Université de Montréal

Learning-Based Matheuristic Solution Methods for Stochastic Network Design

par
Fatemeh Sarayloo

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en Informatique option recherche opérationnelle

September 5, 2018

RÉSUMÉ

Cette dissertation consiste en trois études, chacune constituant un article de recherche. Dans tous les trois articles, nous considérons le problème de conception de réseaux multiproduits, avec coût fixe, capacité et des demandes stochastiques en tant que programmes stochastiques en deux étapes. Dans un tel contexte, les décisions de conception sont prises dans la première étape avant que la demande réelle ne soit réalisée, tandis que les décisions de flux de la deuxième étape ajustent la solution de la première étape à la réalisation de la demande observée. Nous considérons l'incertitude de la demande comme un nombre fini de scénarios discrets, ce qui est une approche courante dans la littérature. En utilisant l'ensemble de scénarios, le problème mixte en nombre entier (MIP) résultant, appelé formulation étendue (FE), est extrêmement difficile à résoudre, sauf dans des cas triviaux. Cette thèse vise à faire progresser le corpus de connaissances en développant des algorithmes efficaces intégrant des mécanismes d'apprentissage en matheuristique, capables de traiter efficacement des problèmes stochastiques de conception pour des réseaux de grande taille.

Le premier article, s'intitule "A Learning-Based Matheuristc for Stochastic Multicommodity Network Design". Nous introduisons et décrivons formellement un nouveau mécanisme d'apprentissage basé sur l'optimisation pour extraire des informations concernant la structure de la solution du problème stochastique à partir de solutions obtenues avec des combinaisons particulières de scénarios. Nous proposons ensuite une matheuristique "Learn&Optimize", qui utilise les méthodes d'apprentissage pour déduire un ensemble de variables de conception prometteuses, en conjonction avec un solveur MIP de pointe pour résoudre un problème réduit.

Le deuxième article, s'intitule "A Reduced-Cost-Based Restriction and Refinement Matheuristic for Stochastic Network Design". Nous étudions comment concevoir efficacement des mécanismes d'apprentissage basés sur l'information duale afin de guider la détermination des variables dans le contexte de la conception de réseaux stochastiques. Ce travail examine les coûts réduits associés aux variables hors base dans les solutions déterministes pour guider la sélection des variables dans la formulation stochastique.

Nous proposons plusieurs stratégies pour extraire des informations sur les coûts réduits afin de fixer un ensemble approprié de variables dans le modèle restreint. Nous proposons ensuite une approche matheuristique utilisant des techniques itératives de réduction des problèmes.

Le troisième article, s'intitule "An Integrated Learning and Progressive Hedging Method to Solve Stochastic Network Design". Ici, notre objectif principal est de concevoir une méthode de résolution capable de gérer un grand nombre de scénarios. Nous nous appuyons sur l'algorithme Progressive Hedging (PHA), ou les scénarios sont regroupés en sous-problèmes. Nous intégrons des methodes d'apprentissage au sein de PHA pour traiter une grand nombre de scénarios. Dans notre approche, les mécanismes d'apprentissage developpés dans le premier article de cette thèse sont adaptés pour résoudre les sous-problèmes multi-scénarios. Nous introduisons une nouvelle solution de référence à chaque étape d'agrégation de notre ILPH en exploitant les informations collectées à partir des sous problèmes et nous utilisons ces informations pour mettre à jour les pénalités dans PHA. Par conséquent, PHA est guidé par les informations locales fournies par la procédure d'apprentissage, résultant en une approche intégrée capable de traiter des instances complexes et de grande taille.

Dans les trois articles, nous montrons, au moyen de campagnes expérimentales approfondies, l'intérêt des approches proposées en termes de temps de calcul et de qualité des solutions produites, en particulier pour traiter des cas très difficiles avec un grand nombre de scénarios.
Mots clés: Conception de réseaux multiproduits avec coût fixe et capacité, demandes stochastiques, matheuristique, apprentissage.

ABSTRACT

This dissertation consists of three studies, each of which constitutes a self-contained research article. In all of the three articles, we consider the multi-commodity capacitated fixed-charge network design problem with uncertain demands as a two-stage stochastic program. In such setting, design decisions are made in the first stage before the actual demand is realized, while second-stage flow-routing decisions adjust the first-stage solution to the observed demand realization. We consider the demand uncertainty as a finite number of discrete scenarios, which is a common approach in the literature.

By using the scenario set, the resulting large-scale mixed integer program (MIP) problem, referred to as the extensive form (EF), is extremely hard to solve exactly in all but trivial cases. This dissertation is aimed at advancing the body of knowledge by developing efficient algorithms incorporating learning mechanisms in matheuristics, which are able to handle large scale instances of stochastic network design problems efficiently.

In the first article, we propose a novel *Learning-Based Matheuristic for Stochastic Network Design Problems*. We introduce and formally describe a new optimization-based learning mechanism to extract information regarding the solution structure of a stochastic problem out of the solutions of particular combinations of scenarios. We subsequently propose the *Learn&Optimize* matheuristic, which makes use of the learning methods in inferring a set of promising design variables, in conjunction with a state-of-the-art MIP solver to address a reduced problem.

In the second article, we introduce a *Reduced-Cost-Based Restriction and Refinement Matheuristic*. We study on how to efficiently design learning mechanisms based on dual information as a means of guiding variable fixing in the context of stochastic network design. The present work investigates how the reduced cost associated with non-basic variables in deterministic solutions can be leveraged to guide variable selection within stochastic formulations. We specifically propose several strategies to extract reduced cost information so as to effectively identify an appropriate set of fixed variables within a restricted model. We then propose a matheuristic approach using problem reduction techniques iteratively (i.e., defining and exploring restricted region of global solutions, as guided by applicable dual information).

Finally, in the third article, our main goal is to design a solution method that is able to manage a large number of scenarios. We rely on the progressive hedging algorithm (PHA) where the scenarios are grouped in subproblems. We propose a two phase *integrated learning and progressive hedging (ILPH)* approach to deal with a large number of scenarios. Within our proposed approach, the learning mechanisms from the first study of this dissertation have been adapted as an efficient heuristic method to address the multi-scenario subproblems within each iteration of PHA. We introduce a new reference point within each aggregation step of our proposed ILPH by exploiting the information garnered from subproblems, and using this information to update the penalties. Consequently, the ILPH is governed and guided by the local information provided by the learning procedure, resulting in an integrated approach capable of handling very large and complex instances.

In all of the three mentioned articles, we show, by means of extensive experimental campaigns, the interest of the proposed approaches in terms of computation time and solution quality, especially in dealing with very difficult instances with a large number of scenarios.

Keywords: Multicommodity capacitated network design problem, stochastic demands, matheuristics, learning.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CCP	Chance Constrainted Programming
MCFND	Multi-commodity Capacitated Fixed-charge Network Design
MIP	Mixed Integer Programming
MP	Mathematical Programming
ND	Network Design
NDP	Network Design Problem
RO	Robust Optimization
SNDP	Stochastic Network Design Problem
SPR	Stochastic Problem with Recourse

To my beloved parents and husband.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere appreciations to my advisors, Professors Teodor Gabriel Crainic and Walter Rei for all of their support, expertise and guidance to make my Ph.D. experience productive and stimulating. It has been an honor and a privilege to work with you. I'm deeply grateful. I am also so thankful to my committee members, Professors Jean-Yves Potvin, Bernard Gendron and Walid Klibi, for their time, feedback, and helpful comments.

I would like to thank all the members of CIRRELT for providing a very pleasant research environment and the administrative staffs of DIRO for their various forms of support during my PhD studies. Also, I'm so grateful to all of my friends and PhD colleagues for their friendship and support during these years.

I gratefully acknowledge the Natural Sciences and Engineering Council of Canada (NSERC), the Fonds Québécois de la Recherche sur la nature et les technologies (FQRNT) and Université de Montréal for providing the full financial scholarships during my Ph.D. studies.

I cannot express enough gratitude toward my beloved parents, Masoomeh and Mahmood, for their unconditional and unlimited love, for their patience and wisdom, and for instilling in me the desire for higher education and the principles that have guided my path. I also want to thank my beautiful sister, Faezeh, for her loving support.

Last but not least, I want to specially thank my dearest husband, Majid, for the wonderful time we had together, as well as for all his love, understanding and outstanding support during the last years.

CHAPTER 1

INTRODUCTION

Network design problems are among the most-often studied problems in the operations research community for decades, because of their rich combinatorial structure as well as theoretical significance. The idea is to establish a network of links (roads, optical fibers, electric lines, etc.) that enables the flow of commodities (people, data packets, electricity, etc.) in order to satisfy some demand characteristics. We are particularly interested in *multicommodity capacitated fixed-charge network design (MCFND)*, where, it is required to route, at minimum total cost, a set of given commodities between different pairs of origin and destination nodes respecting the link capacities. In order to use a link, in addition to the variable cost proportional to the amount of flow, one must pay a fixed cost representing, for example, the cost of constructing a road, or installing an electric line, etc.

These problems naturally appear in a large number of contexts including transportation, telecommunications and power systems. Numerous aspects of transportation planning can be represented by *fixed-charge network design* models. They appear in a full hierarchy of planning levels from strategic capital investments to day-to-day operational scheduling. One significant area is the *service network design problem* which arises, for example, in airline and trucking companies [7, 75].The outbreak of new technologies in telecommunications has also provided a fertile ground for the application of network design models. These studies include, for example, the design of a local access network with one or two technologies [96, 97], and the design of terminal layout in a centralized computer network [48, 55]. In power system applications, fixed-charge network design is used to plan the transmission system which carries electricity from the the generation plants to costumer centers [16, 104] and the distribution of energy inside each center [35, 47]. In the latter case, network models can also be used in an operational context to obtain the configuration that minimizes daily loss costs [20].

In any of these applications, classical deterministic models do not reflect the true dynamic behavior of real-world situations because they fail to take into account uncertainty. Unfortunately, critical parameters such as demands, prices, and capacities are quite uncertain in real-life problems. *Stochastic programming* (we refer to [17] for an introduction to stochastic programming) is concerned with the challenging issue of how to make optimal decisions in uncertain environments. In this dissertation, we address the stochastic network design problem with uncertain demands as a two-stage stochastic program [17], 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 solution given in the first stage to the observed demand realization. To take the demand uncertainty explicitly into account, we consider a finite number of discrete scenarios for the values of uncertain demands together with the associated probabilities, which is a common approach in the literature. Our assumption in this dissertation is that the sets of scenarios are pre-determined and are given as the benchmark instances. The general goal of stochastic programming in network design problems is to find a single design solution that performs well at minimum cost, when evaluated in the stochastic environment.

By using the scenario set, the resulting large-scale mixed integer program (MIP), referred to as the extensive form (EF) [17], endures a remarkable complexity making the state-of-the-art solvers incapable to solve real-size problem instances. The complexity comes from two sources: 1) deterministic network design problems are NP-hard in all but trivial cases [82] and 2) modeling uncertainty with scenarios can yield very large instances [29].

One of the simplest traditional ideas to deal with uncertain parameters is to estimate them and then to apply sensitivity analysis afterwards. However, Higle and Wallace [63] indicate that solving many deterministic problems for each possible outcome and applying “what-if-analysis” may lead to arbitrary bad solutions in the case of stochastic problems. More specifically within the context of network design problems, recent studies have shown that cost-effective design solutions obtained within stochastic settings are structurally different from those obtained within deterministic settings [76]. Nevertheless, given the fact that deterministic formulations are generally considered to be simpler to solve as compared to stochastic formulations, a number of studies have closely examined solutions to deterministic variants of stochastic formulations so as to infer information about potential stochastic solutions. Researchers sought to analyze these solutions with the hope of revealing information which could be applicable in solving corollary stochastic models, given previous findings that stochastic solutions retain parts of deterministic solutions. Yet, no systematic procedure to identify these parts, which would provide the basis for efficient algorithmic developments for the MCFND, may be found in the literature.

From the methodological point of view, exact solution approaches are the most frequently used methods applied towards solving Stochastic Network Design Problem (SNDP). These methodologies are effective at finding optimal solutions, but require extensive computational resources and are extremely time-consuming when applied to real-world problems. Heuristic and metaheuristic techniques are alternative problem solving options which are able to produce good solutions in a reasonable time when applied towards difficult problems.

Due to the increasing complexity and dimensionality of network design problems, researchers have been driven to develop more sophisticated approaches. In recent years, *matheuristics* has emerged as an attractive class of method in deterministic problems. These methods involve hybrid problem solving methodologies which exploit both heuristic search frameworks and exact solution methods (see, e.g., [92, 94] for a survey and a taxonomy). These techniques are of particular applicability in solving stochastic problems due to their associated complexity. However, there are relatively few examples of matheuristic techniques applied as potential solution methods for stochastic problems that are reported in the literature. This provides a strong incentive to develop this type of solution approach to design efficient networks considering uncertainty requirements in reasonable amount of time.

The present dissertation seeks to address these research gaps by incorporating intelligence in matheuristic approaches, through innovative *learning mechanisms* enabling the extraction of solution structure in stochastic problems. Within the context of network design, this information takes the form of design decisions which are common to high-quality (i.e., optimal or near optimal) solutions. Learning and memorizing mechanisms in heuristics represent the information extracted and stored during the search for better solutions. We explore a wide range of learning-based solution approaches designed to solve these stochastic network design problems, where the content of these mechanisms varies from one heuristic to the next. We study the design of algorithms that produce solutions to SNDP by iteratively solving restrictions of the problem via MIP technology. Our work is aimed at developing efficient algorithms that incorporate learning mechanisms in matheuristics, which are able to handle large scale instances of stochastic network design problems efficiently.

1.1 Contributions

Consisting of three self-contained studies, the present dissertation intersects two streams of research bridging stochastic network design and learning based matheuristics. Using knowledge derived from solution structures is not a new concept in addressing stochastic network design models, yet how this knowledge is obtained and exploited remains a key factor in ensuring the successful integration of any learned information. The main contributions of this thesis may be summarized as follows.

In the first study, we introduce and formally describe a new optimization-based learning mechanism designed to extract solution structure information from stochastic solutions through particular combinations of scenarios. In fact, a global image of the promising structure of the stochastic solution is built by gradually learning from the partial knowledge produced by the learning mechanism supporting the collection and use of the memory. We subsequently propose the *Learn&Optimize* matheuristic, which makes use of the learning methods in inferring a set of promising design variables, in conjunction with a state-of-the-art MIP solver to address a reduced problem.

The second work is an attempt to study how to efficiently design learning mechanisms based on dual information as a means of guiding variable fixing in the context of stochastic network design. As mentioned previously, deterministic solutions carry useful information (i.e., structural patterns) which can be leveraged to solve stochastic cases.

The present work investigates how the reduced cost associated with non-basic variables in deterministic solutions can be leveraged to guide variable selection within stochastic formulations. We specifically propose different strategies to extract reduced cost information so as to effectively identify an appropriate set of fixed variables within a restricted model. The restriction involves fixing two sets of identified arcs to open or close, so as to subsequently consider only the remaining arcs. We then propose a matheuristic approach which iteratively defines restricted problems constructed by exploiting reduced cost information extracted from multiple solutions.

Finally, in the third study, a solution method which is able to manage a large number of scenarios is proposed. The main contribution in this work is the development of a method able to produce high-quality solutions to large-scale instances of stochastic network design problems, for which standard MIP solvers are unable to find feasible solutions. These MIP solver limitations exist either due to the fact that instances are too big to load into memory, or the time required to solve even the root node relaxation is prohibitive. As the progressive hedging algorithm (PHA) of Rockafellar and Wets [103] is considered as a successful metaheuristic strategy to address the size of problems in stochastic network design problems, we rely on the PHA where the scenarios are grouped in subproblems. Within our proposed *integrated learning and progressive hedging (ILPH)* approach, the learning mechanism forwarded in the first study of this dissertation has been adapted as an efficient heuristic method to address the multi scenario subproblems within each iteration. We introduce a new reference point in the aggregation step of the proposed ILPH by exploiting the information garnered from subproblems, and using this information to update the penalties. Consequently, the ILPH is governed and guided by the local information provided by the learning procedure, resulting in an integrated approach capable of handling extremely large and complex instances.

In all of the three mentioned studies, we show, by means of an extensive experimental campaigns, the interest of the proposed approaches in terms of computation time and solution quality, especially in dealing with very difficult instances with a large number of scenarios.

1.2 Outline of the dissertation

The remainder of this dissertation is organized as follows. In Chapter 2, we start by providing a background on the deterministic network design problems and related contributions on heuristic methodologies. We then provide a comprehensive review on modeling and methodologies applied in the literature of stochastic network design problems.

Chapters 3 through 5 present the three articles that have been produced over the course of these doctoral studies. Chapter 3 presents *A Learning-Based Matheuristic for Stochastic Network Design*, which has been submitted for publication to *INFORMS Journal on Computing*. Chapter 4 presents *A Reduced-Cost-Based Restriction and Refinement Matheuristic for Stochastic Network Design*, which has been submitted for publication to *European Journal of Operation Research*. Chapter 5 presents *An Integrated Learning and Progressive Hedging Method to Solve Stochastic Network Design*, which is expected to be submitted to *EURO Journal on Computational Optimization*. Finally, Chapter 6 provides the conclusions and potential future research directions.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

This chapter provides a background to the area of our research as well as a review of related works in the literature. The network design problem (NDP) and its variants, in both deterministic and stochastic settings, have been the object of numerous studies. This chapter therefore consists of two main parts. In the first part, we present the deterministic formulations and related solutions methods (Section 2.1). We present key elements of a network design problem and focus on a popular setting of the generic formulation, MCFND. We then summarize a number of solution methods focusing on metaheuristic approaches proposed in the literature for this problem. In the second part, we review different existing stochastic modeling approaches as well as the related works in the literature of stochastic network design (Section 2.2).

2.1 Deterministic network design

In this section, we focus on the MCFND problem which is an important classic problem appearing in many applications. Our research works in this thesis are also developed on top of this model. For completeness, detailed reviews on network design problems can be found in [10, 26, 82, 83]. In the following, we first present the mathematical formulation.

2.1.1 Arc-based formulation

Generally, a network design problem is defined on a graph $\mathscr{G} = (\mathscr{N}, \mathscr{A})$ in which $\mathscr{N}$ and $\mathscr{A}$ refer to a set of nodes and arcs (or links), respectively. Arcs (links) correspond to directed connections between given locations (nodes) to carry the flows to satisfy the demands. There is a predetermined *capacity* on each arc u_{ij} that makes the problem to be known as *capacitated network design*. Let $\mathscr{K}$ be the set of commodities where each of them is recognized by a unique pair of origin-destination $(o(k) - d(k))$ with associated demand d^k. It is worth noting that the commodities are distinguishable if they are different physical products or they have different origin-destination pairs. The commodities share the common capacity installed on each arc.

There are two types of variables; discrete *design variables* y_{ij} and continuous *flow variables* x_{ij}^k. Design variables y_{ij} are binary variables indicating whether an arc is chosen in the design ($y_{ij} = 1$) or not ($y_{ij} = 0$). Flow variables x_{ij}^k represent the amount of commodity k distributed on arc (i, j). Then a network design model becomes,

$$\underset{y,x}{\text{minimize}}\ z(y,x) \tag{2.1}$$

$$\text{subject to} \quad \sum_{j\in\mathscr{N}^+(i)} x_{ij}^k - \sum_{j\in\mathscr{N}^-(i)} x_{ji}^k = d_i^k, \qquad \forall i \in \mathscr{N},\ \forall k \in \mathscr{K} \tag{2.2}$$

$$\sum_{k\in\mathscr{K}} x_{ij}^k \leq u_{ij}y_{ij}, \qquad \forall (i,j) \in \mathscr{A}, \forall k \in \mathscr{K} \tag{2.3}$$

$$(y,x) \in S \tag{2.4}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in \mathscr{A} \tag{2.5}$$

$$x_{ij}^k \geq 0, \qquad \forall (i,j) \in \mathscr{A},\ \forall k \in \mathscr{K} \tag{2.6}$$

When the objective function $z(y,x)$ in this formulation is linear as follows,

$$z(y,x) = \sum_{(i,j)\in\mathscr{A}} f_{ij}y_{ij} + \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} c_{ij}^k x_{ij}^k \tag{2.7}$$

the model becomes a MIP. Two types of costs are considered in this problem; a *fixed cost*, f_{ij}, and a *variable cost*, c_{ij}. The former is incurred as soon as a particular arc (i, j) is chosen to be used in the network while the latter is a utilization cost which is proportional to the volume of traffic of commodity k on a given arc (i, j). The objective (2.7) indicates that the goal of the problem is to minimize the total network cost: the sum of the fixed cost of the included arcs in the final design and the variable cost of routing the commodities.

Generally speaking, the *MCFND* problem is to determine a set of arcs and multi-commodity flows of minimal total fixed and variable costs to enable the network to satisfy the demands while the arc capacities are not violated. As a matter of fact, the interplay between fixed and variable cost included in the objective function determines if an arc should be selected or not in the final design. A more complex type of objective function is nonlinear, which could be used to model congestion effects and concave functions other than fixed cost.

Equations (2.2), called the *flow conservation constraints*, ensure that each commodity is routed from its origin node to its destination node. In these relations, the difference between the sum of incoming flows ($\mathscr{N}^+(i)$ is the set of nodes having arcs toward node *i*) and outgoing flows ($\mathscr{N}^-(i)$ is the set of nodes having arcs from node *i*) at each node *i* is equal to the demand volume d_i^k, where

$$d_i^k = \begin{cases} d^k & \text{if } i = o(k) \\ -d^k & \text{if } i = d(k) \\ 0 & otherwise. \end{cases} \tag{2.8}$$

The demand of commodity $k \in \mathscr{K}$ at an *origin node*, $o(k) \in \mathscr{N}$, is a negative value $d^k_{o(k)} = -d^k$, at a *destination node*, $d(k) \in \mathscr{N}$, is a positive value $d^k_{d(k)} = d^k$, and at a *transshipment node*, $t \in \mathscr{N}$, $d_t^k = 0$.

The *capacity or bundle constraints* (2.3) ensure that the total commodities flowing on each arc (i, j) is less than or equal to u_{ij} provided that the arc (i, j) is chosen in the design ($y_{ij} = 1$), and otherwise $y_{ij} = 0$. These constraints actually couple the commodities together. However, in the case of *uncapacitated* network, there is no such constraint and therefore, the problem can be decomposed in $|\mathscr{K}|$ *shortest path problems*. Relations (2.5) and (2.6) indicate that design and flow decision variables should be binary and continuous, respectively.

Other variations and additional restrictions may be included in constraint (2.4) as *side constraints*. Restrictions on the total flow on different arcs or relationships among the flow and design variables can be captured in such constraints.

Topological aspects of the network such as precedence constraints pertaining to the open arcs may be included as a side constraint. Another important additional constraint is known as the *budget constraint*, which indicates that available resources to design arcs are restricted:

$$\sum_{(i,j)\in\mathscr{A}} f_{ij} y_{ij} \leq B \qquad \text{(Budget Constraints)}$$

This budget constraint illustrates a relatively general class of restrictions imposed upon resources shared by arcs. There is another particular form of such constraints called *partial capacity*, which represents restrictions imposed on the use of some facilities by individual commodities:

$$x_{ij}^k \leq u_{ij}^k \; \forall (i,j) \in \mathscr{A}, \; \forall k \in \mathscr{K} \qquad \text{(Partial Capacity Constraints)}$$

If $u_{ij}^k = \min\{u_{ij}, d^k\}$, these constraints are redundant because of constraint set (2.3). Despite its redundancy, these constraints yield a tighter relaxation and improve the lower bound computation as indicated in [49].

Design-balanced constraints are viewed as side constraints as well. These constraints are concerned with full asset-utilization in terminals meaning that the number of vehicles that leave a given node is equal to the number of vehicles that enter the same node:

$$\sum_{j\in\mathscr{N}^+(i)} y_{ji} - \sum_{j\in\mathscr{N}^-(i)} y_{ij} = 0 \; \forall i \in N \qquad \text{(Design-Balance Constraints)}$$

In the above formulation, we assume every unit flow of each commodity consumes one unit of capacity and the capacity consumption is not commodity-dependent. In the commodity-dependent case, the following capacity constraint replaces (2.3) with a given amount of capacity for each commodity, ρ_{ij}^k:

$$\sum_{k \in \mathscr{K}} \rho_{ij}^k x_{ij}^k \leq u_{ij} y_{ij}, \forall (i,j) \in \mathscr{A}, \forall k \in K; \qquad \text{(Commodity-Dependent Capacity)}$$

It is worth mentioning that the flow conservation constraints together with capacity constraints form the main body of the problem. This is the base model for many well-known problems, such as Traveling Salesman Problem, Spanning Tree Problem and Vehicle Routing Problem which are derived by specific definitions of the networks and side constraints. See [81] for a summary of these problems and how one can derive these problems from the generic formulation.

2.1.2 Alternative formulations

In addition to the above *arc-based* directed network model, other formulations have been proposed in the literature. In the case of telecommunication applications, an undirected graph has been used to formulate the problem. It should be noted that even if the network is undirected, flows are generally directed in these models. In the following, we describe two well known alternative formulations that have been applied to network design problems.

Path-based multicommodity capacitated network design This is one of the equivalent formulations that may be stated in the following form:

$$\begin{aligned}
\underset{y,h}{\text{minimize}} \quad & \sum_{(i,j) \in \mathscr{A}} f_{ij} y_{ij} + \sum_{k \in \mathscr{K}} \sum_{l \in \mathscr{L}^k} c_l^k h_l^k \\
\text{subject to} \quad & \sum_{l \in \mathscr{L}^k} h_l^k = w^k, && \forall k \in \mathscr{K} && (2.9) \\
& \sum_{k \in \mathscr{K}} \sum_{l \in \mathscr{L}^k} h_l^k \delta_{ij}^{kl} \leq u_{ij} y_{ij}, && \forall (i,j) \in \mathscr{A} && (2.10) \\
& y_{ij} \in \{0,1\}, && \forall (i,j) \in \mathscr{A} && (2.11) \\
& h_l^k \geq 0, && \forall k \in \mathscr{K}, \ \forall l \in \mathscr{L}^k && (2.12)
\end{aligned}$$

where,
$\mathscr{L}^k$: the set of paths from the origin $o(k)$ to the destination $d(k)$ for commodity k
h_l^k: flow of commodity k on path l;

$$\delta_{ij}^{kl} = \begin{cases} 1 & \text{if arc } (i,j) \text{ belongs to path } l \in \mathscr{L}^k \text{ for commodity } k \\ 0 & otherwise \end{cases}$$

c_l^k: transportation cost of commodity k on path l, $c_l^k = \sum_{(i,j)\in\mathscr{A}} c_{ij}^k \delta_{ij}^{kl}$

The conservation flow constraints (2.9) and capacity constraints (2.10) are modified to handle the path specifications. Side constraints are usually addressed when the paths are built.

Cut-based formulation Peterson [89] formulated the single-commodity fixed charge capacitated network design problem using cut-flow inequalities as follows:

$$\begin{aligned} \underset{y,x}{\text{minimize}} \quad & z = \sum_{(i,j)\in\mathscr{A}} (f_{ij}y_{ij} + c_{ij}x_{ij}) \\ \text{subject to} \quad & f(S,\bar{S}) \geq v(S,\bar{S}), && \forall (S,\bar{S}) && (2.13) \\ & 0 \leq x_{ij} \leq q_{ij}y_{ij}, && \forall (i,j) \in \mathscr{A} && (2.14) \\ & y_{ij} \in \{0,1\}, && \forall (i,j) \in \mathscr{A} && (2.15) \end{aligned}$$

where,

$$\begin{cases} (S,\bar{S}) & \text{is called a cut } S \subset N, \bar{S} \subset N - S \\ f(S,\bar{S}) & \text{is the sum of flows on all links contained in the cut } (S,\bar{S}) \\ v(S,\bar{S}) & \text{is the volume of flow requirements between two sets of nodes } S \text{ and } \bar{S} \end{cases}$$

The cut $(S,\bar{S})$ is defined as the set of arcs with origins in $S \subset N$ and destinations in $\bar{S} \subset N - S$. Then, in the above cut-flow formulation, instead of the conservation flow constraints, we have cut-flow inequalities (2.13) which require that the flow through each cut be at least equal to the total flow requirements between the origins and destinations separated by the cut.

2.1.3 Solution methods

The solution methods proposed for network design problems may be classified into two broad categories: exact methods and heuristic methods. Exact solution methods such as Branch-and-Bound and Branch-and-Price can find optimal solutions, but they are often extremely time-consuming when solving real-world problems. A good overview of exact methods, together with descriptions of some application areas, can be found in [2]. Since the focus of this dissertation is to develop solution approaches that produce high-quality solutions quickly, relevant studies on metaheuristics as well matheuristics within the context of network design problems will be discussed in the following.

Meta-heuristics are strategies to guide the search process which make use of low-level heuristics to find solutions. These methods do not guarantee to find global optimal solutions; however, they can often find good solutions with less computational effort than exact methods. The inherent difficulty of network design, combined with the large size of instances encountered in practical applications, leaves little hope for exact solution approaches that run in reasonable time. Therefore, fast heuristics and metaheuristics appear to be the method of choice. The proposed metaheuristics approaches for network design problems are mostly based on tabu search [53], path relinking and scatter search [27, 100] to tackle large size instances.

Crainic et al. [31] proposed an integrated tabu search method by combining simplex-based moves with column generation. In fact, they used the similarity between the path-based formulation of multi commodity network flow and master problem in column generation to view the problem as a Dantzig-Wolfe decomposition approach. This algorithm considers the impact of changing the flow of just one commodity flow with each move. The results show that the simplex-based tabu search identifies good solutions within reasonable computing effort and it outperforms relaxation-based heuristics in terms of solution quality.

Ghamlouche et al. [50] proposed a new type of cycle-based neighbourhood to be integrated within a tabu search framework for the MCFND. The key feature of the proposed metaheuristic, which has been a drawback in [31], is defining the neighbourhood such that each move can potentially change the flow of several commodities, simultaneously. The idea in the new neighbourhood structure was redirecting the flow through the cycles by closing and opening some arcs. They proposed to move from one solution to another by considering these cycles which examine a broader range of moves because the flow may deviate between paths linking any two nodes and are not just restricted to the origin and destination of actual commodities. They integrated the cycle-based neighborhood in a simple tabu search algorithm to explore the search space more efficiently. The results show that the proposed method provides best approximate solutions for the MCFND in terms of solution quality and computing efficiency.

Ghamlouche et al. in [51] developed a path relinking metaheuristic as a follow up of the earlier work in [50]. After proposing the cycle based neighbourhood structure, they developed a more refined search method to obtain a more powerful heuristic. Since the selection of the best move in the neighbourhood requires exhaustive computations, they proposed an efficient procedure to avoid the complete evaluation of every examined move. As a path relinking method, they proposed and implemented different strategies in identifying the reference set and guiding solutions. The computational experiments show that path-relinking provides better results than tabu search.

Crainic and Gendreau [30] proposed a scatter search algorithm for the MCFND. Scatter search creates new solutions by using existing solutions in a candidate set. A new design, which is a set of open links, is obtained by a linear weighted combination of existing designs in the candidate set. Then, flow information is obtained by solving a multi-commodity network flow problem defined on open links. Extensive computational experiments have shown that, on average, the most effective variants of the scatter search heuristic do not perform better than the best existing method in [51], but they give close results.

In the above-mentioned works, the focus of the proposed solution methods is on local search methods within metaheuristics approaches. As for matheuristics, however, we have so far come across only a couple of papers that propose such hybrid methods in the context of NDP.

A solution framework that combines mathematical programming algorithms and heuristic search techniques is introduced by Hewitt et al. [61]. Their methodology uses very large neighborhood search in combination with an IP solver on an arc-based formulation of the MCFNDP, and LP relaxation of the path-based formulation using cuts discovered during the neighborhood search.

Vu et al. [122] and Chouman and Crainic [23] also proposed new matheuristics for multi commodity capacitated fixed charge network design considering design-balance constraints. Vu et al. [122] developed a three-phase matheuristic that combines tabu search with path relinking and exact methods. While heuristics are used to explore the solution set, the exact algorithm is used to intensify the search in a specific part of the solution set.

Chouman and Crainic [23] proposed a matheuristic combining an exact lower bound computing method and variable fixing heuristic. This study is motivated by previous effort in [24] in which the authors have proved the efficiency of a cutting plane procedure in computing tight lower bounds for the MCFND. Generally, the main idea is to compute a lower bound on the optimal value using the proposed cutting planes in [24] and compiling statistics on solution characteristics. The embedded learning mechanism in the cutting plane method guides the variable fixing heuristic to reduce the size of the problem and use commercial MIP solvers. The results show that their proposed matheuristic not only reduces the computational time but also achieves high quality feasible solutions.

The above mentioned heuristic algorithms and other relevant solution methods for deterministic problems cannot solve stochastic network design problems with satisfactory performance and scalabilities. This can be explained as follows. First, it has been shown that the solution structures of deterministic and stochastic problems are different, and second, the performance of a local search algorithm largely relies on how the neighborhood of a solution is defined. Therefore, there is a need to design some special heuristic methods that are able to extract the solution structure of stochastic problems. This dissertation aims at proposing such specially designed heuristic methods.

2.2 Stochastic network design

So far we just studied deterministic network design models considering the demand and all parameters in the model are known in advance. However, this is not the case in real life problems. The main parameters of the model including demand, cost and capacity could be different sources of uncertainty that affect design decisions significantly. Hence, considering the uncertainty in network design problems is needed for more realistic problems.

Such problems fall into the framework of stochastic programming, an area that consists in modeling and methodology approaches for optimizing the performance while taking the uncertainty explicitly into account. We therefore start this section with a relatively general discussion on various modelling approaches used in the stochastic programming in subsection 2.2.1 and then review the related methodologies in stochastic network design literature in subsection 2.2.2.

2.2.1 Modelling approaches

Regarding modelling approaches in stochastic programming, we may classify the proposed approaches in three major categories; stochastic programming with recourse (SPR), robust optimization (RO) and chance constrained programming (CCP). These methods are the classical optimization frameworks that are used for planning with uncertainty.

2.2.1.1 Stochastic programming with recourse

Two-stage model. In a standard two-stage stochastic programming model, decision variables are classified into two groups; namely, first stage and second stage variables. First stage variables, known as *here-and-now* decisions, are decided upon before the actual realization of the random parameters. Once the uncertain events have unfolded, further design or operational adjustments can be made through values of the second-stage variables, alternatively called *recourse decisions*, at a particular cost.

A standard formulation of a two-stage stochastic program is as follows:

$$\begin{aligned} \min_{x} \quad & c^T y + \mathbb{E}\left[Q(y,\xi)\right] \\ \text{s.t.} \quad & Ay = b, \\ & y \in \mathscr{Y}, \end{aligned} \tag{2.16}$$

where ξ is a random vector defined on a probability space (refer to [15] for a rigorous definition of a probability space) and for a particular realization of ξ, $Q(y,\xi)$ is defined as:

$$\begin{aligned} Q(y,\xi) = \min_{x} \quad & q(\xi)^T x \\ \text{s.t.} \qquad\quad & Wx = h(\xi) - T(\xi)y \\ & x \in \mathscr{X}, \end{aligned} \tag{2.17}$$

Here, $c \in \mathbb{R}^{n_1}$, $b \in \mathbb{R}^{m_1}$, $q(\xi) \in \mathbb{R}^{n_2}$, $A \in \mathbb{R}^{m_1 \times n_1}$, $T(\xi) \in \mathbb{R}^{m_2 \times n_1}$, and $W \in \mathbb{R}^{m_2 \times n_2}$ comprise the data of the stochastic program. In this formulation, at the first-stage one needs to make decisions $y \in \mathbb{R}^{n_1}$ before uncertainty is revealed and then takes recourse actions (second-stage decisions) in response to a particular realization of the random vector ξ. The objective $c^T y + \mathbb{E}\left[Q(y,\xi)\right]$ is to minimize the sum of first-stage cost and the expectation of the second-stage costs. The first stage decisions must satisfy the constraint set $Ay = b$. The second-stage decisions $x \in \mathbb{R}^{n_2}$ are subject to a cost $q(\xi)$ and are restricted by constraint $Wx = h(\xi) - T(\xi)y$. First-stage decisions impose constraints on second-stage decisions through the matrix $T(\xi)$. The nature of y and x decision variables in terms of sign, bounds and integrality restrictions are defined by $\mathscr{Y}$ and $\mathscr{X}$, respectively.

In the above two-stage stochastic program, if we assume ξ as the stochastic parameter vector with finite and discrete support, it can be expressed as a finite number of realizations, called *scenarios*. Here, $\mathscr{S}$ is the set of all scenarios and $|\mathscr{S}|$ is the number of scenarios. Then, ξ^s, $\forall s \in \mathscr{S}$, is a given realization of stochastic parameters, and set $\{\xi^1, \xi^2, \dots, \xi^{|\mathscr{S}|}\}$ is the sample space for stochastic parameters with corresponding probabilities $\{p^1, p^2, \dots, p^{|\mathscr{S}|}\}$. The so-called extensive form of problem (2.16) - (2.17) can be written as:

$$
\begin{aligned}
\min\ & c^T y + \sum_{s\in\mathscr{S}} p^s q(\xi^s)^T x(\xi^s) \\
\text{s.t. } & Ay = b \\
& Wx(\xi^s) = h(\xi^s) - T(\xi^s) y \quad \forall s \in S \\
& x \in \mathscr{X}, y \in \mathscr{Y}
\end{aligned}
\qquad \text{(Extensive form)}
$$

Given that the two-stage modelling approach has been used to formulate the majority of stochastic network design problems in the literature and our research is also developed on top of this model, we will particularly focus on the two-stage stochastic network design model and its related works in Section 2.2.2.

Multi-stage stochastic programming The previous section concerned stochastic programs with two stages. However, most practical decision problems entail a sequence of decisions that react to outcomes that move forward over time.

In this section, we will examine multistage stochastic problems. In essence, the multistage stochastic program with recourse can be treated as a natural extension of the two-stage stochastic programming model. In a two-stage SP model, a set of decisions is made and kept until the end. However, in a multistage stochastic program, a sequence of recourse decisions, which make up a decision process, are made consecutively over time.

In the general T stages program, one considers a sequence of random parameters $\xi_1, \xi_2, \ldots, \xi_{T-1}$ defined on a probability space. A *scenario* is defined as a realization of random parameters $\xi_1, \xi_2, \ldots, \xi_{T-1}$ and a *scenario tree* is a computationally viable way of discretizing the underlying stochastic parameter over time. In other words, a scenario tree is an explicit representation of the branching process for progressive observation of $\xi_1, \xi_2, \ldots, \xi_{T-1}$ under the assumption that these stochastic parameters have a discrete support. Figure 2.1 illustrates a scenario tree including 8 scenarios for a four-stage stochastic program. Each arc represents a realisation of a random vector of parameters between the two stages. A path from the root to a leaf node represents an individual scenario.

In a multi-stage stochastic program, the random parameters $\xi_1, \xi_2, \ldots, \xi_{t-1}$ are ob-

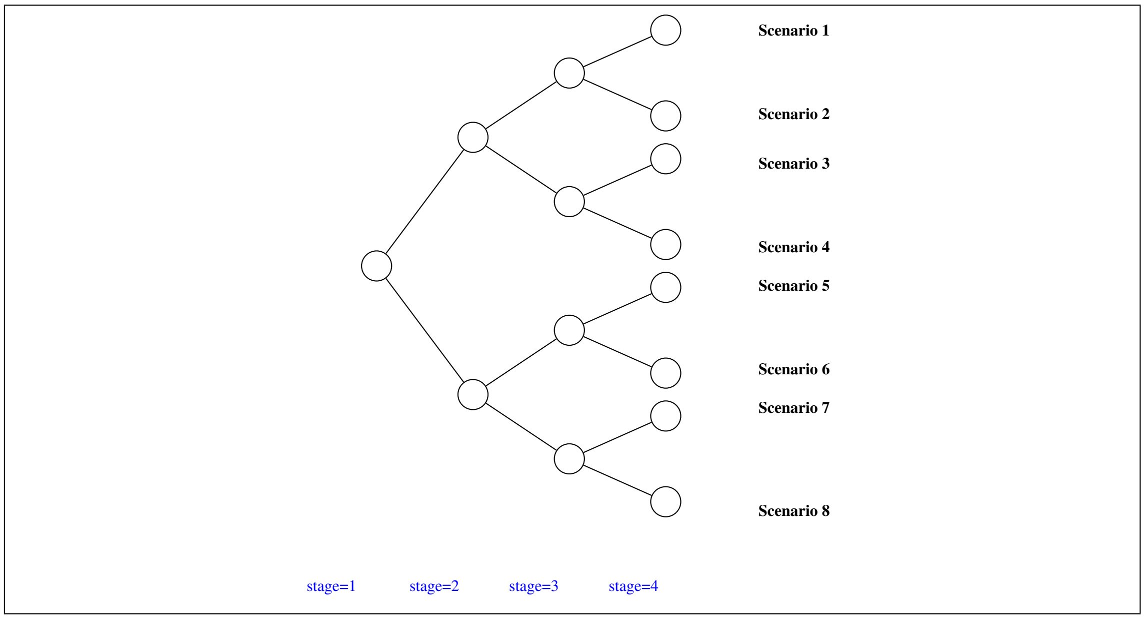

Figure 2.1: Example of scenario tree

served just before taking the decision at stage t and the residual uncertainty includes the random parameters $\xi_t, \dots, \xi_{T-1}$. However, the distribution of these residual stochastic parameters $\xi_t, \dots, \xi_{T-1}$ is conditioned upon the realization of random parameters in previous stages, i.e., $\xi_1, \dots, \xi_{t-1}$ [38]. Considering decision stages numbered from $t = 1$ to $t = T$ with the corresponding decision variables $x^1, x^2, \dots, x^T$, Figure 2.2 represents the sequence of decisions and realizations of random parameters for each stage of a T-stage stochastic program. It is assumed that at each stage $t \geq 1$ the decisions at previous stages $x^1, \dots, x^{t-1}$ and the realisations of the random vectors $x^1, \dots, x^{t-1}$ are known.

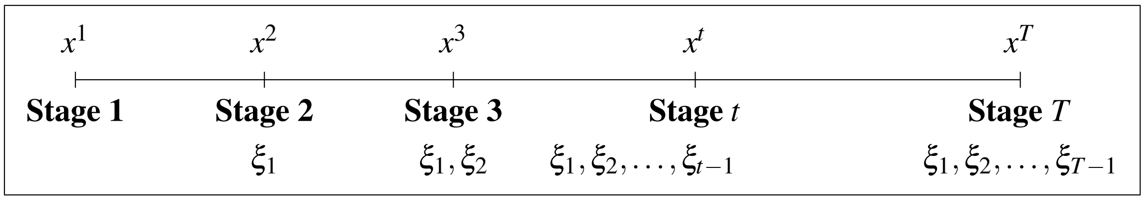

Figure 2.2: Sequence of decisions and realizations of random parameters for each stage of a T-stage stochastic program

In a multi-stage stochastic program, the decision process has to be nonanticipative in the sense that decisions taken in any stage of the process are not dependent on observations relative to subsequent stages or on future decisions. As explained in [40], there are two common approaches to impose nonanticipativity constraints in a multi-stage stochastic programming formulation. In the first approach, non-anticipativity constraints are accounted for implicitly by formulating a multi-stage stochastic program as a sequence of nested two-stage stochastic programs. In this approach, the total objective function is calculated through a recursive evaluation. The second approach imposes non-anticipativity constraints explicitly by introducing a set of decision variables for each stage and each scenario.

Generally, multi-stage stochastic programs have been rarely applied in the stochastic network design literature. There are a limited number of studies in this area such as [3, 54, 85, 90].

2.2.1.2 Robust optimization

Despite the great influence and theoretical impact of stochastic programming, the traditional models described earlier are powerless to handle risk aversion or the decision-maker's preference directly. Moreover, stochastic problems in which the expected total cost is minimized assume that the decision maker is concerned with the average performance of the system. However, there are situations where the decision maker may be worried about the worst-case.

To overcome these drawbacks, Mulvey et al. in [84] proposed an alternative modelling approach called robust optimization by defining different robustness measures for the optimization problem. In this approach, a deterministic worst-case formulation of the original problem is considered in which the worst-case is calculated over all possible values that the input parameters may take within their uncertainty sets. The primary goal of robust optimization is to produce optimum and relatively insensitive solutions ([84]).

In robust optimization problems, uncertain parameters may be continuous or specified via some discrete scenarios. For continuous ones, it is often assumed that these uncertain parameters could be varied within a predefined interval called interval-uncertainty. Generally, robust optimization with interval-uncertain parameters has been applied in order to protect optimization problems against infeasibility due to perturbations of uncertain parameters and also to retain computational tractability. We note that there are only a few studies, for example [39, 91], in robust network design problems with interval-uncertainty.

In the case of discrete scenarios, different robustness measures with or without probability distributions may be considered. Minimax cost and minimax regret are the two popular measures for obtaining a robust network in scenario-based robust optimization programs. The minimax cost measure seeks a solution minimizing the maximum cost over all scenarios. In the minimax regret, the difference between the cost of a solution and that of the optimal solution is defined as the absolute or relative regret for a scenario [114]. The minimax cost is used in [95, 99] and minimax regret in [1, 56]. It is worth mentioning that most studies have used commercial solvers to solve the proposed mathematical models. Also, using robustness measures usually yields multi-objective optimization problems in several studies.

2.2.1.3 Chance constrained program

Chance Constrained Programming (CCP), originally developed by Charnes and Cooper [21], is another approach to model stochasticity in stochastic programs. Sometimes, in optimization problems, one or multiple constraints are not required to be always satisfied. Indeed, these constraints need to hold with some probability or reliability level. Probabilistic or chance-constrained programming is usually applied in such a situation and it is often employed when the distribution probabilities of the uncertain parameters are known by decision makers.

In this approach, the feasibility of a stochastic constraint has to be satisfied with at least a given probability value α. The *chance constraint* (CC) can be expressed as follows:

$$Pr\{a(x,\xi) = a(\xi)x \leq h(\xi)\} \geq \alpha \tag{CC}$$

where, $x \in \mathbb{R}^n$ and for all realizations ξ, $a(\xi) \in \mathbb{R}^n$ and $h(\xi) \in \mathbb{R}$.

The most challenging issue in chance-constrained programs is to obtain a deterministic equivalent formulation. In fact, there are many difficulties associated with transforming chance constraints into deterministic constraints (see [17] and [106] for more details about this issue). In stochastic network design problems, these probabilistic constraints have been developed in a few research studies, such as [125] and [57].

2.2.2 Related works in stochastic network design problems

As already mentioned, due to the two-stage nature of decisions in network design problems, in the majority of the literature, the two-stage modelling approach has been used to formulate the problem. Our research in this dissertation is also based on this model. Hence, we first recall the two-stage formulation of the stochastic network design problem, then review its main applications and uncertainty sources in Section 2.2.2.2 and provide a general review of solution methods and related works proposed for these problems in Sections 2.2.2.3 to 2.2.2.6.

2.2.2.1 Two-stage stochastic formulation for SMCFND problem

Traditionally, in the case of stochastic two-stage network design problems with uncertain demands, the first stage consists of deciding on the network configuration (i.e., design decisions). However, the second-stage consists of commodity flow decisions from origin to destination nodes in an optimal fashion based on the restricted configuration imposed by the first stage and the realized demand variables. The objective of stochastic network design, in general, is to achieve a configuration that performs well under every possible realization of uncertain parameters.

Let us describe the two-stage stochastic formulation for the stochastic MCFND problem [29]. For completeness, we describe all notations although most of them are the same as in the deterministic model. Let $\mathscr{G} = (\mathscr{N}, \mathscr{A})$ be a directed network with $\mathscr{N}$ representing a finite set of nodes and $\mathscr{A}$ a finite set of potential arcs. The set of commodities is represented by $\mathscr{K}$ where each is defined by a unique pair of origin-destination nodes $(o(k), s(k))$. For each design arc $(i, j) \in \mathscr{A}$, we define the fixed cost f_{ij} incurred if the arc is included in the final design and the capacity limit u_{ij} of the total commodity flow that may use the arc (i, j). We also define the unit routing cost c^k_{ij} for each commodity $k \in \mathscr{K}$ and arc $(i, j) \in \mathscr{A}$. Let Ω be the space of random events, where $\omega \in \Omega$ defines a particular realization. Considering that demands are the only stochastic parameters in the model, we let the random vector $\boldsymbol{d}$ define demand distributions. For a given realization $\omega \in \Omega$, assuming that $v^k(\omega)$ is the demand volume of commodity k under the realization ω, the demand of costumer i for commodity k under the realization ω, i.e., $d_i^k(\omega)$, is either set to $v^k(\omega)$ if node i is the origin of commodity k, $-v^k(\omega)$ if node i is the destination of commodity k, or 0 otherwise.

Let the design decision variable $y_{ij} \in \{0,1\}$ indicates if arc (i,j) is included in the network in the first stage. Thus, the two-stage stochastic program for MCFND may be stated as follows:

$$\min \sum_{(i,j)\in\mathscr{A}} f_{ij}y_{ij} + \mathbb{E}_{\boldsymbol{d}}\left[Q(y,d(\omega))\right] \tag{2.18}$$

$$\text{s.t. } y_{ij} \in \{0,1\}, \forall (i,j) \in \mathscr{A} \tag{2.19}$$

where $Q(y,d(\omega))$ is the total routing costs, representing the second-stage recourse function, given the configuration design y and the realized demand vector $d(\omega)$. The objective function (2.18) then minimizes the total cost of the system as the sum of the total fixed costs incurred to build the network and the expected distribution costs associated with using it. Constraints (2.19) impose the integrality requirements on design variables.

The second-stage recourse function may then be formulated as follows:

$$Q(y,d(\omega)) = \quad \min \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} c_{ij}x_{ij}^k \tag{2.20}$$

$$\text{subject to} \quad \sum_{j\in\mathscr{N}^+(i)} x_{ij}^k - \sum_{j\in\mathscr{N}^-(i)} x_{ji}^k = d_i^k(\omega), \quad \forall i \in \mathscr{N}, \forall k \in \mathscr{K} \tag{2.21}$$

$$\sum_{k\in\mathscr{K}} x_{ij}^k \le u_{ij}y_{ij}, \quad \forall (i,j) \in \mathscr{A} \tag{2.22}$$

$$x_{ij}^k \ge 0, \quad \forall (i,j) \in \mathscr{A}, \forall k \in \mathscr{K} \tag{2.23}$$

where continuous decision variables x_{ij}^k represent the amount of commodity k's demand that flows on arc (i,j), while $\mathscr{N}^+(i)$ and $\mathscr{N}^-(i)$ are the sets of outward and inward neighbors of node i, respectively. The objective function (2.20) minimizes the total routing cost, equations (2.21) enforce the flow conservation constraints, and relations (2.22) impose the capacity restrictions on the design arcs of the network. Finally, constraints (2.23) impose non-negativity restrictions on flow variables.

Let $\mathscr{S} \subseteq \Omega$ define a finite set of possible scenarios for the random event, with strictly positive corresponding probabilities of realization $p^1, \ldots, p^{|S|}$. The problem (2.18)-(2.19) may be reformulated as its *extensive formulation*

$$\text{minimize} \quad \sum_{(i,j)\in\mathscr{A}} f_{ij}y_{ij} + \sum_{s\in\mathscr{S}} p^s \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} c_{ij}x_{ij}^{ks} \tag{2.24}$$

$$\text{subject to} \quad \sum_{j\in\mathscr{N}^+(i)} x_{ij}^{ks} - \sum_{j\in\mathscr{N}^-(i)} x_{ji}^{ks} = d_i^{ks}, \qquad \forall i\in\mathscr{N},\ \forall k\in\mathscr{K},\ \forall s\in\mathscr{S} \tag{2.25}$$

$$\sum_{k\in\mathscr{K}} x_{ij}^{ks} \leq u_{ij}y_{ij}, \qquad \forall (i,j)\in\mathscr{A},\ \forall s\in\mathscr{S} \tag{2.26}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j)\in\mathscr{A} \tag{2.27}$$

$$x_{ij}^{ks} \geq 0, \qquad \forall (i,j)\in\mathscr{A},\ \forall k\in\mathscr{K},\ \forall s\in\mathscr{S} \tag{2.28}$$

where the commodity flow variables, x_{ij}^{ks}, and the demands d_i^{ks} are scenario specific. The model (2.24)-(2.28) is a large-scale mixed integer program with a block-diagonal structure, where each block, defined by constraints (2.25) and (2.26), represents the deterministic MCFND for scenario s. By solving problem (2.24)-(2.28), one finds a single design $y_{ij}, \forall (i,j)\in\mathscr{A}$ that minimizes the total system cost, consisting of the sum of the fixed cost for the included arcs and the expected routing costs over all the realizations of demand scenarios.

2.2.2.2 Applications and uncertainty sources

Several applications of stochastic network design models can be found in the literature. Most, but not all, of the existing research is focused in the fields of logistics and telecommunications. As already mentioned, two groups of decisions are involved in network design models; design decisions, that define the structure and characteristics of the network, and flow decisions, which relate to how the network is used to perform the considered operational activities. Considering these two groups of decisions, the two-stage stochastic programming modeling framework suits network design problems well. Therefore, this approach has been used by most of the studies in this area. Generally, in most studies, the first stage decisions belong to the strategic planning level, and the second stage decisions are tactical planning decisions. In Table 2.2.1, we review these two different types of decisions made in a number of different applications of two-stage stochastic network design problems.

In stochastic network design problems, one may face a wide range of possible uncertainties. Klibi et al. [73] investigated different existing uncertainties in supply chain network design as well as their sources and impacts. In general, the uncertainty sources may be classified into the two following groups: (1) the existing uncertainty in parameters such as supply, demand, and costs, which are inherently uncertain, and (2) the uncertainty caused by natural or man-made disruptions. A list of uncertain parameters that have been assumed in designing networks in the literature is reviewed in Table 2.2.2. Among the reference papers, a minority of them have addressed disruptions in supply chain network design problems. The influence of disruptions on the physical structure of a network may cause uncertainty in various parameters. The most frequent parameters which have been assumed uncertain in the case of a disruption event in supply chain network design problems are as follows: capacity of facilities, availability of facilities and their connections, and the amount of disrupted products in facilities [73].

Most of solution methods developed to address stochastic problems under different sources of uncertainty (e.g., costs and capacities) solve the extensive form (or deterministic equivalent) of the two stage stochastic program . It is worth to mention that these

Table 2.2.1: Applications of SNDP

Application			
	Description	First-stage	Second-stage
Transportation	(Service) Network Design [29, 76]	Structure of network	Routing decisions
	Two tier city logistic [28]	First tier service network	Design & routing
Logistic	Multi-echelon supply chain [107, 110, 119]	Topology of supply chain	Flow decisions + Network use
Telecommunication	Internet-based information service [46]	What equipment to install in the network	Operations to perform the service
	Capacity planning problem [101]	Capacity allocation	Flow decisions
	Ring design problem for optical network [112]	Assignment of the client nodes to rings	Demand transit

proposed approaches (e.g., the branch-and-fix with coordination method [4] or the integrated SAA and Benders method proposed in [107]) solve instances where the size of the resulting deterministic equivalent problem is still limited, compared to the size of MIP problem obtained by considering the demand uncertainty in the instances considered in this dissertation. Our goal in this dissertation is to propose efficient solution methodologies that are able to handle very large problems.

Table 2.2.2: Uncertainity sources

Parameters	**Refrences**
Demand	[4, 12, 98, 105, 107, 115]
Parameters of demand distribution	
Costs	[4, 9, 52, 107]
Costs of activities (e.g., transportation, production)	
Capacity	[58, 107]
Capacity of network facilities/ transportation links	
Required capacity for producing products	
Capacity coefficients	
Supply	[9, 12, 69, 98, 107, 115]
Supply quantity for network facilities	
Price	[4, 52, 69]
Selling price of finished products	
Buying price of raw materials	

2.2.2.3 Scenario generation

A crucial step during the implementation of stochastic program models is modeling the random parameters to well reflect the available knowledge on the randomness at hand. Stochastic parameters in SND may be represented by either continuous parameters or discrete scenarios. In a small portion of the literature, the stochastic parameters are described using a known continuous probability distribution where numerical integration is employed over the random continuous probability space. The main disadvantage of this approach is that the computation is difficult to carry out since multidimensional integration is required. In these studies, the most popular stochastic parameter is the customer's demand volume, which is modeled through the normal distribution with known mean and variance. The paper presented by Daskin et al [36] is a foundation for many studies in the area of SND where demands have normal distribution with known mean and variance (e.g., [57, 88]).

Compared with continuous stochastic parameters, approximation by a discrete set of outcomes (scenario approach) yields more manageable models. The dependency among stochastic parameters may be captured by using the scenario approach. In the implementation of multi-stage or two-stage stochastic programs, the discrete scenarios are usually organized in the form of a scenario tree or scenario fan, respectively. In such approaches, not only the parameters can be correlated with each other, but also they can be correlated across the time units and, therefore, the generation of an appropriate set of scenarios would be a difficult task.

The literature on scenario generation is rich and various methods have been developed over the past decades. They range in scope from sampling methods to simulation, from statistical methods (such as principal component analysis technique, regression methods, moment matching) to other methods (e.g. clustering approaches, neural networks) [13]. The most common approach is to generate a scenario tree from the probability distribution by sampling. In the case of correlated random variables, it is then necessary to specify the marginal distributions and the correlation matrix [19]. If the probability distributions are not available, scenarios could be generated with required moments e.g., mean, variance, skewness, etc. These studies may be found for example in [65, 78, 79, 113].

It is easy to perceive that the main problem in all these methods is that the size of the tree grows exponentially with the dimension of the random vector and leads to difficulties in solving the model. Subsequently, scenario reduction techniques [42, 60] can be applied to reduce the size of the tree with the minimum loss of accuracy.

Evaluating the scenario generation methods in terms of quality and stability is the main concern. In this regard, there are two important requirements for an efficient scenario generation procedure including in-sample and out of sample stabilities. We refer to [68], for more information about quality and stability measures in scenario generation methods.

We note that, in the context of SND, there are only a few studies that develop an appropriate scenario generation procedure to obtain a set of scenarios (e.g., [44], [56], [72] and [110]). Typically most research studies exploited a predetermined small set of scenarios with associated probabilities for their stochastic programs.

2.2.2.4 Sampling-based method

In contrast to using a static set of scenarios, sampling-based solution methods dynamically generate sets of representative scenarios for the problem. These methods are usually used for the stochastic programs with a prohibitively large number of scenarios. By applying sampling based approaches, the objective function is approximated through a random sample of scenarios. Sampling techniques can be typically classified in two categories: Interior sampling and exterior sampling methods [121].

In interior sampling methods, sampling is performed and modified during the chosen optimization procedure. These samples may be modified by adding to previously generated samples, by taking subsets of previously generated samples, or by regenerating new sample from scratch. This type of interior sampling based approaches may be found in several studies. For example [62] and [66] developed methods for stochastic linear programming that modify samples within the L-shaped algorithm. For discrete stochastic problems, interior sampling branch-and-bound methods were proposed by Norkin et al in [86, 87].

In the exterior sampling approach, a sample of scenarios is generated according to probability distribution, and then a deterministic optimization problem is developed for the generated samples and solved. The procedure of generating samples and solving deterministic problems may be repeated several times. One of the well-known example of exterior sampling method is the Sample Average Approximation (SAA) approach. In the network design problems, SAA methods have been broadly developed to reduce the size of stochastic programs through repeatedly solving the problem with a smaller set of scenarios. These studies include, e.g., [8, 22, 71, 72, 107].

2.2.2.5 Decomposition-based methods

Computation in stochastic programs with recourse has focus on two-stage problems with finite numbers of realizations. Using a finite number of second-stage realizations, we can always form the full deterministic equivalent program (i.e., the extensive form). Due to the the large scale of the resulting problem, taking advantage of their structure through decomposition-based approaches is especially beneficial and is the focus of much of the algorithmic work in this area. The basic idea behind decomposition methods is to divide a large-scale stochastic problem into several subproblems. Such decomposition strategies can be categorized into two types. The first type decomposes the problem by stages, while the second type decomposes the problem by scenarios. The former category (referred to the L-shape method introduced in [120]) is a cutting-plane method which is the application of the Benders decomposition strategy to the solution of the extensive form of stochastic program. When Benders decomposition is applied to two-stage stochastic linear problems, the first stage is formulated as the master problem providing lower bounds, and a subproblem is formed for each scenario. All the subproblems together generate upper bounds and cuts for the master problem. The lower bound and upper bound eventually converge to the optimal solution.

Following this strategy, the stochastic network design model is first projected onto the space defined by the first stage variables (i.e., the design variables). By doing so, the problem decomposes according to the considered scenarios (i.e., a flow model for each scenario). The problem is then solved by reformulating the scenario subproblems using an outer linearization approach and then applying a relaxation algorithm on the resulting equivalent model. For completeness, detailed review on this type of decomposition approach for stochastic network design problems may be found in [32] and [93].

In the second category of decomposition strategies, referred to as scenario decomposition, the original problem is decomposed per scenarios by applying Lagrangian relaxation to the non-anticipativity constraints (i.e., the constraints ensuring that a single first stage solution is used under all considered scenarios). Once the problem is decomposed, then each scenario becomes a deterministic problem to be solved (i.e., a single-scenario subproblem (SSP) defined for each scenario). The resulting scenario subproblems can then be used to obtain a general lower bound, by solving the Lagrangian dual as in [110], or as a means to produce more efficient solution approaches, e.g., [4, 43] or the PHA proposed in [103].

In the context of network design problem, a PHA-based metaheuristic is proposed by Crainic et al. [29]. In this method, one first need to apply a scenario decomposition technique to separate the stochastic problem following the possible scenarios. To do so, they reformulate the model by creating copies of first-stage variables associated with each scenario and forcing all of them to be the same in all scenarios by introducing a new constraint. By doing so, the resulting problem is scenario-separable as follows:

$$\underset{x}{\text{minimize}} \quad \sum_{s\in\mathscr{S}} p^s \Big(\sum_{(i,j)\in\mathscr{A}} f_{ij} y^s_{ij} + \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} c^k_{ij} x^{ks}_{ij} \Big) \tag{2.29}$$

$$\text{subject to} \quad \sum_{j\in\mathscr{N}^+(i)} x^{ks}_{ij} - \sum_{j\in\mathscr{N}^-(i)} x^{ks}_{ji} = d^{ks}_i, \qquad \forall i \in \mathscr{N},\ \forall k \in \mathscr{K},\ \forall s \in \mathscr{S} \tag{2.30}$$

$$\sum_{k\in\mathscr{K}} x^{ks}_{ij} \le u_{ij} y^s_{ij}, \qquad \forall (i,j) \in \mathscr{A},\ \forall s \in \mathscr{S} \tag{2.31}$$

$$y^s_{ij} \in \{0,1\}, \qquad \forall (i,j) \in \mathscr{A},\ \forall s \in \mathscr{S} \tag{2.32}$$

$$y^s_{ij} = \bar{y}_{ij}, \qquad \forall (i,j) \in \mathscr{A},\ \forall s \in \mathscr{S}, \tag{2.33}$$

$$x^{ks}_{ij} \ge 0, \qquad \forall (i,j) \in \mathscr{A},\ \forall k \in \mathscr{K},\ \forall s \in \mathscr{S} \tag{2.34}$$

Constraints (2.33) are referred to as nonanticipativity constraints. The latter ensure that design decisions are not tailored for each particular scenario and aim towards a "single design". The other constraints were described earlier.

Following the decomposition scheme proposed in [103], constraints (2.33) are relaxed using an augmented Lagrangian relaxation strategy which produces a series of single-scenario subproblem. When applied to network design, each single-scenario subproblem solved at each iteration of the progressive hedging procedure represents a deterministic network design problem yielding a (potentially different) design [29, 34]. These designs are aimed to be reconciled to create a single reference point. Then, at the beginning of the next iteration, the fixed cost associated with each arc is altered through an augmented Lagrangian-type technique to hopefully induce the resulting subproblems to yield solutions closer to the current reference point [34]. In this way, differences between the designs are thus reconciled indirectly.

A main limitation of PHA is that the convergence is guaranteed only in the convex case. However, in the case of mixed-integer stochastic problem, the presence of integer variables eliminates that guarantee. Therefore, to obtain a unique design, Crainic et al. [29] proceed in two phases. If the progressive hedging procedure performed in the first stage does not result in a single design solution over all scenarios, the second phase then solves the restricted problem obtained by fixing all design arcs for which a consensus has been achieved.

In order to deal with a larger number of scenarios, Crainic et al. [34] introduced a new meta-heuristic, named mS-PHA that solves subproblems that may comprise multiple scenarios produced by the scenario-grouping strategies. This new meta-heuristic generalizes the method proposed in [29] for the stochastic MCFND problem. It is shown that by solving multi-scenario sub-problems, the meta-heuristic produces better results in terms of solution quality and computation efficiency. The results also indicate that grouping scenarios is always beneficial and doing it the smart way is even more so, as the manner in which the multi-scenario subproblems are constructed has a definite impact on the performance of the algorithm. The third article of this dissertation in Chapter 5 is developed based on this type of decomposition approach to solve instances with very a large number of scenarios.

2.2.2.6 Investigation on the solution structures

Due to the inherent difficulty and complexity of stochastic network design problems and given the fact that deterministic problems are much easier to solve, it is a common approach to consider the simpler deterministic program in which random parameters are replaced by their expected values (known as *Expected Value Problem (EVP)*), with a loss in terms of solution quality.

Although, the solution to the deterministic model behaves badly in stochastic settings [63, 123], it has been shown that there are situations where the deterministic solution shares some properties with the corresponding stochastic solution [76, 116–118].

Maggioni and Wallace in [80] point out that, while stochastic programs are appropriate, one may only have access to deterministic solutions for difficult instances. So, they seek a deeper understanding of the expected value solution in order to investigate its relationship with its stochastic counterpart. For example, they investigate if the stochastic optimal solution inherit properties from the deterministic one, or if they are totally different. They indicate that a qualitative understanding of the behavior of the deterministic solution relative to the stochastic one could be very useful because it could reveal some general properties of the underlying problem and help predict how the stochastic model will perform in two important cases. Firstly, when the stochastic model is actually solvable, but since it is solved repeatedly (daily), we would rather like to solve the deterministic one, if we could understand its quality and how to interpret it, and secondly, when it is not even solvable. They carry out a number of experiments to examine if a good (if not optimal) solution to the stochastic problem can be obtained by, somehow, updating the deterministic one.

Related studies in stochastic service network design problems e.g., [76, 116–118] show that the deterministic solution carries useful information (i.e. some structural patterns) that can be leveraged to solve the stochastic counterpart. The work in [124] was the first attempt to thoroughly analyze the quality of a deterministic solution in terms of its structure and upgradeability to a solution to the stochastic service network design problem. In particular, [25] showed that the *reduced-cost* associated with the non-basic variables in deterministic solution can be used to identify a skeleton of variables to exclude from the stochastic formulation. The second article in this dissertation is inspired by this contribution. We investigate how the *reduced cost* associated with non-basic variables in deterministic solutions can be used to identify an appropriate set of fixed variables, thus producing a restricted model.

2.3 Conclusions

This chapter has introduced a brief background on deterministic network design problems. We provided the standard notation used in mathematical formulation of fixed charge capacitated multi-commodity network design problem. We reviewed some of metaheuristics and matheuristic solution methods proposed for the deterministic network design problem. We then provided different modelling approaches in stochastic programming and reviewed the related works and methodologies in the context of stochastic network design. We noticed that efficient approaches to deal with large instances are scarce in stochastic network design problems. With these observations and given the fact that realistic problems of this nature are of large scale, this dissertation aims at filling this gap by proposing specially designed algorithms which explicitly treat stochasticity.

CHAPTER 3

ARTICLE 1: A LEARNING-BASED MATHEURISTIC FOR STOCHASTIC MULTICOMMODITY NETWORK DESIGN

Chapter notes: The article in this chapter has been submitted to the INFORMS *Journal on Computing*. The published technical report is as follows: F. Sarayloo, T.G. Crainic and W. Rei, A Learning-based Matheuristic for Stochastic Multicommodity Network Design. Technical report, Publication CIRRELT-2018-12, Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT), Montreal, Canada.

Preliminary work was presented at the following conferences:

- INFORMS and CORS joint conferences, Montreal, 2015
- Optimization Days 2016, Montreal, Canada, May, 2016
- INFORMS Annual meeting, Houston, USA, Oct 2017

Abstract

This paper proposes a solution approach for the Multicommodity Capacitated Fixed-charge Network Design problem with uncertain demand, modeled as a two-stage stochastic program. The proposed learning-based matheuristic combines heuristic-search techniques with mathematical programming. It provides a systematic approach to identify structures of good-quality solutions by gradually considering scenarios and their influences on design decisions. Extensive computational experiments illustrate the efficiency of the proposed matheuristic in obtaining high-quality solutions with limited computational efforts.

Keywords: Stochastic capacitated network design, uncertain multicommodity demand, two-stage formulation, matheuristic

3.1 Introduction

Network Design (ND) defines an important class of combinatorial optimization problems that naturally appear in a large number of applications including transportation, logistics and telecommunications. Given a network where all or a subset of arcs may be used only if selected (“opened") by paying a so-called fixed cost, the *Multicommodity Capacitated Fixed-charge Network Design* (*MCND*) formulation arises when it is required to route at minimum total cost a set of given commodities making up the demand between different pairs of origin and destination nodes of the network. The optimization aims to determine the arcs to select and the commodity flows within the resulting network, the total cost being computed as the sum of the fixed costs of the selected arcs and the cost of transporting the commodities. Surveys on network design may be found in [26, 82, 83].

Critical problem parameters such as demands, costs, and capacities are often uncertain, and many applications require the uncertainty to be explicitly considered when modeling, yielding *Stochastic MCND* (*SMCND*) formulations. We focus on demand uncertainty addressed through two-stage stochastic programs [17], where design decisions are made in the first stage before the actual demand is realized, while second-stage flow-routing decisions adjust the first-stage solution to the observed demand realization. The general goal of SND formulations is to find a single optimal design solution for the range of possible demand realizations. To address such problems, the demand uncertainty is often represented through scenario decomposition, i.e., through a set of values for the uncertain demands, the *scenarios*, together with the associated probabilities. The resulting large-scale mixed integer program (MIP), referred to as the extensive form (EF) in Birge and Louveaux [17], is difficult to solve exactly in most cases as, first, deterministic network design problems are NP-hard in all but trivial cases [82] and, second, modeling uncertainty with scenarios generally yields very large instances [29]. Heuristic solution methods are thus proposed to identify “good”-quality solutions within reasonable computing efforts.

Solving deterministic formulations is generally easier, compared to stochastic ones.

Consequently, a number of studies attempted to analyze the information provided by solutions to deterministic variants of stochastic formulations to infer information about the stochastic solutions and simplify addressing the stochastic models. Although solving deterministic formulations cannot replace addressing the stochastic one and solutions to the former may be very bad for the latter, it has been observed that stochastic solutions retain parts of deterministic solutions. Yet, no systematic procedure to identify these parts, providing the basis for efficient meta-heuristics for the MCND may be found in the literature.

We aim to contribute addressing these challenges by proposing an innovative *systematic learning mechanism* to extract information regarding the solution structure of a stochastic problem out of the solutions of particular combinations of scenarios. In the context of network design, this information takes the form of design decisions that are common to high-quality (i.e., optimal or near optimal) solutions obtained by gradually considering scenarios and their interactions.

We also propose the *Learn&Optimize* matheuristic, which jointly makes use of the learning mechanism to infer a set of promising design variables, and a state-of-the-art MIP solver to address a reduced problem. To explore the search space more efficiently and achieve improved results, we gradually enlarge the reduced problems to be solved by the MIP solver.

Learning, that is, deriving knowledge relative to the solution structure, is not a new concept in addressing stochastic network design models. Yet, how this knowledge is obtained and exploited are key success factors. The contribution of this paper, therefore, is threefold. First, we introduce and formally describe a new optimization-based learning mechanism to identify the solution structure of stochastic network design problems. Although presented in the context of this complex problem setting, the proposed mechanism can be adapted to many other combinatorial optimization problems. Second, we propose a new matheuristic framework, integrating the proposed learning mechanism, which efficiently obtains high-quality solutions, outperforming a state-of-the-art commercial MIP solver in solution quality and computational efforts, particularly as the instance dimension increases. Third, we present the results of extensive computational experiments to asses the merit and limits of the proposed methodology.

The rest of paper is organized as follows. We recall the two-stage formulation of the SMCND and briefly review relevant literature in Section 3.2. Section 3.3 introduces the main ideas of the proposed learning mechanism, while Section 3.4 details the matheuristic. We present and analyze the experimental results in Section 3.5, and provide concluding remarks in Section 3.6.

3.2 Problem description and literature review

We first recall the two-stage stochastic formulation, also known as the *a priori* optimization model [17], of the stochastic multi-commodity capacitated fixed-cost network design problem. We then present a brief review of the relevant literature.

3.2.1 Two-stage SMCND formulation

One distinguishes two sets of decisions in an a priori formulation, according to the moment in time, the *stage*, the decision is taken and the type of information available. The first stage corresponds to decisions that need to be taken here-and-now, based on estimations of future demand and prior to the realization of uncertainty. The second stage and its recourse variables correspond to the decisions made repeatedly, given the restrictions imposed by the first stage, once new information is revealed and the uncertainty is resolved. Traditionally, in the case of stochastic two-stage network design problems with uncertain demands, the first stage consists of deciding on the configuration of network, i.e., the design decisions, and the second-stage consists in optimizing the commodity flow distribution of the observed demand, on the restricted configuration imposed by the first stage.

We use the two-stage stochastic formulation of the SMCND by [29]. Let $\mathscr{G} = (\mathscr{N}, \mathscr{A})$ be a directed network with $\mathscr{N}$ representing a finite set of nodes and $\mathscr{A}$ a finite set of potential arcs. Let $\mathscr{K}$ be the set of commodities each characterized by a unique pair of origin-destination nodes. Let $\mathscr{S}$ be the set of scenarios, where p^s is the probability of scenario $s \in \mathscr{S}$. We introduce binary variable y_{ij}, which indicates if arc

(i, j) is included, or not, in the network in the first stage, where f_{ij} is the fixed cost of doing so. Once the design variables are decided upon, in the second stage, x_{ij}^{ks} represents the amount of commodity k's demand that flows on arc (i, j) in the solution for scenario s. The variable cost per unit of flow on arc (i, j) is denoted by c_{ij}. The mathematical formulation is:

$$\text{minimize} \quad \sum_{(i,j)\in\mathscr{A}} f_{ij}y_{ij} + \sum_{s\in\mathscr{S}} p^s \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} c_{ij}x_{ij}^{ks} \tag{3.1}$$

$$\text{subject to} \quad \sum_{j\in\mathscr{N}^+(i)} x_{ij}^{ks} - \sum_{j\in\mathscr{N}^-(i)} x_{ji}^{ks} = d_i^{ks}, \qquad \forall i \in \mathscr{N},\ \forall k \in \mathscr{K},\ \forall s \in \mathscr{S} \tag{3.2}$$

$$\sum_{k\in\mathscr{K}} x_{ij}^{ks} \leq u_{ij}y_{ij}, \qquad \forall (i, j) \in \mathscr{A},\ \forall s \in \mathscr{S} \tag{3.3}$$

$$y_{ij} \in \{0, 1\}, \qquad \forall (i, j) \in \mathscr{A} \tag{3.4}$$

$$x_{ij}^{ks} \geq 0, \qquad \forall (i, j) \in \mathscr{A},\ \forall k \in \mathscr{K},\ \forall s \in \mathscr{S} \tag{3.5}$$

The objective function (3.1) accounts for the total system cost, consisting of the fixed cost for the included arcs and the expectation of routing costs taken over all the realizations of demand scenarios. Constraints (3.2) represent the flow conservation equations in each scenario, requiring that demand of commodity $k \in \mathscr{K}$ be routed from its origin node to its destination. Therefore, assuming that ω^{ks} is the demand volume of commodity k in scenario s, the parameter d_i^{ks} is set to ω^{ks} if node i is the origin of commodity k, $-\omega^{ks}$ if node i is the destination of commodity k, and 0 otherwise. Linking Constraints (3.3) ensure that the same design is used in each scenario and that arc capacity u_{ij} is never violated. Constraints (3.4) and (3.5) are integrality and non-negativity constraints, respectively.

3.2.2 Literature review

The stochastic network design problem belongs to the class of stochastic mixed integer programs, its complexity stemming not only from stochasticity, but also from the lost convexity of the integrality property.

Most of the existing literature on stochastic network design problems models uncertainty through a given set of scenarios, (see Dupačová et al. [41] and King and Wallace [70] for an overview of scenario generation methods). The deterministic network design problem is NP-Hard [82] and computationally complex. Scenarios compound the difficulty by significantly increasing the dimension of the instances. Hence, some methods, e.g., the Sample Average Approximation (SAA) approach, includes sampling into the iterative solution procedure (see e.g., [9, 14, 107, 111], for applications to network design).

Once the set of scenarios is defined, standard MIP solvers can be used for small problem instances to directly solve the extensive form of the problem in which all scenarios are considered simultaneously (e.g., [9, 119]). For most network design settings and cases, this approach is not appropriate however, as formulations are either too large, or too complex, or both. Decomposition is then often called upon.

The basic idea behind decomposition methods is to divide a large-scale SMIP problem into several subproblems, along scenarios or stages, and solve the smaller-sized subproblems separately in a decomposition-coordination manner. Decomposition approaches generally fall into two groups based on how the two-stage formulation is decomposed. The first group consists of vertical-directive methods that decompose the problem according to scenarios. The Progressive Hedging (PH) method, proposed originally by Rockafellar and Wets [103], belongs to this category and is the foundation of a number of meta-heuristics for stochastic network design (e.g., [29, 34, 74]). In the second group, horizontal-directive methods decompose a stochastic network design problem stage-wise into a master problem for the first-stage variables (i.e., the design variables) and a number of subproblems for the second stage variables (i.e., a network flow problem for each scenario). The L-shaped method [120] belongs to this category and has been used for the stochastic network design (e.g., [14, 32, 102, 107, 112]).

The impact of stochasticity on the solution structure and the role solutions to some deterministic problem variants may play in identifying this structure is another significant research direction that has been explored for stochastic network design. It is well known that solutions derived from stochastic programs are, in general, structurally different from those derived from their deterministic counterparts defined by using the mean of demand distributions, [63, 76, 123]. A number of studies focusing on discerning similarities and differences between solution structures in deterministic and stochastic problem variants [e.g., 80, 116–118, 124] have also shown, however, that components of the deterministic solutions can be found in stochastic ones. Despite these investigations into using the partial information provided by deterministic solutions in addressing stochastic models, there is still no methodology to systematically explore it and identify the elements that would guide a meta-heuristic towards good-quality design solutions to large stochastic instances. We propose such a methodology in the next sections.

3.3 The learning mechanism and heuristic

Several approaches in the literature could be qualified as integrating “learning". Knowledge relative to a stochastic solution may thus be derived from a single deterministic solution only (using the expected value in most cases), as in [80] and [124]. One notices, however, that there is little learning in these approaches from the stochastic information present in the scenarios. Scenario-decomposition based methods [e.g., 29, 34, 103], on the other hand, learn from the multiple solutions provided by solving the deterministic formulations resulting from decomposing along scenarios. Most of the information is lost, however, as solutions are aggregated to compute a so-called consensus solution.

We propose a mechanism to overcome these limitations. We believe that one can derive a good deal of knowledge out of the scenarios, and that better solutions can be obtained more efficiently with a higher level of learning. We aim to provide such a higher level of learning by systematically exploiting the design information obtained by considering not only individual scenarios but also scenario combinations and interactions.

We introduce the concept of *Artificial Demand Scenario* (*ADS*) built out of particular combinations of scenarios. One then learns by iteratively building ADSs, solving the associated problems, and gradually building an image of design variables potentially belonging to good solutions to the stochastic formulation. The result of this learning mechanism is then used to guide the search to higher quality solutions.

We define the Artificial Demand Scenario in Section 3.3.1, while Sections 3.3.2 and 3.3.3 describe the algorithmic components making up the proposed learning mechanism.

3.3.1 Artificial Demand Scenario

We define an *Artificial Demand Scenario* in this paper as a combination of the demands of two scenarios. Let $\boldsymbol{d(s_\alpha)}$ and $\boldsymbol{d(s_\beta)}$ be the demand vectors of size $|\mathscr{K}|$ of scenarios $s_\alpha, s_\beta \in \mathscr{S}$:

$$\boldsymbol{d(s_\alpha)} = \begin{pmatrix} d_1(s_\alpha) \\ d_2(s_\alpha) \\ d_3(s_\alpha) \\ \vdots \\ d_{|\mathscr{K}|}(s_\alpha) \end{pmatrix} \text{ and } \boldsymbol{d(s_\beta)} = \begin{pmatrix} d_1(s_\beta) \\ d_2(s_\beta) \\ d_3(s_\beta) \\ \vdots \\ d_{|\mathscr{K}|}(s_\beta) \end{pmatrix}.$$

An *Artificial Demand Scenario* $\boldsymbol{\delta(s_\alpha, s_\beta)} \in \Delta(s_\alpha, s_\beta),\ \forall s_\alpha, s_\beta \in \mathscr{S}$ is then defined as a demand vector of the same size, where the element $k = 1, 2, 3, \ldots, |\mathscr{K}|$ contains the demand value associated with commodity k in scenario s_α or scenario s_β, that is,

$$\boldsymbol{\delta(s_\alpha, s_\beta)} = \begin{pmatrix} \delta_1(s_\alpha, s_\beta) \\ \delta_2(s_\alpha, s_\beta) \\ \delta_3(s_\alpha, s_\beta) \\ \vdots \\ \delta_{|\mathscr{K}|}(s_\alpha, s_\beta) \end{pmatrix}, \text{ such that } \delta_k(s_\alpha, s_\beta) = d_k(s_\alpha) \vee d_k(s_\beta), \forall k \in \mathscr{K}.$$

To illustrate, consider two demand vectors $\boldsymbol{d(s_1)}$ and $\boldsymbol{d(s_2)}$ containing six commodities each. Three possible artificial demand scenarios are:

$$\boldsymbol{d(s_1)}=\begin{pmatrix}10\\20\\60\\30\\90\\70\end{pmatrix},\quad \boldsymbol{\delta_1(s_1,s_2)}=\begin{pmatrix}45\\20\\60\\30\\90\\70\end{pmatrix},\quad \boldsymbol{\delta_2(s_1,s_2)}=\begin{pmatrix}10\\25\\75\\30\\90\\70\end{pmatrix},\quad \boldsymbol{\delta_3(s_1,s_2)}=\begin{pmatrix}10\\20\\75\\85\\95\\70\end{pmatrix},\quad ...,\quad \begin{pmatrix}45\\25\\75\\85\\95\\15\end{pmatrix}=\boldsymbol{d(s_2)}$$

One generates an ADS by selecting a number of demand elements in one of the two scenarios and the rest in the other one. The generation process is thus determined by how many elements are selected and how this selection is performed. In all generality, there are several possible ways to perform the selection and we define $\mathscr{R}$ as the set of *selection rules*. An ADS $\boldsymbol{\delta^{mr}(s_\alpha,s_\beta)} \in \Delta(s_\alpha,s_\beta)$ is then generated by applying the selection rule $r \in \mathscr{R}$ to select m elements in $d(s_\beta)$ and $|\mathscr{K}|-m$ elements in $d(s_\alpha)$. Let $\Delta^{mr}(s_\alpha,s_\beta)$ be the set of ADSs of type *m-combination*, $m = 1,2,3,\ldots,\mathscr{K}-1$, generated by the selection rule r. Define the *operator* $\oplus$ such that

$$\boldsymbol{\delta^{'''}(s_\alpha,s_\beta)} = \boldsymbol{\delta^{'}(s_\alpha,s_\beta)} \oplus \boldsymbol{\delta^{''}(s_\alpha,s_\beta)},\ \boldsymbol{\delta^{'}(s_\alpha,s_\beta)},\boldsymbol{\delta^{''}(s_\alpha,s_\beta)} \in \Delta^{mr}(s_\alpha,s_\beta)$$

yields

$$\delta_k^{'''}(s_\alpha,s_\beta) = \begin{cases} d_k(s_\beta) & \text{if } (\delta_k^{'}(s_\alpha,s_\beta) = d_k(s_\beta)) \vee (\delta_k^{''}(s_\alpha,s_\beta) = d_k(s_\beta)), \\ d_k(s_\alpha) & \text{otherwise,} \end{cases} \tag{3.6}$$

for all $k \in \mathscr{K}$.

A set $\Delta^{mr}(s_\alpha,s_\beta) = \{\boldsymbol{\delta}_\theta^{mr}(s_\alpha,s_\beta)\}_{\{\theta=1,2,\ldots,N_m\}}$, of cardinality N_m, will then ensure that the demand value of each commodity in $d(s_\beta)$ appeares in at least one of its ADSs, i.e.,

$$\boldsymbol{d(s_\beta)} = \boldsymbol{\delta}_1^{mr}(s_\alpha,s_\beta) \oplus \boldsymbol{\delta}_2^{mr}(s_\alpha,s_\beta) \oplus \ldots \oplus \boldsymbol{\delta}_{N_m}^{mr}(s_\alpha,s_\beta). \tag{3.7}$$

Note that the minimum cardinality of set $\Delta^{mr}(s_\alpha,s_\beta)$ satisfying relation (3.7) is $N_m =$

$\lceil\frac{|\mathscr{K}|}{m}\rceil$. To illustrate, in the following case of two scenarios and three ADSs, the set $\{\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \boldsymbol{\delta}_3\}$ satisfies relation (3.7), while the set $\{\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \boldsymbol{\delta}_4\}$ does not.

$$\boldsymbol{d(s_1)} = \begin{pmatrix}10\\20\\60\\30\\90\\70\end{pmatrix}, \quad \boldsymbol{\delta}_1 = \begin{pmatrix}45\\25\\60\\30\\90\\70\end{pmatrix}, \quad \boldsymbol{\delta}_2 = \begin{pmatrix}10\\20\\75\\85\\90\\70\end{pmatrix}, \quad \boldsymbol{\delta}_3 = \begin{pmatrix}10\\20\\60\\30\\95\\15\end{pmatrix}, \quad \boldsymbol{\delta}_4 = \begin{pmatrix}45\\20\\60\\30\\90\\15\end{pmatrix}, \quad \begin{pmatrix}45\\25\\75\\85\\95\\15\end{pmatrix} = \boldsymbol{d(s_2)}$$

Several methods can be used to construct the set $\Delta^{mr}(s_\alpha, s_\beta)$ but, to avoid introducing “noise" in evaluating the learning mechanisms, we use a simple procedure in this paper. Algorithm 1 builds a set $\Delta^{mr}(s_\alpha, s_\beta)$ of minimum cardinality $N_m = \lceil\frac{|\mathscr{K}|}{m}\rceil$, through the random selection of the demand values (i.e., $\mathscr{R} = \{r = \text{random selection}\}$) of two given scenarios. The procedure first builds $N_m - 1$ ADSs (lines 3 - 5) by iteratively selecting m commoditiy values to copy from $d(s_\beta)$. The last ADS (line 6) is built from the remaining commodity values, if any.

Algorithm 1 $Construct(s_\alpha, s_\beta, m, r = \text{random selection})$

```
1: Initialization: $N_m \leftarrow \lceil\frac{|\mathscr{K}|}{m}\rceil$, $\bar{\mathscr{K}} \leftarrow \mathscr{K}$, $\theta \leftarrow 1$;
2: repeat
3:     Randomly choose m commodities $\mathscr{K}^m \subseteq \bar{\mathscr{K}}$, and create ADS $\boldsymbol{\delta}^{mr}_\theta(s_\alpha, s_\beta)$ with
   the corresponding demand values from scenario $s_\beta$; $\theta \leftarrow \theta + 1$;
4:     Update $\bar{\mathscr{K}}$: $\bar{\mathscr{K}} := \bar{\mathscr{K}} \setminus \mathscr{K}^m$;
5: until  $\theta = N_m$
6: Generate $\boldsymbol{\delta}^{mr}_\theta(s_\alpha, s_\beta)$ by choosing the remaining commodities in $\bar{\mathscr{K}}$ out of $\boldsymbol{d(s_\beta)}$;
7: return  $\Delta^{mr}(s_\alpha, s_\beta)$
```

3.3.2 Partial learning - the scenario-pair case

Given a pair of scenarios $s_\alpha, s_\beta \in \mathscr{S}$ and a given design $\bar{y}$, the *partial-learning* mechanism proceeds by exploring the solution characteristics associated to each ADS $\boldsymbol{\delta}(s_\alpha, s_\beta) \in \Delta(s_\alpha, s_\beta)$ to extract information regarding promising design variables. The exploration is performed by solving an *Artificial-Recourse Problem*, $ARP(\boldsymbol{\delta},\bar{y})$, for each artificial demand scenario $\boldsymbol{\delta} \in \Delta(s_\alpha, s_\beta)$.

To define the $ARP(\boldsymbol{\delta},\bar{y})$, we separate the set of arcs $\mathscr{A}$ according to the given design $\bar{y}$. Then, $\mathscr{A} = \mathscr{A}^0 \cup \mathscr{A}^1$, where $\mathscr{A}^0 = \{(i,j)|(i,j) \in \mathscr{A}, \bar{y}_{ij} = 0\}$ and $\mathscr{A}^1 = \{(i,j)|(i,j) \in \mathscr{A}, \bar{y}_{ij} = 1\}$ are the sets of closed and open arcs in $\bar{y}$, respectively. We then define a modified arc variable cost $\bar{c}_{ij}$ by linearizing the fixed cost of the closed arcs

$$\bar{c}_{ij} = \begin{cases} c_{ij} + \frac{f_{i,j}}{u_{ij}}, \ \forall (i,j) \in \mathscr{A}^0, \\ c_{ij}, \ \forall (i,j) \in \mathscr{A}^1 \end{cases} \tag{3.8}$$

and solve the $ARP(\boldsymbol{\delta},\bar{y})$ multi-commodity network flow problem

$$ARP(\boldsymbol{\delta},\bar{y}) : \text{ minimize } \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} \bar{c}_{ij} x_{ij}^k \tag{3.9}$$

$$\text{subject to} \quad \sum_{j\in\mathscr{N}^+(i)} x_{ij}^k - \sum_{j\in\mathscr{N}^-(i)} x_{ji}^k = \delta_k^i, \quad \forall i \in \mathscr{N}, \forall k \in \mathscr{K} \tag{3.10}$$

$$\sum_{k\in\mathscr{K}} x_{ij}^k \le u_{ij}, \quad \forall (i,j) \in \mathscr{A} \tag{3.11}$$

$$x_{ij}^k \ge 0, \quad \forall (i,j) \in \mathscr{A}, \forall k \in \mathscr{K} \tag{3.12}$$

where

$$\delta_k^i = \begin{cases} \delta_k & \text{if } i = o(k) \\ -\delta_k & \text{if } i = d(k) \\ 0 & \textit{otherwise.} \end{cases} \tag{3.13}$$

Solving $ARP(\boldsymbol{\delta},\bar{y})$ yields the solution $x_{ij}(\boldsymbol{\delta}) = \sum_{k\in\mathscr{K}} x_{ij}^k$, $\forall (i,j) \in \mathscr{A}$, with $\mathscr{A}_{\boldsymbol{\delta}} = \{(i,j) \mid x_{ij}(\boldsymbol{\delta}) > 0\}$. We define a corresponding design solution as $y_{ij}^{\boldsymbol{\delta}} = 1$, when $x_{ij}(\boldsymbol{\delta}) > 0$, and 0, otherwise. It is noteworthy that some of the arcs in $\mathscr{A}^0$, closed in $\bar{y}$, may be open in $y_{ij}^{\boldsymbol{\delta}}$ to satisfy the demand vector $\boldsymbol{\delta}$. This modification to the design vector following the modification of the recourse problem inspired the procedure and the *ARP* name. These modifications capture the interactions occurring in the integration of two scenarios within $\boldsymbol{\delta}$, yielding partial information regarding the design arcs required to address the uncertainty captured by the two scenarios. Repeating this procedure for different artificial demand scenarios builds the knowledge we seek.

The partial-learning approach is described in Algorithm 2. The set of Artificial Demand Scenarios, $\Delta^{mr}(s_\alpha, s_\beta)$, was constructed previously using Algorithm 1. We define the *frequency memory*, F'_{ij}, representing how often arc (i,j) has been used in the solutions of the different *ARP*$(\boldsymbol{\delta}, \bar{y})$. We also define $\mathscr{A}^\Delta$, the set of design arcs used in at least one *ARP*, and $\mathscr{A}^{\alpha\_\beta}$, the set of promising design variables to be identified by the procedure.

Algorithm 2 PartialLearning $(\bar{y}, \Delta^{mr}(s_\alpha, s_\beta))$

```
 1: Initialization: $F'_{ij} \leftarrow 0, \forall (i,j) \in A$, $\mathscr{A}^{\alpha\_\beta} \leftarrow \emptyset$; $\Delta^{mr}(s_\alpha, s_\beta); \leftarrow \emptyset$; $\mathscr{A}^\Delta \leftarrow \emptyset$
 2: repeat
 3:     Randomly choose an artificial demand scenario $\delta \in \Delta^{mr}(s_\alpha, s_\beta)$;
 4:     Solve ARP$(\boldsymbol{\delta}, \bar{y})$ yielding $x_{ij}(\boldsymbol{\delta}), \forall (i,j) \in \mathscr{A}$;
 5:     Identify $\mathscr{A}_{\boldsymbol{\delta}}$ and compute $y^{\boldsymbol{\delta}}_{ij}$, $\forall (i,j) \in \mathscr{A}_{\boldsymbol{\delta}}$;
 6:     Update $\mathscr{A}^\Delta := \mathscr{A}^\Delta \bigcup \mathscr{A}_{\boldsymbol{\delta}}$ and $F'_{ij} := F'_{ij} + 1$, for all $(i,j) \in \mathscr{A}_{\boldsymbol{\delta}}$;
 7:     Remove $\boldsymbol{\delta}$ from $\Delta^{mr}(s_\alpha, s_\beta)$;
 8: until $\Delta^{mr}(s_\alpha, s_\beta) = \emptyset$;
 9: Normalize frequencies $F'_{ij} := F'_{ij}/max\{F'_{ij} | (i,j) \in \mathscr{A}^\Delta\}$, $\forall (i,j) \in \mathscr{A}^\Delta$;
10: for all $(i,j) \in \mathscr{A}^\Delta$ do
11:     if $F'_{ij} \geq \tau'$ then $\mathscr{A}^{\alpha\_\beta} \leftarrow \mathscr{A}^{\alpha\_\beta} \cup \{(i,j)\}$;
12:     end if
13: end for
14: Return the set of promising design variables $\mathscr{A}^{\alpha\_\beta}$.
```

The main loop (lines 3 to 7) iterates over the artificial demand scenarios in $\Delta^{mr}(s_\alpha, s_\beta)$, each being discarded, line 7, after it has been examined. The procedure stops when the set of artificial demand scenarios becomes empty. The *ARP*$(\boldsymbol{\delta}, \bar{y})$ is solved for each $\boldsymbol{\delta} \in \Delta^{mr}(s_\alpha, s_\beta))$, to distribute the demand of δ (line 4). The corresponding design vector is created (line 5), while the set of used design arcs and the frequency memory are updated on line 6. Once artificial demand scenarios in $\boldsymbol{\delta} \in \Delta^{mr}(s_\alpha, s_\beta)$ are treated, the procedure returns the most frequently used arcs for the scenario pair (s_α, s_β), given a threshold τ' (lines 10-13).

3.3.3 The learning procedure

The *learning* mechanism repeatedly applies the partial-learning procedure to various pairs of scenarios to extract global information on promising design variables. The goal is to then use the collected information obtained by the learning mechanism to identify appropriate parts of the solution space where an exact solver may intensify the search (Section 3.4).

The Learning Procedure, described in Algorithm 3, consists of two phases. The partial-learning procedure (Algorithm 2) is first used for each pair of scenarios in a given set Π (all pairs are considered in the mechanism of this paper), identifying the corresponding promising design variables and frequency vectors. This information is used to gradually build the global set of promising design variables, $\mathscr{A}^{\Pi}$, and frequency vector F. The second phase identifies the set of most promising design arcs, $\mathscr{A}^{\star}$, as the most frequently selected ones (given a threshold τ).

Algorithm 3 Learning$(\bar{y}, \Pi, m, r)$

```
 1: $Initialization$: $F_{ij} \leftarrow 0\ ,\forall (i,j) \in \mathscr{A}$, $\mathscr{A}^{\Pi} \leftarrow \emptyset$, $\mathscr{A}^{\star} \leftarrow \emptyset$;
 2: for all pairs $(s_\alpha, s_\beta) \in \Pi$ do
 3:     $\Delta^{mr}(s_\alpha, s_\beta) \leftarrow Construct(s_\alpha, s_\beta, m, r)$;
 4:     $\mathscr{A}^{\alpha\_\beta} \leftarrow PartialLearning(\bar{y}, \Delta^{mr}(s_\alpha, s_\beta))$;
 5:     $Update$ the frequency memory $F_{ij} := F_{ij} + 1, \forall (i,j) \in \mathscr{A}^{\alpha\_\beta}$;
 6:     $Update\ \mathscr{A}^{\Pi} := \mathscr{A}^{\Pi} \bigcup \mathscr{A}^{\alpha\_\beta}$;
 7: end for
 8: $Normalize$ frequencies $F_{ij} := F_{ij}/max\{F_{ij} | (i,j) \in \mathscr{A}^{\Pi}\}$, $\forall (i,j) \in \mathscr{A}^{\Pi}$;
 9: for arc $(i,j) \in \mathscr{A}^{\Pi}$ do
10:     if $F_{ij} \geq \tau$ then $\mathscr{A}^{\star} := \mathscr{A}^{\star} \cup \{(i,j)\}$
11:     end if
12: end for
13: $Return$ the set of promising design variables $\mathscr{A}^{\star}$.
```

3.4 The Matheuristic

The *Learn&Optimize* matheuristic we propose iteratively executes a *learning step*, to identify a promising set of design variables, and a *partial-optimization step*, where a number of promising variables are fixed and the reduced-size formulation is solved exactly.

Algorithm 4 details the matheuristic, where $\bar{y}$ is an initial design solution, Π, the set of scenario pairs, $\mathscr{M} = \{m_1, m_2, \ldots, m_{MMAX}\}$, the set of *m-combinations*, $\mathscr{R} = \{r_1, r_2, \ldots, r_{RMAX}\}$, the set of *selection rules*, p, the initial percentage of promising variables to fix, $\Delta(p)$, the reduction of the current value of p in the diversification step, and q, the number of consecutive iterations with no improvement activating a diversification step. A computational time limit is the stopping criterion in this version of the algorithm.

Algorithm 4 Learn&Optimize ($\bar{y}$, Π, $\mathscr{M}$, $\mathscr{R}$, p, $\Delta(p)$, q)

```
1:  Initialization: $\nu \leftarrow 1$, $y^{best} \leftarrow \bar{y}$, $l \leftarrow 1$, $i \leftarrow 1$;
2:  repeat
3:      Learn&Memorize: $\mathscr{A}^{\star}_{\nu} \leftarrow Learning(y^{best}, \Pi, m_i \in \mathscr{M}, r_l \in \mathscr{R})$;
4:      Fix&Optimize: $\bar{y}_{\nu} \leftarrow SubMIPSolve(\mathscr{A}^{\star}_{\nu}, p)$;
5:      if Global update: $\phi(\bar{y}_{\nu}) \leq \phi(y^{best})$ then  $y^{best} \leftarrow \bar{y}_{\nu}$;
6:      end if
7:      if Diversification: $y^{best}$ has not been improved in the last $q$ iterations then
8:          $i \leftarrow (i+1) mod\ MMAX$;
9:          $l \leftarrow (l+1) mod\ RMAX$;
10:         $p \leftarrow p - \Delta(p)$;
11:     end if
12:     $\nu \leftarrow \nu + 1$ ;
13: until Stopping criterion not satisfied
14: Return the best solution $y^{best}$
```

At each iteration ν, the matheuristic first performs the procedure of Section 3.3.3 with current $m_i \in \mathscr{M}$ and $r_l \in \mathscr{R}$ parameters (line 3), to learn and build statistics on solution characteristics, yielding the set of promising design variables, $\mathscr{A}^{\star}_{\nu}$. The subproblem obtained by fixing p percent of variables in $\mathscr{A}^{\star}_{\nu}$ to the value 1 is then solved, yielding the solution $\bar{y}_{\nu}$ (line 4). The global solution y^{best} is updated when an improvement occurs, i.e., $\phi(\bar{y}_{\nu}) \leq \phi(y^{best})$ where $\phi(.)$ is the objective function (lines 5-6). A diversification step is performed when no improvement is achieved after q consecutive iterations (lines 7-10), by changing the m-combination and the selection rule in the learning step, as well as by decreasing the percentage of variables to fix in the partial optimization step. The last operations expands the solution space of the partial problem to hopefully identify improved solutions.

3.5 Computational results

This section presents the results of the computational experiments performed to assess the performance of the proposed matheuristic. We first describe the test instances and experimental settings (Section 3.5.1). Section 3.5.2 is dedicated to a sensitivity analysis of the algorithm to different initial solutions and ADS types. The performance of the matheuristic in addressing larger instances is analyzed in Section 3.5.3. Comparative results to CPLEX and a *Local Branching* (*LBr*) matheuristic are presented throughout the section.

3.5.1 Data and experimental settings

We used 12 problem classes, R4-R15, from the instances of stochastic CMND problem introduced in Crainic et al. [29]. The attributes of each class, in terms of nodes, arcs, and commodity set cardinalities, are given in Table 3.5.1. Each class contains five networks with different "ratio" index values 1, 3, 5, 7, and 9, indicating continuously increasing ratios of fixed-to-variable-cost and total-demand-to-total-network-capacity.

For each of these networks, we use the demand scenarios generated in Crainic et al. [29]. The triangular demand probability distributions are assumed, with the mode c of each commodity set to the demand value in the original instance, while the minimum and maximum values of the distributions were set to $a = 0$ and $b = 1.35c$, respectively. 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. Finally, scenario trees were generated using the procedure proposed in [64], and instances with 16, 32, and 64 scenarios were created.

Notice that, while the small R instances, groups R4-R10, were used in previous studies of stochastic network design [e.g. 29, 34], the large ones, groups R11-R15, were not. We thus used a subset of instances from classes R4-R10 to perform the sensitivity analysis of the algorithm to parameter values, while a more detailed analysis was conducted

Table 3.5.1: Characteristics of instances

Problem	$\|\mathscr{N}\|$	$\|\mathscr{A}\|$	$\|\mathscr{K}\|$	Problem	$\|\mathscr{N}\|$	$\|\mathscr{A}\|$	$\|\mathscr{K}\|$
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

on classes R11-R15.

Algorithms were implemented in C++. The numerical experiments were performed on a Cisco UCS C200 cluster of 26 computers; each has two 3.07 GHz Intel(R) processors and 96 Gigabytes of RAM, operating under Linux.

We used CPLEX version 12.6.1.0 to solve the deterministic MIP problems, complete and partial. The Local Branching (*LBr*) matheuristic [45] is based on defining a neighborhood of the current incumbent solution by allowing only a few binary variables to flip their value, through the addition of a local branching constraint. We implemented *LBr* by turning on the parameter "LBHeur" in CPLEX, which invokes a local branching heuristic when it finds a new incumbent. The Learn&Optimize matheuristic was run with the random selection rule in creating artificial demand scenarios. Finally, preliminary experiments helped set the τ, q, and $\Delta(p)$ parameters to 0.8, 4, and 0.05, respectively.

3.5.2 Sensitivity analysis

Our main goal with this phase of the experiments is to evaluate the impact of using different initial solutions and different types of ADSs on the behavior of the algorithm. We performed this analysis on 22 representative small R instances, R4-R10, which were solved to optimality by CPLEX within 1000 seconds of CPU time. The instances and the computational results (CPU time) for four variants of the matheuristic, corresponding to four combinations of characteristics, are displayed in Table 3.5.2. The CPU time required by the final Learn&Optimize metaheuristic (*L&Opt*) and CPLEX and the *LBr* matheuristic are also displayed. Notice that the solution values are not displayed in Table 3.5.2 because all variants of *L&Opt* and *LBr* found the optimal solution reported by CPLEX. Hence, we report the time needed to hit these known optimal solutions only.

Two initial solutions were considered by solving the deterministic network design models with the expected demand, the “Exp” case, and the maximum demand, the “Max” case. Two types of ADS were also considered varying the value of the m parameter, $m = 2$ and $m = \lfloor \frac{\mathscr{K}}{2} \rfloor$, used in Algorithm 4 with the random selection rule. Notice that, $m = 2$ yields a larger cardinality for the generated set of ADSs, $N_m = \lceil \frac{|\mathscr{K}|}{2} \rceil$), compared to $m = \lfloor \frac{|\mathscr{K}|}{2} \rfloor$ with a cardinality of 2. The four variants of the *L&Opt* matheuristic then were:

- Variant1: initial solution “Exp”, $m = 2$;
- Variant2: initial solution “Exp”, $m = \lfloor \frac{|\mathscr{K}|}{2} \rfloor$;
- Variant3: initial solution “Max”, $m = 2$;
- Variant4: initial solution “Max”, $m = \lfloor \frac{|\mathscr{K}|}{2} \rfloor$.

The results in Columns 2 to 5 of Table 3.5.2 indicate that the proposed learning mechanism and matheuristic are generally not sensitive to the initial solution and the type of ADS, with respect to computational efficiency and solution quality (the optimal solution on the instances used). A more detailed analysis indicates, however, that using “Exp” in computing the initial solution provides slightly better efficiency compared to using the maximum demand. A similar remark may be made with respect to the parameter m, the variants with $m = \lfloor \frac{|\mathscr{K}|}{2} \rfloor$ examining fewer ADS in the course of the algorithm. Following these observations and the preliminary results, the final *L&opt* algorithm initiates the search from the expected-value solution, and sets the parameter m to iterate over $m = \{\lfloor \frac{|\mathscr{K}|}{2} \rfloor, \lfloor \frac{|\mathscr{K}|}{3} \rfloor, \lfloor \frac{|\mathscr{K}|}{4} \rfloor\}$).

The results of of the *L&Opt*, with these settings are displayed in Column 6, while Columns 7 and 8 display those of CPLEX and the *LBr* matheuristic. The last two columns of Table 3.5.2 display the comparative performance of the *L&Opt* matheuristic with respect to CPLEX and *LBr*, by providing the time ratios calculated as $\frac{t_{CPLEX}}{t_{L\&Opt}}$ and $\frac{t_{LBr}}{t_{L\&Opt}}$. The summation of computation times over all instances for each variant is shown on the last line as “Total" (the average is displayed for the last two columns).

Table 3.5.2: Comparison of CPU times

Prob	Variant1	Variant2	Variant3	Variant4	L&Opt	CPLEX	LBr	Time Ratio	
	Time	Time	Time	Time	Time	Time	Time	CPLEX	LBr
R4.1-16	0.51	0.51	0.51	0.51	0.51	0.79	0.38	1.49	0.74
R4.1-32	1.9	1.8	1.95	1.97	1.4	1.4	0.7	1	0.51
R4.1-64	6.70	5.6	6.46	6.71	5.01	5.03	4.3	1.01	0.85
R4.3-16	19.96	14.28	20.33	18.33	7.2	5.20	2.37	0.72	0.3
R4.3-32	50.21	53.92	60	58	37.02	30.23	7.53	0.81	0.19
R4.3-64	184.11	243.94	201	195	181.02	183.3	29.57	1.01	0.15
R4.5-16	53.67	23.86	40	39.54	20.01	19.12	13.86	0.95	0.65
R4.5-32	150.34	67.25	210	202	54.84	196	42.84	3.57	0.77
R4.5-64	398.74	339.7	580	579.1	339.1	685.5	176.73	2.02	0.51
R5.7-16	2.49	2.21	2.32	2.62	2.6	5.53	6.35	2.12	2.6
R5.7-32	8.56	9.20	10.10	10.50	8.56	15.96	18.2	1.86	2.12
R5.7-64	29.40	30	31.3	31.2	29.96	50.73	75.56	1.69	2.5
R7.1-16	0.65	0.63	0.69	0.63	0.61	0.32	0.47	0.52	0.7
R7.1-32	2.43	2.44	2.44	2.23	1.83	0.83	1.18	0.45	0.64
R7.1-64	9.27	8.56	8.9	10.94	4.61	3.64	4.77	0.78	1.03
R7.3-16	23.21	22.68	24.1	22.8	20.21	5.33	8.05	0.26	0.3
R7.3-32	61	59.2	67	65	46.21	19.67	30.5	0.42	0.65
R7.3-64	148	134	160	155	100.89	116.19	197.66	1.15	1.97
R7.5-16	18.88	18.17	18.9	18.86	16.8	15.82	15.91	0.94	0.94
R7.5-32	5.56	5.19	5.51	5.33	5.01	35.02	53.94	6.99	10.6
R7.5-64	17.92	17.85	19.78	17.59	17.01	153.25	315.41	9.01	18.52
R8.5-16	308.20	300.45	350.21	320.51	290.07	402.87	250.61	1.38	0.86
Total	1193.51	1060.99	1471.29	1443.86	900.41	1549.64	1256.89	-	
Avg								1.82	2.17

We observe that the proposed *L&Opt* performs very efficiently by changing the value of parameter *m* throughout the algorithm, rather than fixing it to a single value. The proposed matheuristic is also faster than CPLEX and *LBr*, except for a few very-easy to solve instances requiring less than 30 seconds of running time. *L&Opt* outruns the two other methods by an average factor of 1.82 and 2.17, respectively.

3.5.3 Experiments on larger instances

The second set of experiments aimed to assess the behavior and performance of the learning mechanism and the *L&Opt* metaheuristic on larger instances, not yet tackled in the literature. We 1) compare the performances of *L&Opt* and CPLEX on those instances; 2) analyze in more depth the behavior of the proposed algorithm, in particular with respect to the impact of demand correlations and instance characteristics (fixed cost and capacity ratios); and 3) illustrate the efficiency of *L&Opt* in improving the solution through time.

A total of 225 (5*5*3*3) instances, derived from the sets R11 to R15 (Table 3.5.1), were generated for these experiments. As an illustration of the challenge to address such instances, consider that the deterministic equivalent problems of R15 instances with 64 scenarios consists of 5 375 800 variables and 282 880 constraints. Thus, 500 minutes of CPU time was allocated to the three methods for these experiments.

Table 3.5.3 gives a general overview of the the computational performance of CPLEX for the instances considered. Column "# Opt." indicates the number of instances solved to optimality (out of the 45 instances in each class) within 500 minutes of CPU time, while Column "# Failures" reports the number of instances for which CPLEX could not solve the root LP relaxation problem within the same time. The difficulty increases with the instance size, class R15 representing the most difficult ones (failure to solve the LP relaxation in 20 out of 45 instances). These results underline the difficulty of even very good commercial MIP solvers to address a large portion of these instances (i.e., 180 out of 225).

Table 3.5.3: CPLEX performance on large R instances

Prob.	# Inst.	# Opt.	# Failures
R11	45	18	0
R12	45	18	5
R13	45	9	0
R14	45	0	6
R15	45	0	20
Total	225	45	31

To analyze the performance of *L&Opt*, we first focus on the 45 instances solved to optimality by CPLEX. Table 3.5.4 displays the comparison results, which show that *L&Opt* was able to find the optimal solutions of all these 45 instances. It indicates that *L&Opt* performs as well as CPLEX in terms of the number of instances solved to optimality. The computational time to hit the optimal solution are, on average, 56.1 and 126.3 minutes for CPLEX and *L&Opt*, respectively. Figure 3.1 illustrates, however, that *L&Opt* reaches high quality solutions fast, e.g., solutions with average optimality gaps ($\frac{L\&Opt-CPLEX}{CPLEX} * 100$) of 1.2%, 0.8%, and 0.74% after 23, 30, and 63.1 minutes, respectively.

Turning to the 180 instances that CPLEX was not able to solve to optimality, we

Table 3.5.4: Performance comparison on easy instances

# Inst.	Opt. CPLEX	Sol. Time	Opt. L&opt	Sol. Time
45	45	56.1	45	126.3

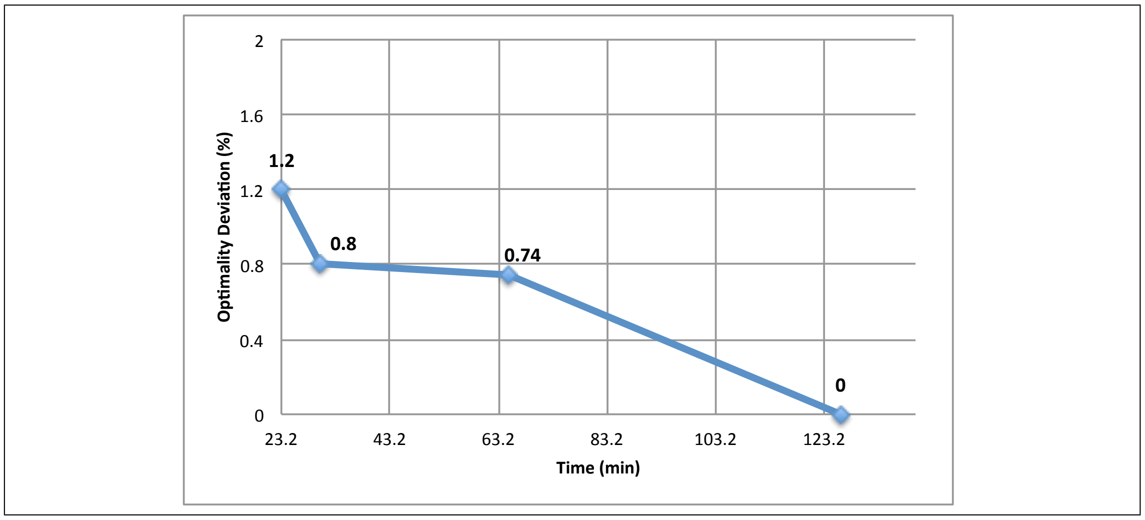

Figure 3.1: Optimality gap of L&Opt through time

compared the best results of *L&Opt*, CPLEX and the *LBr* matheuristic after 500 minutes of CPU time. Table 3.5.5 displays the results. Each line corresponds to a class of instances, Column "# of Ins" indicating the number of instances in each. The two "Wins" column report the percentage of instances for which *L&Opt* provided better solutions compared to CPLEX and *LBr*, respectively. Columns 4 and 6 report the average gaps, in %, between *L&Opt* and the two other methods computed as $\frac{L\&Opt-CPLEX}{L\&Opt} * 100$ and $\frac{L\&Opt-LBr}{L\&Opt} * 100$, respectively. Negative values indicate the superiority of *L&Opt* over CPLEX and *LBr* in terms of solution quality.

Table 3.5.5: Comparative performance of L&Opt and CPLEX on difficult instances

Prob.	# of Inst.	Wins(%)	L&Opt/CPLEX Gap(%)	Wins(%)	L&Opt/LBr Gap(%)	Time to beat(min) CPLEX	LBr
R11	27	88.8	-9.23 %	88.8	-9.60 %	64.94	61.32
R12	27	77.7	-6.13 %	81.4	-6.77 %	43.51	40.02
R13	36	83.3	-18.58 %	83.3	-23.36 %	60.79	55.2
R14	45	86.6	-15.16 %	86.6	-15.35 %	62.25	60.21
R15	45	100	-23.30 %	100	-23.79 %	70.05	68.32

We also report, in the last two columns, the average time for *L&Opt* to find a better solution than CPLEX and *LBr* within 500 min. We observed that, for more than 80% of instances, *L&Opt* found a better solution compared to those CPLEX and *LBr* found within 500 minutes, in 70 and 68.4 minutes, respectively, .

Overall, we noted that, for 119 out of 180 instances, *L&Opt* provided relative improvements of 17.20% and 18.4% in average over the solution found by CPLEX and *LBr*, respectively. The three methods provided the same solution quality for 30 out of 180 instances. Worth noticing, for the 31 instances for which CPLEX and *LBr* could not provide any solution after 500 minutes, *L&Opt* provided a feasible solution within 63 minutes (on average) and continued to improve it through time.

3.5.3.1 Algorithm behavior

We continue to analyze the behavior of the learning-based matheuristic algorithm, given various instance characteristics.

We studied the performance of the *L&Opt* matheuristic and that of CPLEX according to the ratio index values. Each line of the Table 3.5.6 presents the aggregated results of the 15 instances for a given combination of ratio level and number of scenarios (results computed only when a CPLEX lower bound or feasible solution, as required, was available). Column "Failure" represents the number of instances for which CPLEX could not solve the LP relaxation and, thus, no lower bound, within 500 minutes of CPU time. Column "OptGap" represents the average optimality gap of CPLEX after 500 minutes CPU time. The last column displays the average gap (in %) between the best solutions found by *L&Opt* and CPLEX, computed as $\frac{L\&Opt-CPLEX}{L\&Opt} * 100$.

The results underline the impact of these instance characteristics on the behavior of the algorithms. In particular, they indicate that the instances that are the most difficult to address are not those with the highest fixed cost and capacity ratio, an observation often made for deterministic CMND problems e.g., [50], but rather those with intermediate ratios (e.g., 3 and 5). This observation is in line with the results of Crainic et al. [29]. This behavior may be caused by the higher number of similar optimal designs that exist for such intermediary deterministic CMND instances, compared to instances with low or high ratios. Then, when demand is stochastic, such instances would display broader

Table 3.5.6: Algorithm performance by instance type

Instance		#	CPLEX		L&Opt/CPLEX
Type	|S|	Inst.	Failure	OptGap	Gap
1	16	15	0	7.6%	-3.75 %
1	32	15	0	12.21%	-5.12 %
1	64	15	2	19.63%	-4.49 %
3	16	15	0	49.2%	-26.38 %
3	32	15	3	58.7%	-30.87 %
3	64	15	9	70.3%	-33.89 %
5	16	15	0	29.1%	-13.42 %
5	32	15	3	38.6%	-22.63 %
5	64	15	7	44.9%	-23.45 %
7	16	15	0	3.63%	-0.42 %
7	32	15	0	8.53%	-2.79 %
7	64	15	3	7.66%	-4.95 %
9	16	15	0	10.77%	-5.40 %
9	32	15	1	22.44%	-11.25 %
9	64	15	3	24.39%	-9.25 %

differences among scenarios requiring more effort to identify an overall satisfactory, hopefully optimal, solution.

We also analyzed the impact of the scenario correlations, and observed that the demand correlation has little impact on the difficulty to address the problem. Finally, as expected, increasing the number of scenarios makes the problem more difficult to address for both *L&Opt* and CPLEX.

3.5.3.2 Improvement through time

It is also interesting to note how the quality of the solutions evolves over time. We thus let *L&Opt* and CPLEX run for eight hours and compared the evolution of the two methods. Table 3.5.7 displays the comparative results of *L&Opt* after two and eight hours (noted *L&Opt*(2h) and *L&Opt*(8h), respectively), with respect to those of CPLEX at the end of the eight hours (CPLEX(8h)). The relative gaps were computed as $\frac{L\&Opt - CPLEX}{L\&Opt} * 100$ (negative values indicate superiority of *L&Opt*) for the instances for which CPLEX found a feasible solution. Columns “Max” and “Avg” display the maximum and average gaps, respectively, of the two comparisons.

The results clearly show that *L&Opt* not only outperforms CPLEX in producing better solutions for difficult instances, but that it also does so in less computation time. Significant improvements are already observed after two hours of *L&Opt*, 5.36% on average and up to a maximum of 21.61% for the largest instances. These results are better when the matheuristic is given the same computation time, as shown in the last two columns, with a 15.61% average relative gap and a maximum of 48.76% for the largest instances.

Table 3.5.7: Performance comparison between L&Opt and CPLEX through time

Pro	L&Opt(2h)/CPLEX(8h)		L&Opt(8h)/CPLEX(8h)	
	Max(%)	Avg(%)	Max(%)	Avg(%)
R11	-9.86	-2.27	-17.44	-9.23
R12	-5.01	-3.21	-14.27	-6.13
R13	-15.5	-4.71	-28.59	-18.58
R14	-9.82	-6.37	-36.25	-15.16
R15	-21.61	-10.25	-48.76	-23.30

We complete this analysis by illustrating the behavior of the method we propose on the large instances for which CPLEX failed to provide any information (not even a lower bound) in eight hours of CPU time. Figure 3.2 displays the improvement in solution value obtained in time by *L&Opt* relative to the initial solution for two instances, 15.3-0-32 and 15.3-0-64. The relative improvement after *t* hours relative to the initial solution, L&Opt(1h), was calculated as $\frac{L\&Opt(t)-L\&Opt(1h)}{L\&Opt(t)} * 100$. The figure shows that the largest improvement occurs quite rapidly at the beginning of the search, but *L&Opt* continues to find improving solutions as more time is given.

3.6 Conclusions

We introduced a learning-based matheuristic for the stochastic fixed charge multi-commodity network design problem with uncertain demands. The innovative learning mechanism systematically explores combinations of scenarios to extract information regarding the solution structure of the stochastic problem. The matheuristic builds a global

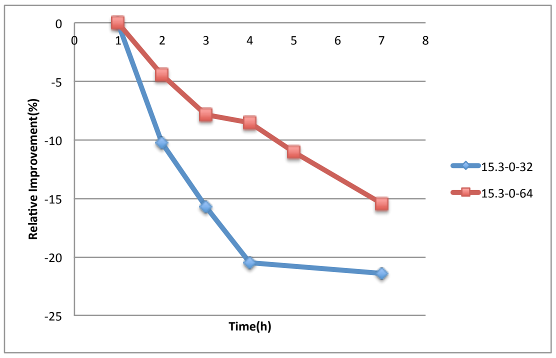

Figure 3.2: Relative improvement over initial solution in time

image of the promising structure of the stochastic solution out of the partial knowledge produced by the learning mechanism, and exploits it to define reduced-size problems that are solved by a MIP solver.

The results of extensive computational experiments showed that the proposed matheuristic performs very well, being highly effective in finding good-quality solutions for the large stochastic network design instances. This is very promising as, although presented in the context of the complex network-design problem setting, the proposed learning mechanism and matheuristic can be adapted to many other stochastic combinatorial optimization 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.

CHAPTER 4

ARTICLE 2: A REDUCED-COST-BASED RESTRICTION AND REFINEMENT MATHEURISTIC FOR STOCHASTIC NETWORK DESIGN

Chapter notes: This chapter has been submitted for publication to the *European Journal of Operation Research*. The published technical report is as follows: F. Sarayloo, T.G. Crainic and W. Rei, A Reduced Cost-based Restriction and Refinement Matheuristic for Stochastic Network Design. Technical report, Publication CIRRELT-2018-32, Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT), Montreal, Canada. Preliminary work was presented at the following conference:

- Optimization Days 2017, Montreal, Canada, May, 2017

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

4.1 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 [26, 82, 83]. 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 [63, 123], there are situations in which the deterministic solution shares some properties with the corresponding stochastic solution [76, 116–118]. 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. [25] 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 fixing performance by proposing a number of strategies in which reduced cost information is 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 fixing strategies, exploiting reduced cost information, into an iterative matheuristic approach, to efficiently deal with difficult stochastic instances.

The contributions of this paper are 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 sacrificing 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 the 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 a 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 instances available 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 4.2, and briefly review some relevant literature in Section 4.3. Section 4.4 introduces the proposed matheuristic. Finally, we present and analyze the experimental results in Section 4.5 and provide concluding remarks in Section 4.6.

4.2 Problem description

The two-stage stochastic formulation, or the *a priori optimization model* [17], 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 [29]. Let $\mathscr{G} = (\mathscr{N}, \mathscr{A})$ be a directed network with $\mathscr{N}$ representing a finite set of nodes and $\mathscr{A}$ a finite set of potential arcs. The set of commodities is represented by $\mathscr{K}$ where each is recognized by a unique pair of origin-destination nodes $(o(k), s(k))$. For each design arc $(i, j) \in \mathscr{A}$, 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^k_{ij} for each commodity $k \in \mathscr{K}$ and arc $(i, j) \in \mathscr{A}$.

We assume the finite scenario set $\mathscr{S}$ with the strictly positive corresponding probabilities of $p^1, \ldots, p^{|\mathscr{S}|}$. For a given scenario $s \in \mathscr{S}$, 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^{ks}_i, 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^{ks}_{ij} 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:

$$\text{minimize} \quad \sum_{(i,j)\in\mathscr{A}} f_{ij}y_{ij} + \sum_{s\in\mathscr{S}} p^s \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} c_{ij}x_{ij}^{ks} \tag{4.1}$$

$$\text{subject to} \quad \sum_{j\in\mathscr{N}^+(i)} x_{ij}^{ks} - \sum_{j\in\mathscr{N}^-(i)} x_{ji}^{ks} = d_i^{ks}, \qquad \forall i \in \mathscr{N}, \forall k \in \mathscr{K}, \forall s \in \mathscr{S} \tag{4.2}$$

$$\sum_{k\in\mathscr{K}} x_{ij}^{ks} \leq u_{ij}y_{ij}, \qquad \forall (i,j) \in \mathscr{A}, \forall s \in \mathscr{S} \tag{4.3}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in \mathscr{A} \tag{4.4}$$

$$x_{ij}^{ks} \geq 0, \qquad \forall (i,j) \in \mathscr{A}, \forall k \in \mathscr{K}, \forall s \in \mathscr{S} \tag{4.5}$$

The objective function (4.1) 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 (4.2) represent the flow conservation equations in each scenario, requiring that demand of commodity $k \in \mathscr{K}$ is routed from its origin node to its destination. Constraints (4.3) ensure that the same design is used in each scenario, and that arc capacity u_{ij} is never violated. Constraint (4.4) and (4.5) impose integrality and non-negativity restrictions on decision variables. We refer to this problem as the MCFND(S).

4.3 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 [120], 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 [32] and [93]. The progressive hedging (PH) method for addressing stochastic linear programs is a scenario-wise decomposition technique that was originally proposed in Rockafellar and Wets [103]. The PH algorithm is the foundation of a number of heuristic methods for SMCFND problems (e.g., [29, 34]).

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. [116, 117, 118] 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 [17] which measures the expected gain from solving a stochastic model rather than its deterministic counterpart). Similar observations were made by Wang et al. [124] for scheduled network design problems. Maggioni and Wallace [80] 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 a follow-up work to analyze the quality of the deterministic solution, Crainic et al. [25] 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 an efficient algorithmic approach. This approach, referred to as *matheuristics*, appears very promising by exploiting the synergies of mathematical programming and metaheuristics (see, Puchinger and Raidl [92], Raidl [94] 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 [11] and the kernel search proposed by Angelelli et al. [5]) and in the context of routing problems (e.g., Archetti et al. [6] and De Franceschi et al. [37]), 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, [108] 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. [108], 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.

4.4 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 subsets of design variables to be fixed to 1 and 0, leading to the reduced-size model. The resulting restricted problems are then solved by a MIP solver. We believe that using a refined approach in the selection of 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 expected value (EV) problem can be leveraged so as to guide variable fixing within MCFND(S) formulation.

In the following section 4.4.1, we present a number of strategies to examine how we can identify the desired set of fixed variables based on reduced cost information. The detailed algorithm will then be explained in Section 4.4.2.

4.4.1 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 by 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, including the choice of solution from which we extract the reduced costs, 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. [25]. Let $\bar{\mathbf{s}}^{lp} = (\bar{y}^{lp}, \bar{x}^{lp})$ be the optimal solution of LP relaxation of the 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):

$$\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in\mathscr{A}} f_{ij}y_{ij} + \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} c_{ij}x_{ij}^k && && (4.6)\\
\text{subject to} \quad & \sum_{j\in\mathscr{N}^+(i)} x_{ij}^k - \sum_{j\in\mathscr{N}^-(i)} x_{ji}^k = d_i^k, && \forall i\in\mathscr{N},\ \forall k\in\mathscr{K} && (4.7)\\
& \sum_{k\in\mathscr{K}} x_{ij}^k \le u_{ij}y_{ij}, && \forall (i,j)\in\mathscr{A} && (4.8)\\
& y_{ij}\in\{0,1\}, && \forall (i,j)\in\mathscr{A} && (4.9)\\
& x_{ij}^k \ge 0 && \forall (i,j)\in\mathscr{A},\ \forall k\in\mathscr{K} && (4.10)
\end{aligned}$$

Thus, in this strategy, the considered solution is $\bar{s}^{lp}$ which is derived from the LP relaxation of DSSP (4.6)-(4.10) and the choice of variable is the design variables $\bar{y}^{lp}$. Let $\mathscr{J}_0^{\bar{s}^{lp}} = \{1,2,\ldots,|\mathscr{J}_0^{\bar{s}^{lp}}|\}$ represent the index set of zero design variables $\bar{y}_j^{lp} = 0$ in the solution $\bar{\mathbf{s}}^{lp}$ and $\mathscr{R}_y^{\mathscr{J}_0^{\bar{s}^{lp}}} = \{r_1,\ldots,r_j,\ldots,r_{|\mathscr{J}_0^{\bar{s}^{lp}}|}\}$ be the set of *reduced cost* with respect to the components $\bar{y}_j^{lp}, j\in\mathscr{J}_0^{\bar{s}^{lp}}$. The set $\mathscr{R}_y^{\mathscr{J}_0^{\bar{s}^{lp}}}$ is then sorted in non-decreasing order. Let $r^{max} = \max_{j\in\mathscr{J}_0^{\bar{s}^{lp}}}\{r_j : r_j\in\mathscr{R}_y^{\mathscr{J}_0^{\bar{s}^{lp}}}\}$ and $r^{min} = \min_{j\in\mathscr{J}_0^{\bar{s}^{lp}}}\{r_j : r_j\in\mathscr{R}_y^{\mathscr{J}_0^{\bar{s}^{lp}}}\}$ be respectively the maximum and the minimum of the reduced costs of the set $\mathscr{R}_y^{\mathscr{J}_0^{\bar{s}^{lp}}}$. To determine the groups of variables to be fixed, the difference $r^{max} - r^{min}$ is divided into N_0 classes of constant size $\frac{r^{max}-r^{min}}{N_0}$. We then solve the model (4.1)-(4.5) by fixing to 0 the variables belonging to the classes p_0 to N_0 where $1\le p_0\le N_0$.

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. To do so, we use the expected value solution as an input to the MCFND(S) model (4.1)-(4.5) by adding the constraints $y\ge\bar{y}$ and then solve its LP relaxation, yielding the solution $\bar{\mathbf{s}}' = (\bar{\mathbf{y}}', \bar{\mathbf{x}}'^{s_1},\ldots,\bar{\mathbf{x}}'^{s_{|S|}})$. Thus, in this strategy, the considered solution is $\bar{\mathbf{s}}'$ and the choice of variable is the design variables $\bar{\mathbf{y}}'$. Let $\mathscr{J}_0^{\bar{s}'}$ represent the index set of zero design variables , i.e., $\bar{\mathbf{y}}'_j = 0$, in the solution $\bar{\mathbf{s}}'$, and $\mathscr{R}_y^{\mathscr{J}_0^{\bar{s}'}}$ be the set of reduced costs with respect to the components $\bar{\mathbf{y}}'_j\ j\in\mathscr{J}_0^{\bar{s}'}$. The set $\mathscr{R}_y^{\mathscr{J}_0^{\bar{s}'}}$ is then sorted in non-decreasing order. Let r^{max} and r^{min} be respectively the maximum and the minimum of the reduced costs of the set $\mathscr{R}_y^{\mathscr{J}_0^{\bar{s}'}}$. To determine the group of variables to be fixed, the difference $r^{max} - r^{min}$ is divided into N_0 classes of constant size $\frac{r^{max}-r^{min}}{N_0}$. We then solve the model (4.1)-(4.5) by fixing to 0 the variables belonging to the classes p_0 to N_0 where $1 \leq p_0 \leq N_0$.

Strategy 3. In this strategy, we try to upgrade the solution of the EV problem, $\bar{\mathbf{s}} = (\bar{y}, \bar{x})$, 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 (4.1)-(4.5). To do so, we use the solution obtained by the EV problem as an input to the model (4.1)-(4.5) by adding the constraints $y \geq \bar{y}$ and then solve the problem to obtain the upgraded solution $\bar{\mathbf{s}}'' = (\bar{\mathbf{y}}'', \bar{\mathbf{x}}''^{s_1}, \ldots, \bar{\mathbf{x}}''^{s_{|S|}})$. Thus, in this strategy, the considered solution is $\bar{\mathbf{s}}''$ and the choice of variables is the design variables $\bar{\mathbf{y}}''$. Let $\mathscr{J}_0^{\bar{s}''}$ represent the index set of zero design variables, i.e., $\bar{\mathbf{y}}_j'' = 0$, in the solution $\bar{\mathbf{s}}''$, and $\mathscr{R}_y^{\mathscr{J}_0^{\bar{s}''}}$ be the set of reduced costs with respect to the components $\bar{\mathbf{y}}_j''$, $j \in \mathscr{J}_0^{\bar{s}''}$. It should be noted that, given the fact that we are solving the MCFND(S) model (4.1)-(4.5) with the integrality requirements, we need to perform one additional step to obtain the reduced cost values. Once the problem (4.1)-(4.5) is solved and its optimal (integer) solution, $\bar{\mathbf{s}}''$, is obtained, we will then need to solve the LP relaxation of the problem (4.1)-(4.5) 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. The set $\mathscr{R}_y^{\mathscr{J}_0^{\bar{s}''}}$ is then sorted in non-decreasing order. Let r^{max} and r^{min} be respectively the maximum and the minimum of the reduced costs of the set $\mathscr{R}_y^{\mathscr{J}_0^{\bar{s}''}}$. Following this strategy, the difference $r^{max} - r^{min}$ is divided into N_0 classes of constant size $\frac{r^{max}-r^{min}}{N_0}$. We then solve the model (4.1)-(4.5) by fixing to 0 the variables belonging to the classes p_0 to N_0 where $1 \leq p_0 \leq N_0$.

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 excluding (or including) an arc using the specific reduced costs associated with all flow variables of that arc, one may hopefully provide 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 $\bar{\mathbf{s}}'' = (\bar{\mathbf{y}}'', \bar{\mathbf{x}}''^{s_1}, \ldots, \bar{\mathbf{x}}''^{s_{|S|}})$. However, we investigate the benefit of using reduced costs associated with the flow variables to identify the set of variables to be fixed. Let $\mathscr{J}_0^{\bar{s}''}$ represents the index set of the design variables set to zero in solution $\bar{\mathbf{s}}''$ and r_j^{ks} be the reduced costs with respect to the flow variables of commodity k in scenario s on arc index j (i.e., $\mathbf{x}''^{ks}_j$). We define $\bar{r}_j = \sum_{s \in S} p^s \sum_{k \in \mathscr{K}} (1/|\mathscr{K}|) r_j^{ks}$ to aggregate all reduced costs associated with the flow variables assigned to arc index j.

Let $\mathscr{R}_x^{\mathscr{J}_0^{\bar{s}''}}$ represents the set of aggregated reduced costs corresponding to index set $\mathscr{J}_0^{\bar{s}''}$ using the flow variables. The set $\mathscr{R}_x^{\mathscr{J}_0^{\bar{s}''}}$ is then sorted in non-decreasing order. Let $\bar{r}^{max}$ and $\bar{r}^{min}$ be respectively the maximum and the minimum of the reduced costs of the set $\mathscr{R}_x^{\mathscr{J}_0^{\bar{s}''}}$. We divide the difference $\bar{r}^{max} - \bar{r}^{min}$ into N_0 classes of constant size $\frac{\bar{r}^{max} - \bar{r}^{min}}{N_0}$. We then solve the model (4.1)-(4.5) by fixing to 0 the variables belonging to the classes p_0 to N_0 where $1 \leq p_0 \leq N_0$.

Strategy 5. In this strategy, the considered solution is again $\bar{\mathbf{s}}'' = (\bar{\mathbf{y}}'', \bar{\mathbf{x}}''^{s_1}, \ldots, \bar{\mathbf{x}}''^{s_{|S|}})$. However, we consider the reduced cost corresponding to both the design and the flow variables used in the previous strategies, thus obtaining a *composite reduced cost*; $r_j^+ = r_j + \bar{r}_j$. Let $\mathscr{J}_0^{\bar{s}''}$ represent the index set of zero design variables, i.e., $\bar{\mathbf{y}}''_j = 0$, in the solution $\bar{\mathbf{s}}''$, and $\mathscr{R}_{xy}^{\mathscr{J}_0^{\bar{s}''}}$ represents the set of composite reduced costs corresponding to index set $\mathscr{J}_0^{\bar{s}''}$ using both of the design and flow variables. The set $\mathscr{R}_{xy}^{\mathscr{J}_0^{\bar{s}''}}$ is then sorted in non-decreasing order. Let r^{+max} and r^{+min} be respectively the maximum and the minimum of the reduced costs of the set $\mathscr{R}_{xy}^{\mathscr{J}_0^{\bar{s}''}}$. We divide the difference $r^{+max} - r^{+min}$ into N_0 classes of constant size $\frac{r^{+max} - r^{+min}}{N_0}$. We then solve the model (4.1)-(4.5) by fixing to 0 the variables belonging to the classes p_0 to N_0 where $1 \leq p_0 \leq N_0$.

To determine the desirable variables to be fixed to 1 (i.e., open arcs), one may use the 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 equal to 1 in the solutions considered in Strategies 1-5). Thus, instead of fixing the last classes p_0 to N_0 (with the largest reduced cost values), we fix the variables belonging to the classes 1 to p_1, where $1 \leq p_1 \leq N_1$, which have the smallest reduced costs values.

4.4.2 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. Algorithm 5 sums up the entire procedure. We refer to problem *P* as the MCFND(S) problem (4.1)-(4.5) including all binary design variables. On the other hand, the restricted problem *RP* represents the MCFND(S) problem restricted by subsets of the design variables that are fixed to 0 or 1. In the following subsections, each component of Algorithm 5 is described in details.

Algorithm 5 Reduced cost-based restriction and refinement matheuristic

```
1: Initialization:                                                          ▷ Section 4.4.2.1
2: $k := 0$; construct initial solution $y^{Ini}$; set $\mathbf{y}^{best} := y^{Ini}$; let $\mathscr{J}_{\mathbf{1}}^{best}$ be the index set of design variables which are 1 in $\mathbf{y}^{best}$;
3: $k := 1$;
4: repeat
5:     Constructing the restricted problem:                                  ▷ Section 4.4.2.2
6:     phase 1:
7:     Construct $AP^k$ by fixing $\mathscr{J}_{\mathbf{1}}^{best}$ in the problem $P$ as $AP^k := P|_{\mathscr{J}_{\mathbf{1}}^{best}}$;
8:     Generate solution pool $\mathscr{P}^k = \{\mathbf{s}^1, \mathbf{s}^2, \ldots, \mathbf{s}^N\}$ for $AP^k$ considering parameter $\alpha$;
9:     phase 2:
10:    Perform Algorithm 6 to establish two subsets $\mathscr{J}_{\mathbf{0}}^{\mathscr{P}^k}$ and $\mathscr{J}_{\mathbf{1}}^{\mathscr{P}^k}$ using reduced cost information associated with $\mathscr{P}^k$;
11:    Solving the restricted problem:                                       ▷ Section 4.4.2.2
12:    Solve $RP^k$ which is constructed by fixing $\mathscr{J}_{\mathbf{0}}^{\mathscr{P}^k} \cup \mathscr{J}_{\mathbf{1}}^{\mathscr{P}^k}$ in problem $P$ as $RP^k := P|_{\mathscr{J}_{\mathbf{0}}^{\mathscr{P}^k} \cup \mathscr{J}_{\mathbf{1}}^{\mathscr{P}^k}}$ yielding the solution $\mathbf{y}^*_{RP^k}$ with objective value $z^*_{RP^k}$;
13:    Improvement check and Diversification:                                ▷ Section 4.4.2.3
14:    if $z^*_{RP^k} < z_{best}$ then
15:        $\mathbf{y}^{best} := \mathbf{y}^*_{RP^k}$ and update $\mathscr{J}_{\mathbf{1}}^{best}$;
16:        $z_{best} := z^*_{RP^k}$;
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 $q$ attempts then
21:            Let $\alpha \leftarrow \alpha + \Delta(\alpha)$ and go to line 8;
22:        else
23:            Enlarge the search space of $RP^k$ by reducing the size of $\mathscr{J}_{\mathbf{0}}^{\mathscr{P}^k}$ and $\mathscr{J}_{\mathbf{1}}^{\mathscr{P}^k}$ and go to line 12;
24:        end if
25:    end if
26:    $k := k + 1$;
27: until stopping criteria
28: return $\mathbf{y}^{best}$.
```

4.4.2.1 Initialization

At the beginning of Algorithm 5, we construct an initial solution y^{Ini} using the procedure described in Strategy 3. To do so, we use the expected value solution as an input to the model (4.1)-(4.5) and then solve the problem to obtain the solution y^{Ini}. Let y^{Ini} be the current best solution (i.e., $\mathbf{y}^{best} := y^{Ini}$), and $\mathscr{J}_{\mathbf{1}}^{best}$ be the index set of design variables which are equal to 1 in solution $\mathbf{y}^{best}$.

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

At each iteration of Algorithm 5, 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 RP^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 the MCFND(S) model would provide better information as compared to solutions obtained by DSSP (4.6)-(4.10). Therefore, to generate multiple good quality solutions, we aim to create solutions obtained by the MCFND(S) model and store them as a pool of solutions at each iteration of Algorithm 5. 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 $j \in \mathscr{J}_{\mathbf{1}}^{best}$ in problem P (i.e., $AP^k := \mathrm{P}|_{\mathscr{J}_{\mathbf{1}}^{best}}$) in order to reduce the problem size. We note that feasible solutions for AP^k are feasible for the MCFND(S) problem as well.

As stated in Algorithm 5, line 8, we generate multiple solutions for AP^k and store them in the solution pool $\mathscr{P}^k$. The solution pool $\mathscr{P}^k$, 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 AP^k and its objective function value, respectively .

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

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

The steps of this phase are stated in Algorithm 6. For each generated solution $\mathbf{s}^i \in \mathscr{P}^k$, we represent the index set of design variables which are equal to 0 by $\mathscr{J}_{\mathbf{0}}^{\mathbf{s}^i}$; the index set of design variables which are equal to 1 by $\mathscr{J}_{\mathbf{1}}^{\mathbf{s}^i}$; the value of objective function by $z(\mathbf{s}^i)$; and the weight by $w(\mathbf{s}^i) = \frac{1}{z(\mathbf{s}^i) - min_{\mathbf{s}^i \in \mathscr{P}} z(\mathbf{s}^i)}$, which indicates the relative quality of $\mathbf{s}^i$. The index sets of desirable variables $\mathscr{J}_{\mathbf{0}}^{\mathscr{P}^k}$ and $\mathscr{J}_{\mathbf{1}}^{\mathscr{P}^k}$ are created according to the desirability factor l_j associated with each arc j, measured using all solutions in $\mathscr{P}^k$. We first define the desirability factor l_j^i associated with each arc j in 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, we consider the reduced cost values associated with y variables, i.e., r_j, as the desirability factor, $l_j^i := r_j$; 2) if the choice of variable is x, it means we consider $\bar{r}_j$ as the desirability factor, $l_j^i := \bar{r}_j$ (recall that $\bar{r}_j = \sum_{s \in S} p^s \sum_{k \in K} (1/|\mathscr{K}|) r_j^{ks}$, where r_j^{ks} is the reduced costs with respect to the flow variables of commodity k in scenario s on arc index j (i.e., $\mathbf{x}''^{ks}_j$)); 3) if the choice of variable is both x and y, we consider the composite reduced cost, $r^+ = r_j + \bar{r}_j$, as the desirability factor, i.e., $l_j^i := r_j^+$. Once, the l_j^i values associated with each solution i are computed, we aggregate the desirability factors over all solutions (in line 7) as follows: $l_{0j} = \sum\limits_{\mathbf{s}^i \in \mathscr{P}} w(\mathbf{s}^i) * l_j^i$ for $j \in \bigcap_{\mathbf{s}^i \in \mathscr{P}^k} \mathscr{J}_{\mathbf{0}}^{\mathbf{s}^i}$ and $l_{1j} = \sum\limits_{\mathbf{s}^i \in \mathscr{P}} w(\mathbf{s}^i) * l_j^i$ for $j \in \bigcap_{\mathbf{s}^i \in \mathscr{P}^k} \mathscr{J}_{\mathbf{1}}^{\mathbf{s}^i}$. Let $\mathbf{L}_0^k = \{(j, l_{0j}) | j \in \bigcap_{\mathbf{s}^i \in \mathscr{P}^k} \mathscr{J}_{\mathbf{0}}^{\mathbf{s}^i}\}$ and $\mathbf{L}_1^k = \{(j, l_{1j}) | j \in \bigcap_{\mathbf{s}^i \in \mathscr{P}^k} \mathscr{J}_{\mathbf{1}}^{\mathbf{s}^i}\}$. We then sort $\mathbf{L}_0^k$

according to $l_{0,j}$ in non-decreasing order. Let l_0^{max} and l_0^{min} the maximum and minimum values in $\mathbf{L}_0^k$. 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 $\frac{l_0^{max}-l_0^{min}}{N_0}$ and store the index of variables belonging to the classes p_0 to N_0 $(1 \leq p_0 \leq N_0)$ in $\mathscr{I}_{\mathbf{0}}^{\mathscr{P}^k}$. We perform the same sorting procedure for $\mathbf{L}_1^k$ according to $l_{1,j}$. Let l_1^{max} and l_1^{min} be the maximum and minimum values in $\mathbf{L}_1^k$, respectively. We then divide the difference $l_1^{max} - l_1^{min}$ in N_1 classes of constant size $\frac{l_1^{max}-l_1^{min}}{N_1}$ and store the index of variables belonging to the classes 1 to p_1 ($1 \leq p_1 \leq N_1$) in $\mathscr{I}_{\mathbf{1}}^{\mathscr{P}^k}$. The two sets $\mathscr{I}_{\mathbf{0}}^{\mathscr{P}^k}$ and $\mathscr{I}_{\mathbf{1}}^{\mathscr{P}^k}$ are returned as the index sets of the most desirable arcs, at iteration k, to be closed and opened, respectively.

Algorithm 6 Reduced cost-based learning procedure

```
 1: *Initialization* Define the sets $\mathscr{I}_{\mathbf{0}}^{\mathbf{s}^i}$ and $\mathscr{I}_{\mathbf{1}}^{\mathbf{s}^i}$ for $\mathbf{s}^i \in \mathscr{P}^k$, let $w(\mathbf{s}^i)$ be the weight of
    solution $\mathbf{s}^i$;
 2: **for all** $\mathbf{s}^i \in \mathscr{P}^k$ **do**
 3:     if the choice of variable is $y$, then $l_j^i := r_j$ for $j \in \mathscr{I}_{\mathbf{0}}^{\mathbf{s}^i}$ and $j \in \mathscr{I}_{\mathbf{1}}^{\mathbf{s}^i}$;
 4:     if the choice of variable is $x$, then $l_j^i := \bar{r}_j$ for $j \in \mathscr{I}_{\mathbf{0}}^{\mathbf{s}^i}$ and $j \in \mathscr{I}_{\mathbf{1}}^{\mathbf{s}^i}$ ;
 5:     if the choice of variable is both $x$ and $y$, then $l_j^i := r_j^+$ for $j \in \mathscr{I}_{\mathbf{0}}^{\mathbf{s}^i}$ and $j \in \mathscr{I}_{\mathbf{1}}^{\mathbf{s}^i}$;
 6: **end for**
 7: Aggregate the desirability factor $l_j^i$ over all solutions as follows:
        $l_{0,j} = \sum_{\mathbf{s}^i \in \mathscr{P}} w(\mathbf{s}^i) * l_j^i$ for $j \in \bigcap_{\mathbf{s}^i \in \mathscr{P}^k} \mathscr{I}_{\mathbf{0}}^{\mathbf{s}^i}$ and $l_{1,j} = \sum_{\mathbf{s}^i \in \mathscr{P}} w(\mathbf{s}^i) * l_j^i$ for $j \in$
    $\bigcap_{\mathbf{s}^i \in \mathscr{P}^k} \mathscr{I}_{\mathbf{1}}^{\mathbf{s}^i}$;
 8: Let $\mathbf{L}_0^k = \{(j, l_{0j}) | j \in \bigcap_{\mathbf{s}^i \in \mathscr{P}^k} \mathscr{I}_{\mathbf{0}}^{\mathbf{s}^i}\}$. Sort $\mathbf{L}_0^k$ in non-decreasing order according to
    $l_{0j}$ and then create the set $\mathscr{I}_{\mathbf{0}}^{\mathscr{P}^k}$ (as in Section 4.4.2.2);
 9: Let $\mathbf{L}_1^k = \{(j, l_{1j}) | j \in \bigcap_{\mathbf{s}^i \in \mathscr{P}^k} \mathscr{I}_{\mathbf{1}}^{\mathbf{s}^i}\}$. Sort $\mathbf{L}_1^k$ in non-decreasing order according to
    $l_{0,j}$ and then create the set $\mathscr{I}_{\mathbf{1}}^{\mathscr{P}^k}$ (as in Section 4.4.2.2);
10: **return** $\mathscr{I}_{\mathbf{0}}^{\mathscr{P}^k}$ and $\mathscr{I}_{\mathbf{1}}^{\mathscr{P}^k}$.
```

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

4.4.2.3 Improvement check and Diversification

In this part of the algorithm, we check the improvement and, if needed, perform the diversification step (lines 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 the new incumbent ($\mathbf{y}^{best} := \mathbf{y}^*_{RP^k}$), and the search continues from this 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 performing the diversification step (lines 19-25) as follows. If $\mathbf{y}^{best}$ 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 by freeing more variables in the current restricted problem RP^k. To do so, we remove v_0 (v_1) percent of variables with the largest (smallest) values of l_{0j} (l_{1j}) from $\mathscr{J}_{\mathbf{0}}^{\mathscr{P}k}$ ($\mathscr{J}_{\mathbf{1}}^{\mathscr{P}k}$) to reduce the number of variables that are fixed in RP^k and then go to line 12 to find a better solution. The stopping criterion is the maximum computation time denoted as t_{max}.

4.5 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 4.5.1 and then provide a comparative analysis of the different proposed strategies in Section 4.5.2. We then detail the numerical results of the proposed matheuristic (denoted by RCHeur) by analyzing 1) in Section 4.5.3.1, the impact of the various features of the proposed RCHeur, and 2) in Section 4.5.3.2, the power of the proposed RCHeur in dealing with difficult instances through a comparative analysis of its performance versus the results of CPLEX and the Learn&Optimize (denoted by L&Opt) procedure proposed in Sarayloo et al. [108].

4.5.1 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. [29]. Each class is characterized by a number of nodes $|\mathscr{N}|$, number of arcs $|\mathscr{A}|$, and number of commodities $|\mathscr{K}|$, specified in Table 4.5.1. Each of these classes contains five networks with different "ratio" index values 1, 3, 5, 7, and 9, which indicate continuously increasing ratios of fixed to variable costs and total demand to total network capacity [29]. 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 4.5.1: Characteristics of instances

Problem	$\|\mathscr{N}\|$	$\|\mathscr{A}\|$	$\|\mathscr{K}\|$	Problem	$\|\mathscr{N}\|$	$\|\mathscr{A}\|$	$\|\mathscr{K}\|$
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. To evaluate the quality of solutions produced by the proposed heuristic approach, we also solve these instances with CPLEX version 12.2. The time limit is set to 500 minutes, when calling CPLEX in the following experiments.

4.5.2 Analyzing different strategies when using reduced cost information

In this section, we analyze and compare different strategies proposed in Section 4.4.1. In this part of the experiments, 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 qualify the quality of solutions obtained by different strategies, as the optimal solution of the majority of these instances can be obtained by CPLEX.

4.5.2.1 Comparing the strategies: fixing design 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., $\bar{\mathbf{s}}^{lp}$). Strategy 1 is denoted by $Str1(p_0,N_0)$ where the sorted set of reduced cost values $\mathscr{R}_y^{\mathscr{I}_0^{\bar{s}^{lp}}}$ 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 suitable values of $p_0{}^\star$ and $N_0{}^\star$. These values $(p_0{}^\star, N_0{}^\star)$ are then fixed and used for the remaining strategies to compare their performance. We present the comparative results according to the following measures: feasibility, solution quality, and computational efforts. Given the fact that fixing design variables to 0 may result in infeasibility issues, we report in Table 4.5.2 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 computation time, we provide a comparative analysis versus CPLEX in Table 4.5.3. The Gap and Time values reported for CPLEX refer, respectively, to the optimal gap and the computation time in seconds. As for "$Str1$", Gap and Time represent the optimality gap relative to the lower bound of CPLEX and the total computation time, respectively.

Table 4.5.2: 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

Table 4.5.2 shows that the total number of infeasible instances increases from 3 instances (in the case of $Str1(9,9)$) to 19 instances (in the case of $Str1(4,9)$). However, the sharp increase in the number of infeasible instances happens in the case of $Str1(3,9)$).

As shown in Table 4.5.3, the results in the case of $Str1(3,3)$, i.e., fixing one out of 3 classes of variables, are as follows. The number of infeasible instances is 9, the average optimality gap is 2.12% which is better than CPLEX with an average gap of 2.57% and the average computation time is reduced by almost 10% compared to CPLEX. Considering that $Str1(3,3)$ provides a little reduction in time, fixing less variables in $Str1(p_0,9), p_0 = 8,9$ does not seem reasonable since they cannot provide much fixed variables. However, in the case of $Str1(2,3)$, i.e., fixing two out of 3 classes of variables, the number of infeasible instances is 19, the average optimality gap is 1.59% which is better than CPLEX with an average of 2.49%, and the computation time is reduced by almost 50% relative to CPLEX. We note that fixing more variables in $Str1(p_0,9), p_0 = 1,2,3$ results in a significant increase in the number of infeasible instances (more than 54 out of 180 instances) as shown in Table 4.5.2. Therefore, it seems that $Str1(2,3)$ is able to provide a good performance in terms of improvement in solution quality and reduction in computation time, both compared to CPLEX, and is a good compromise for the considered instances. In the following, our goal is to examine if it is possible to improve the 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 the other strategies proposed in Section 4.4.1 and compare them with the values obtained by $Str1(2,3)$.

Table 4.5.4 displays the comparative results of performing Strategies 1 to 5 considering $(p_0^\star, N_0^\star) = (2,3)$, i.e, fixing to 0 almost two thirds of the non-basic variables with the highest reduced costs relative to $\mathscr{R}_y^{\mathscr{I}_0^{\bar{s}^{lp}}}$,$\mathscr{R}_y^{\mathscr{I}_0^{\bar{s}'}}$, $\mathscr{R}_y^{\mathscr{I}_0^{\bar{s}''}}$ $\mathscr{R}_x^{\mathscr{I}_0^{\bar{s}''}}$, and $\mathscr{R}_{xy}^{\mathscr{I}_0^{\bar{s}''}}$ 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 computation time in seconds (between parenthesis). As for the different strategies "$Str1$" to "$Str5$", Gap and Time represent the optimality gaps relative to the lower bound of CPLEX and the total computation time in seconds, respectively. Column "INF" indicates the number of infeasible

Table 4.5.3: Performance comparisons of $Str1(p_0,N_0)$ vs. CPLEX when fixing variables 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)	

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 upgrade the solutions in Strategies 2 to 5 (with an average optimality gap of at most 1.7%), compared to Strategy 1 (with an 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., $\mathscr{R}_x^{\mathscr{I}_0^{s''}}$), as defined in Strategy 4, provides the least computation time compared to the other strategies.

4.5.2.2 Comparing the strategies: fixing design 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 4.4.1 on the non-basic variables at their upper bound (i.e., 1). We first examine the performance of Strategy 1. In this strategy, denoted by $Str1(p_1,N_1)$, we use the optimal solution of the

Table 4.5.4: Performance comparisons of Strategies 1 to 5 when fixing variables to 0

Pro	Ins	CPLEX	Str1	$\mathscr{R}_y\mathscr{I}_0^{\bar{s}^{lp}}$	Str2	$\mathscr{R}_y\mathscr{I}_0^{\bar{s}'}$	Str3	$\mathscr{R}_y\mathscr{I}_0^{\bar{s}''}$	Str4	$\mathscr{R}_x\mathscr{I}_0^{\bar{s}''}$	Str5	$\mathscr{R}_{xy}\mathscr{I}_0^{\bar{s}''}$
		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)	

LP relaxation of the EV problem, i.e., $\bar{\mathbf{s}}^{lp}$. The set of reduced cost values $\mathscr{R}\mathscr{I}_1^{\bar{s}^{lp}}$ 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 variables to 1, we only present the comparison results on optimality gaps and computation times versus CPLEX in Table 4.5.5 to qualify the results obtained by Strategy 1. As shown in Table 4.5.5, in the case of fixing only one out of 3 classes ($Str1(1,3)$), Strategy 1 perform as well as CPLEX in terms of both optimality gaps and computation time. Moreover, in the case of fixing two out of 3 classes ($Str1(2,3)$), Strategy 1 performs slightly better than CPLEX by providing optimality gaps of 2.53% (in 9538 seconds) versus 2.56% (in 10229 seconds). The results show that Strategy 1 is not as effective in identifying variables fixed to 1 when compared to variables fixed to 0. This means that the LP relaxation of the EV problem (i.e., $\bar{\mathbf{s}}^{lp}$) does not provide many variable fixing choices, since there are too few design variables that are equal to 1 in the solution $\bar{y}^{lp}$ (there are a maximum of 3 design variables which are equal to 1 in the $\bar{y}^{lp}$ for the instances with ratios 1, 3 and 5). These results indicate 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 solutions over all instances. We will now examine whether we can improve the performance of Strategy 1 by upgrading the solution and using different choices of variables in Strategies 2 to 5.

Table 4.5.5: 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 4.5.6 shows the comparative results of performing Strategies 1 to 5 with $(p_1,N_1) = (2,3)$, i.e, fixing to 1 two thirds of the non-basic variables with the smallest reduced costs relative to $\mathscr{R}_y^{\mathscr{I}_1^{\bar{s}^{lp}}}$,$\mathscr{R}_y^{\mathscr{I}_1^{\bar{s}^{l}}}$, $\mathscr{R}_y^{\mathscr{I}_1^{\bar{s}^{ll}}}$, $\mathscr{R}_x^{\mathscr{I}_1^{\bar{s}^{ll}}}$, and $\mathscr{R}_{xy}^{\mathscr{I}_1^{\bar{s}^{ll}}}$ in Strategies 1 to 5, respectively. The table reports the same information as Table 4.5.4. The results show that, in terms of solution quality and computation time, the performance of using reduced cost is enhanced when we upgrade the solution in Strategies 2 to 5 compared to strategy 1 which uses the solution of the LP relaxation of the EV problem. Furthermore, using the reduced costs associated with flow variables (i.e., $\mathscr{R}_x^{\mathscr{I}_1^{\bar{s}^{ll}}}$) in Strategy 4 provides the least computation time compared to the other strategies. However, when assessing the optimality gaps obtained, we observe that all strategies 2 to 5 seem to be equivalent (i.e., the difference is at most 0.07%).

4.5.3 Numerical results of proposed matheuristic

In this section we present the results of the proposed matheuristic by evaluating 1) the effects of various components of the algorithm in Section 4.5.3.1, and 2) its power to deal with very difficult instances reported in the literature in Section 4.5.3.2. We note

Table 4.5.6: Performance comparisons of Strategies 1 to 5 for fixing to 1

Pro	Ins	CPLEX	Str1	$\mathscr{R}_y^{\mathscr{I}_1^{\bar{s}^{dp}}}$	Str2	$\mathscr{R}_y^{\mathscr{I}_1^{\bar{s}'}}$	Str3	$\mathscr{R}_y^{\mathscr{I}_1^{\bar{s}''}}$	Str4	$\mathscr{R}_x^{\mathscr{I}_1^{\bar{s}''}}$	Str5	$\mathscr{R}_{xy}^{\mathscr{I}_1^{\bar{s}''}}$
		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

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 6 corresponds to the flow variables. A preliminary analysis was conducted to fine tune the v_0, v_1, α, q and N parameters, which were set at the following values .05, .05, .02, 3 and 3, respectively.

4.5.3.1 Impact analysis

The purpose of this section is to evaluate the effects of two features of the proposed matheuristic, which are using solutions obtained by the 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 computation time in seconds, respectively.

Impact of using solutions obtained by the MCFND(S) model. To evaluate the impact of using feasible solutions obtained by the MCFND(S) model rather than those obtained by DSSP on performance of the proposed matheuristic, we show in Table 4.5.7 a comparison of both versions. In version "Deter-sols", to generate the solution pool $\mathscr{P}^k$ in Algorithm 5, we first choose randomly N scenarios and then solve their corresponding DSSP (4.6)-(4.10). However, in the version "Stoch-sol", we use the solutions obtained by the MCFND(S) problem, as explained in Section 4.4.2.2. 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 1.73% when using the solution obtained by DSSP) in almost half the time, which highlights the importance of using good quality solutions to identify the set of fixed variables.

Table 4.5.7: 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 solution? To answer this question and evaluate the impact of using multiple solutions (here $[N = 3]$ in Algorithm 5) versus a single solution on the performance of the proposed matheuristic, we show in Table 4.5.8 a 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 computation time. The fact 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 more refined information provided by multiple solutions leads to better solutions faster. These results strengthen the idea of generating multiple solutions at each iteration of the algorithm.

4.5.3.2 Performance on difficult instances

To evaluate the quality and power of the proposed matheuristic, and to address very difficult instances in the literature, we present the computational results performed on large R instances (i.e., R11-R15, as described in Table 4.5.1). We focus on 180 in-

Table 4.5.8: 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

stances that CPLEX was not able to solve to optimality after 500 minutes of computation time. We compare the performance of the proposed matheuristic, the *Learn&Optimize* (L&Opt) matheuristic proposed in Sarayloo et al. [108], and the MIP algorithm of CPLEX 12.8 to deal with these difficult instances. Table 4.5.9 provides a general view of the effectiveness of RCHeur by displaying the average improvement gap (negative values indicate better results) and percentage of instances with improved solutions (column "Win") obtained by RCHeur over those of the other methods. Columns "RCHeur/CPLEX" and"RCHeur/L&Opt" report the average improvement gap relative to L&Opt and CPLEX computed as $\frac{RCHeur-CPLEX}{RCHeur} * 100$ and $\frac{RCHeur-L\&Opt}{RCHeur} * 100$ after 500 minutes of computation time. We considered the best solution provided by CPLEX and L&Opt with a time of 500 minutes to assess the improvement provided by the proposed RCHeur. Overall, regarding the comparisons "RCHeur/CPLEX", we observed that CPLEX failed to provide any information after 500 minutes for 31 instances, and so we report the improvements only over the remaining 149 instances. We observed that RCHeur provides better solutions for all instances with a relative average improvement of 19.95%, compared to the solutions produced by CPLEX. This clearly shows the difficulty of these instances. Regarding the comparisons "RCHeur/L&Opt", both procedures were able to provide feasible solutions within the time limit, and so we report the improvements over all 180 instances. We observed that, on average, RCHeur is superior to L&Opt for more than 90% of the instances, with a relative average improvement of 6.07%, indicating the ability of RCHeur to deal with these very difficult instances. These results confirm that the proposed matheuristic is the method of choice for such difficult instances, for which the most powerful integer programming solvers are unable to even solve the LP relaxation of the problem.

Table 4.5.9: Performance comparison between RCHeur and L&Opt on difficult instances

Pro	# of	RCHeur/CPLEX		RCHeur/L&Opt	
	Ins	Gap(%)	Win(%)	Gap(%)	Win(%)
R11	27	-18.42	100	-8.01	100
R12	27	-9.79	100	-3.20	100
R13	36	-22.23	100	-3.32	86
R14	45	-21.38	100	-6.49	80
R15	45	-27.97	100	-9.34	86

4.6 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 restrictions in the context of stochastic network design. We specifically proposed different strategies to improve the EV solution and then extract the 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 experiments showed that the proposed algorithm is highly effective in finding good-quality solutions for very large instances of 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 adaptation 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 comes from the fact that most 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.

CHAPTER 5

ARTICLE 3: AN INTEGRATED LEARNING AND PROGRESSIVE HEDGING METHOD TO SOLVE STOCHASTIC NETWORK DESIGN

Chapter notes: The article in this chapter is expected to be submitted to the *EURO Journal on Computational Optimization*. More comparative analysis will be prepared for testing the solution approach before submission.

Abstract

In this paper we address *Multicommodity Capacitated Fixed-charge Network Design* problem with uncertain demands, modeled as a two-stage stochastic program. We rely on the progressive hedging algorithm (PHA) of Rockafellar and Wets where the scenarios are grouped in subproblems. We propose a two phase *integrated learning and progressive hedging (ILPH)* approach to deal with large number of scenarios. In our proposed approach, the *Learn&Optimize* procedure is adapted and applied as an efficient heuristic method to address the multi-scenario subproblems. We exploit the knowledge learned through the *Learn&Optimize* and particularly introduced a new reference point in each aggregation step of ILPH by exploiting the knowledge regarding the promising design variables which are built through the *Learn&Optimize* applied in the subproblems. In this way, we inject the knowledge learned through the heuristic procedure into the PHA leading to the proposed *ILPH*, which is considered as the main contribution in this paper. Given the fact that PHA may not converge to a single solution in the case of integer problems, the algorithm proceeds to the second phase if a consensus solution is not obtained. In phase II, we fix the design variables for which a consensus is obtained and solve the restricted problems to obtain the final solution. Extensive computational experiments illustrate that the proposed approach should be the method of choice when high-quality solutions to very large instances of stochastic network problems need to be found quickly.

5.1 Introduction

Multicommodity Capacitated Fixed-charge Network Design (*MCFND*) models represent a generic model that have been used to address many important planning problems in a variety of applications, such as transportation, logistics and telecommunications [26, 82, 83]. In these applications, it is required to design a network (i.e., choose a set of available arcs with associated capacity) that is to be used to route a given set of commodities in order to satisfy known demands between origin-destinations pairs. In doing so, one pays not only a routing cost proportional to the number of units of each commodity over a network arc, but also the cost that has to be paid whenever an arc is used. The objective of MCFND is to find the optimal design (i.e., selected arcs to be included (open) in the final network) that minimizes the total cost, computed as the sum of the fixed and routing costs.

In the real world, we are faced with the uncertainty in one or more of the elements of MCFND. Demand is, for instance, one of the key sources of uncertainty in any real world applications. Ignoring the demand uncertainty and its impact, that is, solving a deterministic model using a single estimate in replacement of a stochastic parameter, can lead to unfavorable and arbitrarily bad solutions. One should therefore consider demand uncertainty in the design process, which gives rise to the stochastic MCFND considered in this paper. The foremost consideration in incorporating uncertainties into the decision making process is the determination of an appropriate representation of the uncertain parameters. Scenario-based methodology is one of the most common approach in the literature. In this approach, the uncertainty is described by a finite set of discrete scenarios capturing how the uncertainty might play out in the future, together with associated probabilities.

In this paper, we consider the stochastic network design problem as a two stage stochastic program where first-stage decisions, i.e., design decisions, are made prior to realization of demand scenarios. Contingent on these design decisions and the realizations of the uncertain parameters, the second stage (routing) decisions are determined to adapt the first stage solution to the observed demand. Modelling uncertainty with scenarios leads to a very large scale mixed integer program, known as the extensive form (EF), which is too difficult to be handled with exact solution methods and state-of-the-art MIP solvers. Therefore, heuristic approaches are attractive methodologies to produce good-quality solutions within reasonable computing effort.

The PHA of Rockafellar and Wets [103] is considered as a successful meta-heuristic approach, when faced with non-convex integer problems [29, 34, 59]. The method decomposes the problem according to the scenarios (through the application of an augmented Lagrangian strategy) and solves the sub-problems for each scenario separately. When applied to network design, each single-scenario subproblem (SSSP) solved at each iteration of the PHA represents a deterministic network design problem yielding a (potentially different) design [29, 34]. These designs are aggregated (by taking the weighted average over all designs) in order to create a single reference point. Then in the next iteration the fixed cost associated with each arc is modified through a Lagrangian type technique to hopefully induce resulting subproblems that yield solutions closer to the current reference point. Given the fact that PHA may not converge to a single solution in the case of integer problems, the algorithm proceeds to the second phase to produce the final solution.

In this paper, instead of having each sub-problem associated with a single scenario (i.e., SSSP), each subproblem includes multiple scenarios. Grouping scenarios and solving multi-scenario subproblems (MSSP) were successfully applied in the context of network design problem by Crainic et al. [34]. They have shown that by solving multi-scenario subproblems, the proposed PHA-based metaheuristic produces better results in terms of solution quality and computing efficiency. However, the difficulties in solving the subproblems pose big challenges in such setting. We aim to contribute in addressing these challenges and improving the performance of PHA by proposing some significant refinements.

The contribution of this article is threefold. First, we introduce a new progressive hedging-based meta-heuristic to efficiently address the stochastic network design problem. The proposed method takes advantage of specialized methods (referred to as the *Learn&Optimize* procedure in [108]) to solve multi-scenario subproblems. Second, we introduce a new reference point in each aggregation step of PHA by exploiting local information on promising design variables which are built through the *Learn&Optimize* procedure applied in the subproblems. In this way, we integrate the knowledge learned through the subproblems into the PHA leading to the proposed *integrated learning and progressive hedging (ILPH)* approach to guide the overall search mechanism toward a unique design vector, which is considered as the main contribution in this paper. Third, we show, by means of extensive experimental campaign, the interest of the proposed approach in terms of computation time and solution quality, especially in dealing with very difficult instances with large number of scenarios.

The rest of paper is organized as follow. In Section 5.2, we recall the two-stage formulation of stochastic network design problem and briefly review some relevant literature in Section 5.3. Section 5.4 introduces the main ideas and a detailed description of our solution methodology. Finally, we present and analyze the experimental results in Section 5.5 and provide concluding remarks in Section 5.6.

5.2 Problem description

In this section, we present the two-stage stochastic program with recourse (referred to as the *a priori* optimization in [17]) for the MCFND problem proposed by Crainic et al. [29]. In such settings, a set of decisions have to be made a priori in a context where the related environmental information is not completely available, namely the demand volume of each commodity to transport from its origin to its destination. In the first stage of stochastic network design, the model makes the decisions on the network configuration (i.e. the design decisions). However, in the second stage, commodity flow decisions, from origins to destinations, are made in an optimal way based upon the restricted configuration imposed by the first stage and the realized random demands. This model is described in detail in Crainic et al. [29], and we briefly recall it here. The following notations are used:

Sets and indices:

- $\mathcal{N}$: Set of nodes, indexed by $i = 1, \dots, |\mathcal{N}|$.

- $\mathscr{A}$: Set of potential arcs $(i,j) \in \mathscr{A}$.
- $\mathscr{K}$: Set of commodities, indexed by $k = 1, \ldots, |\mathscr{K}|$ where each of them is recognized by a unique pair of origin-destination nodes $o(k) - s(k)$.
- $\mathscr{S}$: Set of scenarios used to model demand uncertainty, indexed by $s = 1, \ldots, |\mathscr{S}|$ with strictly positive corresponding probabilities of realization $p^1, \ldots, p^{|S|}$.

Variables:

- y_{ij}: Binary design variable, which indicates if the arc $(i,j) \in \mathscr{A}$ is included in the network in the first stage.
- x_{ij}^{ks}: Continuous flow variable representing the amount of commodity k's demand that flows on arc $(i,j) \in \mathscr{A}$ under scenario $s \in \mathscr{S}$.

Parameters:

- f_{ij}: Fixed cost incurred if the arc $(i,j) \in \mathscr{A}$ is included in the final design.
- u_{ij}: Capacity on arc $(i,j) \in \mathscr{A}$ limiting the total commodity flow that may use it.
- c_{ij}^k: Unit routing cost for each commodity $k \in \mathscr{K}$ and arc $(i,j) \in \mathscr{A}$.
- d_i^{ks}: Demand volume of commodity $k \in \mathscr{K}$ in node $i \in \mathscr{A}$ under scenario $s \in \mathscr{S}$.

The mathematical formulation is as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in\mathscr{A}} f_{ij} y_{ij} + \sum_{s\in\mathscr{S}} p^s \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} c_{ij} x_{ij}^{ks} && && (5.1)\\
\text{subject to} \quad & \sum_{j\in\mathscr{N}^+(i)} x_{ij}^{ks} - \sum_{j\in\mathscr{N}^-(i)} x_{ji}^{ks} = d_i^{ks}, && \forall i \in \mathscr{N},\ \forall k \in \mathscr{K},\ \forall s \in \mathscr{S} && (5.2)\\
& \sum_{k\in\mathscr{K}} x_{ij}^{ks} \le u_{ij} y_{ij}, && \forall (i,j) \in \mathscr{A},\ \forall s \in \mathscr{S} && (5.3)\\
& y_{ij} \in \{0,1\}, && \forall (i,j) \in \mathscr{A} && (5.4)\\
& x_{ij}^{ks} \ge 0, && \forall (i,j) \in \mathscr{A},\ \forall k \in \mathscr{K},\ \forall s \in \mathscr{S} && (5.5)
\end{aligned}$$

Model (5.1)-(5.5) is a large-scale mixed integer program with a block-diagonal structure, each block, defined by constraints (5.2) and (5.3). Constraints (5.2) represent the flow conservation equations in each scenario, requiring that each commodity's demand be routed from its origin node to its destination node. For a given scenario $s \in \mathscr{S}$, assuming that d^{ks} is the demand volume of commodity k under scenario s, the demand of costumer i for commodity k under 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. Constraints (5.3) ensure that the same design is used in each scenario, and that arc capacity u_{ij} is never violated. Constraints (5.4) and (5.5) impose integrality and non-negativity restrictions on decision variables. The objective function (5.1) 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. We refer to this model as MCFND(S) where its optimal solution is a single design that is cost-effective under all considered scenarios.

5.3 Literature review

There are limited solution methodologies that have been proposed for stochastic network design problems. As mentioned earlier, when a finite set of scenarios is used to estimate the stochastic parameters (see Dupačová et al. [41] and King and Wallace [70] for an overview on scenario generation methods), a stochastic program can be formulated as a equivalent (multi-scenario) deterministic problem. But due to the large scale of the problem, taking advantage of the structure used in decomposition-based approaches is especially beneficial and is the focus of much of the algorithmic work in this area. The goal of decomposition-based approaches is to divide the complex problem into subproblems to be able to solve them more efficiently. Such decomposition strategies can be categorized into two types. The first type decomposes the problem via decisional stages while the second type decomposes by scenarios. The former category (referred to as the L-shape method introduced in [120]) is a cutting-plane method which is the application of Benders decomposition to the solution of the equivalent (multi-scenario) deterministic problem. For completeness, detailed review on this type of decomposition approach for stochastic MCFND may be found in [32] and [93].

In the second category of decomposition strategies, referred to as the scenario decomposition, the original problem is decomposed by scenarios by applying Lagrangian relaxation to the non-anticipativity constraints (i.e., the constraints ensuring that a single design is used under all considered scenarios). Once the problem is decomposed, then each scenario becomes a deterministic problem to be solved (i.e., a single-scenario subproblem (SSP) defined for each scenario). The resulting scenario subproblems can then be used to obtain a general lower bound, by solving the Lagrangian dual as in [110], or as a means to produce more efficient solution approaches, e.g., [43] and [4], or by applying the progressive hedging based meta heuristics proposed in [29]. In the follow up work, Crainic et al. [33] introduced a new progressive hedging based metaheuristic that solves subproblems that may comprise multiple scenarios (i.e., multi-scenario subproblem (MSSP) defined for each group of scenarios).

Applying scenario-decomposition based methods, one could leverage efficient metaheuristics that are available for deterministic network design models (in the case of SSSP), or, for stochastic network design formulated using a reduced number of scenarios to address the subproblems (in the case of MSSP). Although the literature on efficient metaheuristic methods proposed for deterministic MCFND problems is very rich (e.g., [31, 50, 51, 61]), there are only limited contributions on efficient heuristic methods for solving the MCFND(S) problem. For example, Sarayloo et al. [108] proposed a learning based matheuristic approach where the main novelty is to provide a learning heuristic which is able to effectively identify structures of good-quality solutions where the scenarios and their influences on design decisions are gradually considered. In fact, a global image of the promising structure of the stochastic solution is built by gradually learning from the partial knowledge produced by the learning mechanism. The proposed matheuristic produces information on the promising design variables related to the considered scenarios. This can be used in the PHA to gather more refined local information yielded by subproblems to guide the search to a global good solution.

Exploiting common solution structures that exist between deterministic and stochastic solutions is another feature that may be employed in the solution methods based on scenario decomposition. Due to the high complexity and difficulty of stochastic network design problems (NDPs), a number of attempts in the literature on the stochastic NDP has been devoted to investigating how the solution to the deterministic model relates to the stochastic counterpart. It has been shown that, despite the fact that the solution to the deterministic model behaves badly in the stochastic settings [63, 123], there are situations where the deterministic solution shares some properties with the corresponding stochastic solution [25, 76, 116–118]. They show that the deterministic solution carries useful information (i.e., some structural patterns) which can be extracted to simplify the stochastic case. Following this insight, Sarayloo et al. [109] proposed a number of strategies to extract reduced cost information from good quality solutions to be used as a guide for fixing the variables in the MCFND(S) problems (i.e. fixing to 0 and 1).

Revisiting the PHA comprising multi-scenario subproblems appears a methodological avenue worth studying [34]. However, one still needs to iteratively solve a series of multi-scenario subproblems, which remains challenging. We propose to work towards: 1) solving a series of models at each iteration more efficiently 2) extracting more refined local information from subproblems to guide the search toward a global good design solution.

5.4 Solution methodology

In this section, we first re-write the MCFND(S) model (5.1)-(5.5) by partitioning the scenarios into groups and provide the outline of our proposed ILPH method with multiple scenario sub-problems in Section 5.4.1. Then, we describe the proposed methodological developments and strategies applied in each step of ILPH in Sections 5.4.2-5.4.4. Finally, we provide the detailed pseudocode of our proposed ILPH method in Section 5.4.5.

5.4.1 Preliminaries and the outline of the proposed PHA

We first formalize the progressive hedging method applied on the MCFND(S) model (5.1)-(5.5), where subproblems comprise multiple scenarios. To do so, we first need to partition the scenarios into groups in the MCFND(S) model (5.1)-(5.5). The groups are used to define the sub-problems. We then provide the outline of our proposed *integrated learning and PH method.*

Let G be the set of group indices. Suppose that the set of scenarios are partitioned to the $|G|$ groups denoted by $\{C_1,\dots,C_{|G|}\}$ where $C_g \subset S\ \forall g \in G$. Let $p^g = \sum_{s\in C_g} p^s$. The first stage variables y^g_{ij} are subscripted with a group index. This can be seen as creating a copy y^g_{ij} of each y_{ij} for each group g in order to allow design decisions to depend on the group, and yields the following model:

$$\min \quad \sum_{g\in G} p^g \Big(\sum_{(i,j)\in\mathscr{A}} f_{ij} y^g_{ij} + \sum_{s\in C_g} \frac{p^s}{p^g} \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} c^k_{ij} x^{ks}_{ij} \Big) \tag{5.6}$$

$$\text{subject to} \quad \sum_{j\in\mathscr{N}^+(i)} x^{ks}_{ij} - \sum_{j\in\mathscr{N}^-(i)} x^{ks}_{ji} = d^{ks}_i,\ \forall i\in\mathscr{N},\ \forall k\in\mathscr{K}, \forall s\in C_g, \forall g\in G \tag{5.7}$$

$$\sum_{k\in\mathscr{K}} x^{ks}_{ij} \le u_{ij} y^g_{ij},\ \forall (i,j)\in\mathscr{A}, \forall s\in C_g, \forall g\in G \tag{5.8}$$

$$y^g_{ij} = \bar{y}_{ij},\ \ \forall (i,j)\in\mathscr{A},\ \forall g\in G \tag{5.9}$$

$$y^g_{ij} \in \{0,1\},\ \forall (i,j)\in\mathscr{A},\ \forall g\in G, \tag{5.10}$$

$$x^{ks}_{ij} \ge 0,\ \forall (i,j)\in\mathscr{A},\ \forall k\in\mathscr{K}, \forall s\in C_g, \forall g\in G \tag{5.11}$$

Constraints (5.9), called as the non-anticipativity constraints, force all first stage decisions (i.e., design variables) to be equal to a single "overall design vector" denoted by $\bar{y}_{ij}$. We recall that the objective function and the rest of constraints are the ones that were previously introduced.

Following the decomposition scheme proposed in [103] , constraints (5.9) are relaxed using an augmented Lagrangian strategy, which yields the following objective function:

$$\min \sum_{g\in G} p^g \Big(\sum_{(i,j)\in\mathscr{A}} f_{ij} y_{ij}^g + \sum_{s\in C_g} \frac{p^s}{p^g} \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} c_{ij}^k x_{ij}^{ks} + \sum_{(i,j)\in\mathscr{A}} \lambda_{ij}^g (y_{ij}^g - \bar{y}_{ij}) + \frac{\rho}{2} (y_{ij}^g - \bar{y}_{ij})^2 \Big) \tag{5.12}$$

The Lagrangian multipliers $\lambda_{ij}^g \; \forall (i,j) \in \mathscr{A}, \forall g \in G$ are associated with the relaxed constraints (5.9) and ρ is a penalty ratio. Given the binary requirements for the design variables, after rearranging the terms, the function may be reduced to

$$\begin{aligned} \min \sum_{g\in G} p^g \Big(\sum_{(i,j)\in\mathscr{A}} (f_{ij} + \lambda_{ij}^g - \rho \bar{y}_{ij} + \frac{\rho}{2}) y_{ij}^g + \sum_{s\in C_g} \frac{p^s}{p^g} \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} c_{ij}^k x_{ij}^{ks} \Big) \\ - \sum_{(i,j)\in\mathscr{A}} \lambda_{ij}^g \bar{y}_{ij} + \sum_{(i,j)\in\mathscr{A}} \frac{\rho}{2} \bar{y}_{ij} \end{aligned} \tag{5.13}$$

For a given overall design $\bar{y}_{ij}$, the above formulation is decomposable according to the groups, taking the form of MCFND(S) problem with reduced number of scenarios and modified fixed costs $f_{ij} + \lambda_{ij}^g - \rho \bar{y}_{ij} + \frac{\rho}{2}, \forall (i,j) \in \mathscr{A}$. The sub-problem SP_g associated with group g can be expressed as follows:

$$\text{SP}_g\text{: minimize} \quad \sum_{(i,j)\in\mathscr{A}} \big(f_{ij} + \lambda_{ij}^g - \rho \bar{y}_{ij} + \frac{\rho}{2}\big) y_{ij}^g + \sum_{s\in C_g} \frac{p^s}{p^g} \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} c_{ij}^k x_{ij}^{ks} \tag{5.14}$$

$$\text{subject to} \quad \sum_{j\in\mathscr{N}^+(i)} x_{ij}^{ks} - \sum_{j\in\mathscr{N}^-(i)} x_{ji}^{ks} = d_i^{ks}, \; \forall i \in \mathscr{N}, \; \forall k \in \mathscr{K}, , \forall s \in C_g \tag{5.15}$$

$$\sum_{k\in\mathscr{K}} x_{ij}^{ks} \leq u_{ij} y_{ij}, \; \forall (i,j) \in \mathscr{A}, \; \forall s \in C_g \tag{5.16}$$

$$y_{ij}^g \in \{0,1\}, \; \forall (i,j) \in \mathscr{A}, \tag{5.17}$$

$$x_{ij}^{ks} \geq 0, \; \forall (i,j) \in \mathscr{A}, \; \forall k \in \mathscr{K}, \; \forall s \in C_g, \tag{5.18}$$

The PHA proposed for stochastic network design problem in Crainic et al. [34] consists of two main phases. In the first phase, a modified version of the classical PH algorithm is used. It iteratively solves an optimization subproblem for each group of scenarios separately. Using the different subproblem's solutions, it creates a reference point representing the level of consensus among the scenario group subproblems. The PHA adjust the fixed costs of each group subproblem (reflected as "penalties") to incentivize them to eventually produce a single high quality solution, until a stopping criterion is met. In phase II, information obtained during the PH iterations is used to identify a set of design variables for which consensus is obtained. This allows us to fix several variables in the original problem.

The outline of our proposed *ILPH* method is given in Algorithm 7. In our proposed approach, we adapt the *Learn&Optimize* procedure proposed in Sarayloo et al. [108] to be used as an efficient heuristic method to address the multi-scenario subproblems at each iteration (line 3). We aim to exploit the knowledge learned through the *Learn&Optimize* procedure in the aggregation and penalty updates of the *ILPH* to hopefully improve its performance (lines 4-5). In this way, we inject the information obtained by the *Learn&Optimize* into the PHA which results in the integrated approach, which is our main contribution in this paper. Given the fact that the PHA may not converge to a single solution in the case of integer problems, the algorithm proceeds to the second phase if a consensus solution is not obtained, once the stopping criterion is met. In phase II, we fix the design variables for which a consensus is obtained and solve the restricted problems to obtain the final solution (line 8).

Algorithm 7 The outline of the proposed integrated learning and PH method

```
1: Initialization                                             ▷ Section 5.4.2
2: while Stopping criteria is not met do
3:     Solving heuristically multi-scenario subproblems       ▷ Section 5.4.3
4:     Aggregation                                            ▷ Section 5.4.4
5:     Penalty update                                         ▷ Section 5.4.4
6: end while
7: Phase II:
8: Fix the design variables for which consensus is obtained and solve the restricted
   problem
```

In what follows, a step-by-step description of our adaptation of PHA is provided. We first describe the initialization step in Section 5.4.2 and proceed to describe the heuristic *Learn&Optimize* used to heuristically solve the multi-scenario subproblems in Section 5.4.3. We then explain, in Section 5.4.4, how we exploit the information provided by applying the Learn&Optimize procedure in subproblems to create a new reference point and update the Lagrangian multipliers to hopefully improve the performance of PHA.

5.4.2 Initialization

The algorithm is initialized by constructing the list of scenario groups $\bar{C} = \{C_1, \ldots, C_{|G|}\}$. The scenarios within each group are chosen randomly. For each group $g \in G$, we solve heuristically subproblem SP^0_g: minimize $\sum_{(i,j)\in\mathscr{A}} \left(f_{ij}y^g_{ij} + \sum_{s\in C_g} \frac{p^s}{p^g} \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} c^k_{ij}x^{ks}_{ij}\right)$ st. $(5.15)-(5.18)$ using the Learn&Optimize procedure described in Section 5.4.3. Once we heuristically solved the subproblems $g \in G$, we then perform the aggregation step to produce the reference (aggregated) point as well as the other heuristic solution as explained in section 5.4.4.

5.4.3 Solving subproblems heuristically using Learn&Optimize procedure

The most challenging part of the PHA is solving the multi-scenario stochastic network design problem SP^ν_g: (5.14)-(5.18) that occur as subproblems at each iteration ν. Dealing with multiple scenarios in conjunction with the integrality constraints makes the subproblems very hard to solve to optimality. But, there is evidence suggesting that the exact solution of the subproblem is not required [67], and that heuristic solutions can work satisfactorily [77].

We adapt the *Learn&Optimize* procedure, as a heuristic method proposed in Sarayloo et al. [108], to be applied in subproblems of the PHA. It should be noted that there are two important reasons we integrate the so-called *Learn&Optimize* in the PHA. First, we apply it as an efficient heuristic to solve the subproblems heuristically [108]. Our second reason is that we aim to exploit the knowledge learned through the *Learn&Optimize* procedure in the aggregation step of PHA to hopefully improve its performance. In this sense, we inject the the local information obtained by *Learn&Optimize* within the subproblems into the PHA. which results in the integrated approach.

The *Learn&Optimize* procedure, proposed in [108], iteratively executes a *learning step*, to learn and build statistics on solution characteristics. As we need to apply *Learn&Optimize* multiple times at each iteration of PHA to address the multi-scenario subproblems, we will need to make some modifications to efficiently integrate the procedure in the PHA. We recall the learning step and highlight the proposed modifications in Section 5.4.3.1, while Section 5.4.3.2 provides the full description of the adapted *Learn&Optimize* procedure.

5.4.3.1 Learning step

To learn and build statistics on solution characteristics yielding the set of promising design variables, Sarayloo et al. [108] introduced the concept of *Artificial Demand Scenario* (*ADS*) built out of particular combinations of two scenarios. Given the fact that we are faced with multi-scenario subproblem which should be solved multiple times at each iteration of PHA, we introduce *Group-based Artificial Demand Scenario Gb-ADS* and define the associated auxiliary problem which helps us to address the multi-scenario subproblems more efficiently. In this way, we could improve the efficiency of the learning step by decreasing the number of ADSs while treating all scenarios involved in each subproblem. One then learns by iteratively building Gb-ADSs, solving the associated auxiliary problems, and gradually building an image of design variables potentially belonging to good solutions to the multi-scenario subproblem. We proceed by defining the *Gb-ADS* and the associated auxilary problem *AP*.

A *Group-based Artificial Demand Scenario Gb-ADS* $\boldsymbol{\delta}^g$, $g \in G$ under the scenarios in group g i.e., $(s_1^g, \ldots, s_{|g|}^g)$ can be then expressed as

$$\boldsymbol{\delta}^g = \begin{pmatrix} \delta_1^g(s_1^g,\ldots,s_{|g|}^g) \\ \delta_2^g(s_1^g,\ldots,s_{|g|}^g) \\ \delta_3^g(s_1^g,\ldots,s_{|g|}^g) \\ \vdots \\ \delta_{|\mathscr{K}|}^g(s_1^g,\ldots,s_{|g|}^g) \end{pmatrix}, \text{ such that } \delta_k^g(s_1^g,\ldots,s_{|g|}^g) = d_k(s_1^g) \vee d_k(s_2^g) \vee \ldots \vee d_k(s_{|g|}^g), \forall k \in \mathscr{K}. \tag{5.19}$$

Let Δ^g be set of Gb-ADSs, δ^g, generated for the group g containing the scenarios C_g. We use a simple procedure to construct the set Δ^g in this paper. Algorithm 8 builds a set Δ^g of cardinality N_{Δ^g}, through the random selection of the demand values of given scenarios in C_g. In other words, we build the vector of Gb-ADS $\boldsymbol{\delta}^g$, $g \in G$ under the scenarios in group g i.e., $(s_1^g,\ldots,s_{|g|}^g)$ by randomly selecting $s_i \in C_g$ in order to copy its demand value associated to commodity k (i.e., $d_k(s_i)$) and let $\delta_k^g \leftarrow d_k(s_i)$ (line 2-4). The procedure stops when it builds N_{Δ^g} number of Gb-ADSs (line 6).

Algorithm 8 *Construct* Δ^g

```
1: repeat
2:     for all  $k \in \mathscr{K}$ do
3:         Randomly choose $s_i \in C_g$ and let $\delta_k^g \leftarrow d_k(s_i)$
4:     end for
5:     Let $\Delta^g \leftarrow \boldsymbol{\delta}^g \bigcup \Delta^g$
6: until  $|\Delta^g| = N_{\Delta^g}$
7: Return $\Delta^g$
```

As proposed in Sarayloo et al. [108], we aim to explore the solution characteristics associated to each Gb-ADS, $\boldsymbol{\delta}^g \in \Delta^g$, to extract information regarding promising design variables. The exploration is performed by solving an *Auxilary Problem*, $AP(\boldsymbol{\delta}^g, \hat{y})$, for each artificial demand scenario $\boldsymbol{\delta}^g \in \Delta^g$ considering a given design $\hat{y}$.

For completeness, we recall the *Auxilary Problem* proposed in Sarayloo et al. [108]. To define $\widetilde{AP}_g(\boldsymbol{\delta}^g, \hat{y})$, we separate the set of arcs $\mathscr{A}$ according to the given design $\hat{y}$. Then, $\mathscr{A} = \mathscr{A}^0 \cup \mathscr{A}^1$, where $\mathscr{A}^0 = \{(i,j)|(i,j) \in \mathscr{A}, \hat{y}_{ij} = 0\}$ and $\mathscr{A}^1 = \{(i,j)|(i,j) \in \mathscr{A}, \hat{y}_{ij} = 1\}$ are the sets of closed and open arcs in $\hat{y}$, respectively. By considering the fact that the fixed cost f_{ij}^{v} is updated at each iteration of PHA, we then define a modified arc variable cost $\bar{c}_{ij}$ by linearizing the fixed cost of the closed arcs

$$\bar{c}_{ij} = \begin{cases} c_{ij} + \frac{f_{ij}^{v}}{u_{ij}}, \; \forall (i,j) \in \mathscr{A}^0, \\ c_{ij}, \; \forall (i,j) \in \mathscr{A}^1 \end{cases} \tag{5.20}$$

and solve the $\widetilde{AP}(\boldsymbol{\delta}^g, \hat{y})$ multi-commodity network flow problem

$$\widetilde{AP}_g(\boldsymbol{\delta}^g, \hat{y}): \text{ minimize } \sum_{k \in \mathscr{K}} \sum_{(i,j) \in \mathscr{A}} \bar{c}_{ij} x_{ij}^k \tag{5.21}$$

$$\text{subject to } \sum_{j \in \mathscr{N}^+(i)} x_{ij}^k - \sum_{j \in \mathscr{N}^-(i)} x_{ji}^k = \delta_k^{ig}, \quad \forall i \in \mathscr{N}, \forall k \in \mathscr{K} \tag{5.22}$$

$$\sum_{k \in \mathscr{K}} x_{ij}^k \le u_{ij}, \quad \forall (i,j) \in \mathscr{A} \tag{5.23}$$

$$x_{ij}^k \ge 0, \quad \forall (i,j) \in \mathscr{A}, \forall k \in \mathscr{K} \tag{5.24}$$

Solving $\widetilde{AP}_g(\boldsymbol{\delta}^g, \hat{y})$ yields the solution $x_{ij}(\boldsymbol{\delta}^g) = \sum_{k \in \mathscr{K}} x_{ij}^k, \forall (i,j) \in A$, with $\mathscr{A}_{\boldsymbol{\delta}^g} = \{(i,j) | x_{ij}(\boldsymbol{\delta}^g) > 0\}$. We define a corresponding design solution as $y_{ij}^{\boldsymbol{\delta}^g} = 1$, when $x_{ij}(\boldsymbol{\delta}^g) > 0$, and 0, otherwise. It is noteworthy that some of the arcs in $\mathscr{A}^0$, closed in $\hat{y}$, may be open in $y_{ij}^{\boldsymbol{\delta}^g}$ to satisfy the demand vector $\boldsymbol{\delta}^g$. These modifications capture the interactions occurring in the integration of multiple scenarios within $\boldsymbol{\delta}^g$, yielding partial information regarding the design arcs required to address the uncertainty captured by the scenarios involved in group g. Repeating this procedure for different Gb-ADSs builds the knowledge we seek.

5.4.3.2 Learn& Optimize procedure

The adapted *Learn&Optimize* procedure we apply in this paper iteratively executes a *learning step* to identify a promising set of design variables. Then, a *partial-optimization step* is performed where the identified promising variables are fixed and the reduced-size formulation is solved exactly. The original procedure is proposed in Sarayloo et al. [108], however, we make some modification to efficiently apply the procedure in the series of subproblems at each iteration of the PHA.

The Learn&Optimize procedure is described in Algorithm 9. The set of Gb-ADSs, $\Delta^{g\nu}$, is reconstructed at each iteration ν of PHA as described in Algorithm 8. Such an approach, i.e., reconstructing a new set of $\Delta^{g\nu}$ at each iteration of PHA, may allow us to obtain information that would not be available from using a single set of $\Delta^{g\nu}$ in all iterations. We define the *frequency memory*, $F_{ij}^{g\nu}$, representing how often arc (i,j) has been used in the solutions of the different $\widetilde{AP}_g(\boldsymbol{\delta}^g,\hat{y})$ and the normalized frequencies values, $\mathbf{f}_{ij}^{g\nu}$, which is computed as $\mathbf{f}_{ij}^{g\nu} := F_{ij}^{g\nu}/max\{F_{ij}^{g\nu}|(i,j) \in \mathscr{A}\}$ (line 10). We keep the frequency memories built in the previous iterations i.e., $F_{ij}^{g\nu} \leftarrow F_{ij}^{g\nu-1}$ in order to keep track of promising arcs from the beginning. We also define $\mathscr{A}_{\Delta^{g\nu}}^{\nu}$, the set of design arcs used in at least one $\widetilde{AP}_g(\boldsymbol{\delta}^g,\hat{y})$ in iteration ν, and $\mathscr{A}^{g\nu}$, the set of promising design variables to be identified by the procedure.

The main loop (lines 3 to 9) iterates over the Gb-ADSs in $\Delta^{g\nu}$, each being discarded, after it has been examined. The loop stops when the set of artificial demand scenarios becomes empty. The $\widetilde{AP}_g(\boldsymbol{\delta}^g,\hat{y})$ is solved for each $\boldsymbol{\delta}^g \in \Delta^{g\nu}$, to distribute the demand of $\boldsymbol{\delta}^g$ (line 5). The corresponding design vector is created (line 7), while the set of used design arcs and the frequency memories are updated on line 6. Once artificial demand scenarios $\boldsymbol{\delta}^g \in \Delta^{g\nu}$ are treated, then a reduced problem by fixing $\mathscr{A}^{g\nu}$ as the most frequently used arcs (given a threshold τ) is solved using a MIP solver yielding the design solution $y_{ij}^{g\nu}, \forall(ij) \in \mathscr{A}$. The procedure returns the normalized frequencies values $\mathbf{f}_{ij}^{g\nu}$, $\forall(i,j) \in \mathscr{A}$, as well as the design solution $y_{ij}^{g\nu}, \forall(ij) \in \mathscr{A}$.

5.4.4 Aggregation and Penalty updates in the proposed PHA

In this section, we explain how we exploit and inject the local information obtained by the Learn& Optimize procedure into the PHA to produce a new reference point as well as a new heuristic design, in each aggregation step of PHA, to hopefully improve its performance. The new reference point, described in section 5.4.4.1, is created by exploiting the solution characteristics (primal information) and serves as the new reference

Algorithm 9 Learn&Optimize procedure to solve SP_g^ν

```
1:  Initialization: $F_{ij}^{g\nu} \leftarrow F_{ij}^{g\nu-1}, \forall(ij) \in A$, $\mathscr{A}^{g\nu} \leftarrow \emptyset$; construct $\Delta^{g\nu}$;
2:  Learning and memorizing:
3:  repeat
4:      Randomly choose a Gb-ADS $\delta^g \in \Delta^{g\nu}$;
5:      Solve $\widetilde{AP}_g(\boldsymbol{\delta}^g, \hat{y}^\nu)$ yielding $x_{ij}^\nu(\boldsymbol{\delta}^g), \forall(i,j) \in \mathscr{A}$;
6:      Identify $\mathscr{A}_{\boldsymbol{\delta}^g}^\nu$ and compute $y_{ij}^{\boldsymbol{\delta}^g}$, $\forall(i,j) \in \mathscr{A}_{\boldsymbol{\delta}^g}^\nu$;
7:      Update $\mathscr{A}_{\Delta^{g\nu}}^\nu := \mathscr{A}_{\Delta^{g\nu}}^\nu \bigcup \mathscr{A}_{\boldsymbol{\delta}^g}^\nu$ and $F_{ij}^{g\nu} := F_{ij}^{g\nu} + 1$, for all $(i,j) \in \mathscr{A}_{\boldsymbol{\delta}^g}^\nu$;
8:      Remove $\boldsymbol{\delta}^g$ from $\Delta^{g\nu}$;
9:  until $\Delta^{g\nu} = \emptyset$;
10: Normalize frequencies $\mathbf{f}_{ij}^{g\nu} := F_{ij}^{g\nu}/max\{F_{ij}^{g\nu}|(i,j) \in \mathscr{A}\}$, $\forall(i,j) \in \mathscr{A}$;
11: for all $(i,j) \in \mathscr{A}$ do
12:     if $\mathbf{f}_{ij}^{g\nu} \geq \tau$ then $\mathscr{A}^{g\nu} \leftarrow \mathscr{A}^{g\nu} \cup \{(i,j)\}$;
13:     end if
14: end for
15: Partial optimization:
16: Solve $SP_g^\nu$, by fixing variables belonging to $\mathscr{A}^{g\nu}$ to open, yielding solution
    $y_{ij}^{g\nu}, \forall(ij) \in \mathscr{A}$
17: Return the normalized frequencies $\mathbf{f}_{ij}^{g\nu}$, $\forall(ij) \in \mathscr{A}$ and solution $y_{ij}^{g\nu}, \forall(ij) \in \mathscr{A}$.
```

point to update the penalties in the PHA, whereas the proposed heuristic design, in section 5.4.4.2, is created by exploiting the dual information associated with the solutions of subproblems and serves as the initial solution for the subproblems in the following iteration.

5.4.4.1 Introducing a new reference point

As the reference point is used to indicate what appears to be the current trend for opening and closing arcs amongst the subproblem designs, it is obviously vital to have a good aggregated point reflecting such a trend amongst the subproblems. As mentioned earlier, the reference point in the PHA applied on stochastic MCFND in [29, 34], is constructed as follows: $\bar{y}_{ij}^\nu \leftarrow \sum_{g \in G} p^g y_{ij}^{g\nu}, \forall(i,j) \in \mathscr{A}$. The dual prices $\lambda_{ij}^{g\nu}$ are then updated, using the reference solution $\bar{y}_{ij}^\nu$ and the sole external parameter associated with the basic PHA, i.e., ρ, as follows: $\lambda_{ij}^{g\nu} \leftarrow \lambda_{ij}^{g\nu-1} + \rho(y_{ij}^{g\nu} - \bar{y}_{ij}^\nu)$. The mentioned strategy only considers a single design vector obtained by each subproblem to produce the reference point. However, it is worth to exploit more local information obtained by each subproblem, specially when each subproblem presents a multi-scenario structure.

In this paper we introduce a new reference point by exploiting the information derived by the learning mechanisms in the subproblems. In the *Learn&Optimize* procedure performed to solve each subproblem, we create the history of promising design variables identified in each subproblem. We accordingly build frequency memories $F_{ij}^{gv}, \forall(i,j) \in \mathscr{A}$, as well as the normalized values, i.e., $\mathbf{f}_{ij}^{gv} \in [0,1], \forall(i,j) \in \mathscr{A}$. Using the refined information, provided by $\mathbf{f}_{ij}^{gv}$, we have the opportunity to better explore the trend amongst the design variables that becomes available during the iterations of the *Learn&Optimize* procedure.

We then propose $\tilde{y}_{ij}^{v} \leftarrow \sum_{g \in G} p^g \mathbf{f}_{ij}^{gv}$, $\forall(i,j) \in \mathscr{A}$, where $p^g = \sum_{s \in C_g} p^s$, as a reference point in the aggregation step of the ILPH. Therefore, the Lagrangian multipliers λ_{ij}^{gv} $\forall(i,j) \in \mathscr{A}$ are updated using the new reference point $\tilde{y}_{ij}^{v}$ $\forall(i,j) \in \mathscr{A}$ as follows: $\lambda_{ij}^{gv} \leftarrow \lambda_{ij}^{gv-1} + \rho(\mathbf{f}_{ij}^{gv} - \tilde{y}_{ij}^{v})$.

5.4.4.2 Producing a temporary design solution

The second idea is to exploit dual information provided by subproblem solutions to create a temporary design solution at each iteration of PHA. This is motivated by the observation made in Sarayloo et al. [109] suggesting that special knowledge obtained by reduced cost values associated with multiple solutions could allow the identification of a good quality solution in the context of stochastic network design problems.

We extract reduced cost information associated with the solutions of subproblems in the ILPH algorithm to produce a temporary design solution. This temporary design solution $\hat{y}_{ij}^{v}, \forall(i,j) \in \mathscr{A}$ is constructed in the aggregation step of the ILPH algorithm and serves as a initial design solution for the Learn&Optimize procedure in the following iteration. To create the solution $\hat{y}_{ij}^{v}$, we proceed as follows. We initially use the solution $\tilde{y}_{ij}^{v} \leftarrow \sum_{g \in G} p^g f_{ij}^{gv}$, $\forall(i,j) \in \mathscr{A}$ and partition the set of design variables into two disjoint subsets:

- $\hat{\mathscr{A}_1} = \{(i,j) | \tilde{y}_{ij}^{v} \leq l_0 \text{ or } \tilde{y}_{ij}^{v} \geq u_1\}$: the set of design variables for which a consensus has been almost obtained (given thresholds l_0 and u_1) among the groups, or in other words, (almost) all groups agree that these arcs have to be opened.

- $\hat{\mathscr{A}_2} = \{(i,j) | l_0 \leq \tilde{y}_{ij}^{\nu} \leq u_1\}$: the set of the remaining design variables or those for which a consensus has not been obtained.

All design variables in $\hat{\mathscr{A}_1}$ are set to the value 0 or 1 as follows,

$$\hat{y}_{ij}^{\nu} = \begin{cases} 0, \text{ if } \tilde{y}_{ij}^{\nu} \leq l_0, \\ 1, \text{ if } \tilde{y}_{ij}^{\nu} \geq u_1. \end{cases} \tag{5.25}$$

For the rest of variables in $\hat{\mathscr{A}_2}$, the decision is based on reduced cost information. To do so, let $r_{ij}^{g}, \forall (i,j) \in \hat{\mathscr{A}_2}$ be the reduced cost associated with y_{ij}^{g} $\forall (i,j) \in \hat{\mathscr{A}_2}$; $G_{ij}^{1} = \{g | y_{ij}^{g} = 1\}$ be the set of groups where the associated design variables y_{ij}^{g} is one, and $\bar{r}_{ij}^{g\nu} = \sum_{g \in G_{ij}^{1}} p^{g} r_{ij}^{g\nu}, (i,j) \in \hat{\mathscr{A}}$ be the average reduced cost over groups $g \in G_{ij}^{1}$. It should be noted that, given the fact that we are solving the restricted problem SP_{g}^{ν} with the integrality requirements, we need to perform one additional step to obtain the reduced cost values. Once the restricted problem SP_{g}^{ν} is solved and its optimal (integer) solution, $y_{ij}^{g\nu}, \forall (ij) \in \mathscr{A}$, is obtained, we will then need to solve the LP relaxation of the problem SP_{g}^{ν} while the design variables are fixed to the values of the obtained optimal solution $y_{ij}^{g\nu}, \forall (ij) \in \mathscr{A}$. In this way, one can obtain the set of reduced cost values associated with design variables.

We represent the set of reduced cost by $R = \{\bar{r}_{ij}^{g\nu} | (i,j) \in \hat{\mathscr{A}_2}\}$. In order to identify good candidate design variables to be fixed to 1 (open), we choose the variables with the smallest reduced cost values. To do so, we sort R in non-decreasing order according to $\bar{r}_{ij}^{g\nu}$. Let r_{1}^{max} and r_{1}^{min} be the maximum and minimum values in R. We then divide the difference $r_{1}^{max} - r_{1}^{min}$ in N_1 classes and store (i,j) belonging to the first classes 1 to p_1 in $\mathbf{R_1}$. We then set $\hat{y}_{i,j}^{\nu} \leftarrow 1$, $\forall (i,j) \in \mathbf{R_1}$ and $\hat{y}_{ij}^{\nu} \leftarrow 0$, $\forall (i,j) \notin \mathbf{R_1}$. Consequently, the solution $\hat{y}_{ij}^{\nu}$ $\forall (i,j) \in \mathscr{A}$, in each iteration ν, is created by exploiting the reduced cost information associated with multiple solutions obtained by all considered groups.

We note that the reference point created in the previous section may not be used as an initial solution for the Learn&Optimize procedure, because $\tilde{y}_{ij}^{\nu}$, $\forall (i,j), \in \mathscr{A}$ is not a

$\{0,1\}$ design solution. However, the solution $\hat{y}^{\nu}_{ij}$, $\forall(i,j) \in \mathscr{A}$ we created in this section is an actual $\{0,1\}$ design solution.

5.4.5 The algorithm

Algorithm 10 sums up the entire procedure which consists of two main phases (similar to Crainic et al. [34]). In initialization phase, we solve the multi-scenario sub problems $SP^0_g, \forall g \in G$, where the original fixed cost is considered in the objective function. In phase I, the multi-scenario subproblems are solved approximately as explained in Section 5.4.3. In each aggregation step, $\tilde{y}$ and $\hat{y}$ are constructed and accordingly the Lagrangian multipliers are updated as described in Section 5.4.4. To produce a global feasible solution, $y^{M\nu}_{ij}$, $\forall(i,j) \in \mathscr{A}$, at each iteration of PHA, we follow the strategy used in [29, 34] to construct a heuristic feasible network $y^{M\nu}$ at each iteration ν, by setting the design variables $y^{M\nu}_{ij}$, $\forall(i,j) \in \mathscr{A}$ to

$$y^{M\nu}_{ij} = \begin{cases} 1, \text{ if } y^{g\nu}_{ij} = 1, \text{ for any } g \in G, \\ 0, \text{ otherwise .} \end{cases} \tag{5.26}$$

The best network found, i.e., y^{Best}, is updated based on the quality (total cost) of the feasible solution $y^{M\nu}$ obtained at each iteration ν. We use similar stopping criteria (in line 9) as those in Crainic et al. [34]. Namely, we stop after a total of N_{Itr} iterations, N_{Imp} consecutive iterations without improving the best known solution, t_{max} CPU time, or when there are fewer than γ $(0 \leq \gamma \leq 1)$ percent of the arcs for which a consensus has not been reached. When such a situation occurs, *phase II* is used to resolve it. In phase II, we fix the design variables for which a consensus is obtained and solve the original problem to obtain the final design solution y^{Final}. In line 28, we update y^{best}, if needed.

5.5 Experimental results

This section presents results of extensive computational experiments performed to assess the performance of the proposed algorithm. We used two collections of instances

Algorithm 10 The proposed integrated learning and progressive hedging method

```
1: Initialization
2: Let $\nu \leftarrow 0$ $\lambda_{ij}^{g\nu} \leftarrow 0, \forall (i,j) \in \mathscr{A}$, $\rho^{\nu} \leftarrow \rho^{0}$
3: Construct the list of scenario groups $\bar{C} = \{C_1, \ldots, C_{|G|}\}$
4: for each group $g$ do
5:     Solve $SP_g^0$ heuristically by performing Algorithm 8 in Section 5.4.3
6: end for
7: Construct solutions $y_{ij}^{M\nu}$, $\tilde{y}_{ij}^{\nu}$, and $\hat{y}_{ij}^{\nu}$ as stated in lines 18-21
8: Phase I: Seek consensus on the arcs (i,j) that should exist in the design
9: while stopping criteria not met do
10:     Iteration update:
11:     $\nu \leftarrow \nu + 1$
12:     Solving subproblems heuristically:
13:     for each group $g$ do
14:         Solve $SP_g^{\nu}$ heuristically by performing Algorithm 8 in Section 5.4.3, considering Lagrangian multipliers $\lambda_{ij}^{g\nu-1}, \forall (i,j) \in \mathscr{A}$ and solution $\hat{y}_{ij}^{\nu-1}, \forall (i,j) \in \mathscr{A}$, to obtain $\mathbf{f}_{ij}^{g\nu}$ and $y_{ij}^{g\nu} \forall (i,j) \in \mathscr{A}$
15:     end for
16:     Aggregation:
17:     Construct solution $y_{ij}^{M\nu}, \forall (i,j) \in \mathscr{A}$ according to (5.26)
18:     Update the best feasible solution $y^{Best} \leftarrow y^{M\nu}$ , if appropriate;
19:     Let $\tilde{y}_{ij}^{\nu} \leftarrow \sum_{g \in G} p^g \mathbf{f}_{ij}^{g\nu}$, $\forall (i,j) \in \mathscr{A}$ where $p^g = \sum_{s \in C_g} p^s$
20:     Update solution $\hat{y}_{ij}^{\nu}, \forall (i,j) \in \mathscr{A}$ as described in Section 5.4.4.2
21:     Penalty updates:
22:     Adjust penalty values $\lambda_{ij}^{g\nu} \leftarrow \lambda_{ij}^{g\nu-1} + \rho(\mathbf{f}_{ij}^{g\nu} - \tilde{y}_{ij}^{\nu})$ and $\rho^{\nu} \leftarrow \alpha\rho^{\nu-1}$
23: end while
24: Phase II: Solve a restriction as a MIP problem
25: Fix the design variables for which consensus is obtained in MCFND(S) (5.1)-(5.5)
26: Solve the restricted MCFND(S) model (5.1)-(5.5) to obtain a final design $y^{final}$
27: Update best solution, $y^{Best} \leftarrow y^{final}$ if appropriate.
```

which are described in Section 5.5.1. To evaluate the performance of the proposed algorithm (ILPH), we also considered alternative approaches to be tested on the same instances and performed the following algorithms:

- IBM-ILOG CPLEX 12.6.1 with its default settings (CPLEX in the following) on the MILP associated with an instance; and

- The basic progressive hedging with single scenario subproblem (PH-S in the following)

After presenting the two collections of instances, we start by analyzing the results obtained on the first collection of instances containing a smaller number of scenarios. We compare the performance of the proposed method with that of CPLEX in terms of optimality gap and computational time on these easier instances in Section 5.5.2. To assess the power of the proposed algorithm in dealing with a large number of scenarios, we provide a performance comparison of the proposed ILPH versus PH-S and CPLEX on the second collection of instances in Section 5.5.3.

5.5.1 Data and experimental setting

We consider five problem classes (R5-R9) from the set of R instances of the stochastic MCFND problem introduced in Crainic et al. [29]. Each class is characterized by a number of nodes $|\mathscr{N}|$, number of arcs $|\mathscr{A}|$ and number of commodities $|\mathscr{K}|$, specified in Table 5.5.1. For each instance class, we consider five networks (namely, networks 1, 3, 5, 7, and 9) which indicate continuously increasing ratios of fixed to variable costs and total demand to total network capacity. In the first collection of instances, for each of these networks, there are instances with 16, 32 and 64 scenarios with two different levels of correlations 0.2 and 0.8. A total of 150 instances were thus obtained. We followed the strategy proposed in Crainic et al. [34] to generate the groups of scenarios randomly. To do so, we randomly determine the number of groups between $|\mathscr{S}|/2$ and $|\mathscr{S}|/4$ and then randomly assign scenarios to groups. In order to have instances with a larger number of scenarios, we followed the procedure in Boland et al. [18]. In the second collection of instances, for each of the networks, there are 10 instances with 1000 scenarios and two levels of correlations 0.2 and 0.8 which are generated as follows. For each commodity and for each network, the minimum and maximum demand are determined over all scenarios considered in Crainic et al. [29]. Then, the demand is generated for the commodity in each of the $|\mathscr{S}|$ scenarios by drawing uniformly randomly from the interval determined by the minimum and maximum demand. For grouping the scenarios in the instances with 1000 scenarios, we considered 100 scenarios for each subproblem. Therefore, we determined 10 subproblems at each iteration of the proposed algorithm, where the scenarios are chosen randomly within each group.

For the implementation, algorithms were coded in C++ using IBM-ILOG CPLEX 12.6.1 as the MILP solver. We used similar stopping criteria for both of PH-S and ILPH, as presented in Crainic et al. [29]: $N_{Itr} = 1000$, $t_{max} = 8h$, and $\gamma = .1$. The parameter N_{Imp} is set to 10 and 4 for PH-S and ILPH, respectively. A preliminary experiment was conducted on the proposed ILPH to fine tune the τ, N_1, and p_1 parameters to the values .95, 3, and 2, respectively. We also set $N_{\Delta^g} = |\mathscr{K}| * |C_g|$. We let $\hat{y}^0 \leftarrow y^{exp}$ be the initial integer solution at iteration 0 in the Learn&Optimize procedure, where y^{exp} is the optimal solution to the expected value (EV) problem. The EV problem is obtained by replacing the random demand variable by their expected values and solving the deterministic problem. These settings generally worked well on our test problems. To reduce the time required to complete phase I, the optimality tolerance parameter of CPLEX was set to 1 percent when solving the sub-problems. This parameter is set to its default value when solving the restricted problem of the second phase. Unless otherwise specified, all other CPLEX parameters were set to their default values since preliminary experiments indicated that these settings yielded better results. All 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.

Table 5.5.1: Characteristics of instances

Problem	$\vert\mathscr{N}\vert$	$\vert\mathscr{A}\vert$	$\vert\mathscr{K}\vert$
R05	10	60	25
R06	10	60	50
R07	10	82	10
R08	10	83	25
R09	10	83	50

5.5.2 Performance comparison on the first collection of instances

In the first part of experiments, we focus on the first collection of instances (with 16, 32 and 64 scenarios) for which CPLEX is able to provide either the optimal solution or at least a feasible solution for all of them within the time limit of 8 hours. However, this is not the case in the second collection of instances. Table 5.5.2 reports the performance of the proposed ILPH versus CPLEX, in terms of optimality gap and total computational time. The *Gap* and *Time* values reported for CPLEX refer, respectively, to the optimal gap, and the total computation time expressed in seconds. For ILPH, *Gap* and *Time* represent the corresponding optimality gap relative to the lower bound of CPLEX, and the total computation time in seconds, respectively. We observed that the proposed ILPH with an optimality gap of 1.18% performs a little better than CPLEX with an optimality gap of 1.21% on average. However, ILPH is more than 10 times faster than CPLEX on this collection of instances.

The results above are encouraging and demonstrate the potential of ILPH, but the instances are inadequate to fully reveal the power of ILPH. In this collection of instances, most of the instances can be solved to optimality by CPLEX in less than 2 hours. This is not the setting for which a decomposition approach is designed. The ILPH is designed to be used in settings where the instances are large, difficult, and cannot be solved in a reasonable amount of time when providing the MILP formulation to a solver. Indeed, solving the root relaxation may already be computationally prohibitive. In the following subsection, we present results on instances that are (somewhat) more appropriate to show the benefits of ILPH.

Table 5.5.2: Performance comparison versus CPLEX on first collection of instances

Pro	Ins	CPLEX		ILPH	
		Gap	Time	Gap	Time
R05	30	0.00%	1411	0.06%	118
R06	30	1.51%	11076	1.46%	951
R07	30	0.12%	2130	0.84%	105
R08	30	0.96%	7516	1.24%	841
R09	30	3.48%	16415	2.20%	840
Avg		1.21%	7709.6	1.18%	571.3

5.5.3 Performance comparison on the second collection of instances

In the following, we focus on the much more difficult instances with 1000 scenarios. We provide the comparative results of the proposed ILPH versus CPLEX (in Tables 5.5.3

and 5.5.4) and PH-S (in Table 5.5.5) to show the advantage of the proposed method in dealing with such difficult instances. Each row of these tables refers to 10 instances with 1000 scenarios with different characteristics mentioned in Section 5.5.1.

In Table 5.5.3, we first report the average optimality gap, *OptGap*, that CPLEX is able to provide after a time limit of 8 hours. Note that we consider the optimality gap of 100% for those instances for which CPLEX is not able to provide any feasible solution. In column *ILPH*/*CPLEX*, we report the percentage of relative difference between the best solutions found by the two algorithms (i.e., z_{ILPH} and z_{CPX}) by imposing a 8 hour time limit on the two solution methods. The relative difference is computed as $(z_{ILPH} - z_{CPX})/z_{ILPH}$. The negative values refer to the cases where ILPH provides better solutions than CPLEX. In the last columns, we compare the two solution methods based on the percentage of instances for which the considered solution method is able to provide the optimal solution (*Opt*.) and a feasible solution (*Sol*.).

The results in Table 5.5.3 show that ILPH outperforms CPLEX on these difficult instances where the average optimality gap provided by CPLEX is 33.93%. In terms of the percentage of instances with a feasible solution, we observed that ILPH is able to provide a feasible solution in all instances. However, CPLEX is not able to find any feasible solution in 20 percent of instances after 8 hours of computation time, thus indicating the difficulty of these instances. Moreover, ILPH performs as well as CPLEX in terms of the percentage of instances (24%) for which an optimal solution is found. In terms of solution quality, the results indicate that ILPH is able to provide an impressive improvement of -26.6% over CPLEX with a time limit of 8 hours.

Table 5.5.3: Performance comparison on second collection of instances versus CPLEX

Pro	Ins	OptGap(%)	ILPH/CPLEX(%)	ILPH		CPLEX	
		CPLEX	8 h	Opt.	Sol.	Opt.	Sol.
R05	10	17.94%	-11.89%	40%	100%	40%	100%
R06	10	47.87%	-41.38%	40%	100%	40%	100%
R07	10	9.70%	-2.51%	20%	100%	20%	100%
R08	10	27.72%	-15.34%	20%	100%	20%	100%
R09	10	66.37%	-61.89%	0%	100%	0%	40%
Avg		33.93%	-26.60%	24%	100%	24%	80%

In order to show the power of the proposed ILPH in finding good solutions quickly, we report the comparative results of ILPH versus CPLEX in a shorter amount of time for these difficult instances. Table 5.5.4 displays the results obtained by the two methods with 3 hours of time limit. The table reports the same information as Table 5.5.3, but for a time limit of 3 h. The average optimality gap provided by CPLEX is 45.59%, indicating the difficulty of these problems. We observed that ILPH is able to find a feasible solution in all considered instances, while CPLEX is not able to do so in 32% of instances. In terms of the percentage of instances for which the two algorithms can hit the optimal solution after 3 hours of time limit, ILPH performs 5 times better than CPLEX. Furthermore, ILPH provides an improvement, in solution quality of -39.23%, which is facinating.

Table 5.5.4: Efficiency of the ILPH versus CPLEX in finding good solution quickly

Pro	Ins	OptGap(%)	ILPH/CPLEX(%)	ILPH		CPLEX	
		CPLEX	3 h	Opt.	Sol.	Opt.	Sol.
R05	10	22.56%	-12.62%	40%	100%	0%	100%
R06	10	63.90%	-62.31%	20%	100%	20%	40%
R07	10	15.97%	-4.34%	20%	100%	0%	100%
R08	10	41.36%	-35.03%	20%	100%	0%	80%
R09	10	84.18%	-81.84%	0%	100%	0%	20%
Avg		45.59%	-39.23%	20%	100%	4%	68%

Table 5.5.5 displays the performance results of the proposed ILPH compared to PH-S, in terms of solution quality and computation time. In order to investigate how the proposed ILPH performs compared to PH-S, we consider the best solution provided after the first phase (i.e., y^{Best}) by ILPH and PH-S. Column "$ILPH/PH-S$" reports the relative improvement (in percentage), computed as $(z_{ILPH} - z_{PH-S})/z_{ILPH}$, after a time limit of 8 hours. The negative values indicate that ILPH outperforms PH-S. We also report the average computation time in seconds ($Time$) and the percentage of instances with a feasible solution ($Sol.$) for the two algorithms.

We observed that the proposed ILPH provides solutions with an improvement of -15.28% compared to PH-S in almost one third of computation time, on average. Furthermore, PH-S is not able to provide any solution in 8% of instances, while ILPH is able to provide a feasible solution in all considered instances.

Table 5.5.5: Performance comparison on second collection of instances versus PH-S

Pro	Ins	ILPH/PG-S	Time		Sol.	
		Phase1	ILPH	PH-S	ILPH	PH-S
R05	10	-8.45%	582	5325	100%	100%
R06	10	-10.14%	4121	19277	100%	100%
R07	10	-10.16%	1210	3576	100%	100%
R08	10	-5.14%	13755	14822	100%	100%
R09	10	-42.47%	7611	30613	100%	60%
Avg		-15.28%	5456	14728	100%	92%

5.6 Conclusions

This paper explores the development of an efficient optimization approach to address the large and complex stochastic MCFND problems. We proposed a two phase *integrated learning and PH* method as a matheuristic approach to handle a large number of scenarios in the considered context. We adapt the *Learn&Optimize* procedure to be used as an efficient heuristic method to address the multi-scenario subproblems at each iteration of the PHA. We exploited the knowledge learned through the *Learn&Optimize* and particularly introduced a new reference point in each aggregation step of PHA by exploiting the knowledge regarding the promising design variables which are built through the *Learn&Optimize* applied in the subproblems. In this way, we inject the knowledge learned through the heuristic procedure into the PHA leading to the proposed *ILPH* method, which considered as the main contribution in this paper.

Through computational experiments, we have shown that the proposed algorithm performs very well in terms of both solution quality and computation time. We have provided comparative analysis that show the superiority of the proposed approach versus CPLEX and the basic PHA (with single scenario subproblems). The results indicate that the proposed algorithm is able to provide impressive improvements of 26.6% and 15.28% (when dealing with large instances with 1000 scenarios) versus CPLEX and the basic PHA, respectively. The analysis also indicates that ILPH should be the method of choice where high-quality solutions to very complex instances of stochastic network problems containing a large number of scenarios need to be found quickly.

We conclude by providing a few possible directions for future research. This could include investigating whether the algorithm would be as successful or not in solving different optimization problems or other variants of MCFND as it is in solving the current version considered in this paper. Indeed, it is a general-purpose algorithm and can be applicable to different stochastic programs. Tailored implementations of the proposed PHA can lead to quite effective ad-hoc heuristics for very large stochastic programming applications. Other research avenues include considering other strategies for updating the penalties within the PHA and other methods for solving the sub-problems. Finally, another important research direction is to develop parallel PHA strategies, which will further amplify its advantages and benefits.

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.

CHAPTER 6

CONCLUSIONS

The present dissertation addressed the stochastic network design problem under demand uncertainty as a two-stage stochastic program. In such setting, design decisions are made in the first stage before the actual demand is realized, while second-stage flow-routing decisions adjust the first-stage solution to the observed demand realization. The main goal of the stochastic network design formulations is to find a single optimal design solution for the range of possible demand realizations. To represent the uncertainty, we used the well-known scenario approach, where stochastic demands are modeled via a finite number of discrete scenarios together with associated probabilities. This leads to a very large scale mixed integer program which is extremely hard, even without the presence of integrality requirements, to be handled with exact methods and state-of-the-art solvers. Therefore, there is an apparent need to propose and develop heuristic solution methodologies to solve such large stochastic models, efficiently.

In summary, concerning the subject and the scope of the thesis, we have sought to better understand the solution structure of stochastic network design problems to enrich the current research literature by contributing in the solution methodologies.

In Chapter 3, we introduced a learning-based matheuristic for the stochastic fixed charge network design problem. The innovative learning mechanism systematically explores combinations of scenarios so as to extract relevant information regarding the solution structure of the stochastic problem. Using these mechanisms, scenarios and their influences on design decisions were successively considered through the algorithm. In fact, a global image of the promising structure of the stochastic solution is built by gradually learning from the partial knowledge produced by the learning mechanism supporting the collection and use of the memory. This is indeed the main novelty of the proposed approach in dealing with uncertainty. The proposed *Learn&Optimize* matheuristic iteratively exploited the obtained knowledge of learning heuristic to define a reduced size problem to be solved by a MIP solver.

We presented the results of extensive computational experiment using the proposed matheuristic and compared the latter with solutions produced by CPLEX. The results show that the proposed algorithm is highly effective at finding good-quality solutions on the largest available subset of instances for stochastic network design problems.

In Chapter 4, we investigated how to efficiently design learning mechanisms based on dual information as a means of guiding variable fixing within the context of stochastic network design. We looked at 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 selection within stochastic formulations. We particularly explored different strategies to determine the desirable set of variables to be fixed using reduced cost information extracted from a solution obtained by a deterministic expected value problem. The purpose of the proposed strategies was to identify appropriate subsets of design variables (using reduced cost information) to be fixed to open or closed in the stochastic problem, as a restriction, and obtain a good quality solution. We considered two main factors within each strategy, including the choice of solution from which we extract the reduced costs, and the choice of variables (i.e. design variables or flow variables). Each of these strategies utilized a single solution to learn from the associated reduced cost values (corresponding to design or flow variables) so as to create an appropriate set of fixed variables.

An analysis of the proposed strategies showed that the variable fixing process could be significantly improved if the EV solution was reconstructed, upgraded, and if its associated reduced cost information was extracted afterwards. Subsequently, a matheuristic approach was proposed which iteratively defined restricted problems constructed by exploiting reduced cost information extracted from multiple solutions. The main novelty of the proposed approach is its use of primal and dual information to define the restricted problems in a more effective manner. The results show that the proposed algorithm is highly efficient at finding good-quality solutions and even outperforms our proposed method in the first study for very large instances of stochastic network design problems.

In Chapter 5, an efficient solution method was designed which was intended to effectively manage a large numbers of scenarios. We proposed a two-phase solution approach, based on the progressive hedging algorithm of Rockafellar and Wets [103]. In phase I, the PHA was used: the problem is first decomposed by partitioning the set of scenarios into groups, and then the sub-problems associated with each group are addressed iteratively to guide their solutions to a consensus solution. To deal with difficulties in solving multi-scenario subproblems, the *Learn&Optimize* procedure, as an efficient heuristic method, was adapted to address the multi-scenario subproblems at each iteration of ILPH. We also introduced a new reference point within each aggregation step of our PHA (as opposed to of the weighted average solution) by exploiting the information garnered from subproblems, and using this information to update the PHA penalties. In this way, we inject the knowledge learned through the heuristic procedure into the PHA, resulting in the proposed *ILPH* method. In phase II, a reduced size problem is constructed by fixing the design variables for which a consensus is obtained, and the resulting smaller problem is solved to generate the final solution. We showed, by means of extensive comparative analysis, the superiority of the proposed ILPH algorithm as compared against CPLEX and the classical PHA. We are continuing to work on this research work to apply different strategies, in choosing the set of (similar or dissimilar) scenarios in subproblems, like those proposed in [34], to enhance the obtained results. However, the random strategy applied in this dissertation still provides a good performance which underlines the worthiness of a general methodology when a very simple random strategy is involved.

6.1 Future research directions

We conclude this chapter by highlighting some research perspectives.

One possible direction is the continued study of the learning based matheuristic approach which iteratively uses a MIP commercial solver as a subroutine for handling subproblems. In designing such approaches, the key question that arises is how reduced size problems should be constructed by designing effective learning mechanisms. While providing an all-inclusive rule for the learning mechanism does not seem to be a feasible approach, some interesting guidelines emerge from the analysis of solution structures of stochastic problems. Moreover, the nature of the methods proposed in this dissertation also suggests straightforward parallelization strategies that, despite their simplicity, can lead to an attractive reduction in computation time and improvement of the accuracy level.

Apart from the classical network design model addressed in this dissertation, there are still many potential areas to adapt the proposed matheuristic approaches for different variants of this model that have not been addressed so far, like service network design models. It would be worthwhile to develop various metaheuristic and matheuristic approaches which incorporate different learning and memorizing mechanisms to handle such large-scale problems.

The frontiers of stochastic network design research are increasingly dependent on sophisticated modeling approaches. In scenario-based stochastic programs for ND problems, two stage modeling approaches were widely applied due to their relative simplicity. However, time staging information is needed as information arrives over time. Thus, developing multi-stage stochastic programs and developing efficient solution approaches for them will be welcomed by researchers and practitioners.

Another key question for scenario-based stochastic programs is how to generate an efficient set of scenarios to model the underlying stochasticity in SND. Although much literature has studied scenario generation and reduction in the stochastic programing community [41, 70], there is still much space to explore these approaches in this research area. More importantly, evaluating scenario generation methods in terms of stability and quality criteria should be examined in SND problems as well.

Finally, the last conclusion to be drawn from this dissertation is that, while there are several research studies for ND problems under uncertainty, there are only a few papers to cope with real world situations. Possible reasons for the lack of application papers include : (1) preparation and aggregation of data are rather time-consuming to model comprehensive SND problems, and (2) in many real cases there is not enough historical data for the uncertain parameters. Thus, this research area still needs more studies exploring realistic models based on real-world applications and handling computational aspects to solve large-sized problems.

BIBLIOGRAPHY

[1] Ahmadi-Javid, Amir, Amir Hossein Seddighi. 2013. A location-routing problem with disruption risk. *Transportation Research Part E: Logistics and Transportation Review* **53** 63–82.

[2] Ahuja, Ravindra K, Thomas L Magnanti, James B Orlin, et al. 1993. *Network flows: theory, algorithms, and applications*. Prentice hall Englewood Cliffs, NJ.

[3] Albareda-Sambola, Maria, Antonio Alonso-Ayuso, Laureano F Escudero, Elena Fernández, Celeste Pizarro. 2013. Fix-and-relax-coordination for a multi-period location–allocation problem under uncertainty. *Computers & Operations Research* **40**(12) 2878–2892.

[4] Alonso-Ayuso, Antonio, Laureano F Escudero, Araceli Garín, M Teresa Ortuño, Gloria Pérez. 2003. An approach for strategic supply chain planning under uncertainty based on stochastic 0-1 programming. *Journal of Global Optimization* **26**(1) 97–124.

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

[6] 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.

[7] Armacost, Andrew P, Cynthia Barnhart, Keith A Ware. 2002. Composite variable formulations for express shipment service network design. *Transportation Science* **36**(1) 1–20.

[8] Ayvaz, Berk, Bersam Bolat, Nezir Aydın. 2015. Stochastic reverse logistics network design for waste of electrical and electronic equipment. *Resources, conservation and recycling* **104** 391–404.

[9] Azaron, A, KN Brown, SA Tarim, M Modarres. 2008. A multi-objective stochastic programming approach for supply chain design considering risk. *International Journal of Production Economics* **116**(1) 129–138.

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

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

[12] Benyoucef, Lyes, Xiaolan Xie, Guy Aime Tanonkou. 2013. Supply chain network design with unreliable suppliers: a lagrangian relaxation-based approach. *International Journal of Production Research* **51**(21) 6435–6454.

[13] Beraldi, Patrizia, Francesco De Simone, Antonio Violi. 2010. Generating scenario trees: A parallel integrated simulation–optimization approach. *Journal of Computational and Applied Mathematics* **233**(9) 2322–2331.

[14] Bidhandi, Hadi Mohammadi, Rosnah Mohd Yusuff. 2011. Integrated supply chain planning under uncertainty using an improved stochastic approach. *Applied Mathematical Modelling* **35**(6) 2618–2630.

[15] Billingsley, Patrick. 2008. *Probability and measure*. John Wiley & Sons.

[16] Binato, Silvio, Mário Veiga F Pereira, Sérgio Granville. 2001. A new benders decomposition approach to solve power transmission network design problems. *IEEE Transactions on Power Systems* **16**(2) 235–240.

[17] Birge, John R, Francois Louveaux. 2011. *Introduction to stochastic programming*. Springer Science & Business Media.

[18] Boland, Natashia, Matteo Fischetti, Michele Monaci, Martin Savelsbergh. 2016. Proximity Benders: a decomposition heuristic for stochastic programs. *Journal of Heuristics* **22**(2) 181–198.

[19] Cario, Marne C, Barry L Nelson. 1997. Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix. Tech. rep., Northwestern University.

[20] Cavellucci, Celso, Christiano Lyra. 1997. Minimization of energy losses in electric power distribution systems by intelligent search strategies. *International Transactions in Operational Research* **4**(1) 23–33.

[21] Charnes, Abraham, William W Cooper. 1959. Chance-constrained programming. *Management Science* **6**(1) 73–79.

[22] Chouinard, Marc, Sophie D´Amours, Daoud Aït-Kadi. 2008. A stochastic programming approach for designing supply loops. *International Journal of Production Economics* **113**(2) 657–677.

[23] Chouman, Mervat, Teodor Gabriel Crainic. 2014. Cutting-plane matheuristic for service network design with design-balanced requirements. *Transportation Science* **49**(1) 99–113.

[24] Chouman, Mervat, Teodor Gabriel Crainic, Bernard Gendron. 2016. Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design. *Transportation Science* **51**(2) 650–667.

[25] Crainic, Teodor G, Francesca Maggioni, Guido Perboli, Walter Rei. 2017. Reduced cost-based variable fixing in two-stage stochastic programming. *Annals of Operations Research* doi:10.1007/s10479-018-2942-8.

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

[27] Crainic, Teodor Gabriel. 2005. Parallel computation, co-operation, tabu search. *Metaheuristic Optimization via Memory and Evolution*. Springer, 283–302.

[28] Crainic, Teodor Gabriel, Fausto Errico, Walter Rei, Nicoletta Ricciardi. 2015. Modeling demand uncertainty in two-tier city logistics tactical planning. *Transportation Science* **50**(2) 559–578.

[29] 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.

[30] Crainic, Teodor Gabriel, Michel Gendreau. 2007. A scatter search heuristic for the fixed-charge capacitated network design problem. *Metaheuristics*. Springer, 25–40.

[31] Crainic, Teodor Gabriel, Michel Gendreau, Judith M. Farvolden. 2000. A simplex-based tabu search method for capacitated network design. *INFORMS Journal on Computing* **12**(3) 223–236.

[32] Crainic, Teodor Gabriel, Mike Hewitt, Francesca Maggioni, Walter 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.

[33] Crainic, Teodor Gabriel, Mike Hewitt, Walter Rei. 2014. 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.

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

[35] Da Silva, Marcos Carneiro, Paulo Morelato França, Paulo D Bishop Da Silveira. 1996. Long-range planning of power distribution systems: secondary networks. *Computers & Electrical Engineering* **22**(3) 179–191.

[36] Daskin, Mark S, Lawrence V Snyder, Rosemary T Berger. 2005. Facility location in supply chain design. *Logistics systems: Design and optimization*. Springer, 39–65.

[37] 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.

[38] Defourny, Boris, Damien Ernst, Louis Wehenkel. 2013. Scenario trees and policy selection for multistage stochastic programming using machine learning. *INFORMS Journal on Computing* **25**(3) 488–501.

[39] Dubey, Rameshwar, Angappa Gunasekaran, Stephen J Childe. 2015. The design of a responsive sustainable supply chain network under uncertainty. *The International Journal of Advanced Manufacturing Technology* **80**(1-4) 427–445.

[40] Dupačová, Jitka. 1995. Multistage stochastic programs: The state-of-the-art and selected bibliography. *Kybernetika* **31**(2) 151–174.

[41] Dupačová, Jitka, Giorgio Consigli, Stein W Wallace. 2000. Scenarios for multistage stochastic programs. *Annals of Operations Research* **100**(1-4) 25–53.

[42] Dupačová, Jitka, Nicole Gröwe-Kuska, Werner Römisch. 2003. Scenario reduction in stochastic programming. *Mathematical Programming* **95**(3) 493–511.

[43] Escudero, Laureano F, María Araceli Garín, María Merino, Gloria Pérez. 2012. An algorithmic framework for solving large-scale multistage stochastic mixed 0–1 problems with nonsymmetric scenario trees. *Computers & Operations Research* **39**(5) 1133–1144.

[44] Fattahi, Mohammad, Kannan Govindan. 2017. Integrated forward/reverse logistics network design under uncertainty with pricing for collection of used products. *Annals of Operations Research* **253**(1) 193–225.

[45] Fischetti, Matteo, Andrea Lodi. 2003. Local branching. *Mathematical programming* **98**(1-3) 23–47.

[46] Gaivoronski, Alexei A. 2006. Stochastic optimization in telecommunications. *Handbook of Optimization in Telecommunications*. Springer, 761–799.

[47] Gascon, Viviane, Abdelhamid Benchakroun, Jacques A Ferland. 1993. Electricity distribution planning model: A network design approach for solving the master problem of the Benders decomposition method. *INFOR: Information Systems and Operational Research* **31**(3) 205–220.

[48] Gavish, Bezalel. 1983. Formulations and algorithms for the capacitated minimal directed tree problem. *Journal of the ACM* **30**(1) 118–132.

[49] Gendron, Bernard, Teodor Gabriel Crainic. 1994. Relaxations for multicommodity capacitated network design problems. Tech. Rep. CRT-965, Université de Montréal, Centre de recherche sur les transports.

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

[51] Ghamlouche, Ilfat, Teodor Gabriel Crainic, Michel Gendreau. 2004. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. *Annals of Operations Research* **131**(1-4) 109–133.

[52] Giarola, Sara, Nilay Shah, Fabrizio Bezzo. 2012. A comprehensive approach to the design of ethanol supply chains including carbon trading effects. *Bioresource Technology* **107** 175–185.

[53] Glover, Fred. 1997. Tabu search and adaptive memory programming–advances, applications and challenges. *Interfaces in Computer Science and Operations Research*. Springer, 1–75.

[54] Goh, Mark, Joseph YS Lim, Fanwen Meng. 2007. A stochastic model for risk management in global supply chain networks. *European Journal of Operational Research* **182**(1) 164–173.

[55] Gouveia, Luis. 1995. A 2n constraint formulation for the capacitated minimal spanning tree problem. *Operations Research* **43**(1) 130–141.

[56] Govindan, Kannan, Mohammad Fattahi. 2017. Investigating risk and robustness measures for supply chain network design under demand uncertainty: A case study of glass supply chain. *International Journal of Production Economics* **183** 680–699.

[57] Guillén-Gosálbez, Gonzalo, Ignacio E Grossmann. 2009. Optimal design and planning of sustainable chemical supply chains under uncertainty. *AIChE Journal* **55**(1) 99–121.

[58] Hatefi, Seyed Morteza, Fariborz Jolai, S Ali Torabi, Reza Tavakkoli-Moghaddam. 2015. A credibility-constrained programming for reliable forward–reverse logistics network design under uncertainty and facility disruptions. *International Journal of Computer Integrated Manufacturing* **28**(6) 664–678.

[59] Haugen, Kjetil K, Arne Løkketangen, David L Woodruff. 2001. Progressive hedging as a meta-heuristic applied to stochastic lot-sizing. *European Journal of Operational Research* **132**(1) 116–122.

[60] Heitsch, Holger, Werner Römisch. 2003. Scenario reduction algorithms in stochastic programming. *Computational Optimization and Applications* **24**(2-3) 187–206.

[61] Hewitt, Mike, George L Nemhauser, Martin WP Savelsbergh. 2010. Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS Journal on Computing* **22**(2) 314–325.

[62] Higle, Julia L, Suvrajeet Sen. 1991. Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of Operations Research* **16**(3) 650–669.

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

[64] Høyland, Kjetil, Michal Kaut, Stein W Wallace. 2003. A heuristic for moment-matching scenario generation. *Computational optimization and applications* **24**(2-3) 169–185.

[65] Høyland, Kjetil, Stein W Wallace. 2001. Generating scenario trees for multistage decision problems. *Management Science* **47**(2) 295–307.

[66] Infanger, Gerd. 1992. Monte Carlo (importance) sampling within a Benders decomposition algorithm for stochastic linear programs. *Annals of Operations Research* **39**(1) 69–95.

[67] Kall, Peter, Stein W Wallace, Peter Kall. 1994. *Stochastic programming*. Springer.

[68] Kaut, Michal, Hercules Vladimirou, Stein W Wallace, Stavros A Zenios. 2007. Stability analysis of portfolio management with conditional value-at-risk. *Quantitative Finance* **7**(4) 397–409.

[69] Kim, Jinkyung, Matthew J Realff, Jay H Lee. 2011. Optimal design and global sensitivity analysis of biomass supply chain networks for biofuels under uncertainty. *Computers & Chemical Engineering* **35**(9) 1738–1751.

[70] King, Alan J, Stein W Wallace. 2012. *Modeling with stochastic programming*. Springer Science & Business Media.

[71] Klibi, Walid, Alain Martel. 2012. Modeling approaches for the design of resilient supply networks under disruptions. *International Journal of Production Economics* **135**(2) 882–898.

[72] Klibi, Walid, Alain Martel. 2012. Scenario-based supply chain network risk modeling. *European Journal of Operational Research* **223**(3) 644–658.

[73] Klibi, Walid, Alain Martel, Adel Guitouni. 2010. The design of robust value-creating supply chain networks: a critical review. *European Journal of Operational Research* **203**(2) 283–293.

[74] Lanza, G., T.G Crainic, W. Rei, N. Ricciardi. 2017. Service Network Design Problem with Quality Targets and Stochastic Travel Time. Publication CIRRELT-2017, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada.

[75] Lederer, Phillip J, Ramakrishnan S Nambimadom. 1998. Airline network design. *Operations Research* **46**(6) 785–804.

[76] 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.

[77] Løkketangen, Arne, David L Woodruff. 1996. Progressive hedging and tabu search applied to mixed integer (0, 1) multistage stochastic programming. *Journal of Heuristics* **2**(2) 111–128.

[78] Lurie, Philip M, Matthew S Goldberg. 1998. An approximate method for sampling correlated random variables from partially-specified distributions. *Management Science* **44**(2) 203–218.

[79] Lyhagen, Johan. 2001. A method to generate multivariate data with moments arbitrary close to the desired moments. Tech. rep., SSE/EFI Working Paper Series in Economics and Finance.

[80] 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.

[81] Magnanti, Thomas L, P Mireault, Richard T Wong. 1985. Tailoring Benders decomposition for uncapacitated network design. *Mathematical Programming Study* **25** 112–154.

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

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

[84] Mulvey, John M, Robert J Vanderbei, Stavros A Zenios. 1995. Robust optimization of large-scale systems. *Operations Research* **43**(2) 264–281.

[85] Nickel, Stefan, Francisco Saldanha-da Gama, Hans-Peter Ziegler. 2012. A multi-stage stochastic supply network design problem with financial decisions and risk management. *Omega* **40**(5) 511–524.

[86] Norkin, Vladimir I, Yuri M Ermoliev, Andrzej Ruszczyński. 1998. On optimal allocation of indivisibles under uncertainty. *Operations Research* **46**(3) 381–395.

[87] Norkin, Vladimir I, Georg Ch Pflug, Andrzej Ruszczyński. 1998. A branch and bound method for stochastic global optimization. *Mathematical Programming* **83**(1-3) 425–450.

[88] Park, Sukun, Tae-Eog Lee, Chang Sup Sung. 2010. A three-level supply chain network design model with risk-pooling and lead times. *Transportation Research Part E: Logistics and Transportation Review* **46**(5) 563–581.

[89] Peterson, Bruce E. 1980. A cut-flow procedure for transportation network optimization. *Networks* **10**(1) 33–43.

[90] Pimentel, Bruno S, Geraldo R Mateus, Franklin A Almeida. 2013. Stochastic capacity planning and dynamic network design. *International Journal of Production Economics* **145**(1) 139–149.

[91] Pishvaee, Mir Saman, Masoud Rabbani, Seyed Ali Torabi. 2011. A robust optimization approach to closed-loop supply chain network design under uncertainty. *Applied Mathematical Modelling* **35**(2) 637–649.

[92] 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.

[93] 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.

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

[95] Ramezani, Majid, Mahdi Bashiri, Reza Tavakkoli-Moghaddam. 2013. A robust design for a closed-loop supply chain network under an uncertain environment. *The International Journal of Advanced Manufacturing Technology* **66**(5-8) 825–843.

[96] Randazzo, CD, Henrique Pacca Loureiro Luna. 2001. A comparison of optimal methods for local access uncapacitated network design. *Annals of Operations Research* **106**(1-4) 263–286.

[97] Randazzo, CD, Henrique Pacca Loureiro Luna, Philippe Mahey. 2001. Benders decomposition for local access network design with two technologies. *Discrete Mathematics and Theoretical Computer Science* **4**(2).

[98] Rappold, James A, Ben D Van Roo. 2009. Designing multi-echelon service parts networks with finite repair capacity. *European Journal of Operational Research* **199**(3) 781–792.

[99] Realff, Matthew J, Jane C Ammons, David J Newton. 2004. Robust reverse production system design for carpet recycling. *IIE Transactions* **36**(8) 767–776.

[100] Resende, Mauricio GC, Celso C Ribeiro, Fred Glover, Rafael Martí. 2010. Scatter search and path-relinking: Fundamentals, advances, and applications. *Handbook of metaheuristics*. Springer, 87–107.

[101] Riis, Morten, Kim Allan Andersen. 2002. Capacitated network design with uncertain demand. *INFORMS Journal on Computing* **14**(3) 247–260.

[102] Riis, Morton, Kim Allan. Andersen. 2000. Capacitated network design with uncertain demand. Tech. rep., Department of Operations Research University of Aarhus.

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

[104] Romero, R, A Monticelli. 1994. A hierarchical decomposition approach for transmission network expansion planning. *IEEE Transactions on Power Systems* **9**(1) 373–380.

[105] Sabri, Ehap H, Benita M Beamon. 2000. A multi-objective approach to simultaneous strategic and operational planning in supply chain design. *Omega* **28**(5) 581–598.

[106] Sahinidis, Nikolaos V. 2004. Optimization under uncertainty: state-of-the-art and opportunities. *Computers & Chemical Engineering* **28**(6-7) 971–983.

[107] Santoso, Tjendera, Shabbir Ahmed, Marc Goetschalckx, Alexander Shapiro. 2005. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research* **167**(1) 96–115.

[108] 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.

[109] Sarayloo, Fatemeh, Teodor Gabriel Crainic, Walter Rei. 2018. A reduced cost-based restriction and refinement matheuristic for stochastic network design. Publication CIRRELT-2018-32, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada.

[110] Schütz, Peter, Asgeir Tomasgard, Shabbir Ahmed. 2009. Supply chain design under uncertainty using sample average approximation and dual decomposition. *European Journal of Operational Research* **199**(2) 409–419.

[111] Schütz, Peter, Asgeir Tomasgard, Shabbir Ahmed. 2009. Supply chain design under uncertainty using sample average approximation and dual decomposition. *European Journal of Operational Research* **199**(2) 409–419.

[112] Smith, J Cole, Andrew J Schaefer, Joyce W Yen. 2004. A stochastic integer programming approach to solving a synchronous optical network ring design problem. *Networks* **44**(1) 12–26.

[113] Smith, James E. 1993. Moment methods for decision analysis. *Management Science* **39**(3) 340–358.

[114] Snyder, Lawrence V, Mark S Daskin. 2006. Stochastic p-robust location problems. *IIE Transactions* **38**(11) 971–985.

[115] Tanonkou, Guy-Aimé, Lyès Benyoucef, Xiaolan Xie. 2008. Design of stochastic distribution networks using lagrangian relaxation. *IEEE Transactions on Automation Science and Engineering* **5**(4) 597–608.

[116] 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.

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

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

[119] Tsiakis, Panagiotis, Nilay Shah, Constantinos C Pantelides. 2001. Design of multi-echelon supply chain networks under demand uncertainty. *Industrial & Engineering Chemistry Research* **40**(16) 3585–3604.

[120] 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.

[121] Verweij, Bram, Shabbir Ahmed, Anton J Kleywegt, George Nemhauser, Alexander Shapiro. 2003. The sample average approximation method applied to stochastic routing problems: a computational study. *Computational Optimization and Applications* **24**(2-3) 289–333.

[122] Vu, Duc Minh, Teodor Gabriel Crainic, Michel Toulouse. 2013. A three-phase matheuristic for capacitated multi-commodity fixed-cost network design with design-balance constraints. *Journal of Heuristics* **19**(5) 757–795.

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

[124] Wang, X., T.G Crainic, S.W Wallace. 2016. 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.

[125] You, Fengqi, Ignacio E Grossmann. 2008. Design of responsive supply chains under demand uncertainty. *Computers & Chemical Engineering* **32**(12) 3090–3111.