluating 1) the effects of various internal components of the algorithm, in Section 5.3.1, and 2) its power to deal with very difficult instances cited in the literature, in Section 5.3.2. We note that, according to the analysis carried out in the previous section, the parameters (p_0, N_0) and (p_1, N_1) are set to (2,3), in both cases. Also, the choice of variables in Algorithm 2 is the flow variables. A preliminary analysis was conducted to fine tune the ν_0, ν_1, α, q and N parameters, which produced the following values .05, .05, .02, 3 and 3, respectively.

5.3.1 Internal analysis

The purpose of this section is to evaluate the effects of two features of the proposed matheuristic, which are using solutions obtained by MCFND(S) model and also multiple solutions. To do this, we designed two experiments to assess the effect of using stochastic versus deterministic solutions and multiple versus single solutions to generate the pool of solutions. The "Gap" and "Time" represent the optimality gap with respect to the lower bound of CPLEX and computational time in seconds, respectively.

Impact of using solutions obtained by MCFND(S) model. To evaluate the impact of using feasible solutions obtained by the MCFND(S) model rather than those obtained by DSSP on the performance of the proposed matheuristic, we show in Table 7 the performance comparison of both versions. In version "Deter-sols", to generate the solution pool $\mathcal{P}^k$ in Algorithm 1, we first choose randomly N scenarios and then solve their corresponding DSSP (6)-(10). However, in the version "Stoch-sol", we used the solutions obtained by MCFND(S) problem, as explained in Section 4.2.2. We observed that using solutions obtained by MCFND(S) in the proposed matheuristic produces better results, with an average gap of 1.29% (compared with using the solution obtained by DSSP with the average of 1.73%) in almost half the time, which highlights the importance of using good quality solutions to produce the set of fixed variables.

Table 7: Performance comparison on using the DSSP solution vs MCFND(S) solutions

Pro	Ins	CPLEX		Det-Sols		Stoch-Sol	
		Gap(%)	Time	Gap(%)	Time	Gap(%)	Time
R05	30	0.00	1437	0.1	722	0.06	399
R06	30	1.61	11401	1.11	8415	0.94	5607
R07	30	0.10	1745	0.55	673	0.11	558
R08	30	0.98	7217	1.68	4949	1.24	3391
R09	30	4.51	16353	2.44	11818	1.41	7137
R10	30	8.17	23243	4.82	19350	4.03	10131
Avg	180	2.56	10229	1.73	9125	1.29	4725

Impact of using multiple solutions. Is there any value in using multiple solutions versus single solutions ? To answer this question and evaluate the impact of using multiple solutions (here $[N = 3]$ in the Algorithm 1) versus a single solution on the performance of the proposed matheuristic, we show in Table 8 the performance comparison of both versions in columns "SingleSol" and "MultipleSol". The results show that using multiple solutions in the proposed matheuristic leads to better results, with an average gap of 1.29% (versus 1.55% in the case of using a single solution) in less computational time. That using multiple solutions rather than a single solution results in reduced computation time is a surprising observation. This may be explained by the fact that, while generating multiple solutions requires more computational effort at each iteration, the results show that the more refined information provided by multiple solutions leads to identifying better solutions faster. These results strengthen the idea of generating multiple solutions at each iteration of the algorithm.

Table 8: Performance comparison of using single solution vs. multiple solutions

Pro	Ins	CPLEX		SingleSol		MultipleSol	
		Gap(%)	Time	Gap(%)	Time	Gap(%)	Time
R05	30	0.00	1437	0.06	430	0.06	399
R06	30	1.61	11401	0.94	6201	0.94	5607
R07	30	0.10	1745	0.32	673	0.11	558
R08	30	0.98	7217	1.26	4706	1.24	3391
R09	30	4.51	16353	1.85	8218	1.41	7137
R10	30	8.17	23243	4.92	13850	4.03	10131
Avg	180	2.56	10229	1.55	5673	1.29	4725

5.3.2 Performance comparisons on difficult instances

To evaluate the quality and power of the proposed matheuristic, and to assess its performance when solving the challenging instances from the literature, we applied RCHeur to solve a set of large large R instances (i.e., R11-R15 instances described in Table 1). We focus on 180 instances that CPLEX was not able to solve to optimality after 500 min of computational time. We compared the best results (upper bound) of Benders, L&Opt and RCHeur versus that of CPLEX after 500 minutes CPU time. Table 9 displays these results. Each line reports, for each problem class, the number of instances (Column "Ins") in the class and the percentage of these instances where the associated solution method failed to find a feasible solution (Column "F(%)"). In addition, the average optimality gap obtained by CPLEX on those instances for which a feasible solution was found (Column "Gap(%)") is provided.

Columns "Diff" report the relative difference, in %, between the best solution (upper bound) obtained by each considered method and that of CPLEX computed as $100 * \frac{Z-CPLEX}{Z}$, where "Z" represent the best solution (upper bound) provided by the considered method (i.e., Benders, L&Opt and RCHeur) and that of CPLEX (for the instances where they are available). Overall, CPLEX failed to provide feasible solutions after 500 minutes for 31 instances, and so we report the average optimality gaps and improvements over the remaining 149 instances.

Overall, we note that, L&Opt, RCHeur provided the best solutions with the relative improvements being 13.67% and 19.95% , on average over the solution found by CPLEX (for which the average optimality gap is 29.15%), respectively. However, the best solution provided by Benders is 5.94% (on average) worse than that of CPLEX. The performance of the RCHeur is even more impressive when the most difficult instances are solved: i.e., an average relative difference of 27.97% is observed for the R15 instances. We also observed that RCHeur and L&Opt are able to provide a feasible solution for all considered instances, while CPLEX and Benders fail to provide any feasible solution on 15.14% and 15.60% of the instances, respectively. These analysis show the robustness of the proposed

RCHeur in providing good solutions in all considered instances. We observed that, on average, RCHeur outperforms L&Opt on more than 90% of the instances, with an average relative improvement of 6.07%, indicating the clear superiority of RCHeur over all other considered methods.

Table 9: Performance comparison- L&OPT, Benders and PHS versus CPLEX

Pro	Ins	CPLEX		*Benders*		*L&Opt*		*RCHeur*	
		Gap(%)	F(%)	Diff(%)	F(%)	Diff(%)	F(%)	Diff(%)	F(%)
R11	27	26.49	0	-9.22	0	-9.23	0	-18.42	0
R12	27	15.71	18	-1.91	18	-6.58	0	-9.79	0
R13	36	34.67	0	-13.71	0	-18.58	0	-22.23	0
R14	45	33.88	13.3	-8.23	0	-14.24	0	-21.38	0
R15	45	35.48	44.4	3.15	60	-19.79	0	-27.97	0
Avg		29.15	15.14	5.94	15.60	-13.67	0	-19.95	0

To analyze more thoroughly the behaviour of the proposed algorithm, we present the results aggregated according to the cost and capacity ratios used to generate the instances. Ghamlouche et al. (2003) defined two ratios where we briefly recall them here. The *fixed cost ratio* is computed as $|\mathcal{K}| \sum_{(i,j)\in\mathcal{A}} f_{ij} / \sum_{k\in\mathcal{K}} \omega^k \sum_{(i,j)\in\mathcal{A}} c^k_{ij}$, where ω^k is the demand volume of commodity k in the deterministic instances originally proposed in Ghamlouche et al. (2003). Instances were then randomly generated for three values for this ratio: F01= 0.01, F05 = 0.05, and F10 = 0.10, which correspond to increasing levels of the fixed costs compared to the routing costs. As for the *Capacity ratio*, it is computed as $|\mathcal{A}| \sum_{k\in\mathcal{K}} \omega^k / \sum_{(i,j)\in\mathcal{A}} u_{ij}$. Again, three values of this ratio were used to randomly generate instances: C1 = 1, C2 = 2, and C8 = 8, which indicate that the total capacity available becomes increasingly tighter relative to the total demand. The aggregated "ratio" index, 1,3, 5, 7, and 9, captures these two measures, indicating continuously increasing the ratios of fixed-to-variable-cost and total-demand-to-total-network-capacity.

Each line of Table 10 displays the aggregated results over 15 different instances, for a given combination of "ratio" and number of scenarios. The column "Gap" represents the average optimality gap obtained by CPLEX after 500 minutes of CPU time. It should be noted that, for those instances that CPLEX failed to find a feasible solution within the 500 minutes of allotted time (reported in the column "Failure"), a gap of 100the relative difference (in %) between the best solution (upper bound) found by RCHeur and Benders versus that of CPLEX, computed as $100{*}\frac{RCHeur-CPLEX}{RCHeur}$ and $100{*}\frac{Benders-CPLEX}{Benders}$, respectively. We consider that the relative difference is -100% for those cases where the considered method provides a solution but CPLEX fails to do so, and 0% for those cases where both the considered method and CPLEX fail to provide any solution. Table 11 presents the results aggregated according to different levels of fixed cost and capacity ratios.

A first conclusion that emerges from the analysis of Table 10 is that the instances that are the most difficult to address are those with intermediate ratios (e.g., 3 and 5). These results are consistent with the results obtained in previous studies (e.g., Crainic et al. (2011) and Sarayloo et al. (2018)), where similar observations were made. As originally explained in Crainic et al. (2011), this behaviour may be caused by the higher number of alternative optimal designs that may exist for the deterministic variants of these instances when compared to the instances with either higher or lower ratios. In the case where demand are stochastic, there may be broader differences among the scenarios, thus requiring more effort to identify an overall satisfactory, hopefully optimal, solution. Despite the difficulties of solving these instances, the proposed RCHeur is able to improve the solutions of CPLEX by 41.74% (on average). On these instances CPLEX obtains an average optimality gap of 52.84 %. Another conclusion from Table 10 is that the proposed RCHeur significantly outperforms CPLEX independently of ratios and instance characteristics.

We report in Table 11 the average relative difference of solutions (upper bounds) obtained by RCHeur and Benders compared to CPLEX by varying the fixed cost and the capacity ratios. It appears that, as expected, the instances that are the most difficult to address are those with medium levels of fixed cost and capacity ratios (i.e., F5 and C2). It is interesting to note that there is significant improvement achieved by RCHeur when the fixed costs and the capacity levels are set at their mid levels: the average optimality gap of CPLEX for these problems is around 40.66%, while both RCHeur and Benders improve the solutions of CPLEX by about 38.30% and 11.57%, respectively. The results also support the hypothesis that our algorithm provides a more consistent behaviour and appears significantly more robust with respect to these two characteristics overall instances (i.e., improvements are observed for all problem characteristics). In comparison, the performance of the Benders method is weaker when assessed against CPLEX on the C1 and F1 instances.

Finally, Table 12 reports the average results obtained on the instances grouped by correlation levels: Corr= 0.8, 0.2 and 0, and by the different sizes of the scenario set (each group totalling 25 instances). We observe that the demand correlation has little impact on how difficult the associated problems are to solve. RCHeur performs significantly better when compared to CPLEX, regardless of the correlation level considered (the average relative improvement in the quality of the solutions obtained being 24.36% overall instances). Lastly, as expected, increasing the number of scenarios makes the problem more difficult to solve for both RCHeur and CPLEX. Nonetheless, overall instances with 64 scenarios, RCHeur obtains an average improvement of 28.72% in terms of the best found solutions when compared with CPLEX, where the average optimality gap is 31.81%.

Table 10: Aggregated results according to ratios

Instance Ratios	|S|	# Inst.	CPLEX Failure	CPLEX Gap(%)	RCHeur/CPLEX Diff(%)	Benders/CPLEX Diff(%)
1	16	15	0	5.41	-5.92	9.55
1	32	15	0	10.49	-9.04	21.58
1	64	15	2	16.42	-12.42	38.80
3	16	15	0	44.40	-24.61	-21.39
3	32	15	3	57.67	-46.52	-29.66
3	64	15	9	66.32	-49.70	-13.26
5	16	15	0	23.61	-15.62	-2.02
5	32	15	3	43.10	-36.95	-10.13
5	64	15	7	56.70	-58.90	-22.56
7	16	15	0	2.31	-0.22	19.09
7	32	15	0	6.80	-2.08	15.14
7	64	15	3	21.61	-19.20	-0.83
9	16	15	0	7.34	-0.84	3.74
9	32	15	1	17.81	-10.63	-3.63
9	64	15	3	31.90	-25.47	-16.26

Table 11: Relative improvement according to fixed costs and capacity levels

	CPLEX Gap	RCHeur/CPLEX Diff(%)	Benders/CPLEX Diff(%)
F1	10.44	-11.65	7.22
F5	40.66	-38.30	-11.57
F10	37.16	-28.61	-13.63
C1	32.83	-25.27	0.93
C2	40.66	-38.30	-11.57
C8	14.45	-12.02	-2.65

6 Conclusions

In this paper, we investigated how to efficiently use reduced cost information extracted from the solution obtained by the LP relaxation of the EV problem to define good restriction in the context of stochastic network design. We specifically proposed different strategies to improve the EV solution and then extract associated reduced costs. The purpose of each strategy was to identify an appropriate subset of design variables (using reduced cost information) to be fixed in the stochastic problem and obtain a good quality solution. We subsequently proposed a matheuristic approach that iteratively defines restricted problems constructed by exploiting reduced cost information extracted from multiple solutions. The results of extensive computational experiment showed that the proposed algorithm is highly effective in finding good-quality solutions for very large

Table 12: Relative improvement according to scenario correlation

Scen	Corr=0.8		Corr=0.2		Corr=0	
	CPLEX Gap(%)	RCHeur/CPLEX Diff(%)	CPLEX Gap(%)	RCHeur/CPLEX Diff(%)	CPLEX Gap(%)	RCHeur/CPLEX Diff(%)
16	13.85	-9.53	13.73	-9.42	13.31	-9.28
32	22.64	-19.02	22.40	-18.51	22.15	-18.12
64	32.15	-29.25	31.28	-28.73	31.15	-28.29

instances in stochastic network design problems, while reducing the computational effort to obtain them.

We conclude this section with a few possible directions for future research. One possible direction is the adaption of the proposed approach to be applied on more practical variants of the classical network design model like service network design models. The other possible direction is based on the fact that most of solution methods for stochastic network design problems in the literature are based on exact methods. Thus, due to the NP-hardness nature of SND problems, this research area still needs more studies based on heuristic approaches. It would be worthwhile to develop various metaheuristic and matheuristic approaches which incorporate different learning and memorizing mechanisms to handle such large-scale problems.

Acknowledgments

While working on this project, T.G. Crainic was Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal. Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Discovery Grant program, and by the Fonds Québécois de la Recherche sur la nature et les technologies (FQRNT) through its strategic center infrastructure grant program.

References

Angelelli, Enrico, Renata Mansini, M Grazia Speranza. 2010. Kernel search: A general heuristic for the multi-dimensional knapsack problem. *Computers &amp; Operations Research* **37**(11) 2017–2026.

Archetti, Claudia, M Grazia Speranza, Martin WP Savelsbergh. 2008. An optimization-based heuristic for the split delivery vehicle routing problem. *Transportation Science* **42**(1) 22–31.

Balas, Egon, Eitan Zemel. 1980. An algorithm for large zero-one knapsack problems. *operations Research* **28**(5) 1130–1154.

Birge, John R, Francois Louveaux. 2011. *Introduction to stochastic programming*. Springer Science &amp; Business Media.

Crainic, Teodor G, Francesca Maggioni, Guido Perboli, Walter Rei. 2017. Reduced cost-based variable fixing in two-stage stochastic programming. *Annals of Operations Research* 1–37doi:10.1007/s10479-018-2942-8. URL `https://doi.org/10.1007/s10479-018-2942-8`.

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

Crainic, Teodor Gabriel, Xiaorui Fu, Michel Gendreau, Walter Rei, Stein W Wallace. 2011. Progressive hedging-based metaheuristics for stochastic network design. *Networks* **58**(2) 114–124.

Crainic, Teodor Gabriel, Mike Hewitt, Walter Rei. 2014. Scenario grouping in a progressive hedging-based meta-heuristic for stochastic network design. *Computers &amp; Operations Research* **43** 90–99.

Crainic, T.G, M. Hewitt, F. Maggioni, W. Rei. 2016. Partial 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.

De Franceschi, Roberto, Matteo Fischetti, Paolo Toth. 2006. A new ilp-based refinement heuristic for vehicle routing problems. *Mathematical Programming* **105**(2-3) 471–499.

Ghamlouche, Ilfat, Teodor Gabriel Crainic, Michel Gendreau. 2003. Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. *Operations research* **51**(4) 655–667.

Higle, Julia L, Stein W Wallace. 2003. Sensitivity analysis and uncertainty in linear programming. *Interfaces* **33**(4) 53–60.

Lium, Arnt-Gunnar, Teodor Gabriel Crainic, Stein W Wallace. 2009. A study of demand stochasticity in service network design. *Transportation Science* **43**(2) 144–157.

Maggioni, Francesca, Stein W Wallace. 2012. Analyzing the quality of the expected value solution in stochastic programming. *Annals of Operations Research* **200**(1) 37–54.

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

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

Puchinger, Jakob, Günther R Raidl. 2005. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. *International Work-Conference on the Interplay Between Natural and Artificial Computation*. Springer, 41–53.

Rahmaniani, Ragheb, Teodor Gabriel Crainic, Michel Gendreau, Walter Rei. 2017. The benders decomposition algorithm: A literature review. *European Journal of Operational Research* **259**(3) 801–817.

Raidl, Günther R. 2006. A unified view on hybrid metaheuristics. *International Workshop on Hybrid Metaheuristics*. Springer, 1–12.

Rockafellar, R Tyrrell, Roger J-B Wets. 1991. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research* **16**(1) 119–147.

Sarayloo, Fatemeh, Teodor Gabriel Crainic, Walter Rei. 2018. A learning-based matheuristic for stochastic multicommodity network design. Publication CIRRELT-2018-12, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada.

Thapalia, Biju K, Teodor Gabriel Crainic, Michal Kaut, Stein W Wallace. 2011. Single-commodity network design with stochastic demand and multiple sources and sinks. *INFOR: Information Systems and Operational Research* **49**(3) 193–211.

Thapalia, Biju K, Teodor Gabriel Crainic, Michal Kaut, Stein W Wallace. 2012a. Single-commodity network design with random edge capacities. *European Journal of Operational Research* **220**(2) 394–403.

Thapalia, Biju K, Stein W Wallace, Michal Kaut, Teodor Gabriel Crainic. 2012b. Single source single-commodity stochastic network design. *Computational Management Science* **9**(1) 139–160.

Van Slyke, Richard M, Roger Wets. 1969. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics* **17**(4) 638–663.

Wallace, Stein W. 2000. Decision making under uncertainty: Is sensitivity analysis of any use? *Operations Research* **48**(1) 20–25.

Wang, Xin, Teodor Gabriel Crainic, Stein W Wallace. 2018. Stochastic scheduled service network design: The value of deterministic solutions. *INFORMS Jounnal on Computing* .