Exact Algorithms for Network Fortification and Design Problems under Stochastic Interdiction

Shabnam Mahmoudzadeh Vaziri

A Thesis

in

The Department

of

Mechanical, Industrial, and Aerospace Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Industrial Engineering) at

Concordia University

Montréal, Québec, Canada

August 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Shabnam Mahmoudzadeh Vaziri**

Entitled: **Exact Algorithms for Network Fortification and Design Problems under Stochastic Interdiction**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Industrial Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Ciprian Alecsandru

_____ External Examiner
Dr. Cole Smith

_____ External to Program
Dr. Chun Wang

_____ Examiner
Dr. Claudio Contardo

_____ Examiner
Dr. Mingyuan Chen

_____ Supervisor
Dr. Onur Kuzgukaya

_____ Co-supervisor
Dr. Navneet Vidyarthi

Approved by _____
Dr. Muthukumaran Packirisamy
Graduate Program Director

_____ 2024

_____
Dr. Mourad Debbabi
Dean of Faculty of Engineering and Computer Science

Abstract

Exact Algorithms for Network Fortification and Design Problems under Stochastic Interdiction

Shabnam Mahmoudzadeh Vaziri, Ph.D.

Concordia University, 2024

Networks such as telecommunication, transportation, and logistics play a vital role in sustaining social, economic, and industrial operations. However, networks are at risk of natural and man-made disruptions. With the increasing interconnectivity of networks, a disruption in one network can negatively affect other networks. We use network interdiction models to analyze the effects of disruptions on the networks. Interdiction models represent a two-player sequential game between the interdictor, seeking to maximize damage, and the network operator, striving to optimize operations after interdiction. Network interdiction models identify the critical nodes and/or arcs of the network and can be extended to fortify the existing networks or design resilient networks. In this thesis, we study the design of distribution and multicommodity networks and the fortification of spanning trees under stochastic interdictions.

The first paper investigates the distribution network design problem considering the effect of interdictions. Since interdiction outcomes can be uncertain in the real world, we consider the interdiction outcome as an uncertain parameter. We extend the model to consider the correlated facility interdictions where interruptions at one facility affect the nearby facilities. We Benders decomposition algorithm. Improved by two acceleration techniques to solve the model.

In the second paper, we focus on the design of a multicommodity network with interdictions. The designer does not have information about the interdiction resources; therefore, we consider the uncertainty in the number of interdictions by presenting a tri-level stochastic mathematical model. We present a branch-and-Benders-cut (BBC) algorithm enhanced by several acceleration techniques to solve the model.

In the third paper, we study the fortification of minimum spanning tree (MST) and optimum communication spanning tree (OCST) problems under stochastic number of interdictions. The MST aims to connect all nodes in a graph with minimal installation cost while satisfying communication requirements with minimum communication cost. The goal is to find the optimal fortification strategy so that the increase in the MST/OCST costs due to the interdiction of unfortified edges is minimized by presenting a tri-level stochastic model. We use backward sampling framework with acceleration technique to solve the deterministic and stochastic MST/OCST fortification problems.

Acknowledgments

I would like to express my heartfelt gratitude to the following individuals who have played a pivotal role in helping me successfully complete my PhD journey. Their unwavering support, encouragement, and love have been instrumental in this achievement.

First and foremost, I want to acknowledge and thank my supervisors, Dr. Onur Kuzgunkaya and Dr. Navneet Vidyarthi, for their exceptional guidance, mentorship, and patience throughout my doctoral research. Your expertise and dedication were invaluable, and I am deeply grateful for the knowledge and skills I have gained under your guidance. Furthermore, I would like to extend my gratitude to the Fonds de Recherche du Québec – Nature et Technologies (FRQNT) for their invaluable financial support

To my incredible husband and my best friend, Masoud, whose endless love and sacrifice made this journey possible. Your constant encouragement and understanding made it possible for me to focus on my studies. Your unwavering belief in me kept me motivated during the most challenging times, and I am so fortunate to have you by my side. You were my code editor, motivational coach, and proofreader during this journey. Without you, none of this would have been possible. No words can truly capture the depth of my gratitude for everything you have done for me.

I would like to appreciate my parents, Asad and Maryam, for their unconditional love, encouragement, and support. Their belief in my capabilities has been a source of motivation and strength throughout my academic journey. To my sister, Shamim, for her sense of humor, and our long discussions. Thanks for helping mom and dad in my absence.

In loving memory of my late grandmothers, Akram and Fakhri. I want to express my gratitude to both of these remarkable women who have left an indelible mark on my life. I am forever thankful

for the love they shared and the strength they instilled in me. Unfortunately, I was unable to be by my grandmother's side, Fakhri, during her final days as I pursued my PhD in Canada. Her absence weighed heavily on my heart, but her memory and the lessons she imparted continue to inspire me every day.

I would like to appreciate my friends, Ximena and Alejandro, who provided me with unwavering emotional support during the challenging phases of my journey. I would like to thank Ibrahim for his scientific and professional support. To my friend, Maryam, for making me laugh. To Yeganeh, my high school friend, and Yasaman, my childhood friend, thanks for the invaluable presence you've had in my life. Thanks to all my friends whom I did not mention by name. Your friendship provided the necessary balance in my life and made this journey more memorable.

To all of you, I am immensely grateful for being a part of my life and for celebrating this significant achievement with me. Your belief in me has been the driving force behind my success, and I could not have reached this milestone without you. Thank you for being my constant source of inspiration and strength.

To my family

"When it is tough, will you give up, or you will be relentless?"

-Jeff Bezos

Contribution of Authors

This dissertation is presented in a manuscript-based format. The thesis contains three articles that are either under revision in different journals or in preparation for submission to a journal. The first article titled "Distribution Network Design under Stochastic Facility Interdiction" has been submitted to the journal *Computers & Operations Research*. The second article titled "An Exact Algorithm for Multicommodity Network Design under Stochastic Interdictions" has been submitted to *INFORMS Journal on Computing*. Finally, the third manuscript, titled "Fortification of Spanning Trees under Stochastic Interdictions" is being prepared for submission to *European Journal of Operational Research*.

All three manuscripts presented in this thesis were co-authored with Dr. Onur Kuzgunkaya and Dr. Navneet Vidyarthi, as supervisors. The author of this thesis acted as the principal researcher and performed the development of formulations and algorithms, the coding of solution methods, and the analysis of computational results along with writing the first drafts.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Networks are the vital aspects of today's life that appear in many systems such as electric power distribution, transportation and public transit, telecommunication, water supply, and natural gas and petroleum distribution systems. Due to the importance of networks for social and economic aspects of life, it is important to find the critical infrastructures and reduce the vulnerability of the networks. We use the network interdiction model to identify critical infrastructures. Network interdiction models consist of two opposing players: leader and follower. Network interdiction problems are formulated as bi-level mixed-integer programming models where at the upper level, the interdictor (leader) decides to interdict specific network components to cause maximum damage to the network. At the lower level, the follower intends to optimize their decisions in the disrupted network. As an example, in a bi-level shortest path network interdiction problem, the follower's decision variables y determine the shortest path in the network. The follower's decision variables are affected by the interdiction decision x. Let X be the set of feasible solutions to the first level, and $Y(x)$ be the set of feasible solutions to the shortest path problem for a given interdiction decision x. Traversing an arc incurs a cost of c in normal conditions and the cost increases to $(c+dx)$ if the arc is interdicted. The interdictor intends to maximize the shortest path by disrupting some arcs of the network which results in the following max-min optimization problem.

$$\max_{x \in X} \min_{y \in Y(x)} (c+dx)y \tag{1.1}$$

Network interdiction models are studied on networks with well-studied recourse problems like shortest path (Israeli and Wood, 2002; Holzmann and Smith, 2021), maximum flow (Rad and Kakhki, 2013), knapsack (Contardo and Sefair, 2022), vehicle routing problem (Sadati et al., 2020), multicommodity network (Smith et al., 2007), matching (Zenklusen, 2010), hub location (Ramamoorthy et al., 2018), and traveling salesman problem (Lozano et al., 2017). Moreover, network interdiction models have applications in military (McMasters and Mustin, 1970), cybersecurity (Baggio et al., 2021), water resource analysis (Jiang and Liu, 2018), power grid analysis (Wu and Conejo, 2016), and illicit supply chains (Jabarzare et al., 2020).

The interdiction models are able to identify the critical infrastructures and are extendable in two different ways to enhance the reliability of the networks. In the first category, interdiction models are used to fortify the existing networks. In this category, the objective is to mitigate the negative effects of interdictions by protecting the infrastructures of the network e.g., by placing surveillance cameras. In the second category, we consider the effects of interdictions in the design stage of networks. Design models are an extension of interdiction models that aim to build a reliable network which not only operates effectively in normal situations but also functions efficiently under disruptions. Fortification and design models are formulated as tri-level models, a game between defender-interdictor-defender or designer-interdictor-designer, where at the upper level the defender determines the fortification decisions or the designer determines the infrastructures to install.

In the real world, there is uncertainty associated with different parameters of the networks. Different parameters can be uncertain in network interdiction models including interdiction budget (Liberatore et al., 2011), interdiction success (Janjarassuk and Linderoth, 2008), network configuration (Morton, 2010), interdiction place (Holzmann and Smith, 2021), and demand (Hien et al., 2020). In this thesis, we aim to study the effects of uncertain interdiction parameters on the fortification and design of networks. We consider the interdiction outcome and the number of interdictions as uncertain parameters. To incorporate the uncertainty in the models, we present tri-level stochastic mathematical models to fortify or design networks. Considering stochastic interdiction parameters in fortifying or designing networks complicates the problem. In this thesis, we present exact solution methodologies to solve network fortification and design problems with stochastic interdictions.

The contributions of this thesis can be categorized as follows:

- Problem modeling
    - Considering the stochastic interdiction outcome for designing the distribution network
    - Considering the stochastic number of interdictions for designing the multicommodity network
    - Presenting the fortification problem for minimum spanning tree (MST) and optimum communication spanning tree (OCST) under stochastic interdictions
- Algorithmic development
    - Presenting Benders decomposition algorithm accelerated with dual decomposition and supervalid and valid inequalities for distribution network design problem
    - Presenting branch-and-Benders-cut algorithm improved by acceleration techniques for multicommodity network design problem
    - Implementing backward sampling framework for solving the deterministic and stochastic MST/OCST fortification problems
- Managerial insights
    - The resiliency of the networks can be enhanced by a slight increase in the installation cost
    - Showcasing the advantages of stochastic design over deterministic design when the number of interdictions is uncertain

This thesis focuses on the extension of interdiction models for having more reliable networks by considering the effects of interdictions in the design phase of networks and fortifying the existing networks. In what follows, we elaborate on our contributions to network design under stochastic interdictions and spanning tree fortification under stochastic interdictions.

1.1 Network Design under Stochastic Interdictions

Distribution and multicommodity networks have applications in logistics, transportation, and telecommunication systems. Such networks are designed based on the premise of undisrupted functioning with minimal redundancy in their structure. Given the importance of supply chains and telecommunication systems, it is essential to ensure that these systems are resilient. We consider the impact of interdictions in the design stage of networks to create reliable networks that function effectively under both normal and disrupted conditions. It is important to consider disruptions in the design stage of distribution and multicommodity networks because adjusting or changing these systems is generally costly. We present tri-level mathematical models for the distribution and multicommodity network design with interdictions. To have more realistic models, we consider the uncertainty in interdiction parameters in the presented tri-level design models.

In Chapter 2, we present a tri-level stochastic mathematical model for distribution network design under uncertain outcomes of interdictions. To solve the model, we implement the Benders decomposition (BD) algorithm, solving the MILP master problem and stochastic subproblem iteratively. To improve the performance of the BD algorithm, we use the dual decomposition algorithm to solve the stochastic subproblem. We also add valid and supervalid inequalities to the master problem. Our extensive computational experiments show that the BD algorithm improved with dual decomposition and valid and supervalid inequalities outperforms other variants of the BD algorithm. We demonstrate that the stochastic model provides a less conservative design compared to the deterministic counterpart by opening fewer facilities at the expense of a slight increase in the worst-case post-interdiction transportation cost.

In Chapter 3, we present a tri-level stochastic model for multicommodity network design under an uncertain number of interdictions. The designer does not have information about the exact number of interdiction resources. Therefore, we consider the number of interdictions as the stochastic parameter. To solve the model, we use the branch-and-Benders-cut algorithm. We use several acceleration techniques including warm start, variable fixing, cut selection, Pareto-optimal cuts, penalty reformulation, and supervalid and valid inequalities to improve the performance of the branch-and-Benders-cut algorithm. We emphasize the benefits of employing a stochastic design as opposed to

a deterministic one when the number of interdictions is uncertain.

1.2 Spanning Tree Fortification under Stochastic Interdictions

In the last contribution of the thesis, we examine the fortification of two different classes of spanning tree problems. We study the minimum spanning tree (MST) and optimum communication spanning tree (OCST) problems. MST and OCST problems have important applications in transportation and communication networks. Our goal is to find the optimal fortification strategy that hinders the negative impact of interdictions on the MST/OCST costs. In Chapter 4, we propose tri-level stochastic models for fortification of the MST and OCST problems under the stochastic number of interdictions for the first time. We implement the backward sampling framework (BSF) to solve the deterministic and stochastic variants of the MST/OCST fortification problems under stochastic interdictions. To improve the performance of the BSF algorithm, we delay the exploration of fortification strategies if the improvement is not sufficiently large by keeping them in a waiting list. We save the unpromising fortification strategies in a waiting list for subsequent re-evaluation within the algorithm's execution. Our computational experiments illustrate that the waiting list acceleration technique significantly improves CPU time.

Chapter 2

Distribution Network Design under Stochastic Facility Interdiction

Abstract In this paper, we study a robust distribution network design problem that accounts for the impact of disruptions of intermediate facilities. We consider a two-echelon distribution network design problem where the designer's decision is to locate intermediate facilities between a set of supply and demand nodes such that the total installation cost and the transportation costs under normal conditions and after interdictions are minimized. The problem is modeled as a two-player, sequential, designer-attacker-designer Stackelberg game. In the first stage, the designer selects a subset of intermediate facilities to locate, the interdictor next damages a subset of located facilities, after which the designer optimizes a recourse problem over the network of residual facilities. Unlike deterministic network interdiction models that assume that all the model parameters are known with certainty, we present a tri-level stochastic mixed-integer programming model where the interdiction outcomes are uncertain and the interdictor's goal is to maximize the expected damage. We study both cases of independence of interdictions across facilities as well as the correlated interdictions. We present an exact algorithm based on Benders decomposition to solve the model, where the subproblem is a stochastic optimization problem. To improve the computation time, the stochastic subproblem is solved using dual decomposition. Two sets of valid inequalities are proposed to accelerate the Benders decomposition algorithm. Extensive computational experiments are conducted

to assess the efficiency of the proposed refinements. The results show that 91% of instances are optimally solved using algorithmic refinements compared to 67% of instances with the basic Benders decomposition algorithm. In terms of computational time, the refinements reduce the CPU time by 35%. Computational results on large instances with up to 18 facilities, and 2^{18} interdiction scenarios confirm the efficiency of the algorithms. We illustrate that the stochastic model provides a less conservative design compared to the deterministic counterpart by opening fewer facilities at the expense of a slight increase in the worst-case post-interdiction transportation cost.

2.1 Introduction

In recent years, the importance of protection planning for global supply chains and distribution networks has grown with the increasing frequency of major disruptions. Despite lower operational costs, globalization trends, just-in-time delivery, and other lean practices have made supply chains more vulnerable to disruptions. In the short term, disruptions result in higher transportation costs, delayed orders, lost sales, and also have a long-term impact on market share. Supply chains and distribution networks can be at risk of natural disruptions (e.g., earthquake, fire), or premeditated disruptions such as labor strikes, terrorist attacks, and geopolitical conflicts (Ivanov et al., 2017; Xu et al., 2020). For example, the Russia-Ukraine conflict is having a massive influence on the global supply chains (such as the shortage of electronic chips), forcing supply managers to reassess their established supply chain and partner ecosystem (Stackpole, 2022). Toyota had to reduce its production in 2022 due to the chip shortage and halted some of the production lines because of the supply shortage caused by COVID-19 restrictions in China (CNBC, 2022). Many countries placed quarantine orders and closed borders to limit the spread of COVID-19. It is worthwhile noting that 94% of Fortune 1000 companies faced disruptions in their supply chains during the pandemic (Sherman, 2020). These recent events are mounting pressure on supply chain managers to build robustness in their supply chain configurations so that it is possible to operate in both normal and disrupted conditions.

Major disruptions, characterized by a low probability of occurrence and a large magnitude of effect, have significant short-term and long-term cost implications (Jabbarzadeh et al., 2016). A

disruption in one facility can seriously impact the market share and cause extensive interruptions in vital services as it indirectly affects the whole distribution network (Snyder et al., 2016; Aldrighetti et al., 2021). Hendricks and Singhal (2005) show that companies experiencing disruptions have 40% lower stock returns compared with similar industries. Therefore, considering the effects of disruptions in the design stage is essential for a robust network (Snyder and Daskin, 2005).

In considering the impact of major disruptions in networks, approaches in the literature can be categorized into risk-neutral and risk-averse models based on the underlying risk considerations. The literature on the risk-neutral approach relies on stochastic programming models where disruptions are defined as scenarios with different probabilities. The model aims to optimize an expected value of a given performance measure, such as minimizing the expected operational cost or maximizing the expected profit if disruptions occur (Cui et al., 2010). This leads to two important challenges in considering disruptions. First, the disruption probabilities are assumed to be known a priori. Second, decisions based on expected cost minimization can lead to poor performance in rare and high-impact disruption events. Therefore, risk-averse models are better suited to consider worst-case disruptions.

As categorized in Table 2.1, the literature on the risk-averse approach uses robust optimization and network interdiction models to account for the effects of the disruptions. Works on robust optimization use conditional value-at-risk (Yu et al., 2017), p-robustness criterion (Peng et al., 2011), distributionally robust optimization (Che et al., 2024), interval robust optimization (Jabbarzadeh et al., 2016), and two-stage robust optimization (An et al., 2014; Cheng et al., 2021) to design networks with disruptions. On the other hand, network interdiction models are often formulated as a bi-level nested optimization problem based on the leader-follower game theoretic framework. In the first level, the leader interdicts arcs/nodes of the network to maximize the damage. In the second level, the follower optimizes the network operations after interdiction. Both of these approaches result in nested optimization models, defined as min-max-min or max-min-max formulations.

Motivated by the importance of robust supply chains in the recent wave of disruptions, we present a model for designing a two-echelon distribution network that accounts for the effects of interdictions or disruptions. The goal of designing a robust distribution network is to achieve operational efficiency under normal as well as post-interdiction operation conditions. More specifically,

Table 2.1: Classification of the Modeling Approaches for Facility Disruptions

Reference	Modeling Approach		
	Stochastic Optimization	Robust Optimization	Network Interdiction
Cui et al. (2010)	✓		
Li and Ouyang (2010)	✓		
Lim et al. (2010)	✓		
Li et al. (2013)	✓		
Azad and Hassini (2019)	✓		
Chang et al. (2024)	✓		
Peng et al. (2011)		✓	
Baghalian et al. (2013)		✓	
An et al. (2014)		✓	
Jabbarzadeh et al. (2016)		✓	
Yu et al. (2017)		✓	
Cheng et al. (2018)		✓	
Cheng et al. (2021)		✓	
Gicquel et al. (2022)		✓	
Smith et al. (2007)			✓
Laporte et al. (2010)			✓
O'Hanley and Church (2011)			✓
Aksen and Aras (2012)			✓
Medal et al. (2014)			✓
Parvaresh et al. (2014)			✓
Ghaffarinasab and Motallebzadeh (2018)			✓
This paper			✓

we present a tri-level nested design model where in the first level, the designer makes facility location decisions so that the installation cost and weighted sum of transportation costs under normal operations and after interdiction are minimized. Under a limited interdiction budget and uncertain outcome, the interdictor attempts to interdict intermediate facilities in the network with the objective of maximizing the expected minimum transportation costs in the distribution network. Finally, the designer optimizes the flow after interdiction in the third level.

Most interdiction models consider deterministic settings where interdiction outcomes are known with certainty (Smith and Song, 2020). Since interdiction outcomes can be uncertain in practice; it makes sense to consider it as an uncertain parameter rather than a deterministic one. To the best of our knowledge, this is the first paper considering the uncertainty of the interdiction outcomes for designing the two-echelon distribution network using the interdiction modeling framework. Stochastic network interdiction problems are a difficult class of optimization problems. To this end, we present an exact solution approach based on the Benders decomposition algorithm. To improve the performance of the proposed Benders decomposition algorithm, we solve the stochastic subproblem using

dual decomposition. In addition, we strengthen the Benders decomposition by adding supervalid and valid inequalities to the master problem. Our extensive computational experiments show that the Benders decomposition algorithm combined with dual decomposition, supervalid, and valid inequalities outperforms other variants of the Benders decomposition, improving the CPU time by 35% compared to the basic BD algorithm. Results on large-size instances with up to 18 facilities and 2^{18} scenarios confirm the effectiveness of the proposed solution methodology. We analyze the impact of various model parameters such as interdiction budget, interdiction success probability, and the relative weight of post-interdiction cost on the computation time of the proposed algorithm and the obtained design. Moreover, we illustrate that the proposed network design model with stochastic interdiction yields less conservative solutions in comparison with its deterministic variant.

The contributions of this paper are as follows:

- We introduce a tri-level interdiction model for designing robust distribution network that accounts for the effect of disruptions of intermediate facilities under interdiction uncertainty. The problem is modeled as a two-player, sequential designer-attacker-designer framework resulting in a tri-level stochastic mixed-integer program.
- We exploit the structure of the formulation to present an enhanced Benders decomposition algorithm, where we solve the stochastic subproblem using dual decomposition. In addition, we strengthen the Benders decomposition by adding supervalid and valid inequalities to the master problem. Interestingly, our computational results show that with the proposed algorithmic refinements, we are able to solve large instances of an otherwise intractable problem.

The remainder of this paper is organized as follows: In Section 2.2, we review the relevant literature. In Section 2.3, we provide the mathematical model for the stochastic network design problem. Solution methodologies for solving the proposed stochastic design model are described in Section 2.4. The computational experiments and conclusion are presented in Sections 2.5 and 2.6, respectively.

2.2 Literature Review

To consider the impact of disruptions, network interdiction models are one of the widely used models in the literature. Interdiction models were first introduced for military and homeland security applications (McMasters and Mustin, 1970). Since then, interdiction models have been used for shortest path (Israeli and Wood, 2002), maximum flow (Ghosh and Jaillet, 2022), facility location (Scaparra and Church, 2008a; Aksen et al., 2014), hub location (Ramamoorthy et al., 2018; Ullmert et al., 2020), multi-commodity networks (Lim and Smith, 2007), and vehicle routing problem (Sadati et al., 2020).

A common method to solve bi-level interdiction models is to reformulate the model using duality and KKT conditions into a single-level formulation. However, reformulating the bi-level model into a single-level formulation is not an efficient approach for large-size problem instances (Israeli and Wood, 2002). Authors present different heuristic and metaheuristic algorithms (Nandi et al., 2016; Forghani et al., 2020), decomposition algorithm (Tanergüçlü et al., 2019), and cutting plane algorithm (Naoum-Sawaya and Ghaddar, 2017) to solve bi-level interdiction models efficiently. Metaheuristic algorithms might solve large-size instances, but they do not guarantee the optimal solution. Exact algorithms guarantee the optimal solution, but they might not be time-efficient for large-size instances. Moreover, decomposition algorithms like Benders decomposition have slow convergence; therefore, acceleration techniques are required to improve the efficiency of algorithms.

Network interdiction models are further extended to fortify the existing networks (Cappanera and Scaparra, 2011) or design robust networks (Smith et al., 2007). Fortification and design models are expanded to tri-level mathematical models, but they can be modeled as bi-level if the contribution of a single asset to system performance is easy to define (Brown et al., 2006). Fortification models aim to identify critical facilities whose protection would reduce the negative impacts of worst-case interdictions. As the location of facilities is determined, it is costly to change the location and structure of the facilities (Cheng et al., 2018). Therefore, the network will be more robust by considering the effects of disruptions in the design of the network.

The purpose of the network design model with interdictions is to consider the effects of disruptions in the design stage of networks to minimize the disruption costs. Considering network design

with interdictions transforms the bi-level model into a tri-level, increasing the complexity. Authors solve the design model with interdictions using metaheuristic algorithm (Aksen and Aras, 2012; Parvaresh et al., 2014; Ghaffarinasab and Motallebzadeh, 2018), decomposition algorithm (Smith et al., 2007; O'Hanley and Church, 2011; Couedelo, 2018), and binary search algorithm (Medal et al., 2014).

The previous works mentioned in the interdiction literature assume deterministic values for parameters. Table 2.2 summarizes the papers in which uncertainty is considered: interdiction budget (or the number of interdictions) (Liberatore et al., 2011; Chen et al., 2011; Parvaresh et al., 2013; Bhuiyan et al., 2021), network configuration (Pan et al., 2003; Hemmecke et al., 2003; Morton, 2010; Towle and Luedtke, 2018), interdiction success (Cormican et al., 1998; Janjarassuk and Linderoth, 2008; Losada et al., 2012), arc capacities (Pay et al., 2019; Nguyen and Smith, 2022), and demand (Hien et al., 2020). Recently, Holzmann and Smith (2021) consider the randomized interdiction strategy in the shortest path network interdiction problem where the follower only knows the place of interdictions with a given probability.

Table 2.2: Classification of Relevant Literature on Stochastic Network Interdiction

Articles	Problem			Stochastic Parameter				Model Type		Methodology	
	Interdiction	Fortification	Design	Budget	Configuration	Interdiction Success	Other	Bi-level	Tri-level	Exact	Heuristic
Liberatore et al. (2011)		✓		✓				✓		✓	✓
Parvaresh et al. (2013)			✓	✓				✓			✓
Chen et al. (2011)			✓	✓					✓	✓	
Bhuiyan et al. (2021)		✓		✓				✓		✓	
Pan et al. (2003)	✓				✓			✓		✓	
Morton (2010)	✓				✓			✓		✓	
Hemmecke et al. (2003)	✓				✓			✓		✓	
Towle and Luedtke (2018)	✓				✓			✓		✓	
Cormican et al. (1998)	✓					✓		✓		✓	✓
Janjarassuk and Linderoth (2008)	✓					✓		✓		✓	✓
Losada et al. (2012)	✓					✓		✓		✓	
Nguyen and Smith (2022)	✓						✓	✓		✓	
Hien et al. (2020)		✓					✓		✓	✓	
Song and Shen (2016)	✓						✓	✓		✓	✓
Lei et al. (2018)	✓	✓				✓	✓	✓		✓	
Holzmann and Smith (2021)	✓						✓	✓		✓	
This paper			✓			✓			✓	✓	

Considering uncertain parameters as a set of scenarios leads to stochastic programming models, which are more complex than their deterministic counterparts due to the large size of the problem.

Sample average approximation algorithm (Janjarassuk and Linderoth, 2008; Lei et al., 2018), L-shape and Benders decomposition algorithms (Janjarassuk and Linderoth, 2008; Song and Shen, 2016), Jensen's inequality (Cormican et al., 1998; Nguyen and Smith, 2022), and constraint-and-column generation (Bhuiyan et al., 2016) are exact methods to solve the stochastic interdiction models.

Most interdiction papers with stochastic parameters consider the bi-level formulation determining the most critical infrastructures or fortification strategies. Chen et al. (2011) use interdiction budget scenarios to solve the deterministic tri-level network design problem. By enumerating all possible interdiction scenarios, they reformulate the problem into a two-stage stochastic program from the designer's perspective. They present cut generation algorithm using Benders cut to solve the single-level two-stage stochastic model. Hien et al. (2020) present a tri-level mathematical model for fortifying a single-commodity network flow problem with stochastic demand. They also extend the model to multi-commodity network flow. They solve the model using the stochastic approximation method. They mention that the stochastic approximation method works better than the sample average approximation if the parameters are selected carefully.

The closest papers to our work in terms of problem formulation are Chen et al. (2011), Smith et al. (2007), and Cheng et al. (2021). Our work is different from Chen et al. (2011) in two ways: first, we consider node interdiction in contrast to arc interdiction. Second, our proposed objective function considers both pre-interdiction and post-interdiction costs to provide a cost-efficient robust network design. With respect to Smith et al. (2007), we incorporate the stochastic nature of interdictions into the network design as opposed to deterministic assumption. In contrast to Cheng et al. (2021), our paper considers a robust design in an extended two-echelon supply chain network setting and incorporates the uncertainty in the outcome of interdictions. The fact that we consider interdictions to be successful with a given probability results in the inner bi-level optimization model to be a two-stage stochastic optimization. If we assume that interdictions are always successful (i.e., deterministic interdiction outcome version of our mathematical model), it is equivalent to the two-stage robust optimization problem. As such, the model presented in Cheng et al. (2021) is a special case of the model presented in this manuscript. To the best of our knowledge, our proposed model is the first paper that takes into account stochastic interdiction outcomes for distribution network design.

In terms of solution methodology, the closest works are Song and Shen (2016) and Jabarzare et al. (2020). Song and Shen (2016) present branch-and-Benders-cut algorithm to solve a bi-level stochastic shortest path problem. In contrast to their work, we employ classic Benders decomposition algorithm and focus on solving our subproblems efficiently. Jabarzare et al. (2020) use classic Benders decomposition to solve the deterministic bi-level interdiction model for the illicit supply chain. We use Benders decomposition to solve the stochastic design model where the master problem determines the design and pre-interdiction decisions, and the subproblem is the stochastic interdiction problem. We present dual decomposition algorithm to solve the stochastic subproblem more efficiently.

2.3 Model Formulation

In this paper, we design a two-echelon distribution network comprising supply nodes, intermediate facilities, and demand nodes. The goal is to locate intermediate facilities so that the installation and transportation costs before and after interdiction are minimized. We model the distribution network design problem as a Stackelberg game in which the designer makes the first move by selecting the intermediate facilities to install (y) and the routing decisions before interdiction (u' and u). At the second level, the interdictor selects the facilities to interdict (x) to maximize the expected minimum transportation cost of the designer. At the third level, the designer determines the routing decisions given the interdicted facilities in each scenario (w' and w) to minimize post-interdiction cost. We first consider the case where the interdiction on a facility is independent of the interdiction of other facilities, then we relax this assumption to consider the correlated interdiction among facilities.

We make the following assumptions: (i) The designer and the interdictor have complete knowledge of the network structure. (ii) The interdictor is uncertain about the outcome of interdictions on the network. We assume that interdictions are successful with known probability. (iii) The designer makes all the design and pre-interdiction decisions before the interdictor makes interdiction decisions. After the interdictor's decision, the designer makes the post-interdiction decisions. This game is called designer-interdictor-designer game. (iv) The game is played for only one round. (v)

Once a facility is interdicted, and the interdiction is successful, it becomes unavailable since partial interdiction does not occur.

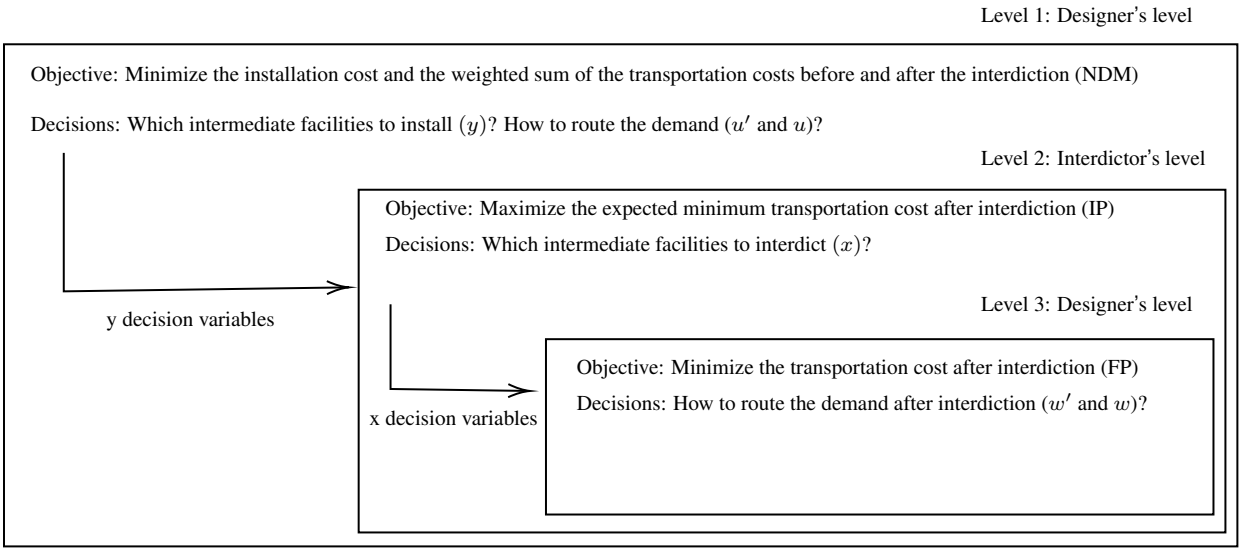


Figure 2.1: Network Design Model as a Tri-level Nested Optimization Model

The nested structure of the mathematical model is shown in Figure 2.1, and the list of notations appears in Table 2.3. Let I be the set of supply nodes, J be the set of potential intermediate facilities, and K be the set of demand nodes. Each supply node $i \in I$ has capacity q'_i and each demand node $k \in K$ has demand d_k. Each facility $j \in J$ has capacity q_j and installation cost f_j. The cost of sending demand from supply node $i \in I$ to facility $j \in J$ is denoted by c'_{ij} and the cost of sending demand from facility $j \in J$ to demand node $k \in K$ is denoted by c_{jk}. We assume the interdictor has a limited interdiction budget B for interdicting intermediate facilities.

Figure 2.2 illustrates the network before and after interdiction.

A scenario $s \in S$ represents a realization of uncertainty set with probability p_s. We define the binary parameter ξ_{js} to represent the uncertain outcome associated with the interdiction of facility j under scenario s. ξ_{js} is 1 if interdiction of facility j in scenario s is successful. In this model, the scenarios being considered do not directly specify whether an interdiction event occurs or not. Instead, they provide information about the potential outcome of an interdiction attempt if the interdictor decides to target facility j in scenario $s \in S$. We assume that interdiction outcomes are all pairwise independent (Janjarassuk and Linderoth, 2008). The interdictor determines the interdiction

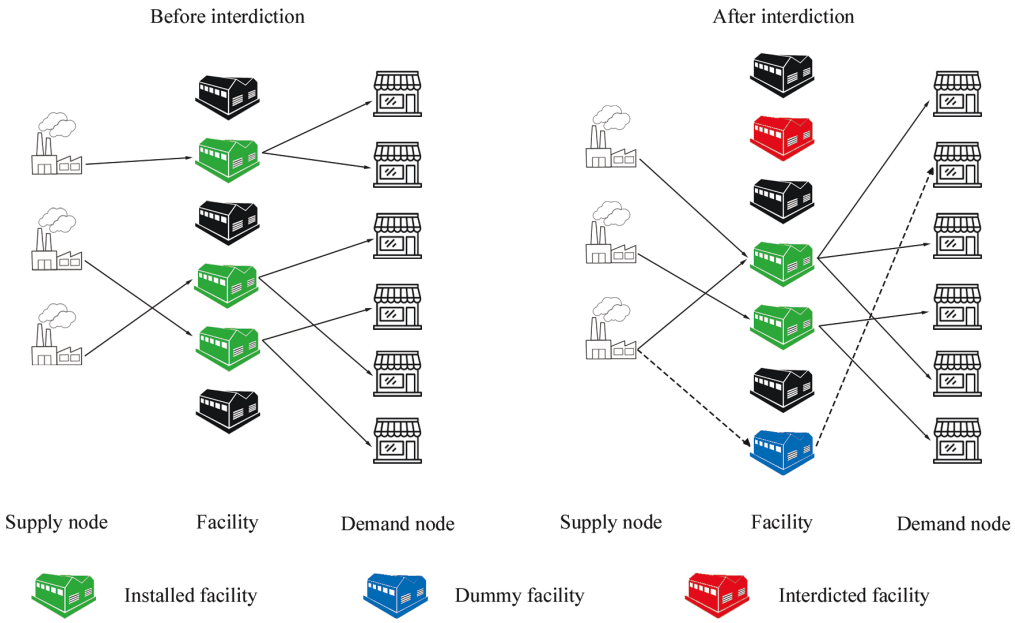


Figure 2.2: An Illustration of Network Design Problem with Interdictions

Table 2.3: Table of Notations

Sets	
I	Set of supply nodes (indexed by i, $i \in I$)
J	Set of candidates for intermediate facilities (indexed by j, $j \in J$)
K	Set of demand nodes (indexed by k, $k \in K$)
S	Set of scenarios (indexed by s, $s \in S$)
Parameters	
d_k	Demand of node k
c'_{ij}	Unit transportation cost from supply node i to facility j
c_{jk}	Unit transportation cost from facility j to demand node k
f_j	Installation cost of facility j
q'_i	Capacity of supply node i
q_j	Capacity of facility j
B	Interdiction budget (Number of facility interdictions)
p_s	Probability of scenario s
ρ_1	The weight of pre-interdiction cost in the objective function
ρ_2	The weight of post-interdiction cost in the objective function
Decision Variables	
y_j	1 if facility j is installed
u'_{ij}	Amount of flow from supply node i to facility j before interdiction
u_{jk}	Amount of flow from facility j to demand node k before interdiction
w'_{ijs}	Amount of flow from supply node i to facility j after interdiction in scenario s
w_{jks}	Amount of flow from facility j to demand node k after interdiction in scenario s
x_j	1 if facility j is interdicted

strategy without the knowledge of the outcome of the interdiction. Therefore, interdiction decisions can be considered as the first-stage decisions in the stochastic programming model. As the uncertainty is revealed for any realization of scenario s, the continuous recourse problem (designer's third level) determines the post-interdiction flows as second-stage decisions. We illustrate the scenarios with an example of three facilities; therefore, there are 2^3 scenarios for the interdiction outcomes. Assume that the interdiction on any given facility would be successful with a probability of 0.75. We present the eight possible scenarios with the associated probabilities in Table 2.4. As an example, in scenario 8, the interdiction on all facilities would be successful (i.e., $\xi_{18} = \xi_{28} = \xi_{38} = 1$) with probability $p_8 = 0.75 * 0.75 * 0.75 = 0.422$. In scenario 5, the interdiction on the first 2 facilities would be successful and the interdiction of the third facility would be unsuccessful (i.e., $\xi_{15} = \xi_{25} = 1, \xi_{35} = 0$) with probability $p_5 = 0.75 * 0.75 * 0.25 = 0.141$.

Table 2.4: Scenario Representation and its Corresponding Probability of a Network with 3 Facilities

Scenario	Facility 1	Facility 2	Facility 3	Probability
1	1	0	0	0.047
2	0	0	1	0.047
3	1	0	1	0.141
4	0	1	0	0.047
5	1	1	0	0.141
6	0	0	0	0.016
7	0	1	1	0.141
8	1	1	1	0.422

The tri-level design model (designer-interdictor-designer) is as follows:

$$(NDM): \quad \min_{u,u',y} \sum_{j\in J} f_j y_j + \rho_1 \Big(\sum_{i\in I}\sum_{j\in J} c'_{ij} u'_{ij} + \sum_{j\in J}\sum_{k\in K} c_{jk} u_{jk} \Big) + \quad \rho_2 T_1(y) \tag{2.1}$$

$$\text{s.t.}: \quad \sum_{j\in J} u_{jk} \geq d_k \qquad \forall k \in K \tag{2.2}$$

$$\sum_{i\in I} u'_{ij} \geq \sum_{k\in K} u_{jk} \qquad \forall j \in J \tag{2.3}$$

$$\sum_{j\in J} u'_{ij} \leq q'_i \qquad \forall i \in I \tag{2.4}$$

$$\sum_{i\in I} u'_{ij} \leq q_j y_j \qquad \forall j \in J \tag{2.5}$$

$$y_j \in \{0,1\} \qquad \forall j \in J \tag{2.6}$$

$$u'_{ij}, u_{jk} \geq 0 \qquad \forall i \in I, j \in J, k \in K \tag{2.7}$$

$$(IP): \quad T_1(y) = \max_{x} \sum_{s\in S} p_s A_s(x,y) \tag{2.8}$$

$$\text{s.t.}: \quad \sum_{j\in J} x_j \leq B \tag{2.9}$$

$$x_j \in \{0,1\} \qquad \forall j \in J \tag{2.10}$$

$$(FP): \quad A_s(x,y) = \min_{w,w'} \sum_{i\in I}\sum_{j\in J} c'_{ij} w'_{ijs} + \sum_{j\in J}\sum_{k\in K} c_{jk} w_{jks} \tag{2.11}$$

$$\text{s.t.}: \quad \sum_{j\in J} w_{jks} \geq d_k \qquad \forall k \in K \tag{2.12}$$

$$\sum_{i\in I} w'_{ijs} \geq \sum_{k\in K} w_{jks} \qquad \forall j \in J \tag{2.13}$$

$$\sum_{j\in J} w'_{ijs} \leq q'_i \qquad \forall i \in I \tag{2.14}$$

$$\sum_{i\in I} w'_{ijs} \leq q_j y_j (1 - x_j \xi_{js}) \qquad \forall j \in J \tag{2.15}$$

$$w'_{ijs}, w_{jks} \geq 0 \qquad \forall i \in I, j \in J, k \in K \tag{2.16}$$

The first level mathematical model (designer's level) is given by (2.1)-(2.7). At this level, the installation decisions are determined to minimize the installation cost and weighted sum of the pre-interdiction and post-interdiction costs. The demand satisfaction is guaranteed by constraint (2.2). The flow balance in the network is ensured by constraint (2.3). Constraints (2.4) and (2.5) impose a capacity limit on each supply node and facility, respectively.

In the proposed stochastic network design model, the uncertainty is modeled at the second level since the outcome of interdiction of intermediate facilities is uncertain. Therefore, the objective function of the interdictor (2.8) is to maximize the expected value of the designer's transportation costs after interdiction, given the uncertainty in scenarios. The interdictor has a limited interdiction budget as shown in constraint (2.9). We should mention that the interdictor should interdict open facilities ($y_j = 1$) to maximize the damage (see Lemma 1 of An et al. (2014)); therefore, we can fix the value of interdiction decisions x_j if the facility is not installed (i.e., $y_j = 0$).

At the third level, the designer minimizes the post-interdiction cost in the residual network for scenario realization s. We can see the effect of interdictions on the right-hand side of constraint (2.15), which states that the capacity of facility j is zero if it is interdicted successfully in scenario $s \in S$. In some cases, the capacity of the residual facilities after interdiction is not sufficient to satisfy demand. To account for the lost demand, we create a dummy facility. The dummy facility is characterized by zero installation and high transportation costs to represent the lost sales cost. The proposed designer-interdictor-designer model results in a tri-level formulation. In the next section, we reduce the lower-level bi-level interdictor-designer model to a single-level model.

2.3.1 Single-level Reduction of the Interdictor's Problem using Dual Formulation

Given a design decision $\hat{y}$, we fix the interdiction decisions in the lower level FP and take the dual of the inner minimization model to get a single-level mixed-integer programming (MIP) model. Associating Γ_{ks}, ϕ_{js}, θ_{is}, and ψ_{js} as dual variables with constraints (2.12)-(2.15), respectively, the resulting single-level MIP problem IP' is as follows:

$$(IP') : \max_{\Gamma,\phi,\theta,\psi,x} \sum_{s\in S} p_s \left[\sum_{k\in K} d_k \Gamma_{ks} - \sum_{i\in I} q'_i \theta_{is} - \sum_{j\in J} q_j \hat{y}_j (1 - x_j \xi_{js}) \psi_{js} \right] \tag{2.17}$$

s.t. $\Gamma_{ks} - \phi_{js} \leq c_{jk}$ $\forall j \in J, k \in K, s \in S$ (2.18)

$\phi_{js} - \theta_{is} - \psi_{js} \leq c'_{ij}$ $\forall i \in I, j \in J, s \in S$ (2.19)

$\sum_{j \in J} x_j \leq B$ (2.20)

$\Gamma_{ks}, \phi_{js}, \theta_{is}, \psi_{js} \geq 0$ $\forall i \in I, j \in J, k \in K, s \in S$ (2.21)

$x_j \in \{0, 1\}$ $\forall j \in J$ (2.22)

The model IP' has bilinear objective function and can be linearized by defining a set of non-negative continuous variables ω_{js} and the following constraints:

$\omega_{js} \leq M x_j$ $\forall j \in J, s \in S$ (2.23)

$\omega_{js} \leq \psi_{js}$ $\forall j \in J, s \in S$ (2.24)

where M is a large number. Variable ω_{js} appears only in the objective function with a positive coefficient in (2.17); therefore, ω_{js} takes the maximum value which is $x_j \psi_{js}$ and there is no need to consider $\omega_{js} - \psi_{js} - M x_j \geq -M$.

With this, the tri-level model (2.1)-(2.16) reduces to a bi-level model NDM' as follows:

$$(NDM') : \min_{u,u',y} \sum_{j \in J} f_j y_j + \rho_1 \left[\sum_{i \in I} \sum_{j \in J} c'_{ij} u'_{ij} + \sum_{j \in J} \sum_{k \in K} c_{jk} u_{jk} \right] + \rho_2 \max_{x,\Gamma,\phi,\theta,\psi,\omega} \sum_{s \in S} p_s \left[\sum_{k \in K} d_k \Gamma_{ks} - \sum_{i \in I} q'_i \theta_{is} - \left(\sum_{j \in J} q_j y_j \psi_{js} - \sum_{j \in J} q_j y_j \xi_{js} \omega_{js} \right) \right] \quad (2.25)$$

s.t. $(2.2) - (2.5), (2.18) - (2.20), (2.23) - (3.9)$

$x_j, y_j \in \{0, 1\}$ $\forall j \in J$

$u'_{ij}, u_{jk} \geq 0$ $\forall i \in I, j \in J, k \in K$

$\Gamma_{ks}, \phi_{js}, \theta_{is}, \psi_{js}, \omega_{js} \geq 0$ $\forall i \in I, j \in J, k \in K, s \in S$

To solve the bi-level formulation, we present Benders decomposition algorithm and two acceleration techniques in Section 2.4.

2.3.2 Extension to the Correlated Facility Disruptions

We extend the model (2.1)-(2.16) to consider the interdependencies among the facility disruptions. Examples of correlation effects are natural disasters (e.g., hurricanes, fire, and earthquakes) where disruption at a facility affects the nearby facilities as well (Liberatore et al., 2012). We assume the interdiction success probability of facilities does not depend on this correlation. The correlation represents the ripple effect of capacity reduction after a successful interdiction. To consider the effects of correlated disruptions, we define correlation matrix $\varrho_{jj'}$ which represents the portion of capacity that facility j loses if facility j' is interdicted. When there is more than one interdiction, the capacity of facility j is reduced by the highest correlation coefficient that affects facility j. Let FC be the set of facility pairs (j, j') with $\varrho_{jj'} \geq 0$. We modify constraint (2.15) to account for the correlated interdictions as follows:

$$\sum_{i \in I} w'_{ijs} \leq q_j y_j f_{jj'}(x_{j'}) \qquad \forall (j, j') \in FC \tag{2.26}$$

where

$$f_{jj'}(x_{j'}) = \begin{cases} 1, & \text{if } \; x_{j'} = 0 \vee (x_{j'} = 1 \wedge \xi_{j's} = 0) \\ 1 - \varrho_{jj'}, & \text{if } \; x_{j'} = 1 \wedge \xi_{j's} = 1 \end{cases}$$

According to constraint (2.26), capacity of facility j does not change if facility j' is not interdicted or in cases where facility j' is interdicted but the interdiction attempt is unsuccessful ($\xi_{j's} = 0$). If facility j' is interdicted and the interdiction is successful ($\xi_{j's} = 1$), the capacity of facility j reduces by a factor of $(1 - \varrho_{jj'})$.

To solve the proposed model considering the correlated facility interdictions, we first take the dual of the inner minimization problem FP with constraint (2.26) instead of constraint (2.15). We define $\tau_{jj's}$ as the dual variable associated with constraint (2.26). The single-level formulation is as

follows:

$$\max_{\Gamma,\phi,\theta,\tau,x} \sum_{s\in S} p_s \left[\sum_{k\in K} d_k \Gamma_{ks} - \sum_{i\in I} q'_i \theta_{is} - \sum_{(j,j')\in FC} q_j \hat{y}_j f_{jj'}(x_{j'}) \tau_{jj's} \right] \tag{2.27}$$

s.t.

$$\Gamma_{ks} - \phi_{js} \leq c_{jk} \qquad \forall j \in J, k \in K, s \in S \tag{2.28}$$

$$\phi_{js} - \theta_{is} - \sum_{j'|(j,j')\in FC} \tau_{jj's} \leq c'_{ij} \qquad \forall i \in I, j \in J, s \in S \tag{2.29}$$

$$\sum_{j\in J} x_j \leq B \tag{2.30}$$

$$\Gamma_{ks}, \phi_{js}, \theta_{is} \geq 0 \qquad \forall i \in I, j \in J, k \in K, s \in S \tag{2.31}$$

$$\tau_{jj's} \geq 0 \qquad \forall (j,j') \in FC, s \in S \tag{2.32}$$

$$x_j \in \{0,1\} \qquad \forall j \in J \tag{2.33}$$

The term $f_{jj'}(x_{j'})\tau_{jj's}$ in the objective function (2.27) makes the problem nonlinear. To linearize the model, we define non-negative continuous variables $\beta_{jj's}$ and we add the following constraints:

$$\beta_{jj's} \geq \tau_{jj's}(1 - \varrho_{jj'}) \qquad \forall (j,j') \in FC, s \in S \tag{2.34}$$

$$\beta_{jj's} \geq \tau_{jj's} - M\xi_{j's}x_{j'} \qquad \forall (j,j') \in FC, s \in S \tag{2.35}$$

To show that the provided linearization is valid, we consider three different cases:

(1) When $x_{j'} = 1$ and $\xi_{j's} = 1$, we obtain $\beta_{jj's} \geq \tau_{jj's}(1 - \varrho_{jj'})$ and $\beta_{jj's} \geq \tau_{jj's} - M$. Since the nonlinear term $f_{jj'}(x_{j'})\tau_{jj's}$ has a negative sign in the maximization objective function, the value of $\beta_{jj's}$ would be $\tau_{jj's}(1 - \varrho_{jj'})$.

(2) When $x_{j'} = 0$, we obtain $\beta_{jj's} \geq \tau_{jj's}(1 - \varrho_{jj'})$ and $\beta_{jj's} \geq \tau_{jj's}$. Since we assume that $\varrho_{jj'} \geq 0$, the value of $\beta_{jj's}$ would be the lowest allowable value $\tau_{jj's}$.

(3) When $x_{j'} = 1$ and $\xi_{j's} = 0$, we obtain $\beta_{jj's} \geq \tau_{jj's}(1 - \varrho_{jj'})$ and $\beta_{jj's} \geq \tau_{jj's}$. Since we assume that $\varrho_{jj'} \geq 0$, the value of $\beta_{jj's}$ would be $\tau_{jj's}$.

The solution methods presented in Section 2.4 are applicable to the correlated facility interdiction model.

2.4 Solution Methodology

Benders decomposition is a technique for solving MIPs by decomposing the problem into an integer master problem and a linear subproblem (Benders, 1962). The value of the specific partition of variables is determined in the master problem first. The value of the other partition of variables is determined by solving the subproblem for a given master problem solution. Due to the structure of interdiction models, Benders decomposition is a common technique for solving such models (Jabarzare et al., 2020). Consider a bi-level interdiction model with integer variables in the first level and continuous variables in the second level. By taking the dual of the inner optimization problem, we obtain a single-level MIP. To solve the MIP using the Benders algorithm, we fix the value of integer variables and take the dual of the single-level problem, which causes the bi-level model to reappear. As a result, the Benders decomposition readily applies to bi-level problems (Brown et al., 2006).

2.4.1 Benders Decomposition

In the Benders reformulation of the model, the master problem (MP) determines the facility location and pre-interdiction flow decisions, and the remaining decision variables are determined in the subproblem (SP). In the proposed model, the SP is the bi-level stochastic interdiction model that is transformed into a single-level MIP model IP' in Section 2.3. By fixing the values of facility location decisions $\hat{y}$, we obtain the following mixed-integer SP.

$$
\begin{aligned}
(SP): \max_{x,\Gamma,\phi,\theta,\psi,\omega} \quad & \sum_{s\in S} p_s \left[\sum_{k\in K} d_k \Gamma_{ks} - \sum_{i\in I} q_i' \theta_{is} - \left(\sum_{j\in J} q_j \hat{y}_j \psi_{js} - \sum_{j\in J} q_j \hat{y}_j \xi_{js} \omega_{js} \right) \right] \\
\text{s.t.} \quad & (2.18) - (3.9)
\end{aligned}
\tag{2.36}
$$

Using continuous variable η for the post-interdiction cost, the Benders reformulation of the

mixed-integer MP is as follows:

$$(MP): \quad \min_{u,u',y,\eta} \sum_{j\in J} f_j y_j + \rho_1 \left[\sum_{i\in I}\sum_{j\in J} c'_{ij}u'_{ij} + \sum_{j\in J}\sum_{k\in K} c_{jk}u_{jk}\right] + \rho_2\eta \tag{2.37}$$

$$\text{s.t.} \quad (2.2)-(2.7)$$

$$\eta \geq \sum_{s\in S} p_s \left[\sum_{k\in K} d_k \hat{\Gamma}_{ks} - \sum_{i\in I} q'_i \hat{\theta}_{is} - \left(\sum_{j\in J} q_j y_j \hat{\psi}_{js} - \sum_{j\in J} q_j y_j \xi_{js}\hat{\omega}_{js}\right)\right] \tag{2.38}$$

$$\eta \qquad \text{unrestricted} \tag{2.39}$$

In each iteration of the Benders decomposition algorithm, we add the optimality cut (2.38) to the MP. As we define the dummy facility to account for lost sales, the SP is always feasible; therefore, we do not need to add a feasibility cut to the MP. The algorithm terminates when the optimality gap between the upper bound and the lower bound is less than a specified threshold value (ϵ). The MP is the relaxation of NDM' since a subset of interdiction scenarios and post-interdiction flows are considered. Therefore, the MP provides a lower bound on the objective function value of the NDM' model. On the other hand, for a design decision $\hat{y}$, the SP determines the optimal decision x_j for the interdictor. This decision is feasible for the NDM' model. Therefore, the sum of the installation cost of the $\hat{y}$ solution, the weighted pre-interdiction transportation cost, and the weighted cost of the SP provides an upper bound on the objective function value of the NDM' model.

Let UB denote an upper bound on the optimal objective value obtained by solving the SP. Let LB indicate a lower bound on the optimal objective value obtained by solving the MP, and t represents the current iteration. $\mathbb{G}^t$ refers to the set of extreme points generated up to iteration t, and ϵ is the desirable optimality gap. The pseudo-code of the classic Benders decomposition algorithm is presented in Algorithm 7.

Algorithm 1 Benders Decomposition Algorithm (BD)

$UB \leftarrow \infty$, $LB \leftarrow -\infty$, $t \leftarrow 0$, $\mathbb{G}^t \leftarrow \varnothing$

While ($\frac{UB - LB}{UB} > \epsilon$) **do**

Solve the MP to obtain y

Update LB

Solve the SP for a given $\hat{y}$

Obtain dual variables $(\Gamma, \phi, \theta, \psi, \omega)$

Update UB

$\mathbb{G}^{t+1} \leftarrow \mathbb{G}^t \cup \{\Gamma, \phi, \theta, \psi, \omega\}$

$t \leftarrow t + 1$

End while

Due to a large number of interdiction scenarios, the SP becomes harder to solve as the number of candidate facilities increases. To overcome this issue, we present dual decomposition algorithm to solve the stochastic SP more efficiently.

2.4.2 Dual Decomposition Algorithm for Solving the Subproblem

Dual decomposition is one of the widely used approaches in stochastic programming as it creates subproblems structurally similar to the original formulation (Märkert and Gollmer, 2008; Guo et al., 2015). It is based on the reformulation of the extensive form of the deterministic equivalent and Lagrangian relaxation (Carøe and Schultz, 1999).The idea of dual decomposition is to decompose the problem by scenarios. To decompose the problem by scenarios, we introduce the copies of x_{js} of the first-stage variables x_j (interdiction decisions) with the non-anticipativity constraint $x_{j1} = x_{j2} = \cdots = x_{jS}$. The non-anticipativity constraint ensures that the first-stage variables take the same value in all scenarios. By relaxing the non-anticipativity constraint, we can decompose the problem into $|S|$ disjoint SPs. We use the subgradient method to solve the resulting Lagrangian dual. We rewrite the SP as MIP SPDD with non-anticipativity constraint (2.44).

$$(SPDD): \max_{x,\Gamma,\phi,\theta,\psi,\omega} \sum_{s\in S} p_s \left[\sum_{k\in K} d_k \Gamma_{ks} - \sum_{i\in I} q_i' \theta_{is} - \left(\sum_{j\in J} q_j \hat{y}_j \psi_{js} - \sum_{j\in J} q_j \hat{y}_j \xi_{js} \omega_{js} \right) \right] \quad (2.40)$$

s.t. $(2.18), (2.19), (3.9)$

$$\omega_{js} \le M x_{js} \qquad \forall j \in J, s \in S \quad (2.41)$$

$$\omega_{js} - \psi_{js} - M x_{js} \ge -M \qquad \forall j \in J, s \in S \quad (2.42)$$

$$\sum_{j\in J} x_{js} \le B \qquad \forall s \in S \quad (2.43)$$

$$\sum_{s\in S} H_{js} x_{js} = 0 \qquad \forall j \in J \quad (2.44)$$

$$\Gamma_{ks}, \phi_{js}, \theta_{is}, \psi_{js}, \omega_{js} \ge 0 \qquad \forall i \in I, j \in J, k \in K, s \in S \quad (2.45)$$

$$x_{js} \in \{0, 1\} \qquad \forall j \in J, s \in S \quad (2.46)$$

The matrices $H_{js} \in \mathbb{R}^{(S-1)\times SJ}$ for the representation of constraint (2.44) are: $H_{j1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$, $H_{j2} = \begin{bmatrix} -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \cdots, H_{jS} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix}$.

By relaxing constraint (2.44) with Lagrange multipliers λ_j, we obtain the following Lagrangian function:

$$D_s(\lambda): \quad \max\ p_s \left[\sum_{k\in K} d_k \Gamma_{ks} - \sum_{i\in I} q_i' \theta_{is} - \sum_{j\in J} q_j \hat{y}_j (\psi_{js} - \xi_{js} \omega_{js}) \right] + \sum_{j\in J} \lambda_j H_{js} x_{js} \quad (2.47)$$

s.t.
$$\Gamma_{ks} - \phi_{js} \leq c_{jk} \qquad \forall j \in J, k \in K \tag{2.48}$$
$$\phi_{js} - \theta_{is} - \psi_{js} \leq c'_{ij} \qquad \forall i \in I, j \in J \tag{2.49}$$
$$\omega_{js} \leq M x_{js} \qquad \forall j \in J \tag{2.50}$$
$$\omega_{js} \leq \psi_{js} \qquad \forall j \in J \tag{2.51}$$
$$\omega_{js} - \psi_{js} - M x_{js} \geq -M \qquad \forall j \in J \tag{2.52}$$
$$\sum_{j \in J} x_{js} \leq B \tag{2.53}$$
$$\Gamma_{ks}, \phi_{js}, \theta_{is}, \psi_{js}, \omega_{js} \geq 0 \qquad \forall i \in I, j \in J, k \in K \tag{2.54}$$
$$x_{js} \in \{0, 1\} \qquad \forall j \in J \tag{2.55}$$

Hence, the Lagrangian dual becomes:

$$(LD) : \min_{\lambda} D(\lambda), \quad \text{where} \quad D(\lambda) = \sum_{s \in S} D_s(\lambda) \tag{2.56}$$

To solve LD, we use the subgradient algorithm as stated in Algorithm 2. At each iteration t of the subgradient method, we compute the new Lagrange multiplier λ^t in the direction of subgradient s^t with the step length of μ^t. The parameter ε controls the step length at every iteration. $\sum_{s \in S} D_s(\lambda^t)$ provides the upper bound (UB') because the non-anticipativity constraint (2.44) may not be satisfied. In order to calculate the lower bound (LB'), we solve SPDD by fixing the values of $(\hat{\Gamma}, \hat{\phi}, \hat{\theta}, \hat{\psi})$ obtained by solving Lagrangian function $D_s(\lambda)$. Due to constraint (2.44) in SPDD, the x solutions would be the same; therefore, we find the lower bound. The algorithm terminates when one of the following stopping criteria has been reached: (i) maximum number of iterations, (ii) the upper bound has not improved after a given number of consecutive iterations, and (iii) optimality gap has been reached ($\frac{UB' - LB'}{UB'} < \epsilon$').

Algorithm 2 Subgradient Method

$UB' \leftarrow \infty, t \leftarrow 0, \lambda^0 \leftarrow 0, \varepsilon^0 \leftarrow 2$

Define LB' as the lower bound on the optimal value.

While (stopping criteria is not satisfied) **do**

Solve the Lagrangian function $D_s(\lambda^t)$

If$(\sum_{s\in S} D_s(\lambda^t) < UB')$ **then**

$UB' \leftarrow \sum_{s\in S} D_s(\lambda^t)$

end if

Evaluate the subgradient s^t

Calculate the step length $\mu^t = \dfrac{\varepsilon^t(\sum_{s\in S} D_s(\lambda^t) - LB')}{\| s^t \|^2}$

$\lambda^{t+1} \leftarrow \lambda^t + \mu^t s^t$

$t \leftarrow t + 1$

End-do

Due to the integrality conditions in the SP, there may be a duality gap between the optimal value of the Lagrangian dual (2.56) and the optimal value of the SP. Generally, the interdiction decisions may not be the same in all scenarios unless the duality gap vanishes. As identified in Algorithm 3, if constraint (2.44) is violated, we first group the scenarios with the same x_j solutions. Let $\mathcal{P}$ denote the groups of distinct solutions. For each solution $P \in \mathcal{P}$, we fix the value of x^P and resolve the SP to update the dual variables $(\Gamma^P, \phi^P, \theta^P, \psi^P, \omega^P)$. Among those solutions, we choose the one that maximizes the value of the objective function.

Algorithm 3 Dual Decomposition Algorithm for Solving the SP

For given $\hat{y}$, solve LD (2.56) with the subgradient algorithm

If $(\sum_{s\in S} H_{js}x_{js} = 0, \forall j \in J)$ **then**

The optimal solution of the SP is found.

Obtain dual variables $(\Gamma, \phi, \theta, \psi, \omega)$ and update UB

Else

Find scenarios with different x_j solutions and let $\mathcal{P}$ consist of the distinct solutions

While $(\mathcal{P} \neq \varnothing)$ **do**

Select and delete a solution P from $\mathcal{P}$.

Solve the SP with fixed x^P to find dual variables $(\Gamma^P, \phi^P, \theta^P, \psi^P, \omega^P)$ and the value of objective function η^P.

End while

Choose the solution with maximum η^P

Update UB and obtain dual variables $(\Gamma, \phi, \theta, \psi, \omega)$.

End if

2.4.3 Supervalid and Valid Inequalities

The convergence of the BD algorithm can be enhanced by adding supervalid inequality (SVI) and valid inequality (VI) to the MP. VI reduces the feasible LP relaxation region without eliminating any integer solutions. In contrast, SVI reduces the integer feasible region by eliminating integer solutions. SVI is guaranteed not to eliminate any optimal solutions unless the incumbent (best solution found so far) is optimal. The presented SVI in this paper is derived from (Wood, 2010) while VI has been derived by exploiting the characteristics of NDM.

Proposition 2.1. *Consider the Benders optimality cut*

$$\eta + \sum_{s\in S} p_s \left(\sum_j q_j y_j \hat{\psi}_{js} - \sum_j q_j y_j \xi_{js} \hat{\omega}_{js} \right) \geq \sum_{s\in S} p_s \left(\sum_k d_k \hat{\Gamma}_{ks} - \sum_i q_i' \hat{\theta}_{is} \right)$$

added to the MP. The following inequality is supervalid.

$$\sum_j I(\hat{\psi}_{js} - \xi_{js}\hat{\omega}_{js})y_j \geq 1 \tag{2.57}$$

where

$$I(\hat{\psi}_{js} - \xi_{js}\hat{\omega}_{js}) = \begin{cases} 1, & \textit{if } \sum_{s\in S}(\hat{\psi}_{js} - \xi_{js}\hat{\omega}_{js}) > 0 \\ 0, & \textit{otherwise} \end{cases}$$

Proof. The statement supposes that feasible solution $(\hat{y}, \hat{\psi}, \hat{\omega}, \hat{\Gamma}, \hat{\theta})$ generates the Benders cut (2.57). Let $\hat{\eta} = \sum_s p_s \left[\sum_k d_k \hat{\Gamma}_{ks} - \sum_i q_i' \hat{\theta}_{is} - \left(\sum_j q_j \hat{y}_j \hat{\psi}_{js} - \sum_j q_j \hat{y}_j \xi_{js}\hat{\omega}_{js}\right)\right]$. We assume that the incumbent leads to $\eta^* < \overline{\eta}$. Let (η^*, y^*) denote the optimal solutions to the MP and note that $\sum_s p_s \sum_j (\hat{\psi}_{js} - \xi_{js}\hat{\omega}_{js}) q_j y_j^* = \sum_j I(\hat{\psi}_{js} - \xi_{js}\hat{\omega}_{js}) y_j^* = 0$ or ≥ 1. When $\sum_s \sum_j I(\hat{\psi}_{js} - \xi_{js}\hat{\omega}_{js}) y_j^* = 0$,

$\eta^* \geq \sum_s p_s \left[\sum_k d_k \hat{\Gamma}_{ks} - \sum_i q_i' \hat{\theta}_{is} - \sum_j q_j y_j^* (\hat{\psi}_{js} - \xi_{js}\hat{\omega}_{js})\right]$ is true for any $(\hat{\psi}, \hat{\omega}, \hat{\Gamma}, \hat{\theta})$,

$= \hat{\eta} - \sum_s p_s \sum_j q_j y_j^* (\hat{\psi}_{js} - \xi_{js}\hat{\omega}_{js}) + \sum_s p_s \left(\sum_j q_j \hat{y}_j \hat{\psi}_{js} - \sum_j q_j \hat{y}_j \xi_{js}\hat{\omega}_{js}\right)$

$\geq \hat{\eta}$ because $\sum_s p_s \sum_j q_j y_j^* (\hat{\psi}_{js} - \xi_{js}\hat{\omega}_{js}) = 0$

$\geq \overline{\eta}$ because $\hat{y}$ does not need to be the incumbent solution and

$> \eta^*$ by assumption; but this is a contradiction.

Therefore, if the incumbent solution is not optimal, $\sum_j I(\hat{\psi}_{js} - \xi_{js}\hat{\omega}_{js}) y_j^* \geq 1$ must be true for every optimal solution (η^*, y^*). Hence, the inequality $\sum_j I(\hat{\psi}_{js} - \xi_{js}\hat{\omega}_{js}) y_j^* \geq 1$ is supervalid. □

In deriving the valid inequality (2.58), we use the observation that the post-interdiction cost is always greater than or equal to the pre-interdiction cost.

$$\eta \geq \sum_{i \in I} \sum_{j \in J} c'_{ij} u'_{ij} + \sum_{j \in J} \sum_{k \in K} c_{jk} u_{jk} \tag{2.58}$$

We add SVI and VI to the MP at each iteration of the Benders decomposition algorithm to improve the computation time. In Section 2.5, we compare the efficiency of acceleration techniques on the performance of the BD algorithm.

2.5 Computational Experiments

Extensive computational experiments are conducted to evaluate the performance of the proposed Benders decomposition algorithm and its algorithmic refinements. First, we analyze the performance of acceleration techniques over different problem instances. Instances vary in terms of size and parameter settings, e.g., the number of potential facilities, demand nodes, interdiction scenarios, the weight of post-interdiction cost, interdiction success probability, and interdiction budget. Next, we analyze the impact of key parameters on the CPU time and the obtained solution.

The algorithms are coded in C and run on a workstation with 3.10 GHz Intel Xeon E5 2687W V3 processor under the Linux environment using the Callable Library for CPLEX 12.10.0. In the following subsections, we summarize the results of our experiments.

2.5.1 Test Instances

The test instances are generated from the 2000 census data comprising 49, 88, and 150 largest cities in the continental United States (Snyder and Daskin, 2005). The 49-node data set consists of the states' capitals of the USA plus Washington D.C. The 88-node data set is obtained by considering the capitals and the 50 largest cities in the USA by omitting duplicates. The 150-node data set comprises the 150 largest cities in the USA. We consider five supply nodes to serve 49, 88, and 150 demand nodes. The set of supply nodes and candidates for intermediate facilities are chosen from demand sets, i.e., each city can simultaneously be the demand node, facility, and/or the supply node. The transportation cost is computed based on the greatest circle distance between two nodes. The capacity of intermediate facilities is set to $q_j = \mathcal{U}[0.1, 0.3] \times \sum_k d_k$. We also set the capacity of each supply node equal to the total demand ($q'_i = \sum_k d_k$). For every instance, we solve a p-median

model to find the location of supply nodes and the candidate locations for intermediate facilities. The interdiction budget B (expressed in terms of the number of facilities to be interdicted) varies from 1 to 5. The number of scenarios is set to $2^{|J|}$. We consider three different values 0.5, 0.75, and 0.9 for the success probability (p) to study the effects of varying success probability on the design and the performance. The weight of post-interdiction cost in the objective function (ρ_2) is varied from 0.2, 0.5 to 0.8. We use these values to analyze the effects of varying post-interdiction costs on the network design when it has low, neutral, and high values in the objective function.

Table 2.5 provides the details of the test problems. These problems are classified into nine sets based on the number of facilities, demand nodes, and the number of scenarios. Each set contains 45 instances obtained by varying the values of ρ_2, p, and B. Thus, our testbed comprises a total of 405 instances. These instances can be downloaded from `https://bit.ly/StochasticDesignInstances`. The CPU time limit for experiments with Sets I to VI is set to 5 hours (18,000 seconds), whereas for Sets VII to IX, it is set to 10 hours (36,000 seconds). The optimality gap for all instances is set to 0.1%.

Table 2.5: Summary of the Instance Sets used in the Computational Experiments

Instance Set	Supply Nodes ($\|I\|$)	Facilities ($\|J\|$)	Demand Nodes $\|K\|$	# Scenarios ($\|S\|$)	Time Limit (s)
I (Small)	5	10	49	1,024	18,000
II (Small)	5	10	88	1,024	18,000
III (Small)	5	10	150	1,024	18,000
IV (Moderate)	5	12	49	4,096	18,000
V (Moderate)	5	12	88	4,096	18,000
VI (Moderate)	5	12	150	4,096	18,000
VII (Large)	5	15	49	32,768	36,000
VIII (Large)	5	15	88	32,768	36,000
IX (Large)	5	15	150	32,768	36,000

The initial value of ε is in the interval (0,2] for the subgradient method used in the dual decomposition algorithm. If the value of the upper bound does not improve in 20 consecutive iterations, we decrease the value of ε to $\varepsilon/2$. Moreover, the algorithm terminates if the upper bound does not improve in 40 consecutive iterations. The maximum number of iterations for the subgradient method is set to 100, and the gap is $\epsilon' = 0.5\%$.

2.5.2 Analysis of Algorithmic Refinements

In the first part of the computational experiments, we analyze the effectiveness of the standard Benders decomposition method and the proposed algorithmic refinements. The four variants of the Benders decomposition algorithm are as follows:

- BD: This is the classic Benders decomposition algorithm, where we solve the MP and the SP iteratively.
- BD-VI: In this variant, we add SVI (2.57) and VI (2.58) to the MP and solve the MP and the SP iteratively.
- BDD: In this variant, the SP is solved using dual decomposition method. However, no valid inequalities are added. We solve the MP and the SP iteratively.
- BDD-VI: In this variant, the SP is solved using dual decomposition method, and we add SVI (2.57) and VI (2.58) to the MP. We solve the MP and the SP iteratively.

To compare the performance of the algorithms, we report the total number of instances solved to optimality using each version of the algorithm in Table 2.6 and the average gap and CPU time for different sets of instances in Table 2.7. The detailed results are provided in Table A.1 - Table A.3 in Appendix A.1.

Table 2.6: Number of Instances Solved to Optimality within the Time Limit

Set	BD	BD-VI	BDD	BDD-VI
I	45	45	45	45
II	45	45	45	45
III	22	45	45	45
IV	45	45	45	45
V	41	45	45	45
VI	14	38	42	45
VII	38	42	43	45
VIII	14	25	28	32
IX	9	10	13	22
Total	273/405 (67%)	340/405 (84%)	351/405 (87%)	369/405 (91%)

Results in Table 2.6 and 2.7 present the summary of improvements in CPU time and optimality

Table 2.7: Summary of the Performance of the Algorithms

	Average Gap (%)				Average Time (s)			
Set	BD	BD-VI	BDD	BDD-VI	BD	BD-VI	BDD	BDD-VI
I	0	0	0	0	1,399	1,216	1,061	867
II	0	0	0	0	4,264	3,261	2,686	1,953
III	2.97	0	0	0	17,143	13,648	12,064	8,708
IV	0	0	0	0	5,787	4,625	3,912	3,035
V	0.63	0	0	0	12,116	10,485	8,343	7,690
VI	4.34	0.69	0.13	0	17,210	15,683	13,496	10,474
VII	0.30	0.12	0.04	0	18,262	16,268	15,083	12,778
VIII	4.27	2.29	1.02	0.49	33,216	31,791	30,554	28,569
IX	13.78	8.07	4.76	1.72	34,014	33,380	32,980	31,974
Average	2.92	1.24	0.66	0.25	15,934	14,484	13,353	11,783

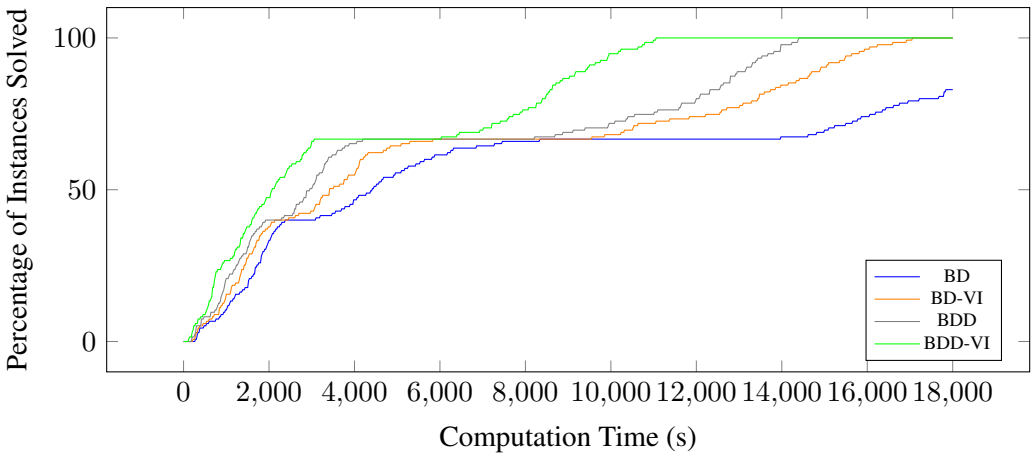
gap achieved by algorithmic refinements across the instance sets. The acceleration techniques provide significant contributions to the average optimality gap and the number of instances solved to optimality within the time limit. Results in Table 2.6 show that we are able to solve 67% (273 out of 405) of the instances to optimality using BD. Note that BD could solve some small and moderate set instances (Sets I, II, and IV) to optimality. The average optimality gap (across all 405 instances) is 2.92%. By adding the valid inequalities to the BD algorithm, we are able to solve 17% more (340 out of 405) instances to optimality. These instances belong to Sets III, V to IX. Moreover, the gap reduces by 57.5% and the CPU time reduces by 9%. On decomposing the SPs (BDD), we are able to solve 20% more instances to optimality compared to BD. Most of these instances belong to moderate and large-size instances (Sets V to IX). The average optimality gap reduces by 77% and the average CPU time reduces by 16%.

In the final version of the algorithm BDD-VI, we add two algorithmic refinements to the BD algorithm. The results show significant improvements in the number of instances solved to optimality, as well as the average optimality gap and CPU time over BD. More specifically, the results in Tables 2.6 and 2.7 show that we are able to solve 91% (369 out of 405) of the instances to optimality within the time limit. These instances belong to small (Sets I, II, and III), moderate (Sets IV, V, and VI), and large (Set VII) instances. Using BDD-VI results in a 91.5% reduction in the average gap and a 26% reduction in average CPU time compared to the BD algorithm. By comparing the results for large instances in Table 2.7, BD has an average gap of 6.1%, while BDD-VI improves the performance significantly and solves the instances with an average gap of 0.7%. The average

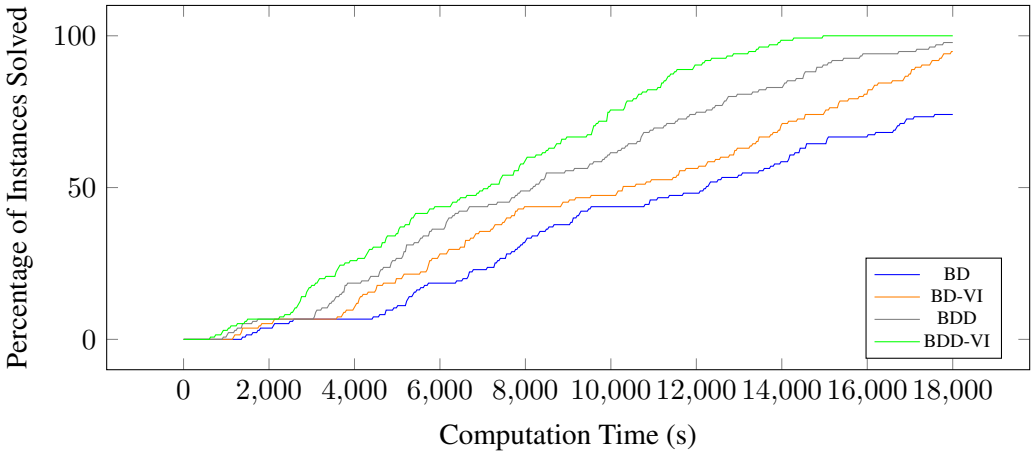
optimality gap for the instances that could not be solved to optimality (Sets VIII and IX) is 2.7% (range: 0.4% and 7.2%) compared to the average optimality gap of 12.1% (range: 0.9% to 45%) with the BD algorithm.

Analyzing the optimality gap of each algorithm across all the instances leads to the following observations: (i) algorithms are sensitive to the number of scenarios. As the number of scenarios increases, the gap increases, (ii) BDD performs better than BD and BD-VI because it solves the scenario SPs separately instead of solving the SP with all scenarios, (iii) the best algorithm is BDD-VI which solves more instances to optimality within the time limit, and (iv) for each $|J|$, as the number of demand nodes $|K|$ increases, the gap increases as well.

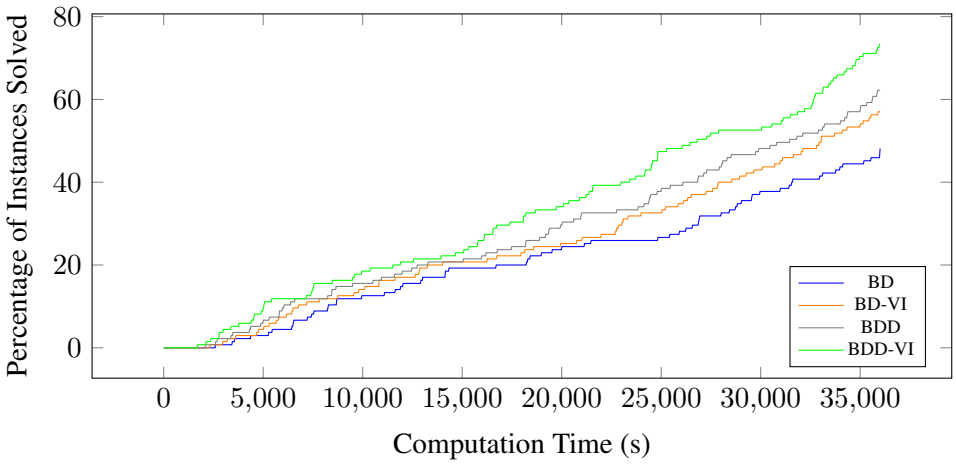
We compare the performance profile of algorithms in terms of CPU time for small, moderate, and large instances in Figure 2.3. The performance profile in Figure 2.3-(a) shows that, for small instances (Sets I, II, and III), BDD-VI solves 64 instances to optimality within 2,000 seconds of CPU time while BD solves only 45 instances (42% more instances). The figure also depicts that BDD-VI solves 56% of instances in 52% less time than the BD algorithm. Moreover, the time limit is reached for 17% of the instances using the BD algorithm, while the other three versions of the algorithm solve all of the small instances within the time limit. For moderate instances (Sets IV, V, and VI), we observe from Figure 2.3-(b) that (i) the BD algorithm can solve 11% of instances in less than 5,000 seconds of CPU time compared to 20%, 26.6%, and 34.8% for BD-VI, BDD, BDD-VI algorithms, respectively, and (ii) 74%, 94.8%, and 97.7% of instances are solved with BD, BD-VI, and BDD, respectively, while BDD-VI solves all instances to optimality in the time limit. For large instances (Sets VII, VIII, and IX), the results show that (i) BDD-VI solves 20% of instances with 35% less time compared with BD algorithm, and (ii) BDD-VI solves 73% of instances compared to BD, which is able to solve only 46% of instances within the time limit. We can conclude that the proposed acceleration techniques improve the BD algorithm's performance substantially. The superiority of the BDD-VI algorithm is demonstrated through the performance profile charts from small to large instances in Figure 2.3.



(a) Small Instances (Sets I, II, III)



(b) Moderate Instances (Sets IV, V, VI)



(c) Large Instances (Sets VII, VIII, IX)

Figure 2.3: Performance Profile of CPU Time for Different Algorithms

2.5.3 Performance on Larger-size Instances

In this section, we compare the performance of BD-VI and BDD-VI over larger-size instances. Table 2.8 provides the details of the larger test instances. For Set X to XIV, we vary the number of facilities from 16 to 20 and the number of customers from 49 to 88, and 150. In addition, we use "Capa" dataset where we vary the number of customers from 200 to 500, and 1000. The "Capa" dataset comprises 100 facilities and 1000 customers[1]. Instances with 200 and 500 customers are generated by selecting the first 200 and 500 customers from the 1000 customers. To choose the suppliers as well as the candidate locations for intermediate facilities, p-median problems are solved. The capacity of intermediate facilities and suppliers are set to $q_j = \mathcal{U}[0.1, 0.3] \times \sum_k d_k$, and $q'_i = \sum_k d_k$ respectively.

Table 2.8: Summary of the Large Instance Sets used in the Computational Experiments

Instance Set	Supply Nodes ($\|I\|$)	Facilities ($\|J\|$)	Demand Nodes ($\|K\|$)	# Scenarios ($\|S\|$)	Time Limit (s)
X	5	16	49, 88, 150	65,536	86,400
XI	5	17	49, 88, 150	130,072	86,400
XII	5	18	49, 88, 150	262,144	86,400
XIII	5	19	49, 88, 150	524,288	86,400
XIV	5	20	49, 88, 150	1,048,576	86,400
Capa-I	5	10	200, 500, 1000	1,024	86,400
Capa-II	5	12	200, 500, 1000	4,096	86,400
Capa-III	5	15	200, 500, 1000	32,768	86,400

To compare the performance of BD-VI and BDD-VI for larger-size instances, we run the experiments for Set X to XIV instances with $\rho_2 = 0.5$ and $p = 0.75$. We report the CPU time (s) and optimality gap (%) for BD-VI and BDD-VI in Table 2.9. The results in Table 2.9 depict the efficiency of the proposed algorithms to solve larger-size instances with up to 18 facilities, 150 customers, and 2^{18} scenarios. For instances with 19 facilities and 2^{19} scenarios, the optimality gap ranges from 0 to 12.4%. For instances with 20 facilities and 2^{20} scenarios, we face insufficient memory. In line with the results of Section 2.5.2, we conclude that BDD-VI solves more instances to optimality (68% compared with 58% of instances solved optimally with BD-VI). BDD-VI also outperforms BD-VI in terms of CPU time and optimality gap. The results in Table 2.10 confirm that BD-VI and BDD-VI can solve the "Capa" dataset instances with up to 1000 customers. As expected, BDD-VI outperforms BD-VI in larger-size instances. More specifically, BDD-VI reduces

[1] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/capinfo.html

the CPU time by 17.6%, 15%, and 13.5% compared with BD-VI for instances with 200, 500, and 1000 customers, respectively.

Table 2.9: Summary of the Performance of BD-VI and BDD-VI for Larger-size Instances

		$\|K\| = 49$				$\|K\| = 88$				$\|K\| = 150$			
$\|J\|$	B	BD-VI		BDD-VI		BD-VI		BDD-VI		BD-VI		BDD-VI	
		Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
16	1	0	27,194	0	22,526	0	40,352	0	32,726	0	55,085	0	45,628
	2	0	34,086	0	29,159	0	45,540	0	37,854	0	63,573	0	50,524
	3	0	46,632	0	38,291	0	53,228	0	42,687	0	69,615	0	60,208
	4	0	48,335	0	42,516	0	58,002	0	48,062	0	75,883	0	67,175
	5	0	55,888	0	47,258	0	64,830	0	52,341	0	82,052	0	77,955
17	1	0	51,438	0	44,753	0	63,857	0	49,869	0	72,960	0	61,217
	2	0	57,823	0	49,707	0	70,542	0	55,403	0	79,931	0	67,083
	3	0	64,742	0	52,843	0	75,634	0	63,500	0	86,257	0	76,910
	4	0	72,943	0	64,287	0	83,582	0	69,385	2.3	86,400	0	84,394
	5	0	81,358	0	66,341	1.5	86,400	0	79,247	3.6	86,400	2.9	86,400
18	1	0	63,826	0	52,633	0	71,837	0	60,854	2.2	86,400	0	85,175
	2	0	68,237	0	59,871	0	78,518	0	68,173	3.4	86,400	2.9	86,400
	3	0	78,524	0	72,370	0	83,694	0	79,237	4.2	86,400	3.3	86,400
	4	0	82,421	0	76,110	2.1	86,400	0	82,145	4.9	86,400	4.2	86,400
	5	2.8	86,400	0	81,754	3.5	86,400	2.8	86,400	5.9	86,400	5.1	86,400
19	1	0	83,706	0	75,259	2.1	86,400	1.8	86,400	3.4	86,400	2.6	86,400
	2	2.2	86,400	0	84,697	3.1	86,400	2.4	86,400	4.5	86,400	3.8	86,400
	3	3.7	86,400	3.1	86,400	4.3	86,400	3.6	86,400	6.7	86,400	5.9	86,400
	4	4.6	86,400	3.9	86,400	5.5	86,400	4.8	86,400	9.3	86,400	8.9	86,400
	5	5.5	86,400	4.8	86,400	7.3	86,400	6.7	86,400	12.4	86,400	10.4	86,400
Average		0.94	67,458	0.61	60,979	1.47	74,041	1.11	66,994	3.14	81,108	2.50	77,013

Table 2.10: Summary of the Performance of BD-VI and BDD-VI for "Capa" Instances

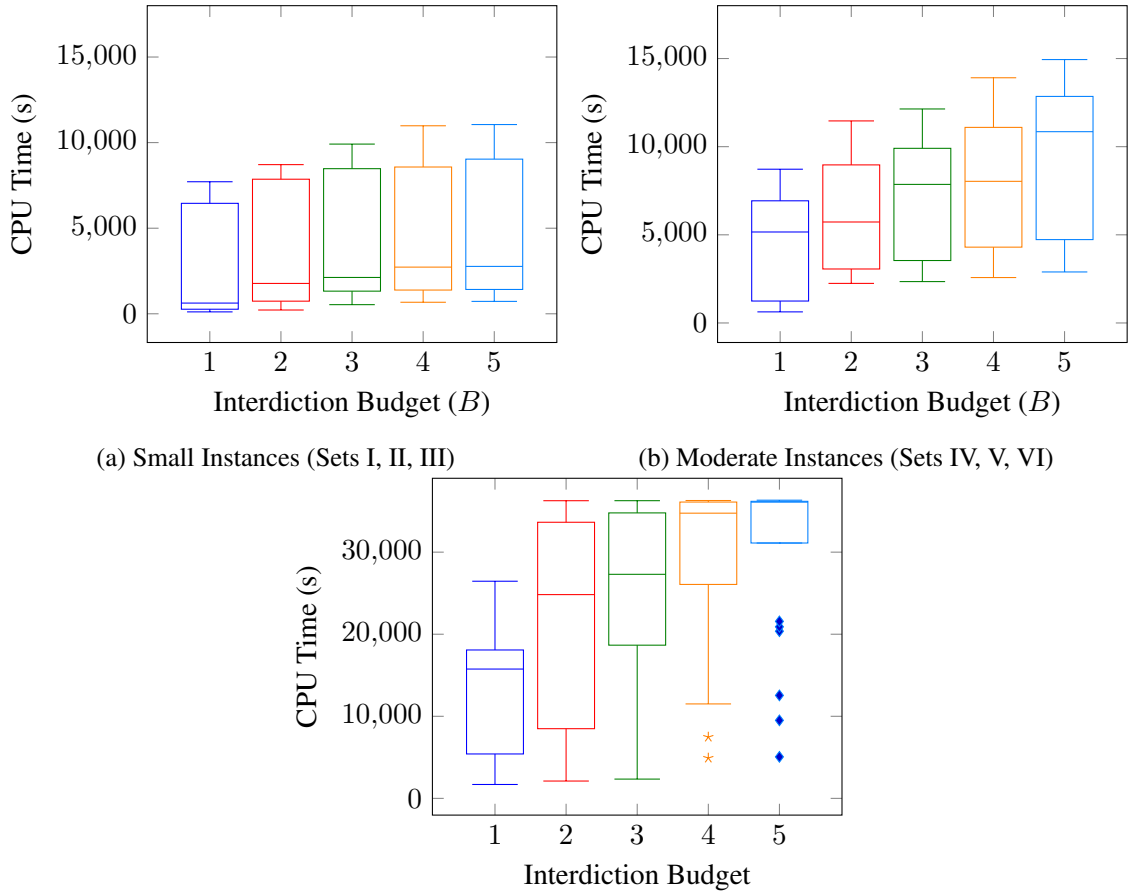
		Computation Time (s)					
		$\|K\| = 200$		$\|K\| = 500$		$\|K\| = 1000$	
$\|J\|$	B	BD-VI	BDD-VI	BD-VI	BDD-VI	BD-VI	BDD-VI
10	1	12,523	9,286	18,259	13,742	24,492	19,724
	2	14,508	10,652	23,638	16,751	31,714	23,754
	3	15,943	11,834	27,937	22,710	36,058	30,054
	4	17,872	12,530	29,630	25,721	40,394	36,841
	5	19,837	15,731	33,712	29,341	43,567	39,151
12	1	18,967	11,842	25,710	18,755	30,333	23,467
	2	20,635	14,637	30,934	23,515	36,069	30,562
	3	22,821	16,341	35,564	29,410	41,810	36,085
	4	26,682	19,751	40,800	34,681	47,375	42,251
	5	29,927	23,741	45,375	39,417	53,381	49,541
15	1	25,413	21,071	36,217	28,978	40,837	33,754
	2	38,716	37,008	42,607	39,014	48,571	43,541
	3	39,937	37,890	48,839	42,184	55,356	48,063
	4	43,873	39,427	51,252	48,370	60,271	54,517
	5	45,594	42,271	55,873	51,762	70,714	60,541
Average		26,217	21,601	36,423	30,957	44,063	38,123

* These instances were solved to optimality within the time limit.

2.5.4 Sensitivity Analysis of Model Parameters on Computation Time

We analyze the impact of various model parameters such as interdiction budget, interdiction success probability, and the relative weight of post-interdiction cost on the computation time using the results obtained with BDD-VI algorithm.

As shown in Figure 2.4, increasing the budget (B) results in an increase in the average CPU time across all the instances. Increasing the interdiction budget results in opening more facilities in the distribution network. As a result, the combination of interdictions to consider in the SP increases, resulting in increased CPU time.



(a) Small Instances (Sets I, II, III)

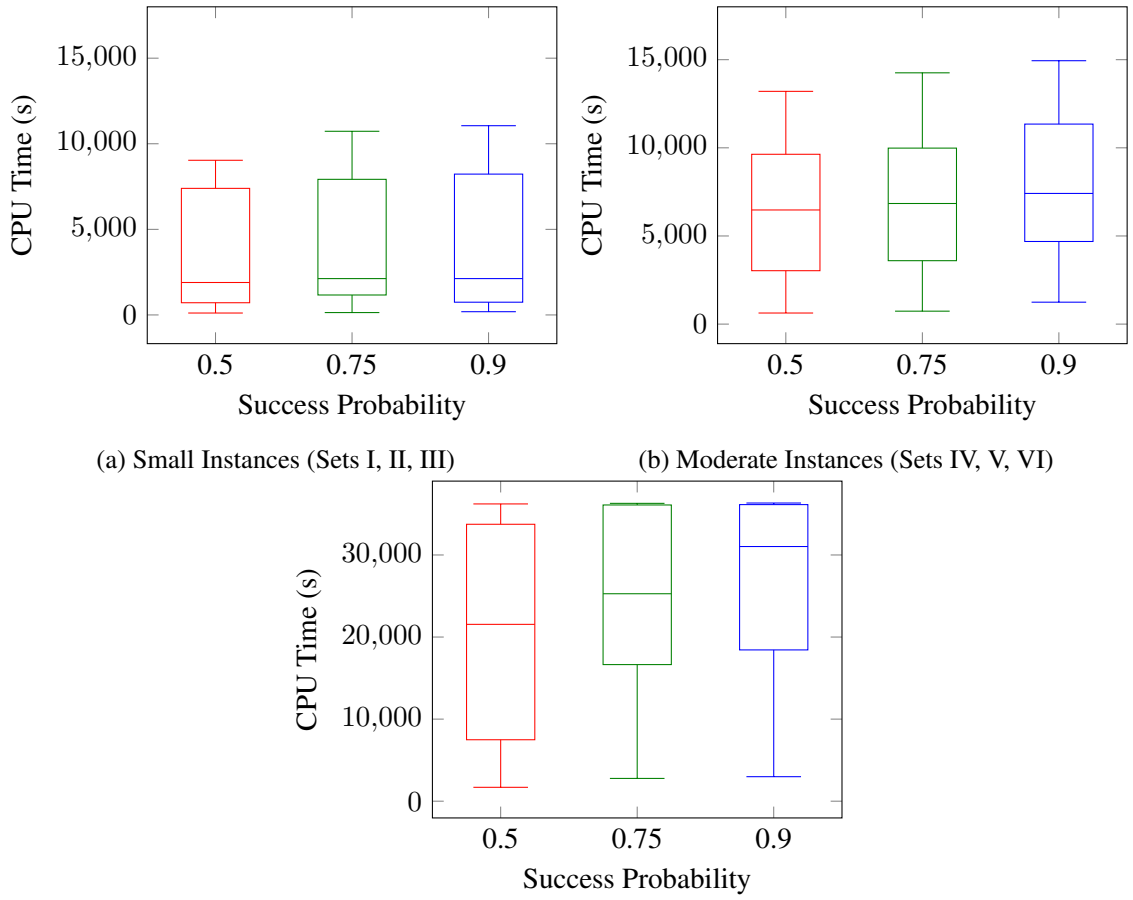
(b) Moderate Instances (Sets IV, V, VI)

(c) Large Instances (Sets VII, VIII, IX)

Figure 2.4: Effects of Varying Interdiction Budget (B) on Computation Time

The effects of varying interdiction success probability (p) on the CPU time are shown in Figure 2.5. The results reveal that increasing interdiction success probability from 0.5 to 0.9 increases the

average CPU time by 21%, 21%, and 28% for small, moderate, and large instances, respectively.



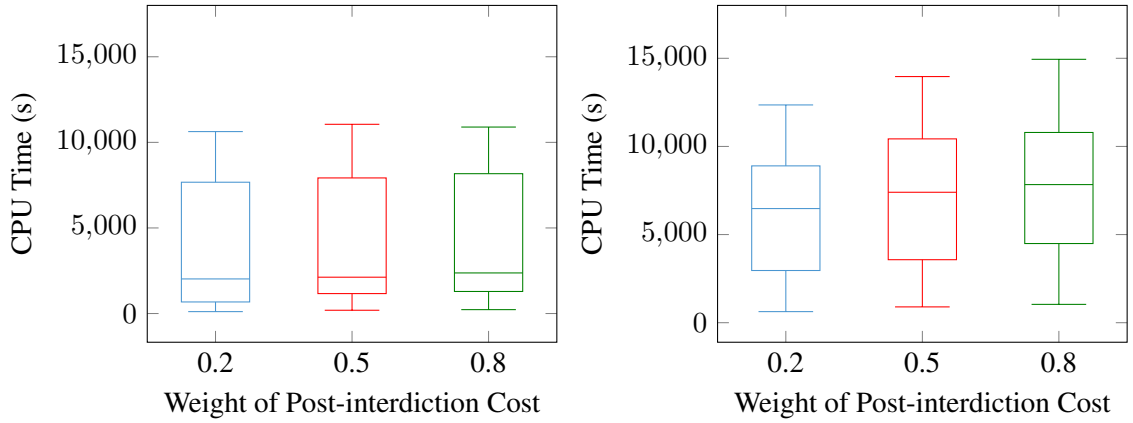
(a) Small Instances (Sets I, II, III)

(b) Moderate Instances (Sets IV, V, VI)

(c) Large Instances (Sets VII, VIII, IX)

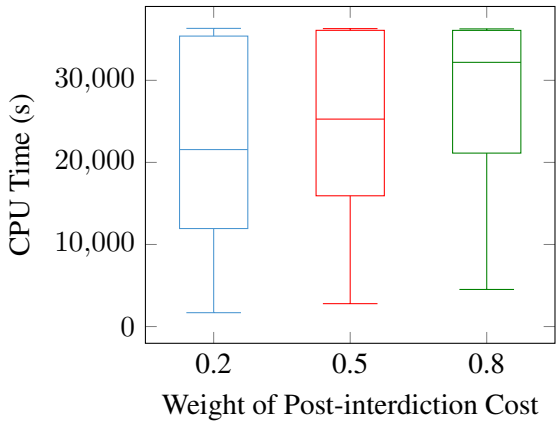
Figure 2.5: Effects of Varying Interdiction Success Probability (p) on Computation Time

The effects of changing the weight of post-interdiction cost (ρ_2) on CPU time are illustrated in Figure 2.6. This analysis reveals that increasing the relative importance of post-interdiction cost leads to an increase in CPU time. For example, increasing the weight of post-interdiction cost from 0.2 to 0.8 increases the average CPU time by 11%, 24%, and 23% for small, moderate, and large instances, respectively. This is because increasing the importance of post-interdiction cost leads to locating more facilities. Such an increase makes the SP difficult.



(a) Small Instances (Sets I, II, III)

(b) Moderate Instances (Sets IV, V, VI)



(c) Large Instances (Sets VII, VIII, IX)

Figure 2.6: Effects of Varying the Weight of Post-interdiction Cost (ρ_2) on Computation Time

2.5.5 Sensitivity Analysis of Model Parameters on Distribution Network Design

In this section of computational experiments, we analyze the effects of interdiction budget, success probability, and the weight of post-interdiction cost on the design of the distribution network. Figure 2.7 depicts the effects of varying interdiction budget on the number of open facilities for Set I at varying levels of success probability while using the weight of post-interdiction cost 0.5 ($\rho_2 = 0.5$). The results show that increasing the interdiction budget results in an increase in the number of open facilities.

In Table 2.11, we analyze the impact of interdiction budget on network costs using instance Set I with $\rho_2 = 0.5$ and $p = 0.5$. The results indicate that an increase in the interdiction budget leads to an increase in the number of open facilities, which also helps reduce pre-interdiction cost. The

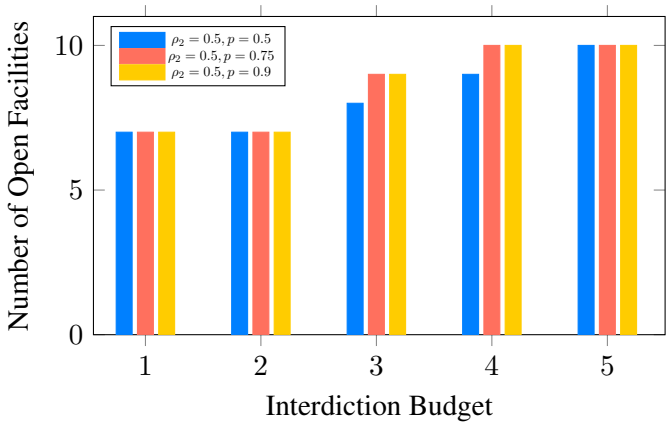


Figure 2.7: Effects of Varying Interdiction Budget on the Number of Open Facilities for Set I

rationale is that as B increases, the model prescribes more facilities, allowing demand nodes to be served by nearby facilities.

Table 2.11: Effects of Varying Interdiction Budget on Network Costs

Interdiction Budget (B)	Installation Cost ($)	Pre-interdiction Cost ($)	Expected Post-interdiction Cost ($)	Total Obj.
1	799,200	1,132,922	1,392,510	2,061,916
2	799,200	1,132,922	1,598,352	2,164,837
3	907,800	1,077,483	1,767,233	2,330,158
4	1,024,500	1,077,483	1,839,460	2,482,972
5	1,169,400	1,076,019	1,784,796	2,599,808

Using the same instance Set I with interdiction budget of 3 and $\rho_2 = 0.5$, we illustrate the implications of varying levels of interdiction success probability on the network design in Figure 2.8. In Figure 2.8, the worst-case happens when all the interdictions are successful ($\xi_{js} = 1$). When probability is 0.75, we open an additional facility compared with probability of 0.5. However, by changing the probability from 0.75 to 0.9, the number of open facilities does not change, but different facilities are chosen. The increase in the interdiction success probability causes more installation cost in the design stage.

We compare the costs for demand satisfaction of each design in Table 2.12. On the one hand, we observe that the installation and pre-interdiction costs for $p = 0.9$ are higher than $p = 0.75$. On the other hand, the post-interdiction cost reduces due to a more robust network design. The

proposed solution provides a robust network since we observe a higher number of open facilities for higher values of interdiction success probability. According to Table 2.12, the worst-case post-interdiction cost for $p = 0.9$ decreases 12.2% by 3.5% more installation cost compared with the costs of $p = 0.75$.

Figure 2.8: Effects of Varying p on the Design for Set I, $\rho_2 = 0.5$ and $B = 3$. (a) Pre-interdiction Flow for $p = 0.5$, (b) Worst-case Post-interdiction Flow for $p = 0.5$, (c) Pre-interdiction Flow for $p = 0.75$, (d) Worst-case Post-interdiction Flow for $p = 0.75$, (e) Pre-interdiction Flow for $p = 0.9$, (f) Worst-case Post-interdiction Flow for $p = 0.9$

The impact of changing the weight of post-interdiction costs, ρ_2, is presented in Figure 2.9

Table 2.12: Effects of Varying Interdiction Success Probability on Network Costs

p	Installation Cost	Pre-interdiction		Post-interdiction	
		Transportation Cost	Per Unit Transportation Cost	Worst-case Transportation Cost	Per Unit Transportation Cost
0.50	907,800	1,077,483	441	3,151,137	1,290
0.75	1,024,500	1,077,483	441	2,665,473	1,091
0.90	1,060,800	1,131,318	463	2,340,339	958

where we compare the designs for instance Set I, with an interdiction budget of 4, and an interdiction success probability of 0.5. When the relative weight of ρ_2 is low, the solution obtained prioritizes low installation cost at the expense of lost sales in a worst-case interdiction scenario of 4 disabled facilities as shown in Figures 2.9-(a) and (b). Increasing the relative weight of post-interdiction cost to 0.5 and 0.8 results in a higher number of open facilities and installation cost while decreasing the post-interdiction cost. The results in Table 2.13 indicate that by increasing the weight of the post-interdiction cost from 0.5 to 0.8, the installation and pre-interdiction costs increase by 3.5% and 5%, respectively, while the worst-case post-interdiction cost decreases by 23.3%.

Table 2.13: Effects of Varying the Weight of Post-interdiction Cost on Network Costs

Weight of Post-interdiction Cost	Installation Cost	Pre-interdiction		Post-interdiction	
		Transportation Cost	Per Unit Transportation Cost	Worst-case Transportation Cost	Per Unit Transportation Cost
0.20	907,800	1,077,483	441	6,231,735	2,551
0.50	1,024,500	1,077,483	441	3,339,391	1,367
0.80	1,060,800	1,131,318	463	2,560,058	1,048

2.5.6 Computational Results for the Correlated Facility Interdictions

In this section, we analyze the effects of correlated facility interdictions on the performance of the algorithms and the optimal design. To define the correlation matrix between the facilities, we use the transportation cost information between facilities. We consider five levels of correlation as follows:

Figure 2.9: Effects of Varying ρ_2 on the Design for Set I, $p = 0.5$ and $B = 4$. (a) Pre-interdiction Flow for $\rho_2 = 0.2$, (b) Worst-case Post-interdiction Flow for $\rho_2 = 0.2$, (c) Pre-interdiction Flow for $\rho_2 = 0.5$, (d) Worst-case Post-interdiction Flow for $\rho_2 = 0.5$, (e) Pre-interdiction Flow for $\rho_2 = 0.8$, (f) Worst-case Post-interdiction Flow for $\rho_2 = 0.8$

$$\varrho_{jj'} = \begin{cases} 1, & \text{if } c_{jj'} \leq 250 \\ 0.75, & \text{if } 250 < c_{jj'} \leq 500 \\ 0.5, & \text{if } 500 < c_{jj'} \leq 1000 \\ 0.25, & \text{if } 1000 < c_{jj'} \leq 1500 \\ 0, & \text{if } c_{jj'} > 1500 \end{cases}$$

Using the instance Set I with interdiction budget of 2, $\rho_2 = 0.5$, and $p = 0.75$, we demonstrate the designs of the model without correlation and the model with correlation in Figure 2.10. In Figure 2.10-(d), the capacity of facilities represented by green triangles is impacted by the interdiction of other facilities. The interdiction of the two facilities does not influence the capacity of the red facilities shown in Figure 2.10-(d). We observe that, when correlation exists among facilities, we need to open more facilities in order to meet the demand after interdiction. Furthermore, the interdiction strategy is affected by the correlation. In Figure 2.10-(b), the interdiction strategy involves interdicting facilities that are distant from others, resulting in increased transportation costs to satisfy demand. However, in Figure 2.10-(d), the interdictor interdicts the facilities in close proximity to others because the interdiction of one facility reduces the capacity of the nearby facilities as well. We observe that in Figure 2.10-(d), the demand of multiple customers is satisfied by more than one facility. We conduct a comparison of network costs in Table 2.14. The results indicate that considering the correlations, the installation cost increases significantly to hedge against the effects of correlated facility interdictions. As a result, the worst-case post-interdiction cost increases only 5.6% in the correlated case.

Table 2.14: Effects of Correlation on Network Costs

	Installation Cost	Pre-interdiction Transportation Cost	Worst-case Post-interdiction Transportation Cost	Open Facilities
No Correlation	799,200	1,132,922	2,395,631	CA,NY,TX,PA,IL,OH,CO
Correlation	1,169,400	1,076,019	2,530,067	CA,NY,TX,FL,PA,IL,OH,NC,WA,CO

We report the CPU time (s) and Gap (%) for BD-VI and BDD-VI in Table 2.15 for Set I to Set IX with $\rho_2 = 0.5$ and $p = 0.75$. The results indicate that considering correlation among facilities increases the complexity of the model as the CPU time increases significantly. We also observe that finding the design and interdiction strategies for $B = 2$ is the most challenging among all interdiction budgets. This can be explained by the fact that finding the interdiction strategy to attack two facilities with the highest correlation is harder than finding the strategy for attacking e.g., four facilities with the highest correlation, increasing the complexity of the subproblem at each iteration of the algorithm. Finally, we conclude that BDD-VI outperforms BD-VI in terms of CPU time and gap for the correlated facility interdiction model (improving average CPU time and average gap by

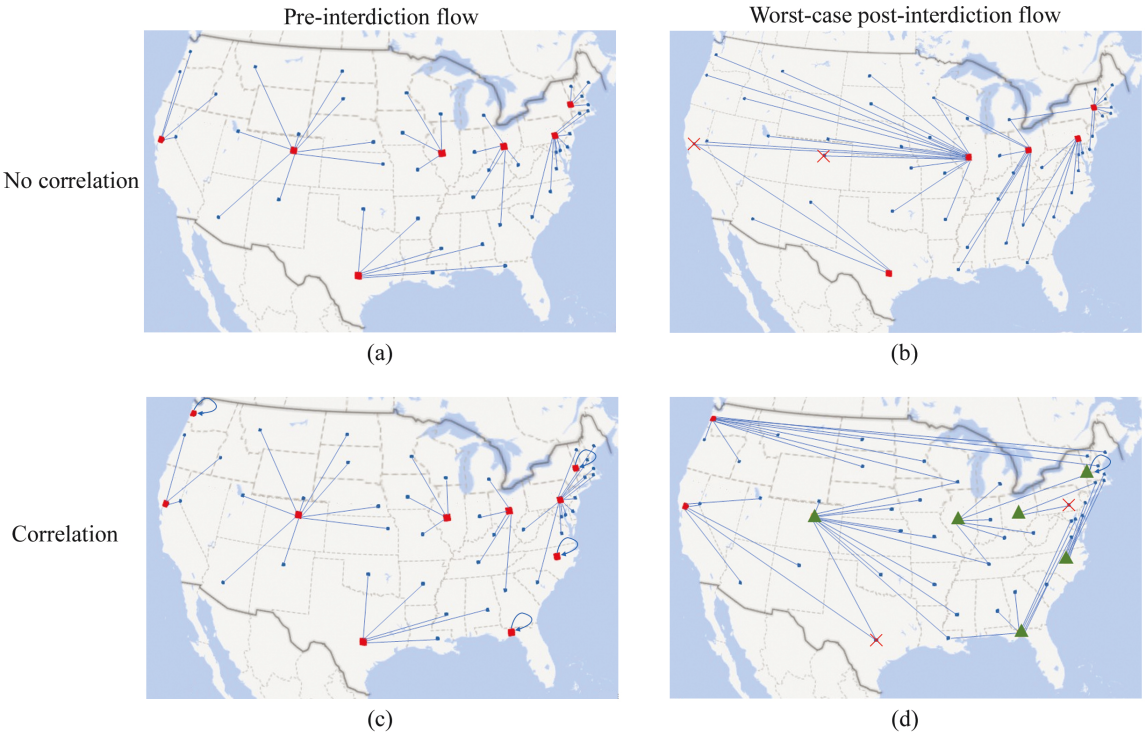


Figure 2.10: Effects of Correlation on Network Design (a) Pre-interdiction Flow without Correlation, (b) Worst-case Post-interdiction Flow without Correlation, (c) Pre-interdiction Flow with Correlation, (d) Worst-case Post-interdiction Flow with Correlation

3% and 17%, respectively).

Table 2.15: Summary of the Performance of BD-VI and BDD-VI for Instances with Correlated Facility Interdictions

$\|J\|$	B	$\|K\| = 49$				$\|K\| = 88$				$\|K\| = 150$			
		BD-VI		BDD-VI		BD-VI		BDD-VI		BD-VI		BDD-VI	
		Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
10	1	0	4,276	0	3,742	0	13,527	0	11,863	0	26,238	0	23,674
	2	0	16,404	0	14,577	0	27,613	0	25,317	0	62,281	0	56,712
	3	0	13,125	0	11,862	0	22,652	0	20,739	0	46,051	0	42,051
	4	0	16,137	0	13,505	0	24,698	0	22,621	0	45,332	0	41,308
	5	0	21,658	0	17,779	0	22,652	0	18,912	0	21,992	0	19,081
12	1	0	28,637	0	24,843	0	47,539	0	41,931	2.2	86,400	1.8	86,400
	2	0	72,419	0	65,831	2.1	86,400	0	83,628	3.3	86,400	2.9	86,400
	3	0	64,953	0	60,374	0	77,234	0	73,711	2.8	86,400	2.2	86,400
	4	0	62,273	0	58,224	1.8	86,400	1.1	86,400	2.3	86,400	1.9	86,400
	5	0	59,214	0	51,760	2.1	86,400	1.7	86,400	3.1	86,400	2.5	86,400
15	1	2.3	86,400	1.7	86,400	3.4	86,400	2.4	86,400	4.8	86,400	4.1	86,400
	2	4.9	86,400	3.8	86,400	6.5	86,400	5.8	86,400	9.1	86,400	8.4	86,400
	3	4.1	86,400	3.3	86,400	4.4	86,400	4.1	86,400	6.6	86,400	5.9	86,400
	4	3.8	86,400	3.1	86,400	5.2	86,400	4.6	86,400	7.7	86,400	6.8	86,400
	5	3.2	86,400	2.3	86,400	5.1	86,400	4.5	86,400	6.9	86,400	6.3	86,400
Avgerage		1.22	52,740	0.95	50,300	2.04	61,808	1.61	60,235	3.25	71,060	2.85	69,788

2.5.7 Value of using Stochastic Design Model

In the last part of the computational experiments, we compare the results of our stochastic model with the deterministic counterpart. Note that when $p = 1$, our stochastic model reduces to the deterministic problem as all interdictions are successful. In Table 2.16, we report the design and costs of the stochastic and the deterministic models for three selected instances from Sets I, IV, and VII. The results of $|J| = 10$ and $|J| = 12$ show that we have to pay more installation cost to reduce the post-interdiction cost in deterministic design compared with the stochastic design with $p = 0.75$ and $p = 0.5$. At $p = 0.9$, the interdiction success probability is high; therefore, the obtained design is the same as the deterministic design for $|J| = 10$ and $|J| = 12$. At $|J| = 15$, we have to open 11 facilities with the deterministic model while the number of open facilities decreases by reducing the interdiction success probability. Since the probability of worst-case interdiction is reduced with lower p, fewer facilities are opened to avoid higher installation costs. Therefore, the presented stochastic model provides a less conservative design compared to the deterministic model by opening fewer facilities at the expense of a slight increase in the worst-case post-interdiction transportation cost.

Table 2.16: Cost Analysis of Stochastic and Deterministic Models

$\|J\|$	ρ_2	B		Open Facilities	Installation Cost	Pre-interdiction Transportation Cost	Worst-case Post-interdiction Transportation Cost	Expected Post-interdiction Transportation Cost
10	0.8	2	Deterministic	1,2,3,4,5,6,7,9,10	1,024,500	1,077,484	1,956,054	
			Stochastic $p = 0.9$	1,2,3,4,5,6,7,9,10	1,024,500	1,077,484	1,956,054	1,823,930
			Stochastic $p = 0.75$	1,2,3,5,6,7,9,10	915,900	1,132,922	2,137,668	1,771,134
			Stochastic $p = 0.5$	1,2,3,5,6,7,10	799,200	1,132,922	2,395,631	1,598,353
12	0.5	2	Deterministic	2,5,6,7,9,10,11,12	567,200	1,168,836	1,870,264	
			Stochastic $p = 0.9$	2,5,6,7,9,10,11,12	567,200	1,168,836	1,870,264	1,789,887
			Stochastic $p = 0.75$	2,5,6,7,9,10,12	490,100	1,168,836	2,196,753	1,757,380
			Stochastic $p = 0.5$	2,5,6,9,10,12	424,100	1,168,967	2,547,331	1,599,521
15	0.2	4	Deterministic	1,2,3,5,6,7,10,11,13,14,15	808,400	719,032	2,027,141	
			Stochastic $p = 0.9$	1,2,3,5,6,8,10,11,13,15	711,800	752,589	2,166,820	1,940,758
			Stochastic $p = 0.75$	1,2,3,5,6,9,10,14,15	684,100	801,052	2,205,598	1,695,312
			Stochastic $p = 0.5$	1,2,3,6,9,10,11,13,115	721,600	724,251	2,262,090	1,398,630

In order to demonstrate the benefit of using our proposed stochastic model, we compare the performance of the networks generated by our proposed model and the deterministic model when they are subject to the same interdiction outcome scenarios. To achieve this, we run the bi-level stochastic interdiction model on the facility decisions obtained by the deterministic model in order to obtain the expected post-interdiction transportation cost. We present the results of experiments conducted

at varying instance sizes and p values in Table 2.17. The comparison is made by comparing the difference between the installation cost and the expected post-interdiction transportation cost between the deterministic model and the stochastic variant. The results show that the increase in installation cost for using the deterministic design is more than the savings in the expected post-interdiction transportation cost. Therefore, one can not justify using the more conservative deterministic design when there is uncertainty in the outcome of interdictions.

Table 2.17: The Cost Increase of using Deterministic Design in Stochastic Interdiction Setting

$\|J\|$	ρ_2	B	p	Installation Cost		Expected Post-interdiction Transportation Cost		Cost Change (Deterministic-Stochastic)	
				Deterministic	Stochastic	Deterministic	Stochastic	Installation Cost	Expected Cost
10	0.8	2	0.9	1,024,500	1,024,500	1,823,930	1,823,930		
			0.75	1,024,500	915,900	1,667,910	1,771,134	108,600	-103,224
			0.5	1,024,500	799,200	1,461,650	1,598,353	225,300	-136,703
12	0.5	2	0.9	567,200	567,200	1,789,887	1,789,887		
			0.75	567,200	490,100	1,693,586	1,757,380	77,100	-63,794
			0.5	567,200	424,100	1,491,122	1,599,521	143,100	-108,399
15	0.2	4	0.9	808,400	711,800	1,847,464	1,940,758	96,600	-93,294
			0.75	808,400	684,100	1,631,968	1,695,312	124,300	-63,344
			0.5	808,400	721,600	1,314,709	1,398,630	86,800	-83,921

2.6 Conclusion

In this paper, we introduce a tri-level interdiction model for designing robust distribution network that accounts for the effect of disruptions of intermediate facilities under interdiction uncertainty. The resulting tri-level stochastic model has been transformed into a bi-level stochastic formulation. To solve the proposed formulation efficiently, we present an exact solution approach based on the Benders decomposition algorithm, where we solve the stochastic subproblem using dual decomposition. In addition, we strengthen the Benders decomposition by adding supervalid and valid inequalities to the master problem. The results show that 91% of instances are optimally solved using algorithmic refinements compared to 67% of instances with the basic Benders decomposition algorithm. In terms of computational time, the refinements reduce the CPU time by 35%. Computational results on large instances with up to 18 facilities, and 2^{18} interdiction scenarios confirm the efficiency of the algorithms. The insights from the experiments show that the correlation effect of interdiction increases the network installation costs. We also illustrate that the proposed

network design model with stochastic interdiction outcomes yields a less conservative design solution in comparison with its deterministic variant.

Chapter 3

An Exact Algorithm for Multicommodity Network Design under Stochastic Interdictions

Abstract In this paper, we study the multicommodity network design problem by considering the effects of worst-case disruptions under the uncertain interdiction budget. The goal is to install links between nodes to satisfy the demand of different commodities with minimum installation cost and the weighted sum of flow costs before and after interdictions. Using the designer-attacker-designer framework, we present a trilevel mixed-integer stochastic network design model. In the first level, the designer selects a subset of links to install and route flows under normal conditions. Most studies in the literature assume that the interdiction budget is known to the decision maker (network designer) with certainty; however, in practice, the designer is not aware of interdiction capabilities. Therefore, the designer's objective is to minimize the installation cost, and the weighted sum of pre-interdiction and expected post-interdiction costs. In the second level, the attacker interdicts a subset of installed arcs with a limited interdiction budget. In the third level, the designer optimizes the flow over the surviving links. Moreover, we extend the model to consider the uncertainty in the demand. We present a branch-and-Benders-cut algorithm to solve the proposed model. The algorithm is enhanced through the use of several features such as the use of multicut

reformulation, warm start, variable fixing, cut selection, penalty reformulation, generation of strong Pareto-optimal cuts, use of supervalid and valid inequalities. Extensive computational experiments were performed to evaluate the efficiency and robustness of the proposed algorithmic refinements. We compare the performance of our algorithm with a state-of-the-art, general-purpose stochastic mixed-integer bilevel linear optimization solver and show that our algorithm is faster by orders of magnitude. Our results demonstrate that the branch-and-Benders-cut algorithm combined with some of these acceleration techniques solves large-scale benchamark instances with up to 20 nodes, 220 arcs and 200 commodities. Furthermore, we highlight the advantages of stochastic design over deterministic design when the number of interdictions is uncertain.

3.1 Introduction

Network design problems (NDPs) are fundamental problems that arise in the strategic design and operational planning of telecommunication networks (Feremans et al., 2003), supply chains (Klibi et al., 2010), power systems (Costa et al., 2011), transportation and logistics systems (Crainic, 2000), among others (Crainic et al., 2021). The goal of most NDPs is to select a set of arcs and/or nodes to install such that a trade-off between the operational cost and design cost is achieved while the demand is satisfied. NDPs can be categorized into single and multicommodity variants based on demand characteristics. In single-commodity variants, the demand at each node can be satisfied by any of the other nodes' supply since they all route the same commodity. However, in the multicommodity variant, demand is expressed as origin-destination pairs, and hence the demand of the destination node must be met by the corresponding supply node (Zetina et al., 2019). The multicommodity network design problem (MCNDP) is a class of NDPs with applications in transportation, logistics, telecommunications, power distribution and production systems, among others. In MCNDPs, the objective is to install a subset of arcs to route different commodities (e.g. people, messages, goods) from specific origins to their destinations at minimal design and flow costs (Gendron et al., 1999). MCNDPs can be categorized into uncapacitated and capacitated variants based on the capacities on the arcs and/or nodes (see Yaghini et al. (2015), Zetina et al. (2019),Crainic et al. (2021) and references therein). Other variants of MCNDPs include dynamic (time-varying)

demand (Fragkos et al., 2017), demand uncertainty (Wang et al., 2019; Sarayloo et al., 2021), uncertain demand and arc capacities (Crainic et al., 2021), uncertain travel times (Lanza et al., 2021), and service requirements like latency, transit time, and transshipment limitations (Balakrishnan et al., 2017; Gudapati et al., 2022). MCNDPs are $\mathcal{NP}$-hard (Paraskevopoulos et al., 2016).

Multicommodity networks are complex networks of several facilities (nodes) connected by links (e.g. roads, railways, cables, pipelines etc.) to facilitate the flow of commodities, passengers and/or information from several origins to destinations. Disruptions of multicommodity networks can have a devastating impact on social well-being and economy. It can cause momentous destruction to properties, losses of human life, and extensive interruptions in vital services. Recent events have exposed the vulnerability of these systems to disruptions due to intentional human actions (e.g. terrorist attacks, labor strikes). With the increasing interconnectivity of networks, a disruption in one network can negatively affect other networks (Cappanera and Scaparra, 2011).

To consider the effects of disruptions, survivable network design (SND) models have been introduced. SND problems aim to minimize the design costs while satisfying specific survivability conditions in the presence of node or arc failures. The survivability conditions can be the amount of demand satisfied after failure, the lengths of the path for demand satisfaction, or the number of paths between any two nodes (Kerivin and Mahjoub, 2005). Garg and Smith (2008) present an SND model by considering reliable and unreliable arcs where unreliable arcs fail with a specific probability. Azizi et al. (2016) present models for designing hub-and-spoke networks with hub failures by minimizing the weighted sum of the transportation cost in normal conditions and the expected transportation cost when a hub fails. Mohammadi et al. (2017) also consider the expected cost for designing the multicommodity hazardous material transportation network. They consider hub failures as well as link failures. These papers assume that failures are random and they intend to minimize the expected cost of failures. Therefore, the design solution may not function efficiently under worst-case disruptions (i.e., optimal interdiction by an intelligent attacker). To overcome this issue, network interdiction models are used to model the effects of worst-case disruptions on networks.

Interdiction models are two-player sequential games between the interdictor and the defender. First, the interdictor attacks the network to maximize the damage. Next, the network operator acts

to optimize the network operations in the residual network. Interdiction models were initially developed for military and homeland security applications (McMasters and Mustin, 1970) and are also used in other applications like illicit supply chain (Jabarzare et al., 2020), water resource analysis (Jiang and Liu, 2018), and power system vulnerability analysis (Wu and Conejo, 2016), cybersecurity (Baggio et al., 2021), among others. Network interdiction models identify the critical nodes and/or arcs of the network and can be extended to fortify the existing networks (Brown et al., 2006) or design resilient networks (Smith et al., 2007). Fortification and design models are trilevel mathematical models, a game between defender-interdictor-defender and designer-interdictor-designer, respectively. The goal of fortification models is to identify an optimal subset of arcs/nodes to protect with a limited budget to reduce the negative effects of interdictions. The fortification models aim to protect the existing networks; however, as the arcs/facilities are installed, it is expensive to change their structure for protection. Therefore, the design model accounts for worst-case interdictions in the design stage.

3.1.1 Contribution

In this paper, we introduce the *multicommodity network design problem under stochastic interdiction* (MCND-SI), by incorporating budget uncertainty into multicommodity flow network interdiction problem. This allows the network designer to make design decisions before the action of the interdictor can be observed. To study this problem, we present a trilevel model using the designer-attacker-designer framework to capture the interactions between the designer and the interdictor. As such, the designer first builds the network, followed by the execution of the max-min network interdiction games. One of the most common modeling assumptions in the network interdiction literature is that the interdiction budget (of the attacker) is known to the decision maker (network designer) with certainty; however, in practice, the designer is not aware of interdiction capabilities in the design phase.

Accounting for stochastic interdiction in designing multicommodity networks makes the problem exceptionally difficult to solve. Even the deterministic version of MCND with (optimal) interdiction, studied by Smith et al. (2007), is difficult and computationally challenging. Therefore, the most significant contribution of this paper is to develop an exact and efficient algorithm that can

find provably optimal solutions within reasonable computational time. We present a branch-and-Benders-cut algorithm enhanced with several acceleration techniques that are tailored based on the problem's characteristics. These include (i) the use of a stronger multicut Benders reformulation, (ii) the generation of stronger Pareto-optimal cuts, (iii) the use of a penalty reformulation, (iv) the generation of supervalid and valid inequalities, and (v) the use of warm start strategies, along with variable fixing, and cut selection. In order to evaluate and assess the robustness, efficiency, and limitations of our proposed algorithm, extensive computational experiments were performed on classical MCND instances. The models and solution methods have been extended to the case when both the demand and interdiction budget are uncertain. We compare the performance of our algorithm with the method presented in Smith et al. (2007) and a general-purpose stochastic mixed-integer bilevel linear optimization solver (MibS), and show that our algorithm is orders of magnitude faster. Results on large-size benchmark instances with up to 20 nodes, 220 arcs, and 200 commodities confirm the effectiveness of our proposed solution methodology. Based on the extensive computational results, we observe that all these acceleration techniques improve the performance of the branch-and-Benders-cut algorithm substantially. Moreover, we demonstrate the advantages of using the stochastic model instead of the deterministic one when the number of interdictions is uncertain.

The remainder of this paper is organized as follows: In Section 3.2, we briefly review the relevant literature. Section 3.3 describes the problem settings, model assumptions, and the trilevel mixed-integer programming (MIP) formulation of the MCNDP under stochastic interdictions. The linear reformulation of the lower bilevel interdiction problem into single-level MIP is also described in this section. Benders reformulation and some aspects of the Benders decomposition algorithm are presented in Section 3.4. Section 4.4.2 introduces several acceleration techniques that improve the convergence and efficiency of the algorithm. In Section 3.6, we present the results of extensive computational experiments. Conclusions follow in Section 3.7.

3.2 Literature Review

In what follows, we provide a brief review of the literature on network interdiction problems. Network interdiction problems are generally bi-level models consisting of two opposing players

(i.e., attacker and defender) in a game theoretic framework. Interdiction problems arise in a variety of real-world applications, including national defense (McMasters and Mustin, 1970), coordinating military logistics (Ghare et al., 1971), flood control (Ratliff et al., 1975), infectious disease control (Assimakopoulos, 1987), counter-terrorism (Salmeron et al., 2004), contrasting nuclear smuggling (Morton et al., 2007), the interception of contraband and illegal items such as drugs (Washburn and Wood, 1995), protection of infrastructure systems (Salmeron et al., 2009), spread of fake news in social media (Baggio et al., 2021), and the kidney exchange problem (Blom et al., 2024). Interdiction problems have been widely investigated for well-known combinatorial optimization problems, including knapsack problem (Caprara et al., 2016; Contardo and Sefair, 2022), shortest path problem (Lozano and Smith, 2017), maximum flow problem(Afshari Rad and Kakhki, 2017), traveling salesman problem (Lozano et al., 2017), facility location (Liberatore et al., 2011), hub location (Ramamoorthy et al., 2018), and MCND problem (Smith et al., 2007), among others.

In one of the seminar papers, Wood (1993) studies single-commodity and multicommodity network interdiction problem. Lim and Smith (2007) studies the complete and partial interdiction of the multicommodity network. They transform the bi-level model for the multicommodity network with complete interdictions into a single-level model using penalty reformulation. They solve the partial interdiction model by partitioning algorithm. Smith et al. (2007) extend the bi-level formulation to a trilevel mathematical model for designing a survivable multicommodity network by considering heuristic and optimal interdictor's actions. Alderson et al. (2011) present a trilevel mathematical model for protecting a municipal transportation network where the government intends to enhance the resiliency of the links of the network in a cost-effective way. Jin et al. (2015) present a trilevel model for protecting the urban rail transit network. Zhang and Fan (2017) study a multicommodity network flow problem with interdictions on nodes and arcs with varying budget levels. Jabarzare et al. (2020) present a multicommodity bi-level interdiction model for illicit supply chains. We refer the readers to Smith and Song (2020) and Hunt and Zhuang (2024) for comprehensive survey on interdiction models and solution methods.

Note that all the mentioned network interdiction papers consider a deterministic model; however, the interdictor/defender/designer may not have complete knowledge about the network parameters or interdiction budget. Different parameters can be uncertain in interdiction models. Liberatore et al. (2011) consider a stochastic number of interdictions for the optimal fortification strategy of r-interdiction median problem. The defender minimizes the expected disruption costs in the first level as the number of interdictions is uncertain. The interdictor, in the second level, maximizes the disruption cost by interdicting r facilities. Bhuiyan et al. (2021) present a risk-averse bi-level stochastic model to protect the network against an uncertain number of cyber-attacks. Interdiction success outcomes can be uncertain as well. Janjarassuk and Linderoth (2008) consider the maximum flow network interdiction problem where interdictions are successful with specific probabilities. Moreover, the configuration of the network (Collado et al., 2017), arc traveling costs (Song and Shen, 2016; Nguyen and Smith, 2022), interdiction place (Holzmann and Smith, 2021), and demand (Hien et al., 2020) can be uncertain. Considering uncertain parameters increases the complexity of the interdiction models. To the best of our knowledge, Hien et al. (2020) is one of the first papers to study the stochastic version of the multicommodity network interdiction problem. They present a trilevel mathematical model for fortifying the single/multicommodity networks with stochastic demand. They solve the trilevel nonlinear stochastic model with a robust stochastic approximation approach. This work differs from Hien et al. (2020) as we design a multicommodity network while they fortify the existing network. Moreover, we consider the uncertain number of interdictions while Hien et al. (2020) consider the uncertainty in the demand.

Table 3.1 presents a summary of the literature on relevant multicommodity network interdiction models. For each paper, we list the problem (interdiction, fortification, or design problem), the number of levels, the deterministic or stochastic nature of the parameters, and the solution methodology. Smith et al. (2007) study the multicommodity network design problem with partial interdictions, where the primary objective is to maximize profit. They solve the problem with the cutting plane algorithm that generates Benders cuts iteratively. We also study the multicommodity network design with interdictions. However, our work is different from Smith et al. (2007) as we consider the complete interdiction, cost minimization, and stochastic number of interdictions. To solve the trilevel

stochastic model, we use the branch-and-Benders-cut algorithm. Note that the cutting plane algorithm presented in Smith et al. (2007) can solve our presented model but the branch-and-Benders-cut algorithm presented in this paper with acceleration techniques has a better performance. To the best of our knowledge, our paper is one of the first that studies a trilevel stochastic model for designing multicommodity networks.

As evident in Table 3.1, interdiction problems are often modelled as bilevel programs, whereas fortification and design problems are modelled as trilevel programs. Bilevel optimization is a a powerful tool for modeling hierarchical decision making processes and real-world problems, however the resulting problems are challenging to solve. Readers are refereed to recent surveys by Beck et al. (2023) and Kleinert et al. (2021). Many state-of-the-art solution approaches for bilevel optimization uses techniques that originate from mixed-integer programming including branch-and-bound methods, cutting plane method (Hemmati and Smith, 2016), branch-and-cut approaches (DeNegre and Ralphs, 2009), or Benders-like decomposition methods among others; see (Kleinert et al., 2021) and references therein. Our approach for solving the reduced bilevel problem is based on an enhanced branch-and-Benders-cut (BBC) method, also called Benders-based branch-and-cut method, where instead of solving one MILP master problem at every iteration, a single branch-and-cut tree is constructed and the Benders cuts are added during the exploration of this tree.

Table 3.1: Summary of Literature on Multicommodity Network Interdiction Problems

Articles	Problem			No. of Levels	Model type		Methodology	
	Interdiction	Fortification	Design		Deterministic	Stochastic	Exact	Heuristic
Wood (1993)	✓			2	✓		✓	
Lim and Smith (2007)	✓			2	✓		✓	
Smith et al. (2007)			✓	3	✓		✓	
Akgün et al. (2011)	✓			2	✓		✓	
Alderson et al. (2011)		✓		3	✓		✓	
Shen et al. (2012)	✓			2	✓		✓	
Jin et al. (2015)		✓		3	✓			✓
Zhang and Fan (2017)	✓			2	✓		✓	
Jabarzare et al. (2020)	✓			2	✓		✓	
Hien et al. (2020)		✓		3		✓	✓	
This paper			✓	3		✓	✓	

3.3 Problem Description and Formulation

In this section, we define the three-stage, sequential, zero-sum game that is used for modeling the MCNDP under stochastic interdiction budget. In particular, we present a *designer–attacker–designer* model for designing a multicommodity network that is resilient against worst-case attacks. In the first stage, the network designer constructs a network in which each arc has a fixed installation cost, a maximum capacity, and a per-unit flow cost. The designer's objective is to minimize the installation cost of arcs, as well as the flow costs incurred in normal operating conditions and after interdictions. The designer does not have full information about the interdiction budget (i.e., the number of arc interdictions). Therefore, the designer minimizes the expected flow cost after interdictions in the first level. In the second stage, the interdictor inflicts damage to the network by destroying certain arcs completely (i.e., reducing the arc capacity to zero). The objective is to maximize the minimum possible post-interdiction flow cost. Finally, in the third stage, the designer maximizes post-interdiction flow cost by solving a multicommodity flow problem on the residual network while satisfying the demand.

In line with network interdiction literature, we assume *perfect information* and *rationality* for the game (see Baggio et al. (2021); Smith and Song (2020) and references therein). In other words, our modeling assumptions are as follows: (i) The designer and the interdictor have complete information about the network topology. (ii) The designer is uncertain about the number of interdictions. (iii) The game is sequential where the designer makes the first move by selecting the design and pre-interdiction flow decisions. Then, the interdictor determines the interdiction decisions. Finally, the designer determines the post-interdiction flow decisions in the remaining network. (iv) The interdictor is aware of the design decisions. (v) The game is played one round by the designer and the interdictor. (vi) We consider complete interdictions which means that the capacity of arcs is reduced to zero if they are interdicted.

3.3.1 Formulation

Let $G(N, A)$ be a directed graph, where N is the set of nodes, A is the set of arcs, and K is the set of commodities. Let $A^+(l)$ represents the set of arcs directed out of node $l \in N$, and $A^-(l)$

Table 3.2: Table of Notations

Sets	
N	Set of nodes (indexed by l, $l \in N$)
A	Set of arcs (indexed by h, $h \in A$)
$A^+(l)$	Set of arcs directed out of node $l \in N$
$A^-(l)$	Set of arcs directed into node $l \in N$
A'	Set of dummy arcs
K	Set of commodities (indexed by k, $k \in K$)
S	Set of scenarios (indexed by s, $s \in S$)
Parameters	
D_k	Demand node of commodity $k \in K$
O_k	Supply node of commodity $k \in K$
d^k	Demand of commodity $k \in K$
r_h^k	Cost of transmitting commodity $k \in K$ through arc $h \in A$
c_h	Installation cost of arc $h \in A$
q_h	Capacity of arc $h \in A$
$f(h)$	From-node of arc $h \in A$
$t(h)$	To-node of arc $h \in A$
B_s	Interdiction budget, i.e., number of interdictions under scenario $s \in S$
p_s	Probability of scenario $s \in S$
M	Penalty cost for using interdicted arcs
Φ	The weight of pre-interdiction flow cost in the objective function
$1 - \Phi$	The weight of post-interdiction flow cost in the objective function
Decision Variables	
w_h	1 if arc $h \in A$ is installed; 0 otherwise
y_h^k	Flow of commodity $k \in K$ on arc $h \in A$ before interdiction
x_{hs}	1 if arc $h \in A$ is interdicted in scenario $s \in S$; 0 otherwise
v_{hs}^k	Flow of commodity $k \in K$ on arc $h \in A$ after interdiction in scenario $s \in S$

represents the set of arcs directed into node $l \in N$. Also, we define the from-node and to-node of arc $h \in A$ as $f(h)$ and $t(h)$, respectively. Each arc $h \in A$ has installation cost c_h and capacity q_h. Each commodity $k \in K$ has a unique supply node O_k and a unique demand node D_k. We assume that transshipment nodes can be used to send commodity $k \in K$ from the supply node O_k to the demand node D_k. The cost of transmitting commodity $k \in K$ through arc $h \in A$ is denoted by r_h^k ($r_h^k > 0$). Since the number of interdictions is uncertain, a scenario $s \in S$ defines a particular realization of the interdiction budget B_s with probability p_s. The notations are listed in Table 3.2.

It is possible that the remaining capacity after interdiction is insufficient to satisfy the demand. Therefore, we create dummy arcs in the network to ensure feasible multicommodity flows. The dummy arcs cannot be interdicted and they represent the unmet demand, stating that all the demand

should be satisfied in all conditions. These arcs have zero installation costs, sufficiently high flow costs, and large capacities. A similar setting has been used in (Smith et al., 2007). Under these assumptions and settings, the nested trilevel optimization problem of the network design can be stated as follows:

$$\text{(MCNDSI)}: \quad \min_{w,y} \sum_{h\in A} c_h w_h + \Phi\left[\sum_{h\in A}\sum_{k\in K} r_h^k y_h^k\right] + (1-\Phi)\sum_{s\in S} p_s T_s(w) \tag{3.1}$$

$$\text{s.t.} \quad \sum_{i\in A^+(l)} y_i^k - \sum_{j\in A^-(l)} y_j^k = \begin{cases} d^k & l = O_k \\ -d^k & l = D_k \\ 0 & \text{otherwise} \end{cases} \qquad \forall k \in K, l \in N \tag{3.2}$$

$$\sum_{k\in K} y_h^k \le q_h w_h \qquad \forall h \in A \tag{3.3}$$

$$w_h \in \{0,1\} \qquad \forall h \in A \tag{3.4}$$

$$y_h^k \ge 0 \qquad \forall h \in A, k \in K \tag{3.5}$$

$$\text{(MCNI}_s\text{)}: \quad T_s(w) = \max_x H_s(x,w) \tag{3.6}$$

$$\text{s.t.} \quad \sum_{h\in A} x_{hs} \le B_s \tag{3.7}$$

$$x_{hs} = 0 \qquad \forall h \in A' \tag{3.8}$$

$$x_{hs} \in \{0,1\} \qquad \forall h \in A \tag{3.9}$$

$$\text{(MCF}_s\text{)}: \quad H_s(x,w) = \min_v \sum_{h\in A}\sum_{k\in K} r_h^k v_{hs}^k \tag{3.10}$$

$$\text{s.t.} \quad \sum_{i\in A^+(l)} v_{is}^k - \sum_{j\in A^-(l)} v_{js}^k = \begin{cases} d^k & l = O_k \\ -d^k & l = D_k \\ 0 & \text{otherwise} \end{cases} \qquad \forall k \in K, l \in N \tag{3.11}$$

$$\sum_{k\in K} v_{hs}^k \le q_h w_h (1 - x_{hs}) \qquad \forall h \in A \tag{3.12}$$

$$v_{hs}^k \ge 0 \qquad \forall h \in A, k \in K \tag{3.13}$$

The designer's objective function (3.1) consists of the fixed arc installation cost plus the weighted sum of the flow cost before interdiction and the expected flow cost after interdiction. As the network is not always under disruption, the designer intends to find a design that performs efficiently in normal conditions and under disruptions. Therefore, the parameter $\Phi \in [0, 1]$ represents the relative importance of the pre-interdiction and post-interdiction flow costs in the objective function. Constraint (3.2) represents the flow balance in node $l \in N$ for commodity $k \in K$ before interdiction. Constraint (3.3) ensures that the capacity limit of arc $h \in A$ is respected. At MCNI$_s$ level, the interdictor's objective is to maximize the post-interdiction flow cost using an interdiction budget of B_s in scenario $s \in S$ stated in constraint (3.7). Constraint (3.8) guarantees that dummy arcs are not interdicted. At MCF$_s$ level, the designer minimizes the network flow cost after interdiction of B_s arcs under scenario $s \in S$. Constraint (3.12) represents the effects of interdictions on the capacity of arc $h \in A$.

3.3.2 Extension to Uncertain Demand and Interdiction Budget

We extend the stochastic trilevel MILP model (3.1)-(3.13) to consider demand uncertainty besides budget uncertainty. We assume that the designer considers $\omega \in \Omega$ scenarios with probability p_ω for the realization of demand in the first level. However, the interdictor interdicts the network based on the expected demand. Therefore, for the third level model, we consider the expected demand $\overline{d}^k$ for each commodity $k \in K$. In the first level, the designer minimizes the sum of installation cost and the convex combination of the expected pre-interdiction cost, and the expected post-interdiction cost. The resulting formulation for MCND with stochastic demand and interdiction

(MCND-SDI) is as follows:

$$\text{(MCND-SDI)}: \quad \min_{w,y} \sum_{h\in A} c_h w_h + \Phi\left[\sum_{\omega\in\Omega} p_\omega \sum_{h\in A}\sum_{k\in K} r_h^k y_{h\omega}^k\right] + \quad (1-\Phi)\sum_{s\in S} p_s T_s(w) \tag{3.14}$$

$$\text{s.t.} \quad \sum_{i\in A^+(l)} y_{i\omega}^k - \sum_{j\in A^-(l)} y_{j\omega}^k = \begin{cases} d_\omega^k & l = O_k \\ -d_\omega^k & l = D_k \\ 0 & \text{otherwise} \end{cases} \quad \forall k\in K, l\in N, \omega\in\Omega \tag{3.15}$$

$$\sum_{k\in K} y_{h\omega}^k \le q_h w_h \qquad \forall h\in A, \omega\in\Omega \tag{3.16}$$

$$w_h \in \{0,1\} \qquad \forall h\in A \tag{3.17}$$

$$y_{h\omega}^k \ge 0 \qquad \forall h\in A, k\in K, \omega\in\Omega \tag{3.18}$$

$$\text{(MCNI}_s\text{)}: \quad T_s(w) = \max_x H_s(x,w) \tag{3.19}$$

$$\text{s.t.} \quad (3.7)-(3.9)$$

$$\text{(MCF}_s\text{)}: \quad H_s(x,w) = \min_v \sum_{h\in A}\sum_{k\in K} r_h^k v_{hs}^k \tag{3.20}$$

$$\text{s.t.} \quad \sum_{i\in A^+(l)} v_{is}^k - \sum_{j\in A^-(l)} v_{js}^k = \begin{cases} \overline{d}^k & l = O_k \\ -\overline{d}^k & l = D_k \\ 0 & \text{otherwise} \end{cases} \quad \forall k\in K, l\in N \tag{3.21}$$

$$\sum_{k\in K} v_{hs}^k \le q_h w_h (1 - x_{hs}) \qquad \forall h\in A \tag{3.22}$$

$$v_{hs}^k \ge 0 \qquad \forall h\in A, k\in K \tag{3.23}$$

The designer-attacker-designer formulation presented in (3.1)-(3.13) and (3.14)-(3.23) are trilevel MIP models. Such trilevel formulations can be solved by transforming them into equivalent bilevel models. In Section 3.3.3, we present the transformation of the trilevel model into the bilevel model

using strong duality.

3.3.3 Single-level Reformulation of the Bilevel Interdiction Problem

We use the strong duality of the bilevel interdiction problem to reformulate the bilevel interdiction problem MCNI$_s$ as a single-level formulation. Zare et al. (2019) have shown that strong duality-based approaches may reduce the number of variables and constraints significantly compared with the KKT-based reformulation. Let π^k_{ls} and α_{hs} denote the dual variables corresponding to flow constraint (3.11) and (3.12), respectively. For fixed design decisions $\hat{w}_h$, we (i) fix the value of x_{hs}, (ii) take the dual of MCF$_s$, and (iii) release x_{hs}, resulting in the following single-level MIP formulation D-MCNI$_s$ for each scenario $s \in S$:

$$\text{(D-MCNI}_s\text{)}: \quad \max_{x,\alpha,\pi} \sum_{k\in K} d^k\left(\pi^k_{O_k s} - \pi^k_{D_k s}\right) - \sum_{h\in A} q_h \hat{w}_h (1 - x_{hs})\alpha_{hs} \tag{3.24}$$

$$\text{s.t.} \quad \pi^k_{f(h)s} - \pi^k_{t(h)s} - \alpha_{hs} \le r^k_h \qquad \forall h \in A, k \in K \tag{3.25}$$

$$\sum_{h\in A} x_{hs} \le B_s \tag{3.26}$$

$$x_{hs} = 0 \qquad \forall h \in A' \tag{3.27}$$

$$x_{hs} \in \{0,1\} \qquad \forall h \in A \tag{3.28}$$

$$\pi^k_{ls} \quad \text{unrestricted} \qquad \forall l \in N, k \in K \tag{3.29}$$

$$\alpha_{hs} \ge 0 \qquad \forall h \in A \tag{3.30}$$

The objective function (3.24) is nonlinear due to the multiplication of the binary variable x and the continuous variable α. To linearize the objective function, we define the auxiliary variable $\varphi_{hs} = x_{hs}\alpha_{hs}$, and add the following linear constraints to D-MCNI$_s$:

$$\varphi_{hs} - \alpha_{hs} \le 0 \qquad \forall h \in A \tag{3.31}$$

$$\varphi_{hs} - \overline{\alpha} x_{hs} \le 0 \qquad \forall h \in A \tag{3.32}$$

Variable φ_{hs} appears only in the objective function with a positive coefficient in (3.24). Therefore, φ_{hs} takes the maximum value which is $\alpha_{hs}x_{hs}$ and there is no need to consider the constraint

$\alpha_{hs} + \overline{\alpha} x_{hs} - \varphi_{hs} \leq \overline{\alpha}$. In constraint (3.32), $\overline{\alpha} = \sum_{h \in A} \sum_{k \in K} r_h^k$ is the upper bound on the value of decision variable α.

After transforming the bilevel MCNI$_s$ into a single-level MIP formulation, the resulting bilevel stochastic MCND model is as follows:

$$\begin{aligned} \text{(MCNDSI')} : \min_{w,y} \sum_{h \in A} c_h w_h + \Phi \left[\sum_{h \in A} \sum_{k \in K} r_h^k y_h^k \right] + (1 - \Phi) \max_{x,\varphi,\alpha,\pi} \sum_{s \in S} p_s \\ \left[\sum_{k \in K} d^k \left(\pi_{O_k s}^k - \pi_{D_k s}^k \right) - \sum_{h \in A} q_h w_h \alpha_{hs} + \sum_{h \in A} q_h w_h \varphi_{hs} \right] \end{aligned} \tag{3.33}$$

s.t. $(3.2) - (3.3), (3.25) - (3.27), (3.31) - (3.32)$

$w_h \in \{0, 1\}$ $\forall h \in A$

$y_h^k \geq 0$ $\forall h \in A, k \in K$

$x_{hs} \in \{0, 1\}$ $\forall h \in A, s \in S$

π_{ls}^k unrestricted $\forall l \in N, k \in K, s \in S$

$\alpha_{hs}, \varphi_{hs} \geq 0$ $\forall h \in A, s \in S$

In MCNDSI' problem, constraints (3.2)-(3.3) belong to the first level, constraints (3.26)-(3.27) belong to the second level, and constraints (3.25), (3.31)-(3.32) belong to the dual of the third level. In Section 3.4, we use the Benders decomposition algorithm to solve MCNDSI'. We present different acceleration techniques to improve the convergence and the efficiency of the Benders decomposition algorithm in Section 4.4.2. Note that the solution methodologies presented in Sections 3.4 and 4.4.2 apply to MCND-SDI as well.

3.4 Benders Decomposition

Benders decomposition is a well-known partitioning technique for solving MILP problems by decomposing the problem into a mixed-integer master problem and a linear subproblem (Benders, 1962). The main idea behind this approach involves simplifying a problem by projecting out some complicating variables, resulting in a formulation with fewer variables but many constraints known

as Benders cuts. Only a few of these constraints are typically necessary to find an optimal solution; therefore, it is common to relax the problem by discarding Benders cuts and generate them as necessary in a cutting-plane fashion.

Due to the slow convergence of the classical Benders decomposition algorithm, several acceleration techniques have been proposed to speed up the convergence of the algorithm. The most widely used acceleration technique is the use of Pareto-optimal cuts (Magnanti and Wong, 1981; Papadakos, 2008). The idea of Pareto-optimal cuts is to add non-dominated optimality cuts to the master problem when there are alternative optimal solutions to the subproblem. Furthermore, one can use multi-cut reformulation when the subproblem can be separated into smaller independent subproblems. Variable fixing is another technique that can reduce the size of the master problem (Taherkhani et al., 2020; Zetina et al., 2021). We refer the readers to the survey by Rahmaniani et al. (2017) for the developments of these acceleration techniques.

Benders decomposition has been widely used for solving several variants of MCNDPs (Costa, 2005; Costa et al., 2009; Rahmaniani et al., 2018; Zetina et al., 2019; Fragkos et al., 2021; Crainic et al., 2021). It has also been proven to be an efficient method for solving interdiction problems (Brown et al., 2006). Examples include interdiction models of illicit supply chain (Jabarzare et al., 2020), shortest path (Israeli and Wood, 2002; Song and Shen, 2016), knapsack (Fischetti et al., 2019), electric power grid (Salmeron et al., 2009), maximum reliability (Bodur et al., 2017), and hub networks (Ramamoorthy et al., 2018), to name a few. In this paper, we exploit the structure of the problem, adapt the acceleration techniques to develop an efficient branch-and-Benders-cut algorithm for solving the model.

3.4.1 Multicut Benders Reformulation

Let $\mathbb{W}$ denote the set of vectors (w, y) satisfying constraints (3.2)-(3.5). For any fixed $\hat{w} \in \mathbb{W}$, the primal subproblem (SP) consists of the bi-level interdiction model, transformed into single-level

formulation D-MCNI$_s$ for each scenario $s \in S$.

$$\text{(SP}_s\text{)}: \quad \max_{x,\varphi,\alpha,\pi} \sum_{k\in K} d^k\left(\pi^k_{O_k s} - \pi^k_{D_k s}\right) - \sum_{h\in A} q_h \hat{w}_h \alpha_{hs} + \sum_{h\in A} q_h \hat{w}_h \varphi_{hs} \tag{3.34}$$

$$\text{s.t.} \quad (3.25) - (3.27), (3.31) - (3.32)$$

$$x_{hs} \in \{0,1\} \qquad \forall h \in A$$

$$\pi^k_{ls} \quad \text{unrestricted} \qquad \forall l \in N, k \in K$$

$$\alpha_{hs}, \varphi_{hs} \geq 0 \qquad \forall h \in A$$

Since we define dummy arcs to account for the unmet demand, the SP is always feasible; therefore, no feasibility cuts are added to the MP.

Let Υ denote the set of extreme points associated with SPs. By introducing continuous variables η_s for the post-interdiction flow cost of each scenario $s \in S$ in the MP, the resulting formulation of Benders master problem (MP) is as follows:

$$\text{(MP)}: \quad \min_{w,y,\eta} \sum_{h\in A} c_h w_h + \Phi\left[\sum_{h\in A}\sum_{k\in K} r^k_h y^k_h\right] + (1-\Phi)\sum_{s\in S} p_s \eta_s \tag{3.35}$$

$$\text{s.t.} \quad (3.2) - (3.3)$$

$$\eta_s \geq \sum_{k\in K} d^k\left(\hat{\pi}^k_{O_k s} - \hat{\pi}^k_{D_k s}\right) - \sum_{h\in A} q_h w_h \hat{\alpha}_{hs} + \sum_{h\in A} q_h w_h \hat{\varphi}_{hs} \quad \forall s \in S, (\hat{\pi}, \hat{\alpha}, \hat{\varphi}) \in \Upsilon \tag{3.36}$$

$$\eta_s \geq lb \qquad \forall s \in S \tag{3.37}$$

$$\eta_s \quad \text{unrestricted} \qquad \forall s \in S \tag{3.38}$$

$$w_h \in \{0,1\} \qquad \forall h \in A$$

$$y^k_h \geq 0 \qquad \forall h \in A, k \in K$$

Note that we use the decomposability of SP by scenarios to generate optimality cuts (3.36). In the multicut reformulation, one optimality cut per scenario is added to the MP instead of aggregating the obtained information of all scenario SPs into one optimality cut. Multicut reformulation provides more information about the SPs and prevents information loss in the aggregation process. As stated

earlier, we relax the optimality cuts (3.36) and add them during the algorithm. Therefore, we add constraint (3.37) to provide an initial lower bound (lb) on η_s to avoid the unboundedness of the MP in the first iteration.

The classic version of Benders decomposition solves MP and SP iteratively until the optimality gap is closed. The MP gives the lower bound of the original problem because the cuts are relaxed and added during the algorithm. By fixing the decision variables of MP in SP, SP provides the upper bound of the original problem by finding a feasible solution. Depending on the solution obtained from SP at each iteration of the algorithm, the optimality or feasibility cuts are added to the MP.

3.4.2 Implementation of Branch-and-Benders-Cut Algorithm

In the classical implementation of the Benders decomposition, the mixed-integer MP is solved to optimality at each iteration of the algorithm, which can be computationally prohibitive. The idea in the recent implementation of Benders decomposition is to solve the mixed-integer MP only once and add the optimality cuts in the branch-and-bound search tree as needed. This implementation, known as branch-and-Benders-cut (BBC), is possible using *callback* functions in commercial solvers such as CPLEX. In the first phase of BBC, we relax the integrality condition of the MP and the optimality cuts (3.36). We add the necessary optimality cuts by solving MP and SP iteratively. In the second phase, we reintroduce the integrality requirement to the MP with the optimality cuts generated in the first phase. This facilitates CPLEX to add more and/or stronger cuts to the model by having more information about the problem based on the constraints of the first phase. The optimality cuts and the CPLEX cuts provide a better starting point for the branch-and-bound algorithm. When an integer solution is found in one of the tree nodes, the SP is solved to find violated optimality cuts, which are added to MP, and the node is resolved. The optimality cuts, at integer solutions, are added as *lazycut* constraints to the MP whereas, at fractional solutions, they are added as *usercut* constraints (Bodur and Luedtke, 2017).

3.5 Acceleration Techniques

In this section, we present several acceleration techniques to improve the convergence and stability of the Benders decomposition algorithm presented in the previous section. In Sections 3.5.1, 3.5.2 and 3.5.3, we use the characteristics of the interdiction model to present three acceleration techniques: generate Pareto-optimal cuts, use of penalty reformulation of the bi-level interdiction model, generate supervalid and valid inequalities.

3.5.1 Pareto-optimal Cuts

The effectiveness of the Benders decomposition algorithm relies on the selection of Benders cuts. One way to improve the convergence of the Benders algorithm is to construct stronger, undominated cuts, known as Pareto-optimal cuts. To address this issue, Magnanti and Wong (1981) proposed a method to generate Pareto-optimal cuts, which are cuts that are not dominated by any other cut.

Given two cuts defined by dual solutions $(\pi^a, \alpha^a, \varphi^a)$ and $(\pi^b, \alpha^b, \varphi^b)$ in the form of $\eta \geq f(\pi^a) + wg(\alpha^a, \varphi^a)$ and $\eta \geq f(\pi^b) + wg(\alpha^b, \varphi^b)$, respectively, the cut defined by $(\pi^a, \alpha^a, \varphi^a)$ dominates the cut defined by $(\pi^b, \alpha^b, \varphi^b)$ if and only if

$$f(\pi^a) + wg(\alpha^a, \varphi^a) \geq f(\pi^b) + wg(\alpha^b, \varphi^b)$$

with strict inequality for at least one feasible solution w of the MP. The cut defined by $(\pi^a, \alpha^a, \varphi^a)$ is Pareto-optimal if it is not dominated by any other cut.

To generate Benders cuts that are Pareto-optimal, we need to solve another auxiliary linear problem at each iteration. This auxiliary linear problem is very similar to SPs, except for two key differences. First, in the objective function (3.34), we replace the MP solution $\hat{w}$ with a core point w_0. A core point is a point in the relative interior of the MP space. Second, an equality constraint is introduced to guarantee that the resulting solution remains within the set of alternative optimal solutions of the SPs for the current MP solution $\hat{w}$ (Magnanti and Wong, 1981).

The disadvantages of the method presented by Magnanti and Wong (1981) are the difficulty of finding the core points and the extensive computation time to solve auxiliary SPs with equality

constraints. Therefore, Papadakos (2008) presents an alternative way to find the Pareto-optimal cuts. In this method, independent Pareto SPs are solved by removing the equality constraint utilized in Magnanti and Wong's method. At each iteration t, the approximate core points are determined using equation (3.39), where $\hat{w}_h^{t-1}$ represents the optimal solution of the MP from the previous iteration $t-1$ and the initial weight value of W_h^t is $W_h^0 = 1$.

$$W_h^t \leftarrow \frac{1}{2} W_h^{t-1} + \frac{1}{2} \hat{w}_h^{t-1} \tag{3.39}$$

Based on Papadakos (2008), the independent Pareto subproblem (PSP_s) defined below is the SP of the original Benders with the approximate core points W_h^t.

$$(\text{PSP}_s): \quad \max \sum_{k \in K} d^k \left(\pi^k_{O_k s} - \pi^k_{D_k s}\right) - \sum_{h \in A} q_h W_h^t \alpha_{hs} + \sum_{h \in A} q_h W_h^t \varphi_{hs} \tag{3.40}$$

$$\text{s.t.} \quad (3.25) - (3.32)$$

3.5.2 Penalty Reformulation

We provide an equivalent reformulation to bilevel MCNI_s where the interdictor's decision variables appear as a penalty term in the objective function of the designer (Lim and Smith, 2007). The interpretation of the reformulation (3.41)-(3.42) is that the designer has to incur a penalty for using interdicted arcs. We can set the penalty term to $M = \sum_{h \in A} \sum_{k \in K} r_h^k$. The resulting formulation is as follows:

$$
\begin{aligned}
(\text{PMCNI}_s): \quad & \max_x H_s(x) \\
\text{s.t.} \quad & \sum_{h\in A} x_{hs} \le B_s \\
& x_{hs} = 0 && \forall h \in A' \\
& x_{hs} \in \{0,1\} && \forall h \in A \\
(\text{PMCF}_s): \quad & H_s(x) = \min_v \sum_{h\in A}\sum_{k\in K} (r_h^k + M\hat{x}_{hs}) v_{hs}^k && (3.41) \\
\text{s.t.} \quad & \sum_{i\in A^+(l)} v_{is}^k - \sum_{j\in A^-(l)} v_{js}^k = \begin{cases} d^k & l = O_k \\ -d^k & l = D_k \\ 0 & \text{otherwise} \end{cases} && \forall k \in K, l \in N \\
& \sum_{k\in K} v_{hs}^k \le q_h \hat{w}_h && \forall h \in A && (3.42) \\
& v_{hs}^k \ge 0 && \forall h \in A, k \in K
\end{aligned}
$$

In what follows, we prove that the reformulation PMCF$_s$ is equivalent to the reformulation MCF$_s$. Lemma 1 and Proposition 1 show that the penalty reformulation correctly models the impact of interdictions on network flow and is equivalent to the original MCNI$_s$ formulation in terms of optimization objectives.

Lemma 3.1. *Let M be a large number, and $\overline{v}(\hat{x})$ be an optimal solution of $PMCF_s$ for any $\hat{x} \in X$. Then we have $\overline{v}(\hat{x})_{hs}^k = 0, \forall k \in K$ if $\hat{x}_h = 1$ for any $\hat{x} \in X$.*

Proof. Since M is a large number, if $\hat{x} = 1$ and $v(\hat{x}) > 0$, a strictly better solution exists in which $v(\hat{x}) = 0$, achieved by reallocating flows from paths containing arc h to the dummy arcs in the network, resulting in lower costs. Therefore, if $\hat{x} = 1$, $v(\hat{x})$ should be zero. This completes the proof. □

Proposition 3.1. *Given any $\hat{x} \in X$, let $MCF_s(\hat{x})$ and $PMCF_s(\hat{x})$ denote the designer's solutions to MCF_s and $PMCF_s$, respectively. Then, $MCF_s(\hat{x})$ has an optimal solution $v(\hat{x})$ if and only if*

$v(\hat{x})$ is optimal to $PMCF_s(\hat{x})$. Additionally, the optimal objective function value of $MCF_s(\hat{x})$ is equal to the objective function value of $PMCF_s(\hat{x})$.

Proof. We first show that the objective function value of $PMCF_s$ remains less than or equal to the objective function value of MCF_s. Consider an optimal solution $v(\hat{x})$ to MCF_s with an objective function value of v^*. Since $PMCF_s(\hat{x})$ is a relaxation of $MCF_s(\hat{x})$, $v(\hat{x})$ is also feasible for $PMCF_s(\hat{x})$. Moreover, we have $\sum_{h\in A}\sum_{k\in K}(r_h^k+M\hat{x}_{hs})v_{hs}^k = v^*$ because $\sum_{h\in A}\sum_{k\in K} r_h^k v_{hs}^k = v^*$ and Lemma 3.1 ensures that $\sum_{h\in A}\sum_{k\in K} M\hat{x}_{hs}v_{hs}^k = 0$. Therefore, the optimal objective function value of $PMCF_s(\hat{x})$ does not exceed v^*.

Furthermore, we show that the optimal objective function value of $MCF_s(\hat{x})$ does not exceed the optimal objective function value of $PMCF_s(\hat{x})$, leading to the equivalence of these two values. Let $v(\hat{x})$ be an optimal solution to $PMCF_s$ with an objective function value of v^*. Since Lemma 3.1 guarantees that $v_{hs}^k(\hat{x}) = 0$ for $\hat{x}_{hs} = 1$, $v(\hat{x})$ is a feasible solution for $MCF_s(\hat{x})$ with objective function value $\sum_{h\in A}\sum_{k\in K} r_h^k v_{hs}^k = v^*$, which provides a lower bound on the optimal objective function value of $MCF_s(\hat{x})$. This completes the proof. □

Now, we take the dual of the inner minimization problem PMCF$_s$, the resulting linear single-level model is as follows:

$$\text{(D-PMCNI}_s\text{)}: \quad \max_{x,\alpha,\pi} \sum_{k\in K} d^k\left(\pi_{O_k s}^k - \pi_{D_k s}^k\right) - \sum_{h\in A} q_h \hat{w}_h \alpha_{hs} \tag{3.43}$$

$$\text{s.t.} \quad \pi_{f(h)s}^k - \pi_{t(h)s}^k - \alpha_{hs} \le r_h^k + M x_{hs} \qquad \forall h \in A, k \in K \tag{3.44}$$

$$\sum_{h\in A} x_{hs} \le B_s$$

$$x_{hs} = 0 \qquad \forall h \in A'$$

$$x_{hs} \in \{0,1\} \qquad \forall h \in A$$

$$\pi_{ls}^k \quad \text{unrestricted} \qquad \forall l \in N, k \in K$$

$$\alpha_{hs} \ge 0 \qquad \forall h \in A$$

Considering Υ' as the set of extreme points associated with D-PMCNI$_s$, the penalty reformulation results in the optimality cut (3.45) instead of (3.36).

$$\eta_s \geq \sum_{k \in K} d^k \left(\hat{\pi}^k_{O_k s} - \hat{\pi}^k_{D_k s}\right) - \sum_{h \in A} q_h w_h \hat{\alpha}_{hs} \qquad \forall s \in S, (\hat{\pi}, \hat{\alpha}) \in \Upsilon' \tag{3.45}$$

3.5.3 Supervalid and Valid Inequalities

To enhance the convergence of the Benders algorithm, we add supervalid inequality (SVI) to the MP. SVI will reduce the integer feasible region by eliminating integer solutions while guaranteeing not to eliminate any optimal solutions unless the current solution is optimal (Israeli and Wood, 2002).

Proposition 3.2. *Consider the Benders cut* $\eta_s + \sum_{h \in A} q_h w_h (\hat{\alpha}_{hs} - \hat{\varphi}_{hs}) \geq \sum_{k \in K} d^k (\hat{\pi}^k_{O_k s} - \hat{\pi}^k_{D_k s})$ *is added to the MP. The following inequality is supervalid.*

$$\sum_{h \in A} I(\hat{\alpha}_{hs} - \hat{\varphi}_{hs}) w_h \geq 1 \tag{3.46}$$

where

$$I(\hat{\alpha}_{hs} - \hat{\varphi}_{hs}) = \begin{cases} 1, & \textit{if } \hat{\alpha}_{hs} - \hat{\varphi}_{hs} > 0 \\ 0, & \textit{otherwise} \end{cases}$$

Proof. The proof employs a contradiction strategy. It demonstrates that if we assume the incumbent solution is not optimal, it leads to a contradiction, thus confirming the necessity of the inequality being at least 1 in an optimal solution. This establishes the supervalidity of the stated inequality in the given context of the Benders cut. We perform the following steps to prove the Proposition:

The statement supposes that feasible solution $(\hat{w}, \hat{\pi}, \hat{\alpha}, \hat{\varphi})$ generates the following Benders cut $\eta_s + \sum_{h \in A} q_h w_h (\hat{\alpha}_{hs} - \hat{\varphi}_{hs}) \geq \sum_{k \in K} d^k (\hat{\pi}^k_{O_k s} - \hat{\pi}^k_{D_k s})$. Let $\hat{\eta}_s = \sum_{k \in K} d^k (\hat{\pi}^k_{O_k s} - \hat{\pi}^k_{D_k s}) - \sum_{h \in A} q_h \hat{w}_h (\hat{\alpha}_{hs} - \hat{\varphi}_{hs})$. We assume that the incumbent leads to $\eta^*_s < \overline{\eta}_s$. Let (w^*, y^*, η^*) denote the optimal solution to the MP and note that $\sum_{h \in A} I(\hat{\alpha}_{hs} - \hat{\varphi}_{hs}) w^*_h = 0$ or ≥ 1. From $\sum_{h \in A} I(\hat{\alpha}_{hs} - \hat{\varphi}_{hs}) w^*_h = 0$, we can conclude that $\sum_{h \in A} q_h w^*_h (\hat{\alpha}_{hs} - \hat{\varphi}_{hs}) = 0$. $\sum_{h \in A} I(\hat{\alpha}_{hs} -$

$\hat{\varphi}_{hs})w_h^* = 0$ when (i) $w_h^* = 0$ which implies that $\sum_{h\in A} q_h w_h^*(\hat{\alpha}_{hs}-\hat{\varphi}_{hs}) = 0$ or (ii) $I(\hat{\alpha}_{hs}-\hat{\varphi}_{hs}) = 0$. In this case, based on the definition of $I(\hat{\alpha}_{hs} - \hat{\varphi}_{hs})$, $\hat{\alpha}_{hs} - \hat{\varphi}_{hs} \leq 0$. Since the maximum value of $\hat{\varphi}_{hs}$ is $\hat{\alpha}_{hs}x_{hs}$, $\hat{\alpha}_{hs} - \hat{\varphi}_{hs}$ would be zero implying that $\sum_{h\in A} q_h w_h^*(\hat{\alpha}_{hs} - \hat{\varphi}_{hs}) = 0$. When $\sum_{h\in A} I(\hat{\alpha}_{hs} - \hat{\varphi}_{hs})w_h^* = 0$,

$\eta_s^* \geq \sum_{k\in K} d^k(\hat{\pi}^k_{O_k s} - \hat{\pi}^k_{D_k s}) - \sum_{h\in A} q_h w_h^*(\hat{\alpha}_{hs} - \hat{\varphi}_{hs})$ is true for any $(\hat{\pi}, \hat{\alpha}, \hat{\varphi})$,

$= \hat{\eta_s} - \sum_{h\in A} q_h w_h^*(\hat{\alpha}_{hs} - \hat{\varphi}_{hs}) + \sum_{h\in A} q_h \hat{w}_h(\hat{\alpha}_{hs} - \hat{\varphi}_{hs})$

$\geq \hat{\eta_s}$ because $\sum_{h\in A} q_h w_h^*(\hat{\alpha}_{hs} - \hat{\varphi}_{hs}) = 0$

$\geq \overline{\eta}_s$ because $\hat{w}$ does not need to be the incumbent solution and

$> \eta_s^*$ by assumption; but this is a contradiction.

Therefore, if the incumbent solution is not optimal, $\sum_{h\in A} I(\hat{\alpha}_{hs}-\hat{\varphi}_{hs})w_h^* \geq 1$ must be true for every optimal solution (w^*, y^*, η^*). Hence, the inequality $\sum_{h\in A} I(\hat{\alpha}_{hs}-\hat{\varphi}_{hs})w_h \geq 1$ is supervalid. □

The fact that the flow cost after interdiction is always greater than or equal to the flow cost before interdiction yields the following valid inequality (3.47) to the MP.

$$\eta_s \geq \sum_{h\in A}\sum_{k\in K} r_h^k y_h^k \quad \forall s \in S \tag{3.47}$$

3.5.4 Additional Acceleration Techniques

We use warm start, variable fixing, and cut selection techniques to further improve the performance of the Benders decomposition algorithm.

3.5.4.1 Warm Start

The initial iterations of Benders decomposition algorithm are ineffective due to weak optimality cuts. To overcome this challenge, we use warm start strategies to generate initial tight cuts (Rahmaniani et al., 2018). To generate tight cuts, we find feasible solutions and add optimality cuts based on feasible binary solutions instead of the optimality cuts obtained from fractional solutions in the first phase of the algorithm.

We add optimality cuts based on the following four warm start strategies to the MP:

- Strategy 1 (WS1): We solve the MCNDP for each instance without considering the interdictions. This solution provides us with the minimum number of arcs to satisfy the demand. We keep the value of the flow variables y_h^k.
- Strategy 2 (WS2): We assume that the interdictor interdicts S arcs having the highest flow.
- Strategy 3 (WS3): We assume that the interdictor has high interdiction budget ($B = \lceil 0.2 * A \rceil$) and interdicts the arcs with the highest multicommodity flow. Then, we solve the MCNDP over the residual network.
- Strategy 4 (WS4): We assume that all arcs are installed.

We solve the SPs based on the design solutions obtained from the above warm start strategies to generate the optimality cuts. We also use the upper bound from feasible binary solutions for fixing the value of variables in Section 3.5.4.2.

3.5.4.2 Variable Fixing

Note that one of the flow balance constraints (3.11) is redundant for each commodity $k \in K$. Therefore, we are able to set the dual variable associated with the origin (destination) of each commodity to zero i.e. $\pi^k_{O_k s} = 0$ (Rahmaniani et al., 2018). We use the information generated in the first phase of the Benders algorithm to fix the value of installed arcs to zero in the optimal solution. Let UB and LB denote the upper bound and LP relaxation lower bound to the MP. Let rc_h be the reduced cost associated with variables w_h. If $LB + rc_h > UB$, then $w_h = 0$. The correctness

of this variable fixing technique results from the fact that $LB + rc_h$ is the lower bound if arc h is installed in the optimal solution. Since $LB + rc_h > UB$, w_h should be zero in any optimal solution (Contreras et al., 2011).

3.5.4.3 Cut Selection

To reduce the size of the LP problem solved at the nodes of the branch-and-bound tree in the second phase of the Benders algorithm, we remove the optimality cuts with large slacks (generated in phase one). In addition, CPLEX removes the *lazycuts* or *usercuts* generated in the second phase if they have large slack.

3.5.5 Enhanced Branch-and-Benders-Cut Algorithm

The pseudo-code of the enhanced BBC algorithm for solving the trilevel stochastic MCNDP is presented in Algorithm 4. First, we use the warm start techniques to generate optimality cuts. Then, we start phase one of the algorithm where the LP relaxation of MP and SPs are solved iteratively until stopping criteria are met. Stopping criteria can be the number of iterations, time limit, and/or the optimality gap. In phase two, we reintroduce the integrality requirement to the MP with the cuts generated in the first phase of the algorithm. We remove the optimality cuts generated in the first phase of the algorithm with high slacks and fix the value of w variables to zero if possible. Considering the tree τ and the global upper bound UB, we start the branch-and-bound procedure (lines 12-49). Whenever we find a node with an integer solution, we check the SPs for violated optimality cuts. If we find any violated cuts, we add them as *lazycut* to the MP and resolve the node (lines 34-45). If no violated optimality cuts are found, we calculate the optimal objective function z^*. If it is better than UB, we update the current upper bound UB and prune the node. We can optionally add the violated optimality cuts at the fractional nodes as *usercut*s to the MP (lines 18-33) and we resolve the node. To reduce the number of cuts added to the MP, we add *usercut*s only at nodes with depth up to Max-Depth. In line 47 of the algorithm, we perform branching to create two new nodes and continue the procedure until the optimal solution is found. Note that optimality cuts can be generated using SP_s/PSP_s/D-$PMCNI_s$ based on the variant of the algorithm.

The BBC algorithm terminates after a finite number of steps as the branch-and-bound method

Algorithm 4 Enhanced Branch-and-Benders-Cut Algorithm

1: Create optimality cuts with warm start techniques
2: **while** Stopping criteria are not met **do** ▷ *Phase 1*
3: Solve the LP relaxation of the MP with optimality cuts generated with warm start techniques
4: **for** $s \in S$ **do**
5: Solve SP_s/PSP_s/D-PMCNI_s
6: Add optimality cut to the MP
7: **end for**
8: **end while**
9: Remove the optimality cuts with large slacks ▷ *Phase 2*
10: Fix the value of w variables if possible
11: Add the obtained MP into the tree τ and set $UB = \infty$
12: **while** $\tau \neq \varnothing$ **do**
13: Select node t from τ
14: Solve this node to obtain the optimal $(\overline{w}, \overline{y}, \overline{\eta})$ with objective value of z^*
15: **if** the node is infeasible **or** $z^* \geq UB$ **then**
16: Prune the node and return to line 12
17: **end if**
18: **if** $\overline{w}$ is not integer **then**
19: **if** depth $\leq$ *Max-Depth* **then**
20: Add the violated cut(s) as the *usercuts* to the MP
21: Solve the node again with the violated cut(s) to find $(\overline{w}, \overline{y}, \overline{\eta})$ with objective value of z^*
22: **if** the node is infeasible **or** $z^* \geq UB$ **then**
23: Prune the node and return to line 12
24: **end if**
25: **if** $\overline{w}$ is integer **then**
26: Return to line 34
27: **else**
28: Return to line 46
29: **end if**
30: **else**
31: Return to line 46
32: **end if**
33: **end if**
34: **if** $\overline{w}$ is integer **then**
35: **while** violating cut(s) can be found **and** $\overline{w}$ is integer **do**
36: Add the violated cut(s) as the *lazycuts* to the MP
37: Solve the node again with the violated cut(s) to find $(\overline{w}, \overline{y}, \overline{\eta})$ with objective value of z^*
38: **if** the node is infeasible **or** $z^* \geq UB$ **then**
39: Prune the node and return to line 12
40: **end if**
41: **end while**
42: **if** $\overline{w}$ is integer **then**
43: Set $UB = \min\{UB, z^*\}$, prune the node and return to line 12
44: **end if**
45: **end if**
46: Choose a fractional $\overline{w}$ variable to branch
47: Create two nodes and add them to τ
48: Remove node t from τ
49: **end while**

operates on a finite search space and the number of Benders cuts is finite. Each time a Benders cut is added to the MP, it eliminates a part of the search space that does not include the optimal solution. Given that there are a limited number of Benders cuts that can be added and that the branch-and-bound method will systematically explore and eliminate suboptimal branches, the process will terminate after a finite number of steps.

3.6 Computational Experiments

We perform extensive computational experiments to assess the performance of our branch-and-Benders-cut algorithm and the efficacy of the proposed acceleration techniques tailored to the problem. In the first set of computational experiments, we analyze the effects of *warm start, variable fixing*, and *cut selection* techniques on the performance of the BBC algorithm. Then, we perform extensive experiments to analyze the effects of the use of *penalty formulation, Pareto-optimal cuts,* and *supervalid and valid inequalities* on the performance of the BBC algorithm over small and moderate size instances. In the third set of experiments, we report the results of solving the large-scale instances using the three most promising variants of our BBC algorithm. We also compare our best variant of the BBC algorithm with the cutting plane algorithm proposed by (Smith et al., 2007). To provide a fair comparison, we have adapted (Smith et al., 2007)'s algorithm for the stochastic variant. The algorithm is outlined in Appendix B.1. Then, we compare the results of our best variant of the algorithm with that of MibS, a state-of-the-art solver for mixed-integer bilevel linear programs. In the next set of computational experiments, we report the performance of our algorithm for the case with uncertain demand and uncertain interdiction budget. Finally, we conduct sensitivity analysis to showcase the impact of considering interdiction budget on the optimal network design. We also present a sensitivity analysis to depict the impact of varying weights of pre-interdiction and post-interdiction flow cost on the optimal network design.

All experiments are run on a workstation with the 3.10 GHz Intel Xeon E5 2687W V3 processor under the Linux environment, using one thread. The algorithms are coded in C using the callable library for CPLEX 22.1.0. We use *lazyconstraintcallback* and *usercutcallback* functions in CPLEX

to add the optimality cuts to the nodes with integer and fractional solutions, respectively. We fine-tune the CPLEX parameters, e.g., the user cut frequency. Adding too many cuts slows down the solution of the LP relaxation at each node while adding a few cuts underestimates the lower bound at the nodes. Therefore, we find the appropriate combination of parameter settings for each algorithm. Based on preliminary experiments, we add user cuts to the nodes with depths up to two, and we generate one round of user cuts in each fractional node. We observe that generating cuts for depths of more than two increases the computation time of the algorithms.

3.6.1 Test Instances

To compare the efficiency of our algorithms, we use the well-known "r" benchmark instances for capacitated MCNDP (Crainic et al., 2001) available at `http://pages.di.unipi.it/frangio/`. There are 18 sets of instances, denoted by "r01" to "r18". Each set consists of 9 problem instances with different capacities and fixed costs. The details of the test instances are summarized in Table 3.3 and Appendix B.2. The time limit in all the experiments is set to 24 hours (86,400 seconds), and the optimality gap is set to 10^{-4} unless otherwise stated. We consider three scenarios ($|S| = 2, 3, 4$). In each scenario $s \in S$, the interdiction budget is set to $B_s = s, s = 1, ..., |S|$ unless otherwise stated. All scenarios have equal probability, and we set the weight of the pre-interdiction flow cost to $\Phi = 0.8$.

Table 3.3: Summary of Test Instances

Set	$\|N\|$	$\|A\|$	$\|K\|$	# Instances	Set	$\|N\|$	$\|A\|$	$\|K\|$	# Instances
r01	10	35	10	9	r10	20	120	40	9
r02	10	35	25	9	r11	20	120	100	9
r03	10	35	50	9	r12	20	120	200	9
r04	10	60	10	9	r13	20	220	40	9
r05	10	60	25	9	r14	20	220	100	9
r06	10	60	50	9	r15	20	220	200	9
r07	10	82	10	9	r16	20	314	40	9
r08	10	83	25	9	r17	20	318	100	9
r09	10	83	50	9	r18	20	315	200	9

3.6.2 Computational Performance of Warm Start, Variable Fixing, and Cut Selection

First, we analyze the efficacy of different warm start techniques on the BBC algorithm. For this, we choose a variant of BBC algorithm (with multicut reformulation, penalty reformulation, variable fixing, and cut selection activated) but without any warm start techniques. In Table 3.4, we report the CPU time (s) of the BBC algorithm under six conditions: (i) without any warm start, (ii-v) with each of the four warm starts activated individually, and (vi) with all warm starts activated. The table also reports the percentage reduction in CPU time as a result of all warm start activated compared to none. Based on the average CPU time, we observe that all the warm start techniques have a positive impact on the performance of the BBC algorithm. If we choose to activate only one of the warm start techniques, warm start 4 has the best performance among all techniques. Warm start techniques 1 and 2 have similar performance.

Table 3.4: Performance of Warm Start Techniques

Set	CPU Time (s) for $\|S\| = 3$							CPU Time (s) for $\|S\| = 4$						
	No WS	WS 1	WS 2	WS 3	WS 4	All WS	% Red	No WS	WS 1	WS 2	WS 3	WS 4	All WS	% Red
r08	6,782	7,023	7,034	7,981	5,902	5,506	20	40,865	39,105	39,599	40,105	33,804	36,798	10
	4,622	3,544	4,306	3,985	3,449	3,387	27	31,935	25,939	25,452	33,733	27,357	21,922	31
	2,696	2,387	2,346	3,315	2,285	2,365	12	21,738	21,310	20,961	24,217	20,771	20,428	6
	12,984	10,502	10,900	12,283	9,848	10,130	22	59,251	56,460	57,273	55,152	55,971	49,036	17
	7,776	7,526	7,282	6,798	5,447	5,314	32	48,671	44,269	45,110	50,595	42,527	45,106	7
	5,855	5,642	6,238	5,989	5,461	5,620	4	23,733	24,200	23,278	21,729	21,154	14,511	39
	1,480	1,482	948	1,384	1,195	1,093	26	1,821	1,787	1,179	1,837	1,801	1,769	3
	1,398	1,183	1,131	1,290	1,110	1,124	20	1,753	1,600	1,981	1,659	1,627	1,752	0
	1,530	1,453	1,395	1,479	1,393	1,391	9	1,770	1,498	1,675	1,661	1,724	1,651	7
Average	5,014	4,527	4,620	4,945	4,010	3,992	19	25,726	24,018	24,057	25,632	22,971	21,441	13
Geo. Mean	3,890	3,536	3,456	3,835	3,165	3,133	16	13,965	12,967	12,817	13,839	12,735	11,922	6

Next, we analyze the performance of variable fixing on the BBC algorithm. For this, we choose a variant of BBC algorithm (with multicut reformulation, penalty reformulation, all warm starts, and cut selection activated) but with and without any variable fixing. In Table 3.5, we report the CPU time (s) of the BBC algorithm under two conditions: (i) without variable fixing (w/o VF), (ii) with variable fixing (w/VF) activated. The table also reports the percentage reduction in CPU time as a result of variable fixing. We observe that variable fixing improves the performance of the BBC algorithm.

Lastly, we analyze the performance of cut selection on the BBC algorithm. For this, we choose

Table 3.5: Performance of Variable Fixing

Set	$\|S\| = 3$			$\|S\| = 4$		
	w/o VF	w/VF	% Red	w/o VF	w/ VF	% Red
r08	6,752	5,506	18	86,400	36,798	57
	6,338	3,387	47	59,779	21,922	63
	6,395	2,365	63	86,400	20,428	76
	17,991	10,130	44	82,044	49,036	40
	9,307	5,314	43	86,400	45,106	48
	43,401	5,620	87	86,400	14,511	83
	5,734	1,093	81	10,085	1,769	82
	5,249	1,124	79	11,828	1,752	85
	7,590	1,391	82	9,591	1,651	83
Average	12,084	3,992	60	57,659	21,441	69
Geo. Mean	9,161	3,049	55	40,791	11,170	67

a variant of BBC algorithm (with multicut reformulation, penalty reformulation, all warm starts, and variable fixing activated) but with and without any cut selection. In Table 3.6, we report the CPU time (s) of the BBC algorithm with the cut selection (w/CS) versus without the cut selection (w/o CS). The results depict that cut selection improves the performance of the BBC algorithm on larger instances compared to the small ones (where it affects adversely).

Table 3.6: Performance of Cut Selection

Set	$\|S\| = 3$			$\|S\| = 4$		
	w/o CS	w/CS	% Red	w/o CS	w/CS	% Red
r08	4,646	5,506	- 19	47,329	36,798	22
	3,176	3,387	- 7	32,923	21,922	33
	2,902	2,365	18	86,037	20,428	76
	9,961	10,130	- 2	74,733	49,036	34
	6,266	5,314	15	86,400	45,106	48
	6,065	5,620	7	86,400	14,511	83
	4,767	1,093	77	1,331	1,769	- 33
	993	1,124	- 13	2,064	1,752	15
	1,073	1,391	- 30	1,439	1,651	- 15
Average	4,428	3,992	5	46,517	21,441	29
Geo. Mean	3,517	3,049		18,817	11,170	

3.6.3 Performance of BBC Algorithms

In this part of the computational experiments, we analyze the effectiveness of the three main acceleration techniques proposed in Section 4.4.2. For presentation purposes, we only include summarized results of all experiments. Interested readers are referred to Appendix B.3 for detailed results. We have implemented five different versions of our BBC algorithm. Based on our preliminary

computational experiments, we activate all warm start techniques, variable fixing, and cut selection techniques in all the five variants. The first one, called BBC1, uses the multicut reformulation. In the second, third, and fourth variants, we activate penalty reformulation, Pareto-optimal cuts, and supervalid and valid inequalities, one at a time respectively. Lastly, we activate all acceleration techniques. Hence, the resulting variants are as follows:

- BBC1: Multicut reformulation
- BBC2: Multicut reformulation with penalty reformulation
- BBC3: Multicut reformulation with Pareto-optimal cuts
- BBC4: Multicut reformulation with supervalid and valid inequalities
- BBC5: Multicut reformulation with all the acceleration techniques

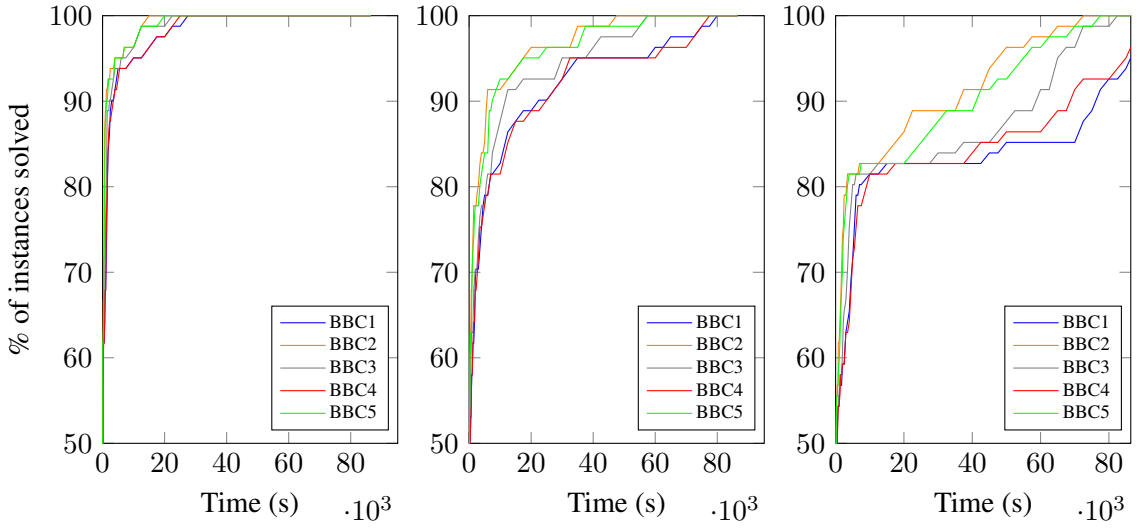
3.6.3.1 Results for Moderate Instances

We compare the performance of the five variants of our algorithm on small to moderate-size instances (sets "r01-r09") for different scenarios (interdiction budgets). The summary of results is reported in Table 3.7 whereas the detailed results of all 81 instances over three budget levels are presented in Appendix B.3. In Table 3.7, we report the average CPU time (seconds) of 9 instances in each set.

We observe that the BBC2 (multicut with penalty reformulation) is the most promising variant of the algorithm as it outperforms all other variants under all scenarios. The use of penalty reformulation (BBC2) reduces the CPU time by 52%, 58%, and 51% compared with BBC1 for $|S|$=2, $|S|$=3, and $|S|$=4, respectively. Moreover, BBC2 outperforms BBC5 on 68% of instances. This is due to the fact that we solve PSP_s to find the Pareto-optimal cuts in BBC5, which increases the CPU time. Note that the BBC5 (multicut with all acceleration techniques) is the next promising variant of the algorithm as it outperforms the three other variants under all scenarios. The use of all the acceleration techniques reduces the CPU time by 46%, 52%, and 42% compared with BBC1 for the three budget levels, respectively.

Note that all the five variants of our algorithm were able to solve 81 instances for low budget levels ($|S|$=2 and $|S|$=3) to optimality. However, BBC1 and BBC4 were unable to solve "r09" instances to optimality for $|S|$=4. BBC1 solves 5 (out of 9) instances of "r09" to optimality with an average gap of 2.7% (maximum of 11.6%). BBC4 solves 6 (out of 9) instances of "r09" to optimality with an average gap of 1.9% (maximum of 9.2%). This shows that adding supervalid and valid inequalities affects the performance of BBC1 for $|S|$=2 and $|S|$=3 slightly. However, as the instance size increases, BBC4 and BBC1 have similar performance in terms of number of instances solved to optimality. Although, BBC4 reduces the average gap by 30% compared with BBC1 for "r09" instances. Furthermore, adding Pareto-optimal cuts (BBC3) improves the CPU time by 28%, 30%, and 23.5% compared with BBC1 for $|S|$=2, $|S|$=3, and $|S|$=4, respectively. The use of all acceleration techniques (BBC5) reduces the CPU time by 46%, 51.5%, and 42% compared with BBC1 for $|S|$=2, $|S|$=3, and $|S|$=4, respectively.

To further compare the efficacy of acceleration techniques, we present the performance profile of acceleration techniques over all 81 instances. Figure 3.1 displays the percentage of instances solved to optimality within the time limit. In summary, BBC2 outperforms all variants of the algorithm by solving all instances in less time across all scenarios. Specifically, BBC2 solves all instances (for $|S|$=2 and $|S|$=3) in 48% and 42% less time than BBC1.



(a) $|S|$=2 (b) $|S|$=3 (c) $|S|$=4

Figure 3.1: Performance Profile of Different Variants of BBC Algorithm

Table 3.7: Summary of the Performance of the Variants of BBC Algorithm on r01-r09 Sets

Set	Average CPU Times (s) for $\|S\| = 2$					% Red.	% Red.
	BBC1	BBC2	BBC3	BBC4	BBC5	BBC2-BBC1	BBC5-BBC1
r01	14	5	14	14	5	66	66
r02	28	8	28	27	8	72	70
r03	74	20	67	72	20	73	73
r04	109	33	64	107	41	70	63
r05	978	387	756	930	454	60	53
r06	1,771	595	1,276	1,823	679	66	62
r07	146	38	100	159	48	74	67
r08	1,510	566	1,044	1,578	636	62	58
r09	10,051	5,359	7,250	10,073	6,056	47	40
Arith. Mean	1,631	779	1,178	1,643	883	66	61
Geo Mean.	284	95	220	285	107	65	60

Set	Average CPU Times (s) for $\|S\| = 3$					% Red.	% Red.
	BBC1	BBC2	BBC3	BBC4	BBC5	BBC2-BBC1	BBC5-BBC1
r01	21	6	20	22	6	73	71
r02	43	11	46	42	12	74	73
r03	109	29	97	100	28	73	74
r04	250	64	152	269	64	74	74
r05	3,302	1,329	2,692	3,208	1,571	60	52
r06	8,365	2,218	4,427	8,534	2,976	73	64
r07	840	269	494	919	299	68	64
r08	11,023	3,992	7,316	11,650	4,446	64	60
r09	38,029	17,942	27,959	37,990	20,571	53	46
Arith. Mean	6,887	2,873	4,800	6,970	3,330	68	64
Geo Mean.	855	267	632	869	295	68	64

Set	Average CPU Times (s) for $\|S\| = 4$					% Red.	% Red.
	BBC1	BBC2	BBC3	BBC4	BBC5	BBC2-BBC1	BBC5-BBC1
r01	34	8	34	38	8	77	76
r02	59	13	65	64	13	78	78
r03	132	31	112	134	34	76	74
r04	329	87	231	369	98	73	70
r05	2,701	1,181	2,283	3,223	1,391	56	48
r06	14,476	5,707	9,980	13,429	8,259	61	43
r07	2,971	1,042	2,040	3,197	1,262	65	57
r08	52,423	21,442	40,811	45,469	27,350	59	48
r09	56,462*	34,620	43,827	55,987**	36,924	39	35
Arith. Mean	14,399	7,126	11,042	13,546	8,371	65	59
Geo Mean.	1,461	484	1,187	1,519	558	64	57

* 5 out of 9 instances are solved to optimality
** 6 out of 9 instances are solved to optimality

Next, we conduct a paired t-test to assess the statistical significance of the means of a pair of different variants of the algorithm. We compare the performance of BBC2, BBC3, BBC4, and BBC5 with that of BBC1. The null hypothesis (H_0) states that there is no significant difference in the performance of the two algorithms, indicating a mean difference of zero. Conversely, the alternative hypothesis (H_1) suggests a significant difference between the two algorithms. To determine the threshold for rejecting the null hypothesis, we set the significance level to $\alpha = 0.05$. We reject the null hypothesis if the p-value is smaller than the significance level of 0.05. As the CPU time among 81 instances of "r01-r09" varies significantly, we first normalize the CPU time as follows:

$$\bar{t}_{bj} = \frac{t_{bj} - \mu_b}{\sigma_b} \tag{3.48}$$

where t_{bj} is the CPU time of instance b using algorithm j, and μ_b, σ_b are the mean and standard deviation of instance b using all algorithms. Based on the results of Table 3.8, there is a significant difference between the results of BBC1 and BBC2, BBC1 and BBC3, as well as BBC1 and BBC5. There is no significant difference between the results of BBC1 and BBC4 under $|S|$=2 and $|S|$=3 but their difference is significant under $|S|$=4.

Table 3.8: Paired T-test of Variants of BBC Algorithm (p-value)

Variant of Algorithm *	$\|S\|$=2	$\|S\|$=3	$\|S\|$=4
BBC2	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$
BBC3	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$
BBC4	0.1997	0.3292	0.0001
BBC5	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$

* All techniques are compared with BBC1

Based on the results in Table 3.7, Table 3.8, and Figure 3.1, we observe that the three most promising variants, in order of their performance, are BBC2, BBC5, and BBC3 as they outperform other variants by reducing the CPU time significantly and solving all the instances to optimality within the time limit.

3.6.3.2 Results for Larger Instances

We now report the performance of our algorithm on large instances (sets "r10 to r18"). For this set of experiments, we focus on the three most promising variants of our algorithm: BBC2, BBC3, and BBC5 only. We present the results of "r10-r18" in Table 3.9 for different budget scenarios. The column "Time" is computed by averaging the CPU time of all 9 instances in each set. The column "Gap" is computed by averaging the instances that reached the time limit with an integer solution. The column "Op/Fe/Nf" shows the number of instances solved optimally within the time limit (Op), the number of instances that reached the time limit with an integer solution (Fe), and the number of instances with no feasible solution (Nf). We observe that BBC2 outperforms BBC3 and BBC5 (as it is able to solve 31% and 10% more instances to optimality than BBC3 and BBC5, respectively). With BBC2, the number of instances with no feasible solution decreases by 26% and 14% compared with BBC3 and BBC5, respectively. Moreover, BBC2 is efficient in terms of average CPU time and average gap compared to BBC3 and BBC5. In conclusion, BBC2 is the most efficient variant of the algorithm (followed by BBC5).

In Table 3.10, we increase the interdiction budget (number of interdictions) in each scenario to test the performance of the algorithm over difficult instances. We set the number of interdictions at levels of 5%, 10%, 15%, and 20% of the total number of arcs, with any fractional values being rounded down. Consequently, when $|S| = 2$, the number of interdictions is set as B_1 being 5% of the arcs and B_2 at 10% of the arcs. In cases where $|S| = 3$, the number of interdictions is B_1 at 5% of the arcs, B_2 at 10%, and B_3 at 15% of the arcs. Lastly, for $|S| = 4$, the number of interdictions is determined as B_1 at 5% of the arcs, B_2 at 10%, B_3 at 15% of the arcs, and B_4 at 20% of the arcs. In Table 3.10, we report the average CPU time (s), the average gap on instances not solved to optimality, and the number of optimal solutions in each "r" set using the BBC2 algorithm. We observe that as the number of interdictions increases, the difficulty of the problem increases, thereby increasing the computation time significantly compared to the results of Table 3.7.

3.6.4 Comparison with Smith et al. (2007)'s Algorithm

Table 3.9: Summary of the Performance of BBC2, BBC3, and BBC5 Algorithms on "r10" to "r18" Sets

(a) $|S|$=2

Set	BBC2			BBC3			BBC5		
	Time (s) [1]	Gap (%) [2]	Op/Fe/Nf [3]	Time (s)	Gap (%)	Op/Fe/Nf	Time (s)	Gap (%)	Op/Fe/Nf
r10	14,508	-	9/0/0	21,867	-	9/0/0	16,338	-	9/0/0
r11	23,631	-	9/0/0	40,089	1.3	7/2/0	26,996	-	9/0/0
r12	63,119	13.5	3/3/3	64,961	30.5	3/1/5	64,164	25.8	3/1/5
r13	39,927	-	9/0/0	55,231	6.7	7/2/0	52,348	4.5	8/1/0
r14	86,185	8.3	1/6/2	86,400	14.6	0/6/3	86,250	7.5	1/5/3
r15	86,400	32.2	0/4/5	86,400	35.5	0/2/7	86,400	35.3	0/3/6
r16	55,854	-	9/0/0	68,581	9.1	8/1/0	62,114	7.4	8/1/0
r17	86,320	27.4	1/4/4	86,400	22.2	0/2/7	86,400	24.8	0/3/6
r18	86,400	-	0/0/9	86,400	-	0/0/9	86,400	-	0/0/9
Total			41/17/23			34/16/31			38/14/29
Arith. Mean	60,260			66,259			63,045		
Geo. Mean	51,789			61,187			55,650		

(b) $|S|$=3

Set	BBC2			BBC3			BBC5		
	Time (s) [1]	Gap (%) [2]	Op/Fe/Nf [3]	Time (s)	Gap (%)	Op/Fe/Nf	Time (s)	Gap (%)	Op/Fe/Nf
r10	29,164	1.1	8/1/0	51,682	2.7	7/2/0	34,090	1.1	8/1/0
r11	38,878	8.3	6/3/0	54,133	15.9	5/3/1	42,584	10	6/3/0
r12	66,041	31.2	3/1/5	69,202	-	3/0/6	67,659	39.2	3/1/5
r13	76,452	10.4	5/4/0	82,993	14.1	3/6/0	81,302	10.9	3/6/0
r14	86,400	16.4	0/7/2	86,400	21.4	0/5/4	86,400	17.5	0/6/3
r15	86,400	27.3	0/3/6	86,400	-	0/0/9	86,400	38.9	0/2/7
r16	73,705	14	4/5/0	82,819	16.8	2/7/0	79,591	13.9	3/6/0
r17	86,400	27.1	0/5/4	86,400	18.3	0/1/8	86,400	27.8	0/3/6
r18	86,400	-	0/0/9	86,400	-	0/0/9	86,400	-	0/0/9
Total			26/29/26			20/24/37			23/28/30
Arith. Mean	69,982			76,270			72,314		
Geo. Mean	65,922			74,896			68,993		

(c) $|S|$=4

Set	BBC2			BBC3			BBC5		
	Time (s) [1]	Gap (%) [2]	Op/Fe/NFe [3]	Time (s)	Gap (%)	Op/Fe/NFe	Time (s)	Gap (%)	Op/Fe/NFe
r10	49,099	10.1	7/2/0	59,731	13.5	4/5/0	53,841	13.6	6/3/0
r11	51,424	12.2	6/3/0	61,218	26.5	5/1/3	55,641	17.3	6/3/0
r12	67,684	-	3/0/6	73,001	-	3/0/6	68,215	-	3/0/6
r13	83,573	14.5	3/6/0	86,047	16.7	1/8/0	84,638	18.1	3/6/0
r14	86,400	20	0/6/3	86,400	28.2	0/4/5	86,400	21	0/5/4
r15	86,400	-	0/0/9	86,400	-	0/0/9	86,400	-	0/0/9
r16	81,136	16.5	3/6/0	86,045	20.4	1/8/0	84,530	18.6	2/7/0
r17	86,400	30	0/4/5	86,400	-	0/0/9	86,400	32	0/2/7
r18	86,400	-	0/0/9	86,400	-	0/0/9	86,400	-	0/0/9
Total			22/27/32			14/26/41			20/26/35
Arith. Mean	75,390			79,071			76,941		
Geo. Mean	73,754			78,261			75,684		

[1] The average time is computed on all 9 instances of each set [2] The average gap is computed on the instances that reached the time limit with an integer solution [3] Op: Number of instances solved to optimality. Fe: Number of instances that reached the time limit with an integer solution. Nf: Number of instances with no feasible solution.

Table 3.10: Summary of Performance of the BBC2 Algorithm for $B_s = \%$ of Arcs

Set	$\|S\| = 2$ $B_s = (5\%, 10\%)$			$\|S\| = 3$ $B_s = (5\%, 10\%, 15\%)$			$\|S\| = 4$ $B_s = (5\%, 10\%, 15\%, 20\%)$		
	Time (s)	Gap (%)	Opt.	Time (s)	Gap (%)	Opt.	Time (s)	Gap (%)	Opt.
r01	4	0.0	9	5	0.0	9	6	0.0	9
r02	7	0.0	9	10	0.0	9	12	0.0	9
r03	18	0.0	9	26	0.0	9	43	0.0	9
r04	143	0.0	9	227	0.0	9	390	0.0	9
r05	1,176	0.0	9	1,380	0.0	9	1,697	0.0	9
r06	4,976	0.0	9	7,283	0.0	9	16,066	0.0	9
r07	1,597	0.0	9	2,065	0.0	9	4,014	0.0	9
r08	20,747	0.0	9	42,243	7.6	7	43,224	8.5	7
r09	39,788	4.8	6	48,539	13.9	5	51,148	11.1	6
Arith. Mean	7,606	0.5		11,309	2.4		12,956	2.2	
Geo. Mean	436			620			878		

Smith et al. (2007) propose a cutting plane algorithm to solve the deterministic variant of the MCND problem under optimal interdiction. We use their algorithm for solving our stochastic MCND problem and compare the results of our BBC2 algorithm over moderate-size instances: set "r07" and set "r08". The pseudo-code of the Smith et al. (2007)'s algorithm is presented in Appendix B.1. The results of Table 3.11 highlight the efficiency and the robustness of our proposed BBC algorithm. As can be observed, while the cutting plane algorithm could solve 76% of 54 instances to optimality within the time limit, BBC2 solves all instances to optimality. Furthermore, our algorithm is an order of magnitude faster, on average, than Smith et al. (2007)'s algorithm.

3.6.5 Comparison with MibS Solver

We compare the performance of the BBC2 algorithm with MibS, a general-purpose mixed-integer bilevel linear optimization solver developed by DeNegre and Ralphs (2009) and Tahernejad et al. (2020). The solver is available at `https://github.com/coin-or/MibS`. MibS solver uses a branch-and-cut algorithm, which recursively partitions the feasible region and dynamically enhances the relaxation by adding valid inequalities. It iterates by generating bounds, pruning nodes, branching on variables, and eliminating infeasible solutions. We use the default settings of MibS solver version 1.2.1. Our preliminary experiments were conducted on "r01" instances, however, MibS was unable to solve any of these instances within the time limit. Therefore, we create new

Table 3.11: Comparison of the Computation Time of BBC2 Algorithm and Smith et al. (2007)'s Algorithm

Set	$\|S\| = 2$		$\|S\| = 3$		$\|S\| = 4$	
	BBC2	Smith et al. (2007)	BBC2	Smith et al. (2007)	BBC2	Smith et al. (2007)
r07	28	94	368	209	1,899	927
	22	134	216	952	2,090	6,832
	27	666	199	11,601	1,592	86,400*
	63	161	689	518	1,070	463
	36	1,498	332	13,310	1,129	12,470
	27	10,770	356	86,400*	1,335	86,400*
	50	33	97	70	73	99
	48	215	71	239	102	259
	40	820	92	861	88	596
Arith. Mean	38	1,599	269	12,684	1,042	21,605
Geo. Mean	36	369	211	1,393	573	2,526
r08	181	166	5,506	3,568	36,798	31,686
	409	5,897	3,387	86,400*	21,922	86,400*
	210	86,400*	2,365	86,400*	20,428	86,400*
	664	630	10,130	10,206	49,036	45,398
	701	14,843	5,314	86,400*	45,106	86,400*
	348	86,400*	5,620	86,400*	14,511	86,400*
	1,159	407	1,093	722	1,769	1,197
	593	2,015	1,124	1,473	1,752	3,090
	829	6,949	1,391	5,846	1,651	7,391
Arith. Mean	566	22,634	3,992	40,824	21,441	48,262
Geo. Mean	483	4,180	3,049	13,252	11,170	23,506

* Time limit reached

small instances from "r" instances with deterministic demand and number of interdictions. The results are presented in Table 3.12 where we compare the CPU time (s) and the objective function value of MibS and BBC2 algorithm. We observe that our proposed algorithm with acceleration techniques improves CPU time by orders of magnitude.

3.6.6 Results of Uncertain Demand and Number of Interdiction

We also analyze the performance of the BBC2 algorithm for instances with stochastic demand and stochastic number of interdictions. We use the benchmark instances with 16, 32, and 64 scenarios for the demand (Sarayloo et al., 2021). The instances can be downloaded from `https://github.com/shabnamvaziri/Instances-MCND-SI`. In Table 3.13, we present the average CPU time and gap over the nine instances in each "r" set. We present the detailed results in Appendix B.3. We can observe that considering the stochastic demand besides uncertain budget increases the problem difficulty; however, BBC2 is capable of solving 284 instances (out of 486 instances) to optimality over all demand and interdiction budget scenarios within the time limit.

Table 3.12: Comparison of the Performance of MibS Solver and BBC2 Algorithm

$\|N\|, \|A\|, \|K\|$	B	MibS		BBC2	
		Time (s)	Obj	Time (s)	Obj
10,15,5	1	11	138,749	< 1	138,749
	2	14	292,166	< 1	292,166
	3	30	444,844	< 1	444,844
	4	52	556,297	< 1	556,297
	5	54	635,497	< 1	635,497
10,15,10	1	6.6	898,918	< 1	898,918
	2	24	1,012,154	< 1	1,012,154
	3	65	1,123,046	< 1	1,123,046
	4	105	1,228,028	< 1	1,228,028
	5	101	1,299,546	< 1	1,299,546
10,20,5	1	260	331,105	< 1	331,105
	2	519	469,584	< 1	469,584
	3	648	573,549	< 1	573,549
	4	671	576,106	< 1	576,106
	5	851	637,286	< 1	637,286
10,20,10	1	176	592,579	< 1	592,579
	2	655	1,038,001	< 1	1,038,001
	3	797	1,164,033	< 1	1,164,033
	4	907	1,270,589	< 1	1,270,589
	5	1,260	1,305,904	< 1	1,305,904

3.6.7 Sensitivity Analysis

In this section, we perform a sensitivity analysis of the effects of the number of interdiction scenarios on the optimal design. We show the importance of stochastic over deterministic variants in cases where the number of interdictions is uncertain. Moreover, we analyze the effect of parameter Φ on the performance of BBC algorithm and the network design.

3.6.7.1 Effect of Varying Interdiction Scenarios on the Network Design

First, we analyze the effects of changing the number of scenarios on the optimal solution. For this purpose, we present the network solution of a specific instance of set "r07", which comprises 10 nodes, 82 arcs, and 10 commodities. Figure 3.2 shows the following cases: (a) solution with no interdiction, (b) solution with one scenario with $B_s = 1$, (c) solution with two scenarios with $B_1 = 1$, $B_2 = 2$, (d) solution with three scenarios with $B_s = s$, $s = 1, ..., 3$, and (e) solution with four scenarios with $B_s = s$, $s = 1, ..., 4$. We observe that the number of installed arcs increases as the number of scenarios increases. Moreover, we notice that the installed arcs remain unchanged across different scenarios, while the number of backup arcs increases as the number of scenarios grows. This depicts that the network design is survivable by maintaining critical arcs consistently

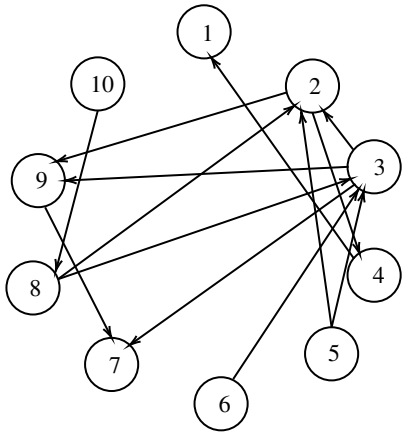
Table 3.13: Summary of Performance of the BBC2 Algorithm for Uncertain Demand and Number of Interdictions

Set	$\|\Omega\| = 16$					
	$\|S\| = 2$		$\|S\| = 3$		$\|S\| = 4$	
	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)
r04	586	0	2,426	0	4,253	0
r05	20,407	0.3	26,391	1.1	22,246	1.6
r06	40,522	2.2	44,872	4.6	48,892	5.4
r07	1,333	0	24,457	0.5	52,502	4.5
r08	16,186	0	76,137	7.3	77,120	16.1
r09	62,350	6.0	65,493	15.6	65,380	42.1
Arith. Mean	23,564	1.4	39,963	4.8	45,065	11.6
Geo. Mean	9,312		26,551		32,708	

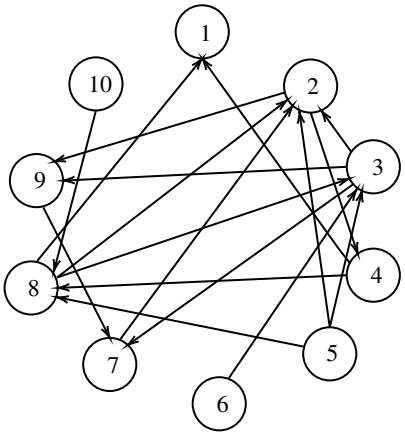
Set	$\|\Omega\| = 32$					
	$\|S\| = 2$		$\|S\| = 3$		$\|S\| = 4$	
	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)
r04	709	0	990	0	4,043	0
r05	31,138	1.9	33,733	2.8	32,122	2.8
r06	50,658	5.2	52,949	8.9	50,211	9.3
r07	7,612	0	41,027	0.4	59,098	5.1
r08	25,035	0.9	77,230	8.7	77,065	16.9
r09	66,796	8.5	72,964	18.6	70,953	48.5
Arith. Mean	30,325	2.7	46,482	6.6	48,915	13.7
Geo. Mean	15,568		27,241		35,805	

Set	$\|\Omega\| = 64$					
	$\|S\| = 2$		$\|S\| = 3$		$\|S\| = 4$	
	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)
r04	1,937	0	7,296	0	5,190	0
r05	39,485	6.6	37,462	6.0	39,006	6.7
r06	56,662	5.5	63,184	10.5	62,134	14.9
r07	8,848	0	55,701	1.0	70,640	6.7
r08	41,438	1.5	77,909	11.9	78,828	23.2
r09	78,602	12.9	78,438	22.2	77,949	44.9
Arith. Mean	37,829	4.4	53,332	8.6	55,624	16.1
Geo. Mean	22,357		42,483		41,963	

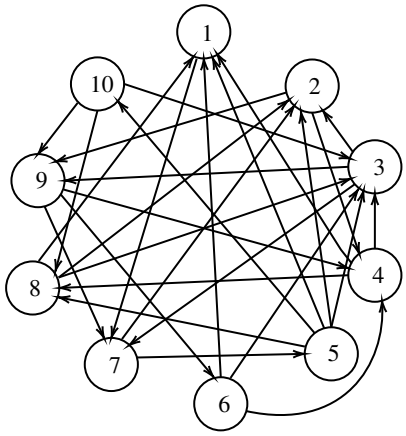
while introducing additional backup arcs to enhance adaptability to various conditions.



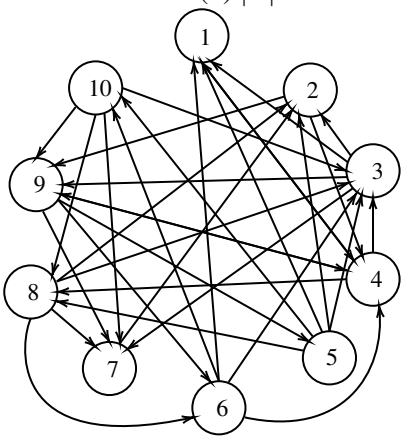
(a) No interdiction



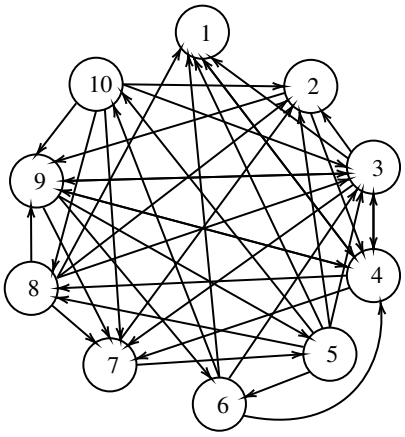
(b) $|S|$=1



(c) $|S|$=2



(d) $|S|$=3



(e) $|S|$=4

Figure 3.2: Effects of Changing the Number of Scenarios on Optimal Design

Second, we show the importance of using the stochastic model over the deterministic variant in cases where the number of interdictions is uncertain. For this, we choose the same instance as above. We find the optimal design for each deterministic number of interdictions. Then, we compute the unmet demand under each design if the actual number of interdictions occurs. The results are depicted in Table 3.14. The first column represents the interdiction budget used for the analysis. The following four columns display the unmet demand for each design, corresponding to the actual number of interdictions mentioned in the column headers. The last column presents the installation cost for each design. The results of Table 3.14 show the robustness of the design obtained with the stochastic model, as the stochastic design satisfies all demands for all numbers of interdictions. However, the stochastic design has a higher installation cost than deterministic designs. Among deterministic designs, the case with $B = 4$ is the same as the stochastic design as the designer accounts for the worst-case disruptions. In contrast, the design of $B = 1$ performs poorly when a higher number of interdictions happen.

Table 3.14: Comparison of Deterministic and Stochastic Designs (Unmet Demand)

	Actual no. of interdictions				Installation cost ($)
	$B = 1$	$B = 2$	$B = 3$	$B = 4$	
Deterministic design, $B = 1$	0	154	242	313	5,372
Deterministic design, $B = 2$	0	0	130	183	7,299
Deterministic design, $B = 3$	0	0	0	154	9,633
Deterministic design, $B = 4$	0	0	0	0	13,187
Stochastic design	0	0	0	0	13,187

3.6.7.2 Effect of Varying Φ on the Performance of BBC Algorithm and the Network Design

In Table 3.15, we compare the number of installed arcs for six randomly selected instances by varying the weight of pre-interdiction and post-interdiction cost. As we can observe, the number of installed arcs may increase as we decrease the weight of pre-interdiction cost or increase the weight of post-interdiction cost, to have a more survivable network design against interdictions. This increases the installation cost of the network.

To evaluate the impact of changing the weight of pre-interdiction and post-interdiction costs on the performance of the BBC algorithm and the obtained design, we select the weight of pre-interdiction cost to be $\Phi = 0.2$, $\Phi = 0.5$, and $\Phi = 0.8$. In Table 3.16, we compare the CPU

Table 3.15: Effects of Varying the Weight of Pre-interdiction Cost (Φ) on the Network Design

$\|N\|$	$\|A\|$	$\|K\|$	$\Phi = 1$	$\|S\| = 2$				$\|S\| = 3$				$\|S\| = 4$			
				$\Phi = 0.8$	$\Phi = 0.5$	$\Phi = 0.2$	$\Phi = 0$	$\Phi = 0.8$	$\Phi = 0.5$	$\Phi = 0.2$	$\Phi = 0$	$\Phi = 0.8$	$\Phi = 0.5$	$\Phi = 0.2$	$\Phi = 0$
10	82	10	29	36	36	36	36	36	37	37	37	36	37	37	37
10	83	25	24	41	41	42	44	49	50	50	51	58	59	59	60
10	83	25	39	48	48	49	50	51	53	53	53	51	55	55	55
10	83	50	41	59	61	63	63	63	64	64	64	64	64	65	66
20	120	40	102	103	105	105	106	104	105	106	107	104	106	106	108
20	120	40	79	84	85	85	86	87	88	90	91	89	91	91	92

time (s) for the BBC2 algorithm applied to set "r07", which consists of 10 nodes, 82 arcs, and 10 commodities. We observe that as the weight of the pre-interdiction cost (Φ) decreases, the CPU time increases. Decreasing Φ implies assigning more importance to post-interdiction costs, which leads to opening more arcs to obtain a more survivable design against interdictions. With the increased number of arcs, the SPs become difficult.

Table 3.16: Effects of Varying the Weight of Pre-interdiction Cost (Φ) on CPU Time (s)

Set	$\|S\| = 2$			$\|S\| = 3$			$\|S\| = 4$		
	$\Phi = 0.8$	$\Phi = 0.5$	$\Phi = 0.2$	$\Phi = 0.8$	$\Phi = 0.5$	$\Phi = 0.2$	$\Phi = 0.8$	$\Phi = 0.5$	$\Phi = 0.2$
r07	28	27	33	368	380	425	1,899	3,025	8,447
	22	23	34	216	288	432	2,090	2,245	2,891
	27	32	38	199	251	282	1,592	2,269	2,474
	63	72	70	689	794	966	1,070	1,726	1,900
	36	47	68	332	434	982	1,129	1,321	2,304
	27	65	87	356	415	460	1,335	1,452	1,472
	50	103	150	97	124	132	73	119	168
	48	96	113	71	100	172	102	124	185
	40	97	123	92	102	118	88	92	125
Arith. Mean	38	62	79	269	321	441	1,042	1,375	2,218
Geo. Mean	36	54	69	211	257	340	573	743	1,047

3.7 Conclusion

We studied a multicommodity network design problem with stochastic budget interdiction. As the designer does not have complete information about the interdictor's budget and attacking capabilities, we considered different scenarios for interdiction budget. Using the defender-attacker-defender framework, we presented a trilevel stochastic MIP model which was reduced to the bilevel model using strong duality. We devised a branch-and-Benders-cut algorithm enriched with several acceleration techniques to improve its efficiency and robustness. Our extensive computational experiments and statistical analysis show that multicut reformulation combined with penalty reformulation, warm start, variable fixing, and cut selection techniques outperforms other variants of

the branch-and-Benders-cut algorithm. Large instances with up to 20 nodes, 220 arcs, 200 commodities, and 4 scenarios were solved to optimality with the proposed variant. Our results clearly demonstrate the efficacy of the several acceleration techniques and the performance of the Benders decomposition algorithm that can find provably optimal solutions within reasonable computational time. Moreover, the proposed algorithm outperformed the general-purpose solver for bilevel problems (MibS) and the solution method proposed in the literature. We also analyzed the effects of changing the number of scenarios on the design and showed the benefits of stochastic design when the number of interdictions is uncertain.

Chapter 4

Fortification of Spanning Trees under Stochastic Interdictions

Abstract In this paper, we study the fortification of minimum spanning tree (MST) and optimum communication spanning tree (OCST) problems by considering the effects of interdictions. The goal of the MST problem is to find a spanning tree with minimum installation cost, and the goal of the OCST problem is to satisfy the communication requirements in a network with minimum communication cost. The fortification problem is modeled as a two-player defender-interdictor-defender game. The goal is to find the optimal fortification strategy in a way that the increase in the MST/OCST costs due to the interdiction of unfortified edges is minimized. We incorporate the uncertainty in the number of interdictions as the defender does not have complete information about the interdiction resources. Therefore, the defender intends to minimize the expected maximum damage in the first level. In the second level, the interdictor interdicts the edges of the graph with a limited budget to maximize the MST/OCST costs. In the third level, the defender optimizes the MST/OCST costs in the remaining graph. We use backward sampling framework to solve both the deterministic and stochastic versions of the MST and OCST problems. We use the waiting list acceleration technique to improve the performance of the backward sampling framework. Our results demonstrate the efficiency of the backward sampling framework algorithm with the waiting list

acceleration technique for solving the deterministic and stochastic MST/OCST fortification problems. We observe that the number of interdictions has a high impact on the computational time of the algorithm. Moreover, we highlight the benefits of using the stochastic model when the number of interdictions is uncertain.

4.1 Introduction

Minimum spanning tree (MST) problem is a well-studied combinatorial problem with applications in the design of transportation, computer, communication, electric grid, and water supply networks. The goal is to connect all the nodes of the undirected, connected graph with minimum cost. We refer the readers to surveys by Graham and Hell (1985) and Bazlamaçcı and Hindi (2001) for the advancements of solution methodologies for solving MST problem. Another problem related to MST is the optimum communication spanning tree (OCST). The goal of the OCST problem is to satisfy the communication requirements between nodes of the network with minimum communication cost. If the communication cost is the only criterion in choosing the optimal solution, the solution would be the union of the shortest paths between communication pairs. This solution has more than $|N| - 1$ edges where $|N|$ is the number of nodes; therefore, the construction cost increases. As a result, OCST makes a trade-off between the minimum number of edges and the minimum communication cost for satisfying the communication requirements. OCST problem has practical applications for transportation and communication network design problems. OCST problem avoids cycles and the edges of OCST have high utilization which is a desired characteristic in transportation networks with expensive installation costs. Moreover, the OCST problem is useful when the building cost of the network depends on the number of established connections e.g., in virtual networks (Mota, 2016).

The feasible solution of OCST involves all possible spanning trees connecting the nodes within the graph. Therefore, OCST has the same feasible solution set as the MST problem, but their objective functions differ. In MST, the goal is to connect all the nodes with minimum installation cost whereas in OCST, the goal is to minimize the communication cost. This difference in the objective function makes OCST a *NP*-hard problem whereas MST is a polynomial-time solvable

problem (Zetina et al., 2019).

Networks are at risk of natural or man-made disruptions. Disruptions have short-term and long-term effects on the performance of networks. It is important to find critical infrastructures of networks and reduce the vulnerability of networks by protection strategies. Network interdiction models are used to find the critical infrastructures of networks, whose interdiction will cause the maximum damage to the network. Network interdiction problems consist of two opposing players, leader and follower, in a game theoretic framework. In the first level, the interdictor finds the nodes or edges of the network to interdict in order to cause the maximum damage to the network. In the second level, the follower optimizes the network after interdiction. Network interdiction problems were first introduced for military applications (McMasters and Mustin, 1970). Network interdiction has applications in power grid analysis, illicit supply chain, and cybersecurity, to name but a few. We refer the readers to the survey by Smith and Song (2020) for the enhancements and applications of the network interdiction field.

To reduce the negative impacts of interdictions on networks, one can fortify the networks e.g., by placing surveillance cameras. The objective of fortification models is to identify infrastructures that, if protected, would mitigate the negative effects of interdictions. If the protection strategy considers protecting the most critical assets, this strategy may result in suboptimal fortification plans (Brown et al., 2006). The reason is that these strategies do not consider how strengthening certain parts of the network can change the set of critical components. Therefore, they do not consider the interactions within the network, which can lead to suboptimal results. Fortification problems are tri-level mathematical models, a game between defender-interdictor-defender. In the first level, the defender determines the defense strategy to minimize the negative impacts of interdictions. In the second level, the interdictor interdicts the unfortified nodes/edges of the network to cause the maximum damage. In the third level, the defender optimizes the residual network after interdiction.

In this paper, we intend to find the most effective defense strategy for the MST and OCST problems considering uncertain number of interdictions. To the best of our knowledge, this is the first work considering the effects of interdiction and fortification on the OCST problem. However, MST problem with interdiction has been studied in the literature. This paper extends the bi-level MST problem with interdiction by considering the fortification of the MST problem. The interdictor

intends to increase the minimum spanning tree cost or minimum communication cost for MST and OCST problems, respectively by interdicting U edges. On the other hand, the defender has the chance to fortify Q edges of the network to minimize the adverse effects of interdictions. We consider the uncertainty in the number of interdictions as the defender does not have complete information about interdiction resources by presenting a tri-level stochastic mathematical model to find the best defense strategy. To solve the tri-level stochastic model, we use the existing exact algorithm based on the backward sampling framework (BSF). We implement the BSF algorithm for two reasons. First, the BSF algorithm can solve problems with mixed-integer model in the third level. As the OCST problem is a mixed-integer program, we cannot use techniques based on dualization. Second, we can solve both the deterministic and stochastic versions of the MST and OCST problems using the BSF algorithm.

4.1.1 Contribution

The contribution of this paper is two-fold. First, MST and OCST problems have important applications for designing computer, communication, and transportation networks. Therefore, it is paramount to find optimal fortification plans to improve their resiliency. To the best of our knowledge, this is the first paper considering the MST and OCST problems with fortification. In practice, the defender does not have prior knowledge of interdiction resources. Consequently, it is more logical to consider interdiction resources as uncertain parameters. In this paper, we extend the deterministic model by considering the uncertainty in the number of interdictions by presenting a tri-level stochastic mathematical model for finding the optimal fortification strategy for MST and OCST problems. Secondly, fortification problems are a difficult class of optimization problems. Considering stochastic parameters in the fortification problem increases the complexity. To this end, our contribution is to develop an exact method based on state-of-the-art solution methods for solving the MST and OCST fortification problems. We implement the existing backward sampling framework to solve the deterministic and stochastic versions of MST/OCST problems with fortification. This is the first paper using the backward sampling framework for the tri-level stochastic fortification problem. We also enhance the performance of the BSF algorithm using the waiting list acceleration technique. Our results demonstrate the efficiency of the backward sampling framework algorithm

with the waiting list acceleration technique for solving the deterministic MST fortification problem for instances with up to 100 nodes, and 4,950 edges. We observe that the BSF algorithm with the waiting list acceleration technique reduces CPU time by an average of 55% compared with the BSF algorithm for large-sized MST instances. The stochastic MST fortification problem is solved for instances with up to 4 scenarios, 40 nodes, and 780 edges. The deterministic OCST fortification problem is solved for instances with up to 50 nodes, 1,225 edges, and 635 communication requests. The stochastic version of OCST is solved for instances with up to 30 nodes for 2 and 3 scenarios, and instances with up to 20 nodes for 4 scenarios. We show that the algorithm is sensitive to the number of interdictions. Moreover, we highlight the benefits of using the stochastic model when the number of interdictions is uncertain.

The remainder of this paper is structured as follows: In Section 4.2, we review the relevant literature. In Section 4.3, we provide the stochastic mathematical models for fortifying the MST and OCST problems. The solution methodology for solving the stochastic model for MST and OCST is presented in Section 4.4. In Sections 4.5 and 4.6, we present the results of our computational experiments and conclude the paper, respectively.

4.2 Literature Review

We provide a brief review of the literature concerning network fortification models, stochastic interdiction/fortification models, MST problems with interdictions, and OCST problem.

4.2.1 Network Fortification and Interdiction

Brown et al. (2006) are among the first authors studying the optimal fortification of networks. They show that protecting the critical assets found in the bi-level interdiction problem provides suboptimal solutions. Scaparra and Church (2008b) reformulate the r-interdiction median problem with fortification (IMF) as a maximal covering problem. They show that the worst-case interdiction occurs by interdicting unfortified facilities. Therefore, they sort all interdiction patterns and try to fortify facilities in worst-case patterns. The basic assumption for IMF reformulation is that they aim to maximize the coverage of the most disruptive interdiction strategies with q fortifications.

Zhang and Ramirez-Marquez (2013) introduce the fortification of the maximum flow network with information asymmetry where the defender does not have complete information about the attack resources available for the interdiction. Jiang and Liu (2018) introduce the fortification model with information asymmetry in water supply networks. They assume that the interdiction budget is not known to the defender, and the interdictor does not have information about the network's operational capacity. Ramamoorthy et al. (2018) and Ghaffarinasab and Atayi (2018) present models and algorithms for hub protection problem. Hien et al. (2020) present a tri-level mathematical model for fortifying single and multicommodity networks with uncertain demand. Sadati et al. (2020) present a tri-level mathematical model for fortifying the depots in vehicle routing problem. Authors use fortification problem to reduce the negative effects of interdictions for median problem (Liberatore et al., 2012), knapsack problem (Leitner et al., 2022), power grid (Costa et al., 2018; Wu et al., 2022), water distribution (Wu et al., 2021), maximum flow (Ghosh and Jaillet, 2022), among others. We summarize some of the fortification papers in Table 4.1.

One of the widely used algorithms to solve the fortification problem is the implicit enumeration technique. Cappanera and Scaparra (2011) implement implicit enumeration for finding the optimal defense strategy of shortest path networks. Parajuli et al. (2017) and Parajuli et al. (2021) implement implicit enumeration for protecting capacitated supply networks. Implicit enumeration is also used for solving hub protection problem (Ghaffarinasab and Atayi, 2018; Ramamoorthy et al., 2018), and r-interdiction median problem with fortification (Scaparra and Church, 2008a; Liberatore et al., 2012). Lozano and Smith (2017) present backward sampling framework for the first time for solving the shortest path fortification problem and capacitated lot sizing interdiction problem with fortification. Lozano and Smith (2017) compare the results of the shortest path fortification problem solved by BSF and implicit enumeration algorithms. They show that the BSF algorithm reduces CPU time by more than two orders of magnitude compared with the implicit enumeration algorithm. Lozano et al. (2017) solve the traveling salesman problem with interdiction and fortification using backward sampling framework. Leitner et al. (2022) present a branch-and-cut algorithm based on fortification cuts for the knapsack fortification problem and shortest path fortification problem. Sadati et al. (2020), Wu et al. (2021), and Ghosh and Jaillet (2022) present heuristic algorithms to solve the fortification problem.

Table 4.1: Classification of Fortification Papers

Articles	Model Type		Methodology		Type of Problem
	Deterministic	Stochastic	Exact	Heuristic	
Brown et al. (2006)	✓		✓		Power grid, Supply chain
Cappanera and Scaparra (2011)	✓		✓		Shortest path
Alderson et al. (2011)	✓		✓		Municipal transportation network
Liberatore et al. (2012)	✓		✓		Median problem
Lozano et al. (2017)	✓		✓		Travelling salesman problem
Lozano and Smith (2017)	✓		✓		Shortest path, Capacitated lot sizing
Parajuli et al. (2017)	✓		✓		Capacitated supply chain
Ramamoorthy et al. (2018)	✓		✓		Hub protection
Hien et al. (2020)		✓	✓		Single and multicommodity
Ghorbani-Renani et al. (2020)	✓		✓		Water, gas, and power utilities
Sadati et al. (2020)	✓			✓	Vehicle routing problem
Parajuli et al. (2021)	✓		✓		Capacitated supply chain
Wu et al. (2021)	✓			✓	Urban water distribution
Leitner et al. (2022)	✓		✓		Knapsack, Shortest path
Ghosh and Jaillet (2022)	✓			✓	Maximum flow
Wu et al. (2022)	✓			✓	Electric power system
This paper		✓	✓		MST, OCST

In network interdiction and fortification models, different parameters can be uncertain e.g., interdiction budget (Liberatore et al., 2011; Bhuiyan et al., 2021), interdiction success (Cormican et al., 1998; Janjarassuk and Linderoth, 2008; Losada et al., 2012), network configuration (Collado et al., 2017), demand (Hien et al., 2020), arc capacities (Lei et al., 2018), and traveling cost (Song and Shen, 2016; Nguyen and Smith, 2022). Liberatore et al. (2011) consider the protection of p-median problem with uncertain interdiction budget. Hien et al. (2020) present tri-level mathematical models for fortifying single-commodity and multicommodity networks under demand uncertainty.

4.2.2 MST and OCST Problems

In this subsection, we review the papers related to the MST problem with interdictions. As the effect of interdictions was not studied for the OCST problem as far as we are aware, we review the solution methodologies for solving the OCST problem.

We refer to the most vital edge in the MST as relevant literature to the MST problem with interdictions. The goal is to find a single edge whose deletion causes the maximum increase in the length of the spanning tree (Hsu et al., 1991). Researchers extend the most vital edge to the k-most vital edges (Lin and Chern, 1993; Frederickson and Solis-Oba, 1999). Bazgan et al. (2012) present the bi-level mathematical model for the MST problem with interdictions for the first time. They

transform the bi-level formulation into single-level formulation. They compare the results of MIP formulation with implicit and explicit enumeration algorithms. Zenklusen (2015) provide an *O(1)*-approximation algorithm for the MST interdiction problem. Linhares and Swamy (2017) extend the work in Zenklusen (2015) by presenting a better approximation algorithm. Wei et al. (2021) present a new single-level mathematical model for the MST interdiction problem. They show that this formulation has a stronger linear relaxation compared to the bi-level model. Wei and Walteros (2022) present a set-covering reformulation for the MST interdiction problem.

Another line of research considering the tree-type graphs is critical node detection as presented by Shen and Smith (2012) and Baggio et al. (2021). The goal is to find a set of nodes whose removal disconnects the network at most. The application of critical node detection can be the control of epidemic diseases by vaccinating the limited population to minimize the spread of the disease.

The *NP*-hard OCST problem was introduced by Hu (1974). Several authors use heuristic and metaheuristic algorithms to solve the problem (Lin and Gen, 2006; Fischer and Merz, 2007; Rothlauf, 2009). Ahuja and Murty (1987) present the first exact algorithm for solving OCST based on branch-and-bound. Contreras et al. (2010) present Lagrangian relaxation for solving the OCST problem. Tilk and Irnich (2018) use column-and-row-generation algorithm to solve the OCST problem and Zetina et al. (2019) propose branch-and-Benders-cut algorithm. Agarwal and Venkateshan (2019) present new valid inequalities for the OCST problem.

To the best of our knowledge, this is the first paper focusing on the optimal fortification strategy for the MST and OCST problems. We consider the uncertain number of interdictions as the defender does not have complete information about the interdiction resources. We solve the deterministic and stochastic fortification models with backward sampling framework. We show that this technique can be used for the stochastic fortification model with a slight change in the algorithm.

4.3 Mathematical Model

In this paper, we study the fortification problem of MST and OCST with interdictions. As the defender does not know the exact amount of interdiction resources, we present a tri-level stochastic

mathematical model for finding the fortification strategy. The goal is to fortify the edges of the network in order to minimize the expected negative effects of interdictions on the network. In the first level, the defender fortifies the limited number of edges to minimize the expected maximum damage in different scenarios. In the second level, the interdictor interdicts the unfortified edges of the network to maximize the network cost. In the third level, the defender solves the MST/OCST problem in the remaining network.

We consider the following assumptions for our mathematical model: (i) the network parameters are completely known by both the defender and the interdictor, (ii) the game is a sequential game between the defender and the interdictor, (iii) the game is only one round, (iv) the defender does not have complete information about the interdiction resources, and (v) we consider complete fortifications and interdictions which means that the fortified edges cannot be interdicted and the interdicted edges cannot be part of the MST/OCST solution. We present the sets, parameters, and decision variables in Table 4.2.

Table 4.2: Table of Notations

Sets	
N	Set of nodes (indexed by $i, i \in N$)
A	Set of directed arcs (i, j)
E	Set of undirected edges $\{i, j\}$
R	Set of communication requests (indexed by r, $r \in R$)
S	Set of scenarios (indexed by $s, s \in S$)
Parameters	
o_r	Origin of request $r \in R$
d_r	Destination of request $r \in R$
W_r	Communication request quantity between (o_r, d_r)
c_{ij}	Installation cost of edge $\{i, j\}$
d_{ij}	Communication cost between edge $\{i, j\}$
Q	Number of fortifications
U_s	Number of interdictions in scenario $s \in S$
p_s	Probability of scenario $s \in S$
Decision Variables	
q_{ij}	1 if edge $\{i, j\}$ is fortified
u_{ijs}	1 if edge $\{i, j\}$ is interdicted in scenario $s \in S$
y_{ijs}	1 if edge $\{i, j\}$ is part of the spanning tree in scenario $s \in S$
x^r_{ijs}	Portion of communication request $r \in R$ that is routed through arc (i, j) in scenario $s \in S$

4.3.1 Stochastic Model for the MST Problem

We consider a connected, complete, and undirected graph $G = (N, E)$ where N is the set of nodes and E is the set of undirected edges. We show the undirected edges by $\{i, j\}$, and the directed arcs by (i, j). We define directed arcs for the routing of communication requests. Each edge has the installation cost c_{ij}. In this paper, we use the network design formulation presented by Magnanti and Wolsey (1995). In this formulation, one of the nodes will be selected as the root node known as the origin of the communication requests o_r. The root node should send one unit of communication request to all other nodes, known as the destination of communication requests d_r. Binary decision variable y_{ijs} is 1 if the edge $\{i, j\}$ is part of the spanning tree in scenario $s \in S$ and 0 otherwise. The continuous decision variable x^r_{ijs} represents the portion of communication request $r \in R$ that is routed through arc (i, j) in scenario $s \in S$. Binary decision variable q_{ij} is 1 if edge $\{i, j\}$ is fortified. The defender can fortify Q edges. The interdictor can interdict U_s unfortified edges of the network in scenario $s \in S$. The interdiction decisions are represented by binary decision variable u_{ijs} which takes the value 1 if the edge is interdicted in scenario $s \in S$.

In the stochastic model, the defender determines the defense strategy under uncertainty about the number of interdictions. In other words, the defender's goal is to minimize the expected maximum damage of the interdictor across possible scenarios. The tri-level stochastic mathematical model is as follows:

$$\text{(DLP-S)}: \quad \min_{q} \sum_{s \in S} p_s H_s(q) \tag{4.1}$$

$$\text{s.t.} \quad \sum_{\{i,j\} \in E} q_{ij} = Q \tag{4.2}$$

$$q_{ij} \in \{0,1\} \qquad \forall \{i,j\} \in E \tag{4.3}$$

$$\text{(ILP-S)}: \quad H_s(q) = \max_{u} A_s(u) \tag{4.4}$$

$$\text{s.t.} \quad \sum_{\{i,j\} \in E} u_{ijs} \leq U_s \tag{4.5}$$

$$u_{ijs} \leq 1 - q_{ij} \qquad \forall \{i,j\} \in E \tag{4.6}$$

$$u_{ijs} \in \{0,1\} \qquad \forall \{i,j\} \in E \tag{4.7}$$

$$\text{(ULP-S)}: \quad A_s(u) = \min_{x,y} \sum_{\{i,j\} \in E} c_{ij} y_{ijs} \tag{4.8}$$

$$\text{s.t.} \quad \sum_{j \in N:(j,i) \in A} x^r_{jis} - \sum_{j \in N:(i,j) \in A} x^r_{ijs} = \begin{cases} -1 & \text{if } i = o_r \\ 1 & \text{if } i = d_r \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in N, r \in R \tag{4.9}$$

$$x^r_{ijs} + x^r_{jis} \leq y_{ijs} \qquad \forall \{i,j\} \in E, r \in R \tag{4.10}$$

$$\sum_{\{i,j\} \in E} y_{ijs} = |N| - 1 \tag{4.11}$$

$$y_{ijs} \leq 1 - u_{ijs} \qquad \forall \{i,j\} \in E \tag{4.12}$$

$$y_{ijs} \in \{0,1\} \qquad \forall \{i,j\} \in E \tag{4.13}$$

$$x^r_{ijs} \geq 0 \qquad \forall (i,j) \in A, r \in R \tag{4.14}$$

In the first level (DLP-S), the defender minimizes the expected maximum damage of interdictions in the objective function (4.1). Constraint (4.2) limits the number of fortifications to Q edges. Constraint (4.3) imposes a binary limitation on the fortification variables. In the second level (ILP-S), the interdictor maximizes the minimum cost of MST in the objective function (4.4). Constraint (4.5) ensures that maximum U_s edges are interdicted. Constraint (4.6) states that the interdictor

cannot interdict the fortified edges. Constraint (4.7) indicates that interdiction variables are binary. In the third level (ULP-S), the defender finds the optimal solution of MST in the residual network by minimizing the installation cost in (4.8). Constraint (4.9) ensures the flow balance for each request $r \in R$ in node $i \in N$. Constraint (4.10) guarantees that the flow passes each edge if the edge is part of the spanning tree. Constraint (4.11) represents the maximum number of open edges in the spanning tree. Constraint (4.12) indicates that the interdicted edges cannot be part of the spanning tree. Constraints (4.13) and (4.14) are the domain constraints.

4.3.2 Stochastic Model for the OCST Problem

To present the OCST interdiction problem with fortification, the first two levels (i.e., DLP-S and ILP-s models) will be the same as (4.1)-(4.7). For the third level, consider the connected, complete, and undirected graph $G = (N, E)$. Each edge has the communication cost d_{ij}. We define the set of communication requests as R and the communication request quantity between origin-destination pairs (o_r, d_r) as W_r. Binary decision variable y_{ijs} is 1 if the edge $\{i, j\}$ is part of the spanning tree in scenario $s \in S$ and 0 otherwise. The continuous decision variable x^r_{ijs} represents the portion of communication request $r \in R$ routed through edge $\{i, j\}$ in scenario $s \in S$. The objective function of OCST minimizes the total communication cost for routing all the requests. The mathematical model of the defender's third level for OCST is as follows:

$$\text{(ULP-S)}: \quad A_s(u) = \min_{x,y} \sum_{r \in R} \sum_{\{i,j\} \in E} W_r d_{ij} (x^r_{ijs} + x^r_{jis}) \tag{4.15}$$

$$\text{s.t.} \quad \sum_{j \in N:(j,i) \in A} x^r_{jis} - \sum_{j \in N:(i,j) \in A} x^r_{ijs} = \begin{cases} -1 & \text{if } i = o_r \\ 1 & \text{if } i = d_r \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in N, r \in R \tag{4.16}$$

$$x^r_{ijs} + x^r_{jis} \leq y_{ijs} \qquad \forall \{i,j\} \in E, r \in R \tag{4.17}$$

$$\sum_{\{i,j\} \in E} y_{ijs} = |N| - 1 \tag{4.18}$$

$$y_{ijs} \leq 1 - u_{ijs} \qquad \forall \{i,j\} \in E \tag{4.19}$$

$$y_{ijs} \in \{0,1\} \qquad \forall \{i,j\} \in E \tag{4.20}$$

$$x^r_{ijs} \geq 0 \qquad \forall (i,j) \in A, r \in R \tag{4.21}$$

As mentioned earlier, the feasible solutions of MST and OCST problems are the same as can be seen in constraints (4.9)-(4.14) for the MST problem and constraints (4.16)-(4.21) for the OCST problem. However, the objective functions of MST (4.8) and OCST (4.15) are different. The objective function (4.15) minimizes the communication cost for satisfying the communication requests while the objective function (4.8) minimizes the installation cost.

4.4 Solution Methodology

In this section, we present the BSF algorithm for solving the tri-level stochastic fortification model. We present the details for using the BSF algorithm for the MST fortification problem in Section 4.4.1.1, and the OCST fortification problem in Section 4.4.1.2. We improve the performance of the BSF algorithm with the waiting list acceleration technique in Section 4.4.2.

We implement the BSF algorithm to solve the tri-level fortification problem for two reasons. First, the third-level OCST model is MIP. Therefore, solution methods based on duality cannot be used. BSF algorithm can find the optimal defense strategy for problems with mixed-integer or

nonlinear third-level models. We should mention that MST formulation is recognized as integral, indicating that the y variables inherently take values of either 0 or 1 without requiring the integrality constraints (Magnanti and Wolsey, 1995). Therefore, we can relax the integrality conditions for the MST problem. Second, the stochastic models of MST and OCST can be solved with the same algorithm for the deterministic models with minor changes in the steps of the algorithm.

4.4.1 BSF Algorithm for the Stochastic Model

To solve the tri-level fortification problem, we implement the BSF algorithm presented by Lozano and Smith (2017). Let $\mathcal{Q}$, $\mathscr{U}$, and $\mathscr{Y}$ be the set of all possible feasible solutions for the first-level, second-level, and third-level problems, respectively. The main idea of the BSF algorithm is to sample the third-level solutions ($\hat{\mathscr{Y}} \subseteq \mathscr{Y}$) iteratively. Therefore, the interdictor determines the interdiction strategy based on the samples $\hat{\mathscr{Y}}$ instead of all possible feasible set $\mathscr{Y}$. We add more samples to the sample set during the algorithm until the optimal solution is found. The BSF algorithm consists of the sampling-based inner approach to solve the bi-level interdiction model embodied by an outer cutting plane algorithm determining the optimal defense strategy. Lozano and Smith (2017) prove that BSF is an exact solution methodology terminating finitely with an optimal solution. Our goal is to find the optimal solution z^* of the problem (4.22):

$$z^* = \min_{q \in \mathcal{Q}} \sum_{s \in S} p_s \left(\max_{u_s \in \mathscr{U}(q)} \min_{y \in \mathscr{Y}(u_s)} f(y) \right) \tag{4.22}$$

First, we explain how to find the optimal solution to the bi-level interdiction problem using the BSF algorithm. For any specific defense solution $q \in \mathcal{Q}$, we want to find the optimal solution to the interdiction problem $z^I(q)$ as follows:

$$z^I(q) = \sum_{s \in S} p_s \left(\max_{u_s \in \mathscr{U}(q)} \min_{y \in \mathscr{Y}(u_s)} f(y) \right) \tag{4.23}$$

To solve the bi-level problem (4.23), we solve the restricted problem (4.24) over the samples of

$\hat{\mathscr{Y}} \subseteq \mathscr{Y}$ which provides an upper bound on the value of $z^I(q)$.

$$z^I(q, \hat{\mathscr{Y}}) = \sum_{s \in S} p_s \left(\max_{u_s \in \mathscr{U}(q)} \min_{y \in \hat{\mathscr{Y}}(u_s)} f(y) \right) \tag{4.24}$$

We call equation (4.24) perceived damage because the interdictor assumes that the defender chooses the optimal solution among feasible solutions $\hat{\mathscr{Y}}$. However, the defender can choose any feasible solution in $\mathscr{Y}$.

Proposition 4.1. *For each $q \in \mathcal{Q}$ and the sample of third-level solutions $\hat{\mathscr{Y}} \subseteq \mathscr{Y}$, $z^I(q, \hat{\mathscr{Y}}) \geq z^I(q) \geq z^*$.*

Proof. As $\hat{\mathscr{Y}}(u_s) \subseteq \mathscr{Y}(u_s)$ for $s \in S$, we can conclude that $min_{y \in \hat{\mathscr{Y}}(u_s)} f(y) \geq min_{y \in \mathscr{Y}(u_s)} f(y)$, $\forall s \in S$ which suggested that $\sum_{s \in S} p_s \left(min_{y \in \hat{\mathscr{Y}}(u_s)} f(y) \right) \geq \sum_{s \in S} p_s \left(min_{y \in \mathscr{Y}(u_s)} f(y) \right)$. Therefore, $z^I(q, \hat{\mathscr{Y}}) \geq z^I(q)$. Moreover, $z^I(q)$ provides a feasible solution for the fortification problem for a fixed value of q, implying that $z^I(q) \geq z^*$. □

Let $\hat{u}_s, \forall s \in S$ be the optimal solution of the perceived damage problem (4.24). We define the real damage of the interdiction $\hat{u}_s$ over all possible feasible solutions $\mathscr{Y}$ as:

$$z^R(\hat{u}_s) = \min_{y \in \mathscr{Y}(\hat{u}_s)} f(y) \qquad \forall s \in S \tag{4.25}$$

The expected real damage over all scenarios provides the lower bound on the value of $z^I(q)$. Let $(\hat{u}_s, \hat{y}_s)$ for $s \in S$ be the optimal solution of $z^I(q, \hat{\mathscr{Y}})$. If $z^I(q, \hat{\mathscr{Y}}) = \sum_{s \in S} p_s z^R(\hat{u}_s)$, then $(\hat{u}_s, \hat{y}_s),\ \forall s \in S$ optimizes $z^I(q)$ (see Proposition 2 of Lozano and Smith (2017)).

Now we extend the algorithm to find the optimal defense strategy q. We call an interdiction critical if its real damage is greater than a target upper bound $\overline{z}$ ($\sum_{s \in S} p_s z^R(\hat{u}_s) \geq \overline{z}$). Therefore, the defender should prevent the critical interdiction by fortifying at least one of the interdicted edges in $\hat{u}_s$ for $s \in S$. We add the covering constraint $q^T \hat{u}_s \geq 1$ to the fortification problem for each critical interdiction $\hat{u}_s, \forall s \in S$. The BSF algorithm terminates when the fortification problem becomes infeasible.

Let $\overline{z}$ be the global upper bound, UB_k be the upper bound obtained by solving the perceived damage (4.24) at iteration k, LB_k be the lower bound obtained by computing the expected value of

the real damage (4.25) at iteration k, and $\mathscr{C}$ represent the set of covering constraints. The pseudo-code of the BSF algorithm is presented in Algorithm 5. The algorithm starts with $\mathscr{C} = \varnothing$ and $\overline{z} \leftarrow \infty$. The main while-loop of Algorithm 5 continues until the fortification problem becomes infeasible with the incumbent solution as an optimal solution ($z^* = \overline{z}$). The algorithm has two main steps. First, we select the defense strategy in step 5. Then, we solve the interdiction problem in steps 6-22 for a specific defense strategy $\hat{q}$. The inner while-loop (6-22) continues until we find the critical interdictions ($LB_k \geq \overline{z}$). As the critical interdictions are found, we add the covering constraints to $\mathscr{C}$ and find a new defense strategy in step 5. For any defense strategy $\hat{q}$ that reduces $\overline{z}$, the algorithm finds the optimal interdiction strategy, and the incumbent solution (the best solution found so far) is updated.

Algorithm 5 BSF for Finding the Optimal Fortification Strategy for Stochastic Model

1: Let $\overline{z} \leftarrow \infty$ be the global upper bound and $\mathscr{C} = \varnothing$ represent the set of covering constraints
2: Let $\hat{\mathscr{Y}}^1 \subseteq \mathscr{Y}$ be a sample of the third-level solution space, and $f(y)$ be the third-level objective function for each solution $y \in \hat{\mathscr{Y}}^1$
3: Iteration $k \leftarrow 0$
4: **while** $\mathcal{Q}(\mathscr{C}) \neq \varnothing$ **do**
5: Select any $\hat{q} \in \mathcal{Q}(\mathscr{C})$ and set $UB_k \leftarrow \infty$ and $LB_k \leftarrow -\infty$
6: **while** $LB_k < \overline{z}$ **do**
7: $k \leftarrow k + 1$
8: Solve $UB_k = \sum_{s \in S} p_s z_s^I(\hat{q}, \hat{\mathscr{Y}}^k) = \sum_{s \in S} p_s [\max_{u \in \mathscr{U}_s(\hat{q})} \min_{y \in \hat{\mathscr{Y}}^k(u)} f(y)]$ and find an optimal solution $(\hat{u}_s, \hat{y}_s)$
9: Solve $LB_k = \sum_{s \in S} p_s z^R(\hat{u}_s) = \sum_{s \in S} p_s [\min_{y \in \mathscr{Y}(\hat{u}_s)} f(y)]$ and find an optimal solution $\hat{y}_s^*$
10: Set $\hat{\mathscr{Y}}^{k+1} = \hat{\mathscr{Y}}^k \cup \{\hat{y}_s^* \quad \forall s \in S\}$
11: **if** $UB_k < \overline{z}$ **then**
12: $\overline{z} \leftarrow UB_k$
13: Remove all solutions with objective value exceeding UB_k from $\hat{\mathscr{Y}}^{k+1}$
14: Select $\hat{\mathscr{Y}}' \subseteq \mathscr{Y}$ as a sampling of the third-level solution space
15: Add to $\hat{\mathscr{Y}}^{k+1}$ all new solutions in $\hat{\mathscr{Y}}' \cap \mathscr{Y}_{UB_k}$
16: **else**$LB_k \geq \overline{z}$
17: We find the critical interdictions. Add the covering constraint $q^T \hat{u}_s \geq 1$ for each scenario $s \in S$ to $\mathscr{C}$
18: **end if**
19: **if** $LB_k = UB_k = \overline{z}$ **then**
20: Update the incumbent solution $(\overline{q}, \overline{u}_s, \overline{y}_s) \leftarrow (\hat{q}, \hat{u}_s, \hat{y}_s)$ for each scenario $s \in S$
21: **end if**
22: **end while**
23: **end while**
24: **return:** $(\overline{q}, \overline{u}_s, \overline{y}_s), \forall s \in S$

In order to apply the BSF algorithm for solving the MST and OCST fortification problems, it is necessary to introduce the sampling strategy, an algorithm to solve the perceived damage over

the samples, and an algorithm tailored to solve the real damage model involving the MST or OCST problems. In what follows, we present the details for implementing the BSF algorithm to solve the MST fortification problem and OCST fortification problem.

4.4.1.1 Implementation of the BSF Algorithm for the MST Fortification Problem

Sampling strategy: If the samples are not diverse and they share many common edges, the interdictor can interdict the MSTs by interdicting a few critical edges which results in a poor upper bound for the BSF algorithm. Moreover, if the sample size is large, we may obtain a tighter upper bound but this increases the complexity of solving the perceived damage. Therefore, we are interested in a sampling strategy that creates varied samples of moderate size. It is important to note that ensuring $\hat{\mathscr{Y}} \subseteq \mathscr{Y}$ guarantees the termination of the BSF algorithm. Therefore, we do not need to find the optimal solution of the samples for the BSF algorithm to converge. To create the samples, we iteratively solve the MST problem with a subset of interdicted edges. Let the first sample be the solution of MST in normal conditions (i.e., there is no interdiction). We randomly choose the number of interdictions (between 1 and $|N| - 2$). The interdiction decisions of sample l depend on the y solution (edges that are part of the spanning tree) of sample $l - 1$. For each sample l, we interdict the edges that are part of the y solution of sample $l - 1$ with the lowest installation cost c_{ij}. Then, we solve MST for sample l considering that the interdicted edges cannot be part of the MST. To have diverse samples, we restrict the number of times an edge can be included in the MST solution. To find the MST for each sample, we implement Prim's algorithm (Prim, 1957) by allocating high cost for the interdicted edges or edges that are unable to be part of the MST due to reaching the maximum allowed occurrences within the MST.

Solving the perceived damage problem: To solve the perceived damage (4.24), we consider Ω^l as the set of edges forming the lth spanning tree of graph G in sample $\hat{\mathscr{Y}} \subseteq \mathscr{Y}$. We define $c(\Omega^l)$ as the cost of the lth MST in sample $\hat{\mathscr{Y}} \subseteq \mathscr{Y}$. We solve the following MIP model (4.26)-(4.31) for $s \in S$ to find the interdiction strategy.

$$\max \quad z \tag{4.26}$$

$$\text{s.t.} \quad z \leq c(\Omega^l) + \sum_{\{i,j\} \in \Omega^l} M u_{ijs} \qquad \forall \Omega^l \in \hat{\mathscr{Y}} \tag{4.27}$$

$$\sum_{\{i,j\} \in E} u_{ijs} \leq U_s \tag{4.28}$$

$$u_{ijs} \leq 1 - \hat{q}_{ij} \qquad \forall \{i,j\} \in E \tag{4.29}$$

$$u_{ijs} \in \{0,1\} \qquad \forall \{i,j\} \in E \tag{4.30}$$

$$z \geq 0 \tag{4.31}$$

The objective function (4.26) maximizes z where constraint (4.27) ensures that the value of z does not exceed the minimum MST cost considering the effects of interdictions. We use a large number M to penalize the interdicted edges ($M = max\{c_{ij}\} + 1$). Constraint (4.29) defines the feasible interdictions based on the defense strategy $\hat{q}$.

Lemma 4.1. *$M = max\{c_{ij}\} + 1$ is a valid value for constraint (4.27).*

Proof. As the interdictor interdicts an edge $\{i,j\}$ that is the part of the spanning tree, the defender replaces that edge with another edge having the cost as much as $max\{c_{ij}\}$. Therefore the value of z increases by $max\{c_{ij}\}$. □

We can tighten constraint (4.27) by defining $M_{ij} = max\{c_{ij}\} - c_{ij}$. We next show that M_{ij} is a valid value for constraint (4.27).

Lemma 4.2. *$M_{ij} = max\{c_{ij}\} - c_{ij}$ is a valid value for constraint (4.27).*

Proof. As the interdictor interdicts an edge $\{i,j\}$ that is the part of the spanning tree, the defender replaces that edge with another edge having the cost as much as $max\{c_{ij}\}$. However, the defender does not use edge $\{i,j\}$ anymore. Therefore the value of z increases by $M_{ij} = max\{c_{ij}\} - c_{ij}$. □

Solving the real damage problem: To solve the real damage (4.25), we use Prim's algorithm, known for its capability to find the optimal solution for the MST problem (Cormen et al., 2022).

4.4.1.2 Implementation of the BSF Algorithm for the OCST Fortification Problem

Sampling strategy: We use the same strategy for finding the samples as in Section 4.4.1.1. To have a different solution in each sample, we interdict the edges that are part of the y solution (edges that are part of the OCST solution) of the sample $l-1$ with the lowest communication cost d_{ij}. To find the solution to the samples with respect to the interdicted edges, we use the Ahuja-Murty local search algorithm (Ahuja and Murty, 1987). This method starts by constructing a tree and improving the solution by 1-edge exchange neighborhood choosing the best neighbor until a local minimum is found.

Solving the perceived damage problem: We solve a similar model to (4.26)-(4.31) to find the interdiction strategy based on the samples where $c(\Omega^l)$ is the cost of lth OCST and $M = \sum_{r\in R}\sum_{\{i,j\}\in E} W_r d_{ij}$.

Solving the real damage problem: To find the optimal objective value of the real damage (4.25), we use the state-of-the-art branch-and-Benders-cut algorithm enhanced with warm-start, cutset inequalities, Pareto-optimal cuts, in-tree heuristics, and cut filtering presented by Zetina et al. (2019). We modify their presented branch-and-Benders-cut algorithm so that the interdicted edges cannot be part of the OCST solution.

4.4.2 Waiting List Acceleration Technique

Finding the optimal solution to the perceived damage model (4.26)-(4.31) can be hard as the number of samples increases throughout the algorithm. To reduce the number of perceived damage models to be solved to optimality, Lozano and Smith (2017) present the acceleration technique that we name it waiting list acceleration technique. The idea of the waiting list acceleration technique is to stop searching $\hat{q} \in \mathcal{Q}$ if the relative enhancement to $\overline{z}$ is small. Therefore, we add the possible covering constraints for each $s \in S$ to the fortification problem, assuming that $\hat{u}_s, \forall s \in S$ is critical for $\hat{q}$. We save $\hat{q}$ in a waiting list for subsequent re-evaluation within the algorithm's execution. During this phase, we either validate that $\hat{u}_s, \forall s \in S$ were critical and remove $\hat{q}$ from the waiting list, or proceed to investigate $\hat{q}$ further if the attacks are considered non-critical.

The pseudo-code of the BSF algorithm with the waiting list acceleration technique is presented

in Algorithm 6. Let $\mathscr{C}^{\phi}$ represent the set of possible covering constraints integrated into the fortification problem. For each possible covering constraint $\mathscr{C}^{\phi}$, we keep the defense strategy $\hat{q}$ that should be revisited, the expected real damage $\sum_{s\in S} p_s z^R(\hat{u}_s)$ that is assumed to be critical, and the covering constraints $\hat{\phi}$ in waiting list $\mathscr{L}$. The algorithm continues until the fortification problem is feasible. For each feasible fortification strategy $\hat{q}$, the algorithm solves the interdiction problem $z^I(\hat{q})$ as in Algorithm 5 in steps 6-20. If the interdiction strategies are not critical, we calculate the $(\overline{z} - LB_k)/\,\overline{z}$ to obtain the percentage reduction to $\overline{z}$ if we continue to solve $z^I(\hat{q})$. If the reduction is less than ϵ, we add $(\hat{q}, \sum_{s\in S} p_s z^R(\hat{u}_s), \hat{\phi})$ to the waiting list $\mathscr{L}$ in step 22, and add the covering constraints $q^T\hat{u}_s \geq 1, \forall s \in S$ to $\mathscr{C}^{\phi}$ in step 23. At this point, we return to step 5. When the fortification problem becomes infeasible (i.e., $\mathcal{Q}(\mathscr{C} \cup \mathscr{C}^{\phi}) = \varnothing$), if the waiting list is not empty ($\mathscr{L} \neq \varnothing$), we should reassess the items stored in the waiting list in steps 28-37. For item i in the waiting list $\mathscr{L}$, if $\sum_{s\in S} p_s z^R(\hat{u}_s^i) > \overline{z}$, it means that these interdictions are critical; therefore, we move the corresponding covering constraints from $\mathscr{C}^{\phi}$ to $\mathscr{C}$, and remove the item from the waiting list $\mathscr{L}$. For the remaining items in steps 33-37, we continue the exploration for $\hat{q}^i$ that is still in $\mathcal{Q}(\mathscr{C})$ with $\epsilon = 0$. In step 38, we clear the waiting list, remove the remaining covering constraints $\mathscr{C}^{\phi}$, and return to the main while-loop (step 4).

4.5 Computational Experiments

To test the performance of the proposed BSF algorithm, we use the benchmark instances presented in Contreras et al. (2010) and Zetina et al. (2019). We also analyze the effects of the number of fortifications and interdictions on the performance of the algorithm. We demonstrate the benefits of using the stochastic fortification model instead of the deterministic fortification model when the number of interdictions is uncertain. We implement the algorithms in the C programming language and execute them on a workstation equipped with a 3.10 GHz Intel Xeon E5 2687W V3 processor within a Linux environment. We utilize the Callable Library for CPLEX version 22.1.0. for our computational experiments.

Algorithm 6 BSF Algorithm with Waiting List Acceleration Technique

Let $\overline{z} \leftarrow \infty$ be the global upper bound and $\mathscr{C} = \mathscr{C}^{\phi} = \varnothing$ represent the set of covering constraints and $\mathscr{L} = \varnothing$ indicating the waiting list
Let $\hat{\mathscr{Y}}^1 \subseteq \mathscr{Y}$ be a sample of the third-level solution space, and $f(y)$ be the third-level objective function for each solution $y \in \hat{\mathscr{Y}}^1$
Iteration $k \leftarrow 0$
while $\mathcal{Q}(\mathscr{C} \cup \mathscr{C}^{\phi}) \neq \varnothing$ **do**
    Select any $\hat{q} \in \mathcal{Q}(\mathscr{C} \cup \mathscr{C}^{\phi})$ and set $UB_k \leftarrow \infty$ and $LB_k \leftarrow -\infty$
    while $LB_k < \overline{z}$ **do**
        $k \leftarrow k + 1$
        Solve $UB_k = \sum_{s \in S} p_s z_s^I(\hat{q}, \hat{\mathscr{Y}}^k) = \sum_{s \in S} p_s [\max_{u \in \mathscr{U}_s(\hat{q})} \min_{y \in \hat{\mathscr{Y}}^k(u)} f(y)]$ and find an optimal solution $(\hat{u}_s, \hat{y}_s)$
        Solve $LB_k = \sum_{s \in S} p_s z^R(\hat{u}_s) = \sum_{s \in S} p_s [\min_{y \in \mathscr{Y}(\hat{u}_s)} f(y)]$ and find an optimal solution $\hat{y}_s^*$
        Set $\hat{\mathscr{Y}}^{k+1} = \hat{\mathscr{Y}}^k \cup \{\hat{y}_s^* \quad \forall s \in S\}$
        if $UB_k < \overline{z}$ **then**
            $\overline{z} \leftarrow UB_k$
            Remove all solutions with objective value exceeding UB_k from $\hat{\mathscr{Y}}^{k+1}$
            Select $\hat{\mathscr{Y}}' \subseteq \mathscr{Y}$ as a sampling of the third-level solution space
            Add to $\hat{\mathscr{Y}}^{k+1}$ all new solutions in $\hat{\mathscr{Y}}' \cap \mathscr{Y}_{UB_k}$
        else$LB_k \geq \overline{z}$
            We find the critical interdictions. Add the covering constraint $q^T \hat{u}_s \geq 1$ for each scenario $s \in S$ to $\mathscr{C}$
        end if
        if $LB_k = UB_k = \overline{z}$ **then**
            Update the incumbent solution $(\overline{q}, \overline{u}_s, \overline{y}_s) \leftarrow (\hat{q}, \hat{u}_s, \hat{y}_s)$ for each scenario $s \in S$
        else$(\overline{z} - LB_k)/\overline{z} \leq \epsilon$ and $LB_k < \overline{z}$
            Add $(\hat{q}, \sum_{s \in S} p_s z^R(\hat{u}_s), \hat{\phi})$ to the waiting list $\mathscr{L}$
            Add covering constraints $q^T \hat{u}_s \geq 1, \forall s \in S$ to $\mathscr{C}^{\phi}$ and return to step 5
        end if
    end while
end while
if $\mathscr{C}^{\phi} \neq \varnothing$ **then**
    for $i \in \mathscr{L}$ **do**
        if $\sum_{s \in S} p_s z^R(\hat{u}_s^i) > \overline{z}$ **then**
            Add covering constraints $\hat{\phi}^i$ to $\mathscr{C}$, eliminate them from $\mathscr{C}^{\phi}$, and eliminate $(\hat{q}^i, \sum_{s \in S} p_s z^R(\hat{u}_s^i), \hat{\phi}^i)$ from waiting list $\mathscr{L}$
        end if
    end for
    for $i \in \mathscr{L}$ **do**
        if $q^i \in \mathscr{Q}$ **then**
            Continue solving $z^I(q^i)$ with $\epsilon = 0$
        end if
    end for
    $\mathscr{C}^{\phi} \leftarrow \varnothing, \mathscr{L} \leftarrow \varnothing$, and return to step 4
end if
return: $(\overline{q}, \overline{u}_s, \overline{y}_s), \forall s \in S$

4.5.1 Test Instances

To evaluate the effectiveness of the BSF algorithm on the MST fortification problem, we execute the algorithm for instances with 10 to 100 nodes. Instances with an identical number of nodes ($|N|$) have different installation costs (shown by a, b, c, and d in the instances). For the OCST fortification problem, we run the algorithm for instances with 10 to 50 nodes. Instances with an identical number of nodes ($|N|$) have different communication costs and communication request quantities. Each instance has $\frac{|N|\times(|N|-1)}{2}$ edges as we consider the complete graphs. We set the number of interdictions to 2, 3, and 4. The number of fortifications is set to 3, 4, and 5. The time limit is 24 hours (86,400 seconds) with 1 hour (3,600 seconds) allocated to the sampling phase. The maximum number of samples is set to 100.

4.5.2 Results of the MST Fortification Problem

4.5.2.1 Deterministic MST Fortification Problem

First, to show the effectiveness of the waiting list acceleration technique on the performance of the BSF algorithm, we execute the BSF algorithm with and without the waiting list acceleration technique ($M = max\{c_{ij}\} + 1$). The value of ϵ for the waiting list acceleration technique is set to 0.25. We compare the performance of the algorithm across instances comprising 10 to 100 nodes, with 2 interdictions ($U = 2$) and 3,4, and 5 fortifications ($Q = \{3, 4, 5\}$). In Table 4.3, column "BSF" presents the CPU time (s) for the BSF algorithm without acceleration technique, column "BSF+WL" presents the results for the BSF algorithm with waiting list acceleration technique, and column "Reduction %" presents the percentage reduction in CPU time by using the BSF+WL algorithm. In Table 4.3, we call instances with 10 to 30 nodes as small-sized instances, 40 to 60 nodes as moderate-sized instances, and 70 to 100 nodes as large-sized instances. The CPU time for small-sized instances varies from less than 1 second to a maximum of 79 seconds. We observe that the BSF+WL improves the CPU time in small-sized instances by an average of 7.6% across all fortification levels where the average time of BSF is 23.6 seconds and the average time of BSF+WL is 21.8 seconds. For moderate-sized instances, the CPU time varies from 282 seconds to 3,254 seconds. In moderate-sized instances, the BSF+WL reduces the CPU time by an average of 6.8%

across all fortification levels. For large-sized instances, we observe that the BSF algorithm is not able to solve instances with 100 nodes and 4,950 edges in the time limit, but the BSF+WL solves these instances to optimality with a maximum of 40,706 seconds. The BSF+WL reduces the CPU time by an average of 55.5% across all fortification levels compared with the BSF algorithm for large-sized instances.

As the BSF algorithm with the waiting list acceleration technique outperforms the BSF algorithm, we use the BSF algorithm with acceleration technique for the rest of the computational experiments. Table 4.4 presents the CPU time (s) for the MST fortification problem with $U = \{3, 4\}$ interdictions and $Q = \{3, 4, 5\}$ fortifications. We observe that the BSF+WL algorithm is sensitive to the number of interdictions. As U increases, CPU time increases significantly. Increasing the number of fortifications increases CPU time slightly. The reason is that the number of interdictions affects the difficulty of the perceived damage (4.24) but the number of fortifications affects the fortification problem solved when the critical interdiction has been found. As the perceived damage (4.24) is solved more frequently in the algorithm, the increase in the number of interdictions has a higher effect on the performance of the algorithm than the number of fortifications. When $U = 2$, the algorithm finds the optimal defense strategy for instances with 100 nodes and 4,950 edges within the time limit. As we increase the number of interdictions to 3, the algorithm solves the instances with up to 60 nodes and 1,770 edges to optimality. When $U = 4$, the algorithm finds the optimal solution for instances with up to 40 nodes and 780 edges.

4.5.2.2 Stochastic MST Fortification Problem

The defender does not know the exact number of interdictions; therefore, we consider the uncertainty in the number of interdictions as a set of scenarios with equal probabilities. First, we show the effects of the number of scenarios on the CPU time in Table 4.5 for the MST fortification problem with two scenarios and Table 4.6 for the MST fortification problem with three and four scenarios. The number of interdictions in each scenario $s \in S$ is set to $U_s = s$ e.g., when $|S| = 3$, we have $U_1 = 1, U_2 = 2, U_3 = 3$. All the scenarios have the same probability. We observe that CPU time increases as we consider the stochastic number of interdictions compared with the results of Table 4.3 and Table 4.4. The reason is that the perceived damage (4.24) and the real damage (4.25) should

Table 4.3: Comparing the Effect of Acceleration Technique on CPU Time (s) for the MST Fortification Problem

Instance	$U = 2, Q = 3$			$U = 2, Q = 4$			$U = 2, Q = 5$		
	BSF	BSF+WL	Reduction %	BSF	BSF+WL	Reduction %	BSF	BSF+WL	Reduction %
10a	0.5	0.4	20.3	0.5	0.4	11.3	0.5	0.4	17.8
10b	0.3	0.2	22.5	0.3	0.3	17.8	0.4	0.3	22.9
10c	0.4	0.2	37.9	0.4	0.3	33.1	0.4	0.3	32.6
10d	0.4	0.3	21.9	0.4	0.3	15.5	0.4	0.3	21.0
20a	9.6	7.4	22.7	9.8	7.8	20.3	10.1	8.3	17.9
20b	6.0	5.4	9.5	6.3	5.9	5.9	6.5	5.9	9.4
20c	9.9	7.9	20.0	10.1	8.0	20.9	10.5	9.2	12.5
20d	9.1	7.0	22.6	9.2	7.2	22.6	9.5	8.0	15.9
30a	57	42	26.2	60	50	17.0	61	51	17.1
30b	60	47	20.3	62	51	17.6	63	56	12.1
30c	56	50	12.2	59	61	-2.5	61	67	-9.5
30d	63	62	2.5	68	79	-15.9	69	78	-12.4
Average	22.7	19.1	19.9	23.9	22.5	13.6	24.4	23.6	13.1
40a	289	287	0.6	302	308	-2.1	305	309	-1.3
40b	343	307	10.8	344	316	8.1	342	319	6.7
40c	313	282	9.9	317	285	10.0	325	289	11.0
40d	310	288	7.0	325	288	11.5	329	296	10.0
50a	1,139	1,147	-0.7	1,162	1,151	0.9	1,170	1,158	1.0
50b	951	888	6.5	969	889	8.3	987	892	9.7
50c	1,052	950	9.7	1,065	953	10.5	1,078	956	11.3
50d	1,140	1,018	10.7	1,140	1,017	10.8	1,147	1,024	10.7
60a	2,697	2,753	-2.1	2,837	2,766	2.5	2,833	2,781	1.8
60b	2,970	2,683	9.7	3,085	2,747	10.9	3,096	2,745	11.3
60c	3,002	2,654	11.6	3,002	2,695	10.2	3,011	2,762	8.3
60d	3,231	3,001	7.1	3,237	3,046	5.9	3,250	3,254	-0.1
Average	1,453	1,355	6.7	1,482	1,372	7.3	1,489	1,399	6.7
70a	6,857	5,780	15.7	6,913	5,826	15.7	7,214	5,851	18.9
70b	6,109	5,860	4.1	6,241	6,109	2.1	6,456	6,135	5.0
70c	6,808	6,582	3.3	6,876	6,522	5.1	7,079	6,516	8.0
70d	6,679	6,536	2.1	6,722	6,546	2.6	7,010	6,580	6.1
80a	18,672	13,802	26.1	18,668	13,886	25.6	19,166	13,733	28.3
80b	20,368	13,302	34.7	20,375	13,299	34.7	20,788	13,360	35.7
80c	21,888	15,888	27.4	21,906	16,017	26.9	22,633	16,003	29.3
80d	24,325	13,180	45.8	24,377	13,299	45.4	24,906	13,252	46.8
90a	49,910	16,532	66.9	49,910	16,536	66.9	51,308	16,650	67.5
90b	54,351	20,335	62.6	54,710	20,014	63.4	56,862	20,419	64.1
90c	57,278	19,090	66.7	57,402	19,061	66.8	58,000	19,221	66.9
90d	51,942	17,420	66.5	52,171	17,580	66.3	52,105	17,776	65.9
100a	86,400*	33,507	61.2	86,400*	34,298	60.3	86,400*	34,774	59.8
100b	86,400*	36,097	58.2	86,400*	36,786	57.4	86,400*	37,028	57.1
100c	86,400*	33,516	61.2	86,400*	34,208	60.4	86,400*	34,440	60.1
100d	86,400*	39,550	54.2	86,400*	40,536	53.1	86,400*	40,706	52.9
Average	41,924	18,561	41.0	41,992	18,783	40.8	42,445	18,903	42.0

* Time limit reached

Table 4.4: CPU Time (s) for Solving MST Fortification Problem with BSF+WL

Instance	$U = 3$			$U = 4$		
	$Q = 3$	$Q = 4$	$Q = 5$	$Q = 3$	$Q = 4$	$Q = 5$
10a	3.8	3.8	3.9	40	42	52
10b	3.4	3.5	3.7	47	49	45
10c	1.7	1.8	2.8	50	47	30
10d	2.1	2.3	3.2	49	27	24
20a	520	521	546	38,115	38,483	38,519
20b	492	494	494	30,184	30,282	30,557
20c	679	680	682	32,254	32,397	32,610
20d	704	728	729	25,577	25,583	25,773
30a	13,532	13,688	13,805	73,481	74,011	74,909
30b	14,865	14,941	14,979	76,819	77,285	77,793
30c	13,948	13,964	13,992	74,558	75,037	75,693
30d	14,302	14,476	14,558	76,704	77,154	77,986
Average	4,921	4,959	4,983	35,657	35,866	36,166
40a	29,257	29,812	30,589	65,289	67,210	68,040
40b	31,837	32,194	33,048	86,400*	86,400*	86,400*
40c	28,080	28,755	29,889	62,384	62,998	63,818
40d	31,286	32,002	33,104	86,400*	86,400*	86,400*
50a	75,867	76,812	77,209	86,400*	86,400*	86,400*
50b	66,358	67,402	67,975	86,400*	86,400*	86,400*
50c	76,341	76,941	77,503	86,400*	86,400*	86,400*
50d	73,800	74,226	74,857	86,400*	86,400*	86,400*
60a	84,414	85,498	86,302	86,400*	86,400*	86,400*
60b	85,409	86,100	86,400*	86,400*	86,400*	86,400*
60c	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*
60d	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*
Average	62,954	63,545	64,140	82,639	82,851	82,988

* Time limit reached

be solved for each scenario separately which increases the CPU time.

Table 4.5: CPU Time (s) for Solving the Stochastic MST Fortification Problem with Two Scenarios

Instance	$\|S\| = 2$			Instance	$\|S\| = 2$		
	$Q = 3$	$Q = 4$	$Q = 5$		$Q = 3$	$Q = 4$	$Q = 5$
10a	0.6	0.6	0.7	60a	3,492	3,623	3,832
10b	0.5	0.5	0.5	60b	3,061	3,210	3,357
10c	0.4	0.5	0.5	60c	3,333	3,409	3,513
10d	0.5	0.9	1.0	60d	3,742	3,816	3,968
20a	13	13	15	70a	6,798	6,869	6,917
20b	9	10	9	70b	6,836	7,016	7,318
20c	14	14	15	70c	10,757	11,090	11,132
20d	13	14	13	70d	9,886	9,990	10,846
30a	65	70	72	80a	18,671	19,099	19,323
30b	75	79	82	80b	20,780	20,909	21,228
30c	72	75	81	80c	24,623	25,074	25,402
30d	84	90	91	80d	22,643	22,734	23,431
40a	355	356	358	90a	17,217	17,328	17,562
40b	355	358	370	90b	23,011	23,226	23,263
40c	386	397	407	90c	19,566	20,028	20,967
40d	306	314	338	90d	20,055	20,088	21,072
50a	1,200	1,258	1,321	100a	40,965	41,004	41,246
50b	999	1,023	1,037	100b	37,528	38,490	39,287
50c	1,008	1,042	1,098	100c	42,725	43,307	44,338
50d	1,407	1,502	1,593	100d	47,666	48,919	50,663

We observe that as the number of scenarios increases, the CPU time increases. The reason is that the algorithm is sensitive to the number of interdictions. As the number of scenarios increases, the perceived damage and real damage problems should be solved for a higher number of interdictions, increasing CPU time. Based on the results of Table 4.6, we find the optimal solution for the MST fortification problem with three and four scenarios for instances with up to 40 nodes.

Next, we demonstrate the benefits of using the stochastic model over the deterministic model in cases where the number of interdictions is uncertain. For this purpose, we randomly choose instance "10a" with $Q = 4$ and we have four scenarios. First, we find the optimal fortification strategy for the deterministic number of interdictions. Then we fix the fortification strategy and compute the MST cost under this deterministic fortification strategy considering that the actual number of interdictions happens. We report the results in Table 4.7. The first column shows the U value used to derive the optimal fortification strategy, and the four subsequent columns represent the MST cost for each fortification strategy corresponding to the actual U mentioned in the column headers. The last column presents the optimal fortified edges of each U. Based on the results of Table 4.7, we observe that the fortification strategy obtained from the stochastic model has a good performance for all of the values of U. The minimum MST cost for each U is obtained when we

Table 4.6: CPU Time (s) for Solving the Stochastic MST Fortification Problem with Three and Four Scenarios

Instance	$\|S\| = 3$			$\|S\| = 4$		
	$Q = 3$	$Q = 4$	$Q = 5$	$Q = 3$	$Q = 4$	$Q = 5$
10a	5.6	5.8	6.5	151	170	174
10b	8.6	8.8	9.5	294	327	329
10c	5.4	5.7	5.9	184	208	213
10d	8.3	9.1	9.2	258	298	302
20a	1,042	1,073	1,102	49,675	51,935	53,470
20b	550	563	587	35,643	35,741	36,899
20c	975	1,131	1,193	50,081	52,206	62,182
20d	954	1,086	1,095	49,908	51,117	51,716
30a	14,967	15,568	16,000	82,736	83,820	84,663
30b	15,856	15,886	16,982	86,400*	86,400*	86,400*
30c	15,572	15,777	17,003	84,039	85,705	86,400*
30d	15,751	16,139	17,729	83,725	84,607	85,594
40a	56,610	57,593	58,084	79,384	80,606	82,067
40b	74,499	75,003	75,942	86,400*	86,400*	86,400*
40c	60,153	61,795	62,404	84,254	85,907	86,309
40d	66,803	67,555	68,449	86,400*	86,400*	86,400*
50a	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*
50b	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*
50c	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*
50d	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*
60a	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*
60b	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*
60c	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*
60d	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*

* Time limit reached

consider that specific U value to find the fortification strategy. We notice that the stochastic model has the minimum MST cost for $U = 1$, $U = 2$, and $U = 4$. When $U = 3$, this fortification strategy increases the MST cost by 2 units. The deterministic fortification considering $U = 2$ performs the same as the stochastic model as they fortify the same edges. However, the deterministic fortification of $U = 1$, $U = 3$, and $U = 4$ do not show good performance when confronted with different U levels.

Table 4.7: Comparison of the MST Cost for Deterministic and Stochastic Fortification Strategies

	Actual no. of Interdictions				Fortified Edges
	$U = 1$	$U = 2$	$U = 3$	$U = 4$	
$U = 1$	241	257	274	282	$\{0,5\} - \{0,8\} - \{4,7\} - \{8,9\}$
$U = 2$	241	250	266	280	$\{0,8\} - \{2,4\} - \{4,7\} - \{8,9\}$
$U = 3$	250	258	264	281	$\{0,5\} - \{2,4\} - \{4,7\} - \{8,9\}$
$U = 4$	244	257	270	280	$\{0,8\} - \{2,3\} - \{6,7\} - \{8,9\}$
Stochastic	241	250	266	280	$\{0,8\} - \{2,4\} - \{4,7\} - \{8,9\}$

4.5.2.3 Effect of Big-M Value on the Performance of BSF Algorithm with Acceleration Technique

We solve the BSF algorithm with acceleration technique with $M_{ij} = max\{c_{ij}\} - c_{ij}$. We present the results of $M_{ij} = max\{c_{ij}\} - c_{ij}$ in Table 4.8 for deterministic model with $U = 2, Q = 3$ and stochastic model with $|S| = 2, Q = 3$. The results of $M = max\{c_{ij}\} + 1$ are presented in Table 4.3 and Table 4.5. We compare the average of CPU time (s) for $M_{ij} = max\{c_{ij}\} - c_{ij}$ and $M = max\{c_{ij}\} + 1$. We observe that the performance of BSF+WL algorithm is better with new big-M value for instances with up to 80 nodes in deterministic model. However, instances with 90 and 100 nodes have better performance with $M = max\{c_{ij}\} + 1$. In stochastic model, instances with up to 50 nodes have better performance with $M = max\{c_{ij}\} + 1$, and instances with 60, 70, 80, 90, and 100 nodes have better performance with $M_{ij} = max\{c_{ij}\} - c_{ij}$.

Table 4.8: Comparing the Effect of Big-M Value on CPU Time (s) for the MST Fortification Problem

Instance	$U = 2, Q = 3$	$\|S\| = 2, Q = 3$	Instance	$U = 2, Q = 3$	$\|S\| = 2, Q = 3$
10a	0.5	0.6	60a	2,036	2,591
10b	0.3	0.5	60b	2,025	2,693
10c	0.3	0.5	60c	1,666	3,555
10d	0.3	0.6	60d	2,042	2,714
20a	9.2	14	70a	4,494	5,998
20b	6.5	9	70b	3,479	5,081
20c	9.3	18	70c	3,440	4,951
20d	8.2	13	70d	3,850	5,141
30a	50	76	80a	9,316	10,342
30b	53	89	80b	9,443	10,394
30c	50	82	80c	10,242	11,396
30d	60	91	80d	7,855	9,027
40a	223	371	90a	19,331	19,228
40b	221	421	90b	24,798	23,063
40c	198	328	90c	28,679	19,532
40d	204	325	90d	23,719	20,210
50a	818	1,519	100a	55,084	33,833
50b	627	1,239	100b	56,023	35,841
50c	589	1,212	100c	59,053	36,451
50d	702	1,309	100d	60,708	36,174
Avg. with $M_{ij} = max\{c_{ij}\} - c_{ij}$	192	356		19,364	14,911
Avg. with $M = max\{c_{ij}\} + 1$	270	318		15,403	19,168

4.5.3 Results of the OCST Fortification Problem

4.5.3.1 Deterministic OCST Fortification Problem

We compare the performance of the BSF+WL algorithm for the OCST fortification problem across instances comprising 10 to 50 nodes, with $U = \{2, 3, 4\}$ and $Q = \{3, 4, 5\}$. Based on the

results of Table 4.9, we observe that: (i) the OCST fortification problem is harder than the MST fortification problem as we reach the time limit for the OCST instances with 50 nodes when $U = 2$ while we find the optimal solution for MST fortification problem with 100 nodes in Table 4.3, (ii) both the number of fortifications and the number of interdictions increase the CPU time. However, the number of interdictions has a higher impact on the CPU time. The reason is that the number of interdictions affects the difficulty of perceived damage and the third-level OCST problems, and (iii) when $U = 2$, the BSF+WL algorithm solves instances with 50 nodes, 1,225 edges, and 635 communication requests. When $U = 3$, we can solve instances with up to 40 nodes, 780 edges, and 420 communication requests within the time limit. For $U = 4$, instances with up to 30 nodes, 435 edges, and 250 communication requests are solved within the time limit.

Table 4.9: CPU Time (s) for Solving OCST Fortification Problem with BSF+WL

Instance	$U = 2$			$U = 3$			$U = 4$		
	$Q = 3$	$Q = 4$	$Q = 5$	$Q = 3$	$Q = 4$	$Q = 5$	$Q = 3$	$Q = 4$	$Q = 5$
10a	7.3	8.7	8.4	22	23	29	46	47	61
10b	1.0	1.1	1.2	3.6	4.2	4.5	16	16	19
10c	0.6	0.8	0.7	3.1	3.0	3.4	15	15	15
10d	2.3	2.4	2.4	9	10	12	30	33	37
20a	1,384	1,399	1,712	9,614	10,190	10,968	47,555	58,696	61,756
20b	192	195	199	1,492	1,690	1,720	18,058	18,477	20,164
20c	1,462	1,480	1,535	9,763	9,874	10,527	55,754	71,750	75,391
20d	67	74	98	583	606	655	6,636	8,184	8,587
30a	12,580	13,903	14,466	39,513	42,034	44,231	80,692	81,732	82,593
30b	38,093	41,144	49,468	76,805	79,528	82,042	86,400*	86,400*	86,400*
30c	20,592	22,736	23,810	45,733	47,023	48,230	84,201	85,309	86,215
30d	22,544	24,048	24,915	49,512	52,815	53,556	86,400*	86,400*	86,400*
40a	36,790	37,704	38,448	78,308	79,440	81,301	86,400*	86,400*	86,400*
40b	39,693	41,178	42,850	79,644	80,249	81,474	86,400*	86,400*	86,400*
40c	42,004	43,552	44,328	82,514	83,412	84,529	86,400*	86,400*	86,400*
40d	67,747	68,307	69,666	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*
50a	80,816	82,514	84,004	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*
50b	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*
50c	70,525	72,303	73,188	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*
50d	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*

* Time limit reached

4.5.3.2 Stochastic OCST Fortification Problem

We present the CPU time (s) for the stochastic OCST fortification problem for $|S| = 2$, $|S| = 3$, and $|S| = 4$ in Table 4.10 for instances with 10, 20, and 30 nodes. As expected, the number of scenarios affects the CPU time significantly. The algorithm finds the optimal fortification strategy for instances with up to 30 nodes, 435 edges, and 250 communication requests for $|S| = 2$ and $|S| = 3$. As we increase the number of scenarios to $|S| = 4$, the algorithm finds the optimal

solution for instances with up to 20 nodes, 190 edges, and 102 communication requests.

Table 4.10: CPU Time (s) for Solving the Stochastic OCST Fortification Problem

Instance	$\|S\| = 2$			$\|S\| = 3$			$\|S\| = 4$		
	$Q = 3$	$Q = 4$	$Q = 5$	$Q = 3$	$Q = 4$	$Q = 5$	$Q = 3$	$Q = 4$	$Q = 5$
10a	9.8	10.2	10.2	106	114	115	399	393	368
10b	2.6	3.1	4.3	21	22	21	134	133	140
10c	1.3	1.3	1.5	7.2	7.8	8.4	85	99	86
10d	6.7	6.9	6.9	33	35	37	200	277	227
20a	4,417	4,565	5,436	33,424	46,698	47,904	86,400*	86,400*	86,400*
20b	399	417	437	6,882	7,002	7,233	59,963	60,887	61,395
20c	2,515	2,541	3,257	78,187	85,884	86,400*	86,400*	86,400*	86,400*
20d	150	177	181	2,305	2,247	2,430	39,967	41,206	42,424
30a	23,270	26,849	28,650	52,751	53,520	54,759	86,400*	86,400*	86,400*
30b	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*
30c	51,552	52,880	54,089	83,764	84,537	85,040	86,400*	86,400*	86,400*
30d	62,815	64,182	65,223	86,400*	86,400*	86,400*	86,400*	86,400*	86,400*

* Time limit reached

To show the advantages of using the stochastic model for finding the optimal fortification strategy when the number of interdictions is uncertain, we analyze the same instance "10a" as Section 4.5.2.2. We present the results in Table 4.11. We observe that the stochastic model performs well for all of the values of U. The deterministic fortification strategy of $U = 3$ is the same as the stochastic fortification strategy. The deterministic fortification strategy of $U = 4$ has the worst performance under the real number of interdictions 2 and 3.

Table 4.11: Comparison of the OCST Cost for Deterministic and Stochastic Fortification Strategies

	Actual no. of Interdictions				Fortified Edges
	$U = 1$	$U = 2$	$U = 3$	$U = 4$	
$U = 1$	75,198	77,679	81,155	84,671	$\{0,8\} - \{1,9\} - \{7,9\} - \{8,9\}$
$U = 2$	75,198	77,679	81,155	84,671	$\{0,8\} - \{1,9\} - \{7,9\} - \{8,9\}$
$U = 3$	75,265	77,746	80,549	84,167	$\{0,8\} - \{3,7\} - \{7,9\} - \{8,9\}$
$U = 4$	75,265	79,307	81,788	83,545	$\{0,8\} - \{4,7\} - \{7,9\} - \{8,9\}$
Stochastic	75,265	77,746	80,549	84,167	$\{0,8\} - \{3,7\} - \{7,9\} - \{8,9\}$

4.6 Conclusion

We present tri-level mathematical models for fortifying the minimum spanning tree problem and the optimum communication spanning tree with uncertain number of interdictions. We solve the tri-level deterministic and tri-level stochastic models using backward sampling framework (BSF). We implement the BSF algorithm as the third-level OCST problem is MIP and we cannot use solution

methods based on duality. Moreover, we improve the performance of the BSF algorithm with the waiting list acceleration technique. The extensive computational experiments show that the performance of the BSF algorithm improves significantly with the waiting list acceleration technique. We observe that the number of fortifications and interdictions affect the performance of the BSF algorithm; however, the number of interdictions has a significant impact on the CPU time. Moreover, as the number of scenarios increases, the CPU time increases as more perceived damage and real damage models should be solved. Finally, we demonstrate the advantages of using the stochastic model in cases where the interdiction resources are uncertain.

Chapter 5

Conclusion

This thesis addressed the effects of stochastic interdictions on the design and fortification of networks. It contributed to the current literature by presenting stochastic models for distribution and multicommodity network design problems, and stochastic fortification models for the minimum spanning tree (MST) and optimum communication spanning tree (OCST) problems. To solve the tri-level stochastic design models, we developed exact algorithms based on the Benders decomposition (BD) algorithm improved by acceleration techniques tailored to the presented models. We used the backward sampling framework (BSF) to solve the tri-level stochastic fortification models. The BSF was used to solve the deterministic fortification problem in the literature. We used the BSF to solve the stochastic fortification models for the first time. We analyzed the effects of stochastic parameters on the design. These contributions collectively highlight the importance of considering stochastic parameters in the design and fortification of networks over deterministic models when the parameters are uncertain.

In Chapter 2, we studied the distribution network design under uncertain outcome of interdictions. In the real world, the interdictions are not always successful; therefore, we consider uncertainty as a set of scenarios. To solve the tri-level stochastic model, we used the BD algorithm. To solve the stochastic subproblems more efficiently, we implemented the dual decomposition algorithm. Moreover, we improved the efficacy of the BD algorithm by valid and supervalid inequalities. The results from extensive computations showed that the BD algorithm improved by dual decomposition and valid and supervalid inequalities reduced CPU time by 35% compared with the BD

algorithm.

Chapter 3 investigated the effects of uncertain interdiction resources on the multicommodity network design. As the designer does not have information about the number of interdictions, we presented a tri-level stochastic model to design the multicommodity network. We proposed the branch-and-Bender-cut (BBC) algorithm to solve the model. The BBC algorithm was enhanced using warm start, variable fixing, cut selection, Pareto-optimal cuts, penalty reformulation, and supervalid and valid inequalities. The best variant of the BBC algorithm, penalty reformulation combined with warm start, variable fixing, and cut selection techniques, solved instances with up to 20 nodes, 120 arcs, 200 commodities, and 4 scenarios to optimality.

Finally, Chapter 4 investigated the fortification problem of MST and OCST problems under an uncertain number of interdictions. As the third level OCST problem is mixed-integer, we could not use solution methods based on duality; therefore, we solved the models using the BSF algorithm. Moreover, we improved the efficacy of the BSF algorithm with the waiting list acceleration technique. Our extensive computational experiments show that the waiting list acceleration technique improves the performance of the BSF algorithm significantly.

Beyond the mentioned contributions, the results of this thesis also represent a first step towards considering more realistic assumptions for network design problems with interdictions. A logical forward step in this area could be the inclusion of the risk behavior of the designer in the model. We considered the risk-neutral designer. Our models can be extended by considering the risk-averse designer. Moreover, the effects of other uncertain parameters can be studied.

Appendix A

Details of Results: Chapter 2

A.1 Tables of Computational Results

The details of experiments for small instances (Sets I, II, III), moderate instances (Sets IV, V, VI), and large instances (Sets VII, VIII, IX) are presented in Tables A.1-A.3, respectively.

Table A.1: Comparison of the Performance of the Algorithms for Set I, Set II, and Set III Instances

				$\|K\|$ = 49								$\|K\|$ = 88								$\|K\|$ = 150							
$\|J\|$	ρ_2	p	B	BD		BD-VI		BDD		BDD-VI		BD		BD-VI		BDD		BDD-VI		BD		BD-VI		BDD		BDD-VI	
				Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
10	0.2	0.5	1	0	263	0	210	0	186	0	112	0	1,029	0	840	0	798	0	508	0	13,969	0	9,548	0	8,205	0	6,031
			2	0	753	0	587	0	402	0	221	0	3,096	0	2,442	0	2,352	0	1,331	0	14,609	0	11,834	0	10,096	0	7,213
			3	0	1,437	0	1,293	0	876	0	534	0	3,186	0	2,617	0	2,300	0	1,739	0	14,995	0	12,201	0	11,093	0	7,924
			4	0	1,506	0	1,310	0	1,067	0	674	0	3,976	0	3,097	0	2,862	0	2,045	0	15,236	0	13,434	0	11,634	0	8,245
			5	0	1,524	0	1,356	0	1,150	0	725	0	4,060	0	3,257	0	3,006	0	2,215	0	15,654	0	14,337	0	12,096	0	8,524
		0.75	1	0	303	0	271	0	195	0	138	0	1,156	0	954	0	849	0	592	0	16,359	0	9,843	0	9,119	0	6,933
			2	0	837	0	675	0	411	0	325	0	3,479	0	2,974	0	2,557	0	1,643	0.5	18,028*	0	12,589	0	11,961	0	8,065
			3	0	1,671	0	1,337	0	922	0	680	0	3,541	0	3,052	0	2,706	0	2,022	2.6	18,084*	0	12,709	0	12,071	0	8,805
			4	0	1,815	0	1,418	0	1,162	0	709	0	4,418	0	3,660	0	3,111	0	2,525	8.8	18,119*	0	14,602	0	12,785	0	9,486
			5	0	1,858	0	1,441	0	1,210	0	750	0	4,511	0	3,823	0	3,378	0	2,637	10.8	18,038*	0	15,497	0	13,172	0	9,934
		0.9	1	0	480	0	368	0	294	0	189	0	1,503	0	964	0	942	0	674	0	16,952	0	10,237	0	9,957	0	7,417
			2	0	1,077	0	824	0	637	0	452	0	4,279	0	3,182	0	2,864	0	1,824	5.2	18,178*	0	13,472	0	12,302	0	8,379
			3	0	1,752	0	1,480	0	1,285	0	885	0	4,460	0	3,205	0	2,950	0	2,143	4.9	18,056*	0	13,596	0	12,639	0	9,180
			4	0	1,891	0	1,621	0	1,486	0	905	0	5,523	0	4,136	0	3,578	0	2,727	7.4	18,033*	0	14,894	0	13,680	0	10,170
			5	0	1,959	0	1,702	0	1,564	0	952	0	6,315	0	4,320	0	3,581	0	2,769	13.8	18,145*	0	15,807	0	13,962	0	10,629
	0.5	0.5	1	0	316	0	278	0	250	0	192	0	1,193	0	991	0	862	0	544	0	14,807	0	10,312	0	8,697	0	6,393
			2	0	866	0	710	0	639	0	575	0	3,778	0	2,686	0	2,599	0	1,451	0	15,485	0	12,544	0	10,399	0	7,574
			3	0	1,509	0	1,358	0	1,222	0	1,100	0	3,696	0	3,062	0	2,635	0	1,895	0	16,045	0	13,099	0	11,648	0	8,241
			4	0	1,626	0	1,415	0	1,274	0	1,146	0	4,573	0	3,406	0	3,091	0	2,168	0	15,845	0	14,643	0	12,565	0	8,493
			5	0	1,661	0	1,462	0	1,316	0	1,184	0	4,947	0	3,844	0	3,247	0	2,437	0	16,436	0	15,054	0	12,943	0	8,865
		0.75	1	0	333	0	300	0	270	0	208	0	1,306	0	1,107	0	976	0	626	0	17,668	0	10,532	0	9,393	0	7,009
			2	0	921	0	838	0	721	0	641	0	3,896	0	3,420	0	2,787	0	1,774	2.6	18,205*	0	13,470	0	12,320	0	8,388
			3	0	1,838	0	1,618	0	1,488	0	1,188	0	3,965	0	3,510	0	3,112	0	2,123	4.8	18,148*	0	13,980	0	12,554	0	9,156
			4	0	1,997	0	1,697	0	1,527	0	1,304	0	5,169	0	4,209	0	3,298	0	2,778	9.3	18,064*	0	15,478	0	13,424	0	9,750
			5	0	2,044	0	1,778	0	1,547	0	1,392	0	5,639	0	4,244	0	3,716	0	2,869	9.5	18,044*	0	16,597	0	13,962	0	10,232
		0.9	1	0	528	0	449	0	404	0	322	0	1,803	0	1,137	0	1,046	0	707	0	17,799	0	11,056	0	10,355	0	7,603
			2	0	1,097	0	987	0	888	0	799	0	5,092	0	3,755	0	3,179	0	1,915	1.4	18,137*	0	14,145	0	12,917	0	8,630
			3	0	1,927	0	1,696	0	1,526	0	1,323	0	4,906	0	3,749	0	3,333	0	2,250	2.5	18,108*	0	14,412	0	13,524	0	9,639
			4	0	2,080	0	1,767	0	1,590	0	1,387	0	6,848	0	4,673	0	3,771	0	2,972	6.4	18,142*	0	16,235	0	14,227	0	10,781
			5	0	2,152	0	1,850	0	1,665	0	1,422	0	7,263	0	4,838	0	3,867	0	2,990	5.7	18,200*	0	17,047	0	14,381	0	11,055
	0.8	0.5	1	0	363	0	327	0	294	0	228	0	1,379	0	1,108	0	918	0	585	0	15,086	0	10,502	0	8,861	0	6,453
			2	0	961	0	865	0	779	0	701	0	4,087	0	3,198	0	2,588	0	1,584	0	15,777	0	13,017	0	11,005	0	7,862
			3	0	1,667	0	1,501	0	1,351	0	1,215	0	4,365	0	3,402	0	2,622	0	2,035	0	16,195	0	13,299	0	11,981	0	8,478
			4	0	1,708	0	1,537	0	1,383	0	1,245	0	5,249	0	4,088	0	3,062	0	2,372	0	16,607	0	14,912	0	12,798	0	8,575
			5	0	1,807	0	1,645	0	1,480	0	1,332	0	5,440	0	4,104	0	3,397	0	2,414	0	16,749	0	16,058	0	13,306	0	9,035
		0.75	1	0	367	0	330	0	297	0	267	0	1,584	0	1,212	0	942	0	680	0	17,831	0	10,630	0	9,940	0	7,209
			2	0	1,013	0	912	0	820	0	738	0	4,662	0	4,015	0	2,814	0	1,922	3.5	18,024*	0	13,722	0	12,798	0	8,630
			3	0	2,022	0	1,820	0	1,638	0	1,474	0	4,637	0	4,029	0	2,868	0	2,386	4.7	18,101*	0	13,853	0	13,157	0	9,420
			4	0	2,196	0	1,977	0	1,779	0	1,583	0	5,876	0	4,758	0	3,298	0	2,803	7.9	18,086*	0	15,624	0	13,397	0	9,960
			5	0	2,248	0	2,023	0	1,821	0	1,603	0	6,180	0	5,085	0	3,479	0	2,901	5.2	18,170*	0	16,913	0	13,962	0	10,729
		0.9	1	0	581	0	523	0	470	0	386	0	1,969	0	1,301	0	999	0	741	0	17,800	0	11,363	0	10,554	0	7,714
			2	0	1,206	0	1,086	0	977	0	879	0	5,820	0	4,041	0	3,038	0	2,061	3.5	18,010*	0	14,684	0	13,286	0	8,714
			3	0	2,120	0	1,908	0	1,717	0	1,545	0	6,245	0	4,102	0	3,265	0	2,465	5.6	18,111*	0	15,088	0	13,808	0	9,914
			4	0	2,287	0	2,058	0	1,852	0	1,667	0	7,455	0	5,294	0	4,043	0	3,000	6.9	18,094*	0	15,937	0	14,364	0	10,984
			5	0	2,377	0	2,130	0	1,917	0	1,709	0	8,336	0	5,832	0	4,190	0	3,046	0	17,202	0	15,359	0	11,586	0	9,452
			Min.	0	263	0	210	0	186	0	112	0	1,029	0	840	0	798	0	508	0	13,969	0	9,548	0	8,205	0	6,031
			Avg.	0	1,399	0	1,216	0	1,061	0	867	0	4,264	0	3,261	0	2,686	0	1,953	2.97	17,143	0	13,648	0	12,064	0	8,708
			Max.	0	2,377	0	2,130	0	1,917	0	1,709	0	8,336	0	5,832	0	4,190	0	3,046	13.8	18,205*	0	17,047	0	14,381	0	11,055

* Time limit reached (5 hours)

Table A.2: Comparison of the Performance of the Algorithms for Set IV, Set V, and Set VI Instances

				$\|K\|$ = 49								$\|K\|$ = 88								$\|K\|$ = 150							
$\|J\|$	ρ_2	p	B	BD		BD-VI		BDD		BDD-VI		BD		BD-VI		BDD		BDD-VI		BD		BD-VI		BDD		BDD-VI	
				Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
12	0.2	0.5	1	0	1,368	0	1,160	0	917	0	629	0	6,779	0	5,736	0	4,589	0	4,067	0	12,153	0	10,949	0	8,222	0	6,413
			2	0	4,429	0	3,574	0	3,056	0	2,245	0	7,766	0	5,945	0	4,840	0	4,424	0	14,564	0	13,240	0	10,708	0	8,246
			3	0	4,583	0	3,720	0	3,070	0	2,347	0	10,793	0	8,860	0	7,123	0	6,475	0	16,757	0	13,964	0	10,764	0	8,396
			4	0	5,268	0	4,126	0	3,537	0	2,576	0	11,684	0	9,013	0	7,652	0	7,008	2.5	18,286*	0	15,023	0	12,644	0	9,989
			5	0	5,359	0	4,320	0	3,635	0	2,894	0	14,163	0	11,540	0	9,966	0	9,442	2.8	18,107*	0	16,565	0	14,045	0	11,236
		0.75	1	0	1,468	0	1,189	0	1,013	0	734	0	7,401	0	6,168	0	5,145	0	4,261	0	13,567	0	11,596	0	8,477	0	6,612
			2	0	4,910	0	3,731	0	3,226	0	2,506	0	8,016	0	6,580	0	5,214	0	4,965	8.4	18,394*	0	14,000	0	11,574	0	8,681
			3	0	5,194	0	4,015	0	3,391	0	2,856	0	11,346	0	9,527	0	7,818	0	6,842	9.2	18,092*	0	14,973	0	11,768	0	8,826
			4	0	5,523	0	4,528	0	3,700	0	2,982	0	12,497	0	10,117	0	7,882	0	7,726	4.6	18,243*	0	15,784	0	13,451	0	10,357
			5	0	6,676	0	5,675	0	4,673	0	3,138	0	15,052	0	12,543	0	10,605	0	9,579	4.2	18,158*	0	16,874	0	14,943	0	11,954
		0.9	1	0	2,074	0	1,763	0	1,369	0	1,244	0	8,326	0	6,784	0	5,736	0	5,057	0	17,094	0	12,893	0	10,240	0	6,930
			2	0	5,249	0	4,094	0	3,569	0	2,754	0	9,529	0	7,765	0	6,635	0	5,859	3.9	18,006*	0	14,365	0	12,238	0	8,967
			3	0	5,620	0	4,665	0	3,765	0	2,939	0	12,594	0	10,575	0	8,172	0	7,403	5.4	18,193*	0	15,099	0	12,616	0	9,546
			4	0	7,290	0	6,196	0	4,957	0	3,572	0	13,443	0	11,736	0	9,496	0	8,535	8.5	18,083*	0	16,146	0	14,796	0	11,541
			5	0	9,094	0	7,457	0	6,184	0	4,729	10.6	18,094*	0	14,550	0	11,508	0	11,376	12.3	18,034*	0	17,471	0	15,839	0	12,354
	0.5	0.5	1	0	1,611	0	1,350	0	1,047	0	897	0	7,676	0	6,596	0	5,157	0	5,037	0	14,559	0	12,235	0	9,867	0	7,400
			2	0	4,722	0	3,784	0	3,075	0	2,602	0	7,804	0	6,813	0	5,388	0	5,326	0	16,700	0	14,152	0	11,993	0	9,549
			3	0	4,735	0	3,966	0	3,116	0	2,691	0	12,240	0	10,278	0	8,118	0	7,736	0	17,570	0	15,549	0	12,702	0	9,903
			4	0	5,716	0	4,515	0	3,772	0	2,744	0	12,320	0	11,606	0	8,316	0	7,849	5.2	18,099*	0	16,008	0	14,793	0	11,095
			5	0	6,679	0	5,142	0	4,341	0	3,606	0	16,185	0	13,386	0	10,709	0	10,344	5.4	18,158*	0	16,958	0	16,714	0	13,204
		0.75	1	0	1,724	0	1,379	0	1,190	0	918	0	8,478	0	6,908	0	5,526	0	5,401	0	15,068	0	12,769	0	9,579	0	7,184
			2	0	5,008	0	4,040	0	3,389	0	2,642	0	8,661	0	7,502	0	6,138	0	5,728	5.9	18,206*	0	15,915	0	12,732	0	9,595
			3	0	5,215	0	4,194	0	3,478	0	2,871	0	12,980	0	11,433	0	8,938	0	8,003	6.5	18,194*	0	16,834	0	13,650	0	10,511
			4	0	7,082	0	5,524	0	4,603	0	3,541	0	13,719	0	12,431	0	9,786	0	8,035	3.6	18,222*	0	17,255	0	15,065	0	11,299
			5	0	7,556	0	5,818	0	5,214	0	3,751	0	16,667	0	14,550	0	12,434	0	10,318	7.9	18,083*	5.4	18,100*	0	17,781	0	13,691
		0.9	1	0	2,471	0	2,099	0	1,606	0	1,433	0	9,200	0	7,666	0	6,133	0	5,297	0	14,285	0	12,954	0	10,718	0	8,467
			2	0	6,378	0	5,103	0	4,146	0	3,062	0	10,912	0	9,162	0	7,330	0	6,247	4.8	18,068*	0	16,010	0	14,319	0	11,169
			3	0	7,536	0	5,728	0	5,200	0	3,542	0	15,055	0	12,267	0	10,148	0	9,925	5.4	18,119*	0	17,695	0	14,508	0	11,317
			4	0	9,108	0	7,244	0	6,093	0	4,645	0	16,602	0	13,731	0	10,985	0	10,610	10.2	18,073*	7.2	18,105*	0	17,459	0	13,444
			5	0	10,894	0	8,824	0	7,626	0	5,011	9.5	18,132*	0	17,024	0	14,548	0	11,917	7.3	18,039*	3.5	18,084*	2.9	18,153*	0	13,963
	0.8	0.5	1	0	1,792	0	1,362	0	1,255	0	1,040	0	7,931	0	6,711	0	5,494	0	5,308	0	14,178	0	13,914	0	9,456	0	7,470
			2	0	5,395	0	4,100	0	3,777	0	2,728	0	8,150	0	7,741	0	5,735	0	5,373	0	16,885	0	15,350	0	12,379	0	9,718
			3	0	5,456	0	4,474	0	3,819	0	3,165	0	12,284	0	10,810	0	8,353	0	7,861	1.9	18,118*	0	16,258	0	12,957	0	9,907
			4	0	7,290	0	5,759	0	4,884	0	4,301	0	13,821	0	12,884	0	8,450	0	7,964	8.5	18,094*	0	17,125	0	15,426	0	12,186
			5	0	7,873	0	6,456	0	5,117	0	4,635	0	15,994	0	13,963	0	11,298	0	11,171	7.4	18,034*	1.3	18,073*	0	17,135	0	12,852
		0.75	1	0	2,122	0	1,804	0	1,379	0	1,082	0	8,535	0	7,278	0	6,285	0	5,161	0	13,958	0	13,303	0	9,918	0	7,835
			2	0	6,527	0	4,960	0	4,373	0	3,316	0	9,265	0	7,962	0	6,442	0	6,297	4.8	18,321*	0	16,134	0	13,310	0	10,648
			3	0	6,631	0	4,974	0	4,576	0	3,655	0	13,061	0	12,042	0	9,143	0	8,841	6.5	18,264*	0	17,510	0	14,121	0	10,732
			4	0	7,991	0	5,993	0	5,594	0	3,892	0	14,138	0	13,443	0	10,323	0	9,650	4.3	18,082*	0	17,956	0	15,872	0	12,698
			5	0	8,281	0	6,594	0	5,797	0	4,724	4.8	18,150*	0	15,303	0	11,825	0	10,851	7.25	18,268*	3.9	18,148*	2	18,101*	0	14,255
		0.9	1	0	2,559	0	2,113	0	1,715	0	1,482	0	9,291	0	7,802	0	6,667	0	5,745	0	14,409	0	13,465	0	10,903	0	8,722
			2	0	7,333	0	5,720	0	4,840	0	4,253	0	11,078	0	10,262	0	7,746	0	7,413	3.5	18,005*	0	17,037	0	14,508	0	11,462
			3	0	9,055	0	7,196	0	6,157	0	4,346	0	14,421	0	13,790	0	10,383	0	9,921	11.4	18,102*	0	17,785	0	15,175	0	12,140
			4	0	9,468	0	7,559	0	6,354	0	4,745	0	16,912	0	15,301	0	11,231	0	10,328	6.8	18,208*	4.8	18,103*	0	17,607	0	13,910
			5	0	12,088	0	10,154	0	8,461	0	7,132	3.5	18,067*	0	17,752	0	14,040	0	13,394	8.9	18,344*	5.1	18,027*	1.1	18,090*	0	14,942
			Min.	0	1,368	0	1,160	0	917	0	629	0	6,779	0	5,736	0	4,589	0	4,067	0	12,153	0	10,949	0	8222	0	6413
			Avg.	0	5,787	0	4,625	0	3,912	0	3,035	0.63	12,116	0	10,485	0	8,343	0	7,690	4.34	17,210	0.69	15,683	0.13	13,496	0	10,474
			Max.	0	12,088	0	10,154	0	8,461	0	7,132	10.6	18,150*	0	17,752	0	14,548	0	13,394	12.3	18,394*	7.2	18,148*	2.9	18,153*	0	14,942

* Time limit reached (5 hours)

Table A.3: Comparison of the Performance of the Algorithms for Set VII, Set VIII, and Set IX Instances

				|K| = 49								|K| = 88								|K| = 150							
|J|	ρ_2	p	B	BD		BD-VI		BDD		BDD-VI		BD		BD-VI		BDD		BDD-VI		BD		BD-VI		BDD		BDD-VI	
				Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
15	0.2	0.5	1	0	2,603	0	2,315	0	2,079	0	1,690	0	18,402	0	16,729	0	16,172	0	13,941	0	18,284	0	16,259	0	15,966	0	14,647
			2	0	3,438	0	2,953	0	2,594	0	2,109	0	28,421	0	25,216	0	24,361	0	21,370	4.3	36,350*	0	35,522	0	34,218	0	32,589
			3	0	3,566	0	3,191	0	2,698	0	2,346	0	29,579	0	25,856	0	24,994	0	21,546	6.4	36,032*	1.7	36,022*	0	35,316	0	32,760
			4	0	8,254	0	6,485	0	5,797	0	4,913	5.9	36,026*	5.3	36,283*	0	35,878	0	31,472	9.3	36,173*	5.7	36,312*	1.3	36,222*	0	33,100
			5	0	8,695	0	6,774	0	6,572	0	5,055	12.4	36,091*	8.9	36,008*	1.8	36,206*	0	34,200	16.1	36,500*	10.6	36,333*	5.8	36,300*	0	33,840
		0.75	1	0	5,436	0	4,750	0	3,453	0	2,774	0	25,891	0	21,980	0	18,184	0	15,320	0	25,411	0	20,753	0	19,924	0	16,096
			2	0	6,433	0	5,788	0	5,002	0	3,752	0.9	36,147*	0	30,088	0	28,033	0	22,978	1.5	36,005*	0.9	36,289*	0.3	36,178*	0	35,812
			3	0	13,019	0	10,805	0	9,508	0	7,540	2.2	36,267*	0	32,928	0	28,510	0	23,420	12.5	36,121*	7.1	36,148*	3.8	36,008*	1.5	36,033*
			4	0	29,527	0	25,027	0	23,825	0	19,725	16.7	36,301*	10.2	36,243*	3	36,227*	0	34,969	28	36,118*	24.5	36,007*	15.1	36,163*	4.9	36,290*
			5	0	32,952	0	27,844	0	24,451	0	20,367	20.4	36,000*	15.3	36,112*	8.1	36,183*	2.6	36,203*	45	36,178*	20.4	36,126*	10.5	36,226*	5.4	36,194*
		0.9	1	0	6,484	0	5,299	0	4,370	0	2,985	0	26,247	0	22,855	0	18,200	0	16,546	0	27,991	0	23,096	0	20,953	0	17,384
			2	0	7,217	0	6,173	0	5,810	0	4,539	7.4	36,041*	0	31,628	0	28,087	0	24,816	18.9	36,153*	7.3	36,259*	2.8	36,109*	1.9	36,266*
			3	0	16,708	0	12,829	0	12,730	0	9,945	8	36,102*	0	33,038	0	29,901	0	24,825	28.2	36,094*	9.8	36,155*	4.2	36,093*	3.7	36,064*
			4	0	33,120	0	27,600	0	24,783	0	21,231	10.5	36,018*	5.9	36,092*	3.8	36,006*	1.8	36,102*	32.5	36,083*	15.9	36,327*	9.6	36,205*	3.9	36,202*
			5	0	33,961	0	29,270	0	26,839	0	21,565	12.8	36,036*	6.8	36,026*	3.5	36,192*	2.3	36,330*	39.5	36,143*	19.3	36,290*	10.2	36,151*	5.6	36,102*
	0.5	0.5	1	0	4,378	0	3,625	0	3,374	0	2,789	0	20,007	0	18,589	0	17,486	0	15,754	0	19,501	0	17,953	0	16,746	0	15,087
			2	0	5,264	0	4,657	0	4,319	0	3,374	0	31,548	0	29,650	0	27,278	0	23,720	6.1	36,172*	3.9	36,074*	0	35,031	0	32,644
			3	0	7,546	0	5,659	0	5,308	0	4,387	0	31,595	0	30,704	0	27,919	0	24,563	7.6	36,056*	4.2	36,330*	0.9	36,277*	0	33,398
			4	0	12,023	0	9,783	0	8,663	0	7,468	1.3	36,139*	1.1	36,288*	0.5	36,148*	0	34,305	9.8	36,431*	4.4	36,298*	2.4	36,044*	0	34,748
			5	0	14,160	0	12,640	0	11,594	0	9,503	2.9	36,002*	1.5	36,212*	1.2	36,204*	0.8	36,153*	19.9	36,361*	8.9	36,303*	6.8	36,051*	0	35,870
		0.75	1	0	8,249	0	7,194	0	6,037	0	4,985	0	24,840	0	22,682	0	19,358	0	16,546	0	26,886	0	22,753	0	20,581	0	16,740
			2	0	11,087	0	10,086	0	8,441	0	7,153	3.2	36,000*	0	31,089	0	28,309	0	24,825	8.4	36,100*	4.3	36,179*	3.7	36,006*	1.3	36,231*
			3	0	26,862	0	23,316	0	19,172	0	16,111	4.7	36,188*	0	33,038	0	31,804	0	25,276	12.3	36,075*	8.3	36,257*	4.1	36,116*	0.9	36,004*
			4	0	35,175	0	26,335	0	24,125	0	20,100	3.7	36,352*	3.3	36,173*	2.4	36,253*	2.3	36,115*	19.5	36,202*	9.7	36,118*	8.2	36,200*	1.6	36,093*
			5	0	35,532	0	28,944	0	25,454	0	20,901	5.9	36,281*	4.2	36,281*	2.9	36,289*	2.4	36,092*	24.2	36,364*	13.8	36,029*	11.1	36,058*	2.2	36,146*
		0.9	1	0	8,674	0	7,830	0	6,381	0	5,409	0	26,932	0	23,022	0	20,020	0	18,200	0	28,803	0	23,972	0	20,983	0	18,079
			2	0	12,902	0	11,628	0	10,949	0	8,488	2.4	36,333*	0	32,122	0	29,760	0	26,811	15.3	36,018*	12.7	36,202*	7.8	36,185*	3.5	36,154*
			3	0	30,972	0	26,105	0	22,770	0	18,664	4.8	36,201*	0	33,710	0	32,120	0	27,298	20.2	36,260*	14.2	36,174*	9.2	36,038*	4.1	36,205*
			4	1.2	36,000*	0	35,449	0	33,103	0	26,065	5.3	36,118*	4.5	36,290*	2.8	36,112*	2.5	36,229*	17.4	36,099*	16.5	36,032*	10.7	36,270*	4.8	36,205*
			5	1.5	36,118*	1.1	36,094*	0.8	36,293*	0	31,847	7.7	36,103*	4.9	36,341*	3.5	36,280*	2.2	36,178*	33.2	36,154*	16.9	36,143*	12.2	36,117*	5.3	36,302*
	0.8	0.5	1	0	6,523	0	5,035	0	4,870	0	4,519	0	21,127	0	21007	0	19,327	0	18,086	0	21,451	0	19,949	0	18,877	0	15,791
			2	0	7,422	0	6,443	0	5,966	0	5,011	0	33,756	0	32,027	0	30,476	0	27,895	7.7	36,189*	2.8	36,063*	0	35,581	0	33,641
			3	0	9,957	0	8,753	0	8,235	0	7,504	1.1	36,037*	0	32,792	0	31,469	0	30,054	9.1	36,431*	4.6	36,189*	1.8	36,110*	0	34,781
			4	0	14,333	0	14,027	0	13,283	0	11,506	2.3	36,402*	2.1	36,248*	0.3	36,224*	0	34,593	12.3	36,028*	10.7	36,310*	3.3	36,251*	0	35,135
			5	0	18,235	0	18,125	0	15,034	0	12,543	3.1	36,138*	2.6	36,156*	1.5	36,185*	0.5	36,220*	18.4	36,190*	15.8	36,263*	9.2	36,180*	0	35,960
		0.75	1	0	11,632	0	9,559	0	8,452	0	7,407	0	26,579	0	26,504	0	24,413	0	24,179	0	29,036	0	27,840	0	26,082	0	24,490
			2	0	14,158	0	12,852	0	11,974	0	10,355	1.6	36,207*	0	35,154	0	34,344	0	32,193	6.6	36,285*	3.1	36,220*	2.2	36,336*	0.4	36,093*
			3	0	31,429	0	30,961	0	30,806	0	24,203	3.2	36,405*	1.8	36,082*	0	35,640	0	33,123	11.8	36,092*	5.3	36,199*	3.5	36,250*	1.2	36,129*
			4	1.9	36,104*	0	34,324	0	32,889	0	30,563	5.2	36,125*	3.5	36,011*	1.9	36018*	0.8	36,007*	16.8	36,350*	14.1	36,038*	10.3	36,353*	3.9	36,240*
			5	2.1	36,074*	0	34,985	0	33,996	0	31,112	8.6	36,083*	5.8	36,299*	2.1	36,027*	1.3	36,118*	22.2	36,398*	14.3	36,181*	11.1	36,280*	5.2	36,193*
		0.9	1	0	11,884	0	10,807	0	10,526	0	9,593	0	28,737	0	28,548	0	27,121	0	26,456	0	29,955	0	27,259	0	26,867	0	24,515
			2	0	18,965	0	13,234	0	12,463	0	11,898	3.8	36,005*	0	35,910	0	34,992	0	32,678	10.2	36,400*	7.4	36,193*	2.5	36,059*	0.8	36,110*
			3	2.4	36,072*	0	34,103	0	33,199	0	27,463	5.4	36,174*	3.6	36,192*	0	35,835	0	33,517	16.9	36,476*	10.7	36,105*	5.7	36,004*	1.8	36,269*
			4	2.5	36,263*	2.2	36,186*	0	34,379	0	31,029	7.2	36,427*	4.7	36,275*	2.1	36,077*	0.7	36,103*	22.8	36,240*	15.3	36,253*	10.2	36,014*	6.1	36,080*
			5	2.1	36,394*	1.9	36,224*	1.2	36,189*	0	32,522	11.7	36,293*	7.1	36,136*	4.4	36,302*	1.8	36,267*	29.2	36,006*	18.1	36,345*	13.5	36,118*	7.2	36,120*
			Min.	0	2,603	0	2,315	0	2,079	0	1,690	0	18,402	0	16,729	0	16,172	0	13,941	0	18,284	0	16,259	0	15,966	0	14,647
			Avg.	0.30	18,262	0.12	16,268	0.04	15,083	0	12,778	4.27	33,216	2.29	31,791	1.02	30,554	0.49	28,569	13.78	34,014	8.07	33,380	4.76	32,980	1.72	31,974
			Max.	2.5	36,394*	2.2	36,224*	1.2	36,293*	0	32,522	20.4	36,427*	15.3	36,341*	8.1	36,302*	2.6	36,330*	45	36,500*	24.5	36,345*	15.1	36,353*	7.2	36,302*

* Time limit reached (10 hours)

Appendix B

Smith et al. (2007)'s Algorithm, Details of Test Instances, and Results: Chapter 3

B.1 Details of Smith et al. (2007)'s Algorithm

We present the pseudo-code for the algorithm presented by Smith et al. (2007). We define UB as an upper bound on the optimal objective value obtained by solving the scenario SPs, LB as a lower bound on the optimal objective value obtained by solving the MP, and t as the current iteration. Let Υ^t be the set of extreme points generated up to iteration t, and ϵ be the desirable optimality gap.

Algorithm 7 Smith et al. (2007)'s Algorithm

$UB \leftarrow \infty$, $LB \leftarrow -\infty$, $t \leftarrow 0$, $\Upsilon^t \leftarrow \varnothing$

While ($\frac{UB - LB}{UB} > \epsilon$) **do**

Solve the MP to obtain w

Update LB

For $s \in S$ **do:**

Solve the SP_s for a given $\hat{w}$

Obtain dual variables (π, α, φ)

End for

Update UB

$\Upsilon^{t+1} \leftarrow \Upsilon^t \cup \{\pi, \alpha, \varphi\}$

$t \leftarrow t + 1$

End while

B.2 Detailed Input Data

In this section, we present the details of "r" set in Table B.1.

B.3 Detailed Results

In this section, we report the detailed results of "r01-r09" on five variants of the branch-and-Benders-cut (BBC) algorithm in Table B.2. We also present the detailed results of "r10-r18" for the best three variants of BBC algorithm in Table B.3. In Table B.2 and B.3, The columns "N", "A", and "K" represent the number of nodes, arcs, and commodities, respectively. In the subsequent columns, we present the CPU time (s) for instances with an optimal solution. If the algorithms reached the time limit for a specific instance, we present the gap (%) instead of the CPU time. In Table B.4, $|\Omega|$ is the number of demand scenarios.

Table B.1: Details of ”r” Set

Set	Node	Arc	Commodity	# of scenarios	Interdiction budget
r01	10	35	10	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r02	10	35	25	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r03	10	35	50	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r04	10	60	10	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r05	10	60	25	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r06	10	60	50	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r07	10	82	10	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r08	10	83	25	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r09	10	83	50	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r10	20	120	40	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r11	20	120	100	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r12	20	120	200	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r13	20	220	40	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r14	20	220	100	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r15	20	220	200	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r16	20	314	40	2	1, 2
				3	2, 3, 4
				4	1, 2, 3, 4
r17	20	318	100	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4
r18	20	315	200	2	1, 2
				3	1, 2, 3
				4	1, 2, 3, 4

Table B.2: CPU Time (s) of Variants of BBC Algorithm on "r01-r09" Sets

#	Set	$\|N\|$	$\|A\|$	$\|K\|$	$\|S\|=2$					$\|S\|=3$					$\|S\|=4$				
					BBC1	BBC2	BBC3	BBC4	BBC5	BBC1	BBC2	BBC3	BBC4	BBC5	BBC1	BBC2	BBC3	BBC4	BBC5
1	r01	10	35	10	17	5	14	17	5	21	6	20	23	8	40	8	46	44	12
2		10	35	10	21	7	19	21	6	29	9	26	30	8	40	11	32	45	12
3		10	35	10	21	8	23	21	7	38	9	27	40	8	58	12	50	62	14
4		10	35	10	14	5	17	14	7	23	6	23	24	8	42	9	39	46	9
5		10	35	10	16	7	17	17	7	28	9	26	28	7	47	13	50	57	9
6		10	35	10	16	6	21	17	6	28	8	27	29	9	40	9	44	44	11
7		10	35	10	3	1	3	3	1	4	1	5	5	1	7	1	8	8	1
8		10	35	10	5	1	5	6	1	10	2	13	10	3	15	3	13	17	3
9		10	35	10	8	2	7	8	2	12	3	12	13	3	20	4	22	23	4
10	r02	10	35	25	21	5	19	20	5	28	7	41	29	7	36	9	44	40	9
11		10	35	25	51	14	57	40	19	47	15	71	50	17	82	18	111	87	15
12		10	35	25	60	17	50	64	17	88	29	91	86	26	100	29	144	148	28
13		10	35	25	26	4	20	25	5	37	8	31	36	7	52	10	49	54	11
14		10	35	25	23	7	29	25	7	58	9	54	56	12	90	15	75	65	15
15		10	35	25	29	5	25	28	6	55	12	56	45	13	84	12	73	87	15
16		10	35	25	18	7	17	17	7	29	7	22	28	7	31	8	27	35	8
17		10	35	25	8	2	8	9	3	13	3	12	15	4	11	4	15	14	5
18		10	35	25	18	8	23	17	6	34	11	34	37	14	47	14	46	48	13
19	r03	10	35	50	110	31	100	110	28	164	34	135	160	39	181	41	151	253	48
20		10	35	50	122	30	140	118	33	230	61	216	170	50	220	49	197	233	56
21		10	35	50	146	29	119	125	36	235	52	186	222	45	407	70	291	314	65
22		10	35	50	39	9	32	42	11	55	13	49	56	18	71	15	57	79	21
23		10	35	50	89	22	69	86	16	99	25	94	97	27	104	26	100	107	28
24		10	35	50	89	21	69	85	16	99	25	94	97	27	104	27	100	107	28
25		10	35	50	25	14	26	27	14	32	16	36	33	16	32	17	38	37	19
26		10	35	50	24	12	25	26	15	33	19	32	33	17	33	16	36	38	20
27		10	35	50	24	12	25	25	14	32	15	34	33	17	32	18	35	36	18
28	r04	10	60	10	109	39	72	135	41	642	102	224	780	118	619	151	351	782	169
29		10	60	10	105	32	75	107	43	461	129	253	518	100	558	149	385	632	228
30		10	60	10	113	36	60	107	52	268	115	234	300	96	573	161	278	573	148
31		10	60	10	298	52	112	260	77	287	52	176	274	68	310	95	200	365	91
32		10	60	10	150	60	115	142	65	271	70	201	252	89	439	117	364	500	124
33		10	60	10	143	60	96	139	70	185	85	180	171	82	290	77	347	304	94
34		10	60	10	29	5	17	35	7	68	12	38	64	7	63	14	49	59	11
35		10	60	10	17	5	14	17	5	31	8	28	27	7	46	11	51	44	9
36		10	60	10	19	6	19	22	5	43	8	32	35	8	63	10	52	65	11
37	r05	10	60	25	1,353	581	883	1,355	596	10,594	3,572	8,702	10,428	4,528	6,555	3,429	4,983	7,588	3,033
38		10	60	25	1,095	543	966	1,153	583	6,958	2,872	5,577	6,191	3,419	5,881	2,454	4,468	6,420	2,779
39		10	60	25	1,191	542	954	1,076	577	6,327	3,351	5,351	6,883	3,529	5,218	2,316	5,974	8,437	3,424
40		10	60	25	2,087	664	1,374	1,836	798	1,693	602	1,352	1,360	810	1,776	796	1,014	1,410	842
41		10	60	25	1,265	553	1,105	1,106	743	1,929	610	1,300	1,815	911	1,855	793	1,843	1,957	1,160
42		10	60	25	1,599	541	1,311	1,625	736	1,910	874	1,647	1,903	876	2,698	751	1,957	2,815	1,209
43		10	60	25	51	15	56	51	13	84	22	77	80	19	98	24	83	107	20
44		10	60	25	71	22	78	71	18	96	30	104	96	21	111	31	100	126	24
45		10	60	25	88	23	75	97	24	124	31	121	114	25	120	31	123	147	31

Table B.2: CPU Time (s) of Variants of BBC Algorithm on "r01-r09" Sets (continued)

#	Set	$\|N\|$	$\|A\|$	$\|K\|$	$\|S\|=2$					$\|S\|=3$					$\|S\|=4$				
					BBC1	BBC2	BBC3	BBC4	BBC5	BBC1	BBC2	BBC3	BBC4	BBC5	BBC1	BBC2	BBC3	BBC4	BBC5
46	r06	10	60	50	1,339	312	748	1,149	337	4,054	904	2,720	4,372	1,033	4,215	1,542	2,128	4,338	1,737
47		10	60	50	1,299	499	800	1,226	381	4,128	1,459	3,681	4,930	1,604	8,410	3,239	4,079	7,545	3,537
48		10	60	50	1,067	584	1,046	1,152	342	28,691	5,085	10,608	28,060	7,573	49,501	18,956	36,189	39,608	30,555
49		10	60	50	2,342	446	1,489	2,028	563	3,511	1,007	1,891	3,454	987	5,075	1,490	2,953	6,230	1,598
50		10	60	50	3,515	1,532	2,797	3,622	1,746	14,871	5,061	7,050	13,836	6,446	13,825	6,601	11,931	16,541	7,441
51		10	60	50	4,477	1,235	3,479	5,311	1,904	16,371	4,917	10,630	18,755	7,309	42,988	16,932	28,730	41,414	26,956
52		10	60	50	57	16	45	61	15	89	48	81	97	34	110	52	86	122	40
53		10	60	50	131	39	99	126	35	174	70	168	182	59	209	59	170	241	63
54		10	60	50	1,713	693	978	1,734	783	3,394	1,408	3,011	3,116	1,741	5,953	2,495	3,554	4,826	2,403
55	r07	10	82	10	110	28	43	108	26	1,043	368	435	1,145	289	5,124	1,899	3,624	5,797	2,574
56		10	82	10	81	22	66	90	34	765	216	460	729	289	4,551	2,090	3,266	4,944	1,920
57		10	82	10	107	27	70	135	39	640	199	426	638	228	5,774	1,592	3,137	5,943	2,229
58		10	82	10	231	63	111	242	59	1,995	689	1,341	2,481	734	3,358	1,070	3,151	4,059	1,388
59		10	82	10	140	36	123	147	59	1,014	332	604	1,058	460	4,069	1,129	2,425	4,209	1,493
60		10	82	10	95	27	76	96	38	1,180	356	579	1,292	402	2,874	1,335	2,062	2,771	1,474
61		10	82	10	248	50	178	329	65	333	97	177	420	107	279	73	165	331	70
62		10	82	10	169	48	125	129	57	295	71	213	289	89	335	102	246	274	98
63		10	82	10	132	40	109	154	54	298	92	212	218	97	380	88	281	442	115
64	r08	10	83	25	776	181	493	786	252	21,775	5,506	8,815	23,685	6,028	83,289	36,798	65,651	80,367	50,005
65		10	83	25	808	409	541	780	341	8,788	3,387	7,019	10,818	4,152	76,940	21,922	64,290	64,934	23,498
66		10	83	25	518	210	339	571	263	4,552	2,365	5,453	5,390	3,044	71,651	20,428	46,083	60,988	21,950
67		10	83	25	2,496	664	1,476	2,821	1,059	31,663	10,130	16,178	31,072	9,742	72,846	49,036	59,412	71,716	68,955
68		10	83	25	1,909	701	1,085	1,921	721	12,246	5,314	11,895	13,821	6,410	78,099	45,106	70,694	68,040	47,403
69		10	83	25	1,227	348	715	1,190	537	11,250	5,620	8,037	11,230	6,492	76,033	14,511	48,714	49,121	28,251
70		10	83	25	2,906	1,159	2,027	3,172	1,121	3,554	1,093	2,680	3,453	1,752	3,978	1,769	3,985	4,951	2,120
71		10	83	25	1,432	593	1,275	1,505	538	2,184	1,124	2,718	2,678	1,021	4,814	1,752	4,634	5,471	1,901
72		10	83	25	1,519	829	1,445	1,458	893	3,195	1,391	3,048	2,701	1,372	4,162	1,651	3,839	3,630	2,064
73	r09	10	83	50	15,014	6,582	9,678	16,578	6,969	79,527	33,700	38,934	77,392	36,774	(7.4)*	42,937	58,795	(5.3)	54,674
74		10	83	50	8,002	3,614	5,782	7,624	3,923	59,176	18,500	40,634	60,106	14,745	85,793	35,880	62,793	83,753	41,470
75		10	83	50	4,982	2,403	3,691	4,895	3,740	27,189	12,639	29,868	26,267	15,297	72,165	42,978	52,165	69,164	40,241
76		10	83	50	25,595	10,616	11,931	22,529	10,957	74,382	45,797	56,467	73,277	56,478	(11.6)	70,164	80,521	(9.2)	76,664
77		10	83	50	20,354	13,423	20,382	22,460	17,669	64,073	33,703	54,281	70,759	35,999	(3.7)	62,608	71,598	(2.9)	61,949
78		10	83	50	13,660	10,714	11,417	13,960	10,177	33,708	16,066	28,605	30,283	24,483	(1.3)	55,606	64,351	86,357	55,739
79		10	83	50	606	189	520	604	195	819	208	620	767	249	1,013	264	880	1,136	303
80		10	83	50	746	246	664	713	313	1,509	269	763	1,135	292	1,067	415	1,170	1,302	506
81		10	83	50	1,503	440	1,188	1,293	562	1,883	597	1,459	1,927	826	2,520	727	2,165	2,971	770

* () indicates the gap at the time limit

Table B.3: CPU Time (s) of Variants of BBC Algorithm on "r10-r18" Sets

#	Set	$\|N\|$	$\|A\|$	$\|K\|$	$\|S\|=2$			$\|S\|=3$			$\|S\|=4$		
					BBC2	BBC3	BBC5	BBC2	BBC3	BBC5	BBC2	BBC3	BBC5
82	r10	20	120	40	16,358	21,935	16,997	41,430	73,387	59,250	81,474	(4.8)*	83,208
83		20	120	40	10,526	19,755	13,994	32,679	66,082	38,693	(11)	(15.8)	(11.1)
84		20	120	40	8,235	9,782	8,392	24,862	51,325	31,439	35,079	66,078	52,316
85		20	120	40	51,360	69,034	55,916	(1.1)	(4.6)	(1.1)	85,039	(17.4)	(10.5)
86		20	120	40	24,454	33,990	28,850	34,090	(0.8)	50,767	(9.1)	(20.3)	(19.2)
87		20	120	40	14,274	20,590	16,656	32,802	67,807	29,639	53,183	(9)	76,177
88		20	120	40	1,494	5,675	1,552	1,647	7,598	1,689	1,962	9,041	2,397
89		20	120	40	1,718	9,055	1,723	3,845	11,549	3,378	5,905	13,346	4,498
90		20	120	40	2,155	6,983	2,958	4,717	14,590	5,554	6,446	17,118	6,770
91	r11	20	120	100	25,138	(1.2)	34,932	(9.2)	(12.2)	(9.5)	(16.2)	(26.5)	(15.8)
92		20	120	100	11,834	21,411	17,038	18,757	43,432	18,138	63,109	73,041	66,462
93		20	120	100	8,779	14,065	9,736	17,423	59,353	24,563	54,008	82,929	67,976
94		20	120	100	38,504	57,015	39,438	34,178	(17.1)	58,690	62,532	(100)	79,739
95		20	120	100	54,953	(1.4)	70,313	(6.7)	(100)	(6.8)	(8.5)	(100)	(13.2)
96		20	120	100	59,696	79,218	58,872	(9.1)	(18.2)	(13.7)	(11.9)	(100)	(22.9)
97		20	120	100	4,229	5,048	3,696	5,224	8,452	4,669	4,396	12,796	5,044
98		20	120	100	5,983	5,715	4,770	4,889	11,314	4,813	6,581	15,454	6,351
99		20	120	100	3,559	5,530	4,166	10,232	19,047	13,184	12,986	21,144	15,992
100	r12	20	120	200	(13.3)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
101		20	120	200	(9.7)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
102		20	120	200	(17.4)	(30.5)	(25.8)	(31.2)	(100)	(39.2)	(100)	(100)	(100)
103		20	120	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
104		20	120	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
105		20	120	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
106		20	120	200	18,459	25,471	23,340	25,043	32,750	26,570	27,877	38,452	27,061
107		20	120	200	16,151	21,689	19,476	28,790	36,456	32,768	30,271	51,201	32,166
108		20	120	200	15,065	19,093	16,256	22,139	35,214	31,197	32,604	48,959	36,311
109	r13	20	220	40	23,257	26,391	25,174	(10.4)	(11.8)	(10.1)	(12.2)	(18.3)	(17.2)
110		20	220	40	26,565	31,405	33,794	60,421	(14.1)	(8.4)	75,341	(5.7)	79,368
111		20	220	40	29,617	33,325	37,754	82,657	86,262	83,048	(9.3)	(12.8)	(10.5)
112		20	220	40	58,635	60,821	56,756	(9.7)	(12.4)	(12.3)	(14.2)	(18.4)	(16.5)
113		20	220	40	36,554	51,869	43,935	62,710	75,193	65,010	78,052	83,219	80,058
114		20	220	40	38,261	48,449	46,167	67,447	67,081	65,258	80,359	(6.9)	83,920
115		20	220	40	52,815	72,022	69,920	(6.7)	(11.1)	(8.4)	(12.3)	(17.7)	(16.8)
116		20	220	40	48,210	(8.7)	(4.5)	(15)	(23.2)	(19.2)	(21.4)	(28.7)	(26.4)
117		20	220	40	45,426	(4.7)	71,231	69,236	(12.2)	(7.2)	(17.4)	(24.8)	(21.1)
118	r14	20	220	100	84,466	(9.8)	85,048	(4.4)	(9.8)	(5.2)	(7.3)	(13.8)	(9.5)
119		20	220	100	(3.9)	(14.7)	(3.6)	(10.7)	(17.8)	(11.6)	(14.3)	(27.8)	(17.7)
120		20	220	100	(5.2)	(11.1)	(9.6)	(100)	(100)	(100)	(100)	(100)	(100)
121		20	220	100	(10.9)	(26)	(14.7)	(33.9)	(100)	(100)	(38.4)	(100)	(100)
122		20	220	100	(21.5)	(100)	(100)	(21.3)	(100)	(27.4)	(27.3)	(100)	(32.6)
123		20	220	100	(6.3)	(16)	(8.1)	(9.4)	(22.6)	(13.5)	(17.2)	(32.7)	(21.2)
124		20	220	100	(100)	(100)	(100)	(25.2)	(36.2)	(31.3)	(100)	(100)	(100)
125		20	220	100	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
126		20	220	100	(1.7)	(9.6)	(1.5)	(9.7)	(20.7)	(16.3)	(15.8)	(38.3)	(24.2)

* () indicates the gap at the time limit (24 hours)

Table B.3: CPU Time (s) of Variants of BBC Algorithm on "r10-r18" Sets (continued)

#	Set	$\|N\|$	$\|A\|$	$\|K\|$	$\|S\|=2$			$\|S\|=3$			$\|S\|=4$		
					BBC2	BBC3	BBC5	BBC2	BBC3	BBC5	BBC2	BBC3	BBC5
127	r15	20	220	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
128		20	220	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
129		20	220	200	(52.1)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
130		20	220	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
131		20	220	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
132		20	220	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
133		20	220	200	(32.2)	(100)	(48.2)	(17.3)	(100)	(100)	(100)	(100)	(100)
134		20	220	200	(20.7)	(33)	(26.1)	(30.4)	(100)	(38.5)	(100)	(100)	(100)
135		20	220	200	(24.1)	(38)	(31.4)	(34.4)	(100)	(39.4)	(100)	(100)	(100)
136	r16	20	314	40	32,129	41,964	38,287	53,525	66,086	63,643	65,286	83,205	77,614
137		20	314	40	36,011	53,860	51,528	56,705	74,485	70,239	73,897	(18.3)	(14.6)
138		20	314	40	46,403	58,601	55,054	60,059	(10.3)	(8.8)	(13.3)	(19.3)	(16.2)
139		20	314	40	61,930	66,893	65,106	(13.5)	(18.4)	(14.2)	(16.5)	(22.6)	(19.4)
140		20	314	40	48,703	59,621	47,479	61,055	(9.2)	64,037	72,637	(9.6)	78,356
141		20	314	40	77,614	82,896	79,040	(8.2)	(16.2)	(10.2)	(13.8)	(20.6)	(17.3)
142		20	314	40	67,043	80,677	63,647	(10.9)	(18)	(13.2)	(16.5)	(24.9)	(21.5)
143		20	314	40	73,190	86,317	72,483	(19.8)	(26.7)	(23.7)	(23.6)	(28.6)	(25.5)
144		20	314	40	59,666	(9.1)	(7.4)	(17.6)	(18.9)	(13)	(15.5)	(19.4)	(15.4)
145	r17	20	318	100	85,679	(16.3)	(11.5)	(7.5)	(18.3)	(14.5)	(11.3)	(100)	(20.3)
146		20	318	100	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
147		20	318	100	(23)	(100)	(100)	(27.6)	(100)	(100)	(33.2)	(100)	(100)
148		20	318	100	(25.3)	(28.1)	(25.1)	(30.1)	(100)	(29.8)	(36.8)	(100)	(43.6)
149		20	318	100	(35.1)	(100)	(37.8)	(39.6)	(100)	(39.3)	(100)	(100)	(100)
150		20	318	100	(26.1)	(100)	(100)	(30.6)	(100)	(100)	(38.8)	(100)	(100)
151		20	318	100	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
152		20	318	100	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
153		20	318	100	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
154	r18	20	315	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
155		20	315	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
156		20	315	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
157		20	315	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
158		20	315	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
159		20	315	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
160		20	315	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
161		20	315	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)
162		20	315	200	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)	(100)

* () indicates the gap at the time limit (24 hours)

Table B.4: CPU Time (s) of BBC2 for Uncertain Demand and Number of Interdictions

Set	$\|\Omega\| = 16$						$\|\Omega\| = 32$						$\|\Omega\| = 64$					
	$\|S\| = 2$		$\|S\| = 3$		$\|S\| = 4$		$\|S\| = 2$		$\|S\| = 3$		$\|S\| = 4$		$\|S\| = 2$		$\|S\| = 3$		$\|S\| = 4$	
	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap
r04	100	0	253	0	291	0	52	0	293	0	369	0	147	0	660	0	649	0
	903	0	4,692	0	4,740	0	117	0	364	0	18,872	0	161	0	461	0	5,828	0
	302	0	12,622	0	27,556	0	217	0	1,586	0	1,534	0	532	0	6,975	0	1,416	0
	174	0	162	0	117	0	543	0	394	0	185	0	1,252	0	709	0	233	0
	1,531	0	1,567	0	897	0	3,716	0	1,638	0	8,560	0	9,989	0	17,312	0	18,043	0
	1,980	0	2,356	0	4,500	0	1,331	0	4,112	0	6,482	0	4,105	0	37,898	0	19,502	0
	12	0	12	0	14	0	38	0	24	0	32	0	155	0	144	0	136	0
	66	0	37	0	42	0	153	0	104	0	109	0	352	0	513	0	299	0
	208	0	132	0	116	0	215	0	391	0	244	0	740	0	995	0	606	0
r05	2,272	0	3,122	0	2,218	0	3,630	0	3,938	0	2,144	0	4,891	0	4,636	0	2,582	0
	21,001	0	49,484	0	10,466	0	34,630	0	86,400	3.1	34,630	0	86,400	6.4	86,400	8.5	86,400	5.9
	49,639	0	86,400	5.5	86,400	9.5	86,400	6.3	86,400	12.4	86,400	13.8	86,400	15.4	86,400	20.0	86,400	22.3
	831	0	1,270	0	1,046	0	3,082	0	1,265	0	1,106	0	3,655	0	1,103	0	1,713	0
	23,383	0	10,687	0	13,536	0	65,814	0	38,884	0	78,051	0	86,400	3.5	70,938	0	86,400	2.2
	86,400	2.6	86,400	4.0	86,400	5.0	86,400	10.8	86,400	9.9	86,400	11.2	86,400	34.2	86,400	25.8	86,400	29.6
	14	0	18	0	22	0	28	0	28	0	36	0	71	0	71	0	81	0
	45	0	65	0	53	0	90	0	115	0	149	0	581	0	523	0	462	0
	83	0	72	0	76	0	171	0	163	0	178	0	567	0	682	0	615	0
r06	1,814	0	8,486	0	4,386	0	11,387	0	23,003	0	6,660	0	21,557	0	68,702	0	41,603	0
	86,400	3.1	86,400	6.6	86,400	8.3	86,400	6.5	86,400	15.8	86,400	14.0	86,400	9.2	86,400	18.1	86,400	27.2
	86,400	7.3	86,400	14.7	86,400	18.1	86,400	16.3	86,400	28.4	86,400	31.6	86,400	11.4	86,400	32.6	86,400	36.9
	3,129	0	3,873	0	3,477	0	12,406	0	21,375	0	12,993	0	56,036	0	67,541	0	85,170	0
	86,400	2.9	86,400	7.1	86,400	6.6	86,400	8.7	86,400	11.9	86,400	15.7	86,400	10.7	86,400	15.3	86,400	17.1
	86,400	6.4	86,400	13.3	86,400	13.7	86,400	12.7	86,400	20.7	86,400	15.4	86,400	10.1	86,400	22.6	86,400	22.4
	19	0	24	0	39	0	25	0	33	0	53	0	46	0	48	0	70	0
	72	0	84	0	122	0	103	0	125	0	190	0	320	0	365	0	362	0
	14,062	0	45,776	0	86,400	1.9	86,400	2.5	86,400	2.9	86,400	7.0	86,400	8.0	86,400	6.0	86,400	30.2
r07	56	0	1,908	0	86,400	1.2	71	0	3,363	0	86,400	2.0	73	0	5,187	0	86,400	2.1
	43	0	3,657	0	86,400	6.1	54	0	17,876	0	86,400	8.8	70	0	49,251	0	86,400	11.3
	53	0	19,114	0	86,400	11.5	76	0	18,015	0	86,400	12.4	80	0	86,400	2.2	86,400	17.5
	60	0	8,639	0	26,299	0	84	0	66,495	0	35,190	0	95	0	86,400	0.8	86,400	1.0
	62	0	86,400	1.1	86,400	10.4	132	0	86,400	1.3	86,400	8.0	339	0	86,400	2.5	86,400	10.5
	65	0	86,400	2.9	86,400	11.8	141	0	86,400	2.0	86,400	14.4	268	0	86,400	4.0	86,400	16.4
	159	0	170	0	169	0	310	0	514	0	499	0	532	0	575	0	854	0
	3,677	0	2,947	0	4,401	0	26,054	0	23,274	0	20,791	0	23,364	0	29,665	0	30,103	0
	7,821	0	10,879	0	9,645	0	41,583	0	66,909	0	43,403	0	54,813	0	71,027	0	86,400	1.1
r08	695	0	86,400	2.2	86,400	8	586	0	86,400	2.4	86,400	8.2	684	0	86,400	3.5	86,400	10.0
	2,418	0	86,400	10.7	86,400	30.0	3,185	0	86,400	12.3	86,400	28.4	9,329	0	86,400	12.0	86,400	35.8
	1,294	0	86,400	14.6	86,400	30.3	1,531	0	86,400	13.5	86,400	30.7	4,379	0	86,400	14.7	86,400	35.6
	5,333	0	86,400	2.4	86,400	5.9	4,218	0	86,400	2.9	86,400	5.6	28,956	0	86,400	2.4	86,400	12.6
	10,829	0	86,400	15.4	86,400	32.0	15,928	0	86,400	16.0	86,400	30.7	56,127	0	86,400	19.8	86,400	40.7
	4,057	0	86,400	16.8	86,400	30.6	21,422	0	86,400	20.4	86,400	32.3	67,081	0	86,400	26.2	86,400	36.4
	2,836	0	2,239	0	2,877	0	5,643	0	3,870	0	2,381	0	33,591	0	9,979	0	18,250	0
	41,357	0	78,196	0	86,400	2.4	86,400	3.9	86,400	2.3	86,400	5.8	86,400	13.0	86,400	14.4	86,400	16.6
	76,853	0	86,400	3.7	86,400	5.7	86,400	4.3	86,400	8.6	86,400	10.2	86,400	0.1	86,400	13.8	86,400	21.3
r09	86,400	1.3	86,400	5.2	86,400	100.0	86,400	2.3	86,400	8.7	86,400	46.9	86,400	3.0	86,400	14.3	86,400	100
	86,400	4.5	86,400	26.7	86,400	48.3	86,400	9.6	86,400	34.7	86,400	100	86,400	15.5	86,400	41.9	86,400	55.1
	86,400	10.3	86,400	29.4	86,400	44.0	86,400	12.3	86,400	35.1	86,400	51.3	86,400	18.4	86,400	42.8	86,400	49.7
	42,605	0	86,400	4.7	86,400	100.0	27,843	0	86,400	7.3	86,400	100	86,400	3.4	86,400	10.7	86,400	100
	86,400	16.7	86,400	35.7	86,400	45.0	86,400	21.9	86,400	41.9	86,400	43.9	86,400	33.8	86,400	39.0	86,400	44.8
	86,400	21.2	86,400	38.8	86,400	41.6	86,400	30.1	86,400	39.1	86,400	45.8	86,400	35.5	86,400	40.2	86,400	48.7
	1,958	0	1,920	0	1,073	0	3,709	0	3,776	0	2,008	0	16,214	0	14,743	0	10,339	0
	25,900	0	20,015	0	24,545	0	59,098	0	48,102	0	31,768	0	86,400	1.1	86,400	1.9	86,400	3.2
	58,683	0	49,099	0	44,406	0	78,513	0	86,400	1.0	86,400		86,400	4.9	86,400	8.7	86,400	2.5
Average	23,564	1.4	39,963	4.8	45,065	11.6	30,325	2.7	46,482	6.6	48,915	13.1	37,829	4.4	53,332	8.6	55,624	16.1

Bibliography

Afshari Rad, M. and H. T. Kakhki (2017). Two extended formulations for cardinality maximum flow network interdiction problem. *Networks 69*(4), 367–377.

Agarwal, Y. K. and P. Venkateshan (2019). New Valid Inequalities for the Optimal Communication Spanning Tree Problem. *INFORMS Journal on Computing 31*(2), 268–284.

Ahuja, R. and V. Murty (1987). Exact and heuristic algorithms for the optimum communication spanning tree problem. *Transportation Science 21*(3), 163–170.

Akgün, İ., B. Ç. Tansel, and R. K. Wood (2011). The multi-terminal maximum-flow network-interdiction problem. *European Journal of Operational Research 211*(2), 241–251.

Aksen, D., S. Ş. Akca, and N. Aras (2014). A bilevel partial interdiction problem with capacitated facilities and demand outsourcing. *Computers & Operations Research 41*, 346–358.

Aksen, D. and N. Aras (2012). A bilevel fixed charge location model for facilities under imminent attack. *Computers & Operations Research 39*(7), 1364–1381.

Alderson, D. L., G. G. Brown, W. M. Carlyle, and R. K. Wood (2011). Solving defender-attacker-defender models for infrastructure defense. Technical report, Naval Postgraduate School Monterey CA Dept Of Operations Research.

Aldrighetti, R., D. Battini, D. Ivanov, and I. Zennaro (2021). Costs of resilience and disruptions in supply chain network design models: a review and future research directions. *International Journal of Production Economics 235*, 108103.

An, Y., B. Zeng, Y. Zhang, and L. Zhao (2014). Reliable p-median facility location problem: two-stage robust models and algorithms. *Transportation Research Part B: Methodological 64*, 54–72.

Assimakopoulos, N. (1987). A network interdiction model for hospital infection control. *Computers in biology and medicine 17*(6), 413–422.

Azad, N. and E. Hassini (2019). A Benders decomposition method for designing reliable supply chain networks accounting for multimitigation strategies and demand losses. *Transportation Science 53*(5), 1287–1312.

Azizi, N., S. Chauhan, S. Salhi, and N. Vidyarthi (2016). The impact of hub failure in hub-and-spoke networks: Mathematical formulations and solution techniques. *Computers & Operations Research 65*, 174–188.

Baggio, A., M. Carvalho, A. Lodi, and A. Tramontani (2021). Multilevel approaches for the critical node problem. *Operations Research 69*(2), 486–508.

Baghalian, A., S. Rezapour, and R. Z. Farahani (2013). Robust supply chain network design with service level against disruptions and demand uncertainties: A real-life case. *European Journal of Operational Research 227*(1), 199–215.

Balakrishnan, A., G. Li, and P. Mirchandani (2017). Optimal network design with end-to-end service requirements. *Operations Research 65*(3), 729–750.

Bazgan, C., S. Toubaline, and D. Vanderpooten (2012). Efficient determination of the k most vital edges for the minimum spanning tree problem. *Computers & Operations Research 39*(11), 2888–2898.

Bazlamaçcı, C. F. and K. S. Hindi (2001). Minimum-weight spanning tree algorithms a survey and empirical study. *Computers & Operations Research 28*(8), 767–785.

Beck, Y., I. Ljubić, and M. Schmidt (2023). A survey on bilevel optimization under uncertainty. *European Journal of Operational Research*.

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

Bhuiyan, T. H., H. R. Medal, A. K. Nandi, and M. Halappanavar (2021). Risk-averse bi-level stochastic network interdiction model for cyber-security risk management. *International Journal of Critical Infrastructure Protection 32*, 100408.

Bhuiyan, T. H., A. K. Nandi, H. Medal, and M. Halappanavar (2016). Minimizing expected maximum risk from cyber-attacks with probabilistic attack success. In *2016 IEEE Symposium on Technologies for Homeland Security (HST)*, pp. 1–6. IEEE.

Blom, D., C. Hojny, and B. Smeulders (2024). Cutting plane approaches for the robust kidney exchange problem. *Computers & Operations Research 162*, 106470.

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

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

Brown, G., M. Carlyle, J. Salmerón, and K. Wood (2006). Defending critical infrastructure. *Interfaces 36*(6), 530–544.

Cappanera, P. and M. P. Scaparra (2011). Optimal allocation of protective resources in shortest-path networks. *Transportation Science 45*(1), 64–80.

Caprara, A., M. Carvalho, A. Lodi, and G. J. Woeginger (2016). Bilevel knapsack with interdiction constraints. *INFORMS Journal on Computing 28*(2), 319–333.

Carøe, C. C. and R. Schultz (1999). Dual decomposition in stochastic integer programming. *Operations Research Letters 24*(1-2), 37–45.

Chang, K.-H., Y.-C. Chiang, and T.-Y. Chang (2024). Simultaneous location and vehicle fleet sizing of relief goods distribution centers and vehicle routing for post-disaster logistics. *Computers & Operations Research 161*, 106404.

Che, A., J. Li, F. Chu, and C. Chu (2024). Optimizing emergency supply pre-positioning for disaster relief: A two-stage distributionally robust approach. *Computers & Operations Research*, 106607.

Chen, R. L.-Y., A. Cohn, and A. Pinar (2011). An implicit optimization approach for survivable network design. In *2011 IEEE Network Science Workshop*, pp. 180–187. IEEE.

Cheng, C., Y. Adulyasak, and L.-M. Rousseau (2021). Robust facility location under disruptions. *INFORMS Journal on Optimization 3*(3), 298–314.

Cheng, C., M. Qi, Y. Zhang, and L.-M. Rousseau (2018). A two-stage robust approach for the reliable logistics network design problem. *Transportation Research Part B: Methodological 111*, 185–202.

CNBC (2022). Toyota to cut global production plan by 100,000 in June.

Collado, R., S. Meisel, and L. Priekule (2017). Risk-averse stochastic path detection. *European Journal of Operational Research 260*(1), 195–211.

Contardo, C. and J. A. Sefair (2022). A progressive approximation approach for the exact solution of sparse large-scale binary interdiction games. *INFORMS Journal on Computing 34*(2), 890–908.

Contreras, I., J.-F. Cordeau, and G. Laporte (2011). Benders decomposition for large-scale uncapacitated hub location. *Operations Research 59*(6), 1477–1490.

Contreras, I., E. Fernández, and A. Marín (2010). Lagrangean bounds for the optimum communication spanning tree problem. *Top 18*(1), 140–157.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein (2022). *Introduction to algorithms*. MIT press.

Cormican, K., D. Morton, and R. Wood (1998). Stochastic network interdiction. *Operations Research 46*(2), 184–197.

Costa, A., D. Georgiadis, T. S. Ng, and M. Sim (2018). An optimization model for power grid fortification to maximize attack immunity. *International Journal of Electrical Power & Energy Systems 99*, 594–602.

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

Costa, A. M., J.-F. Cordeau, and B. Gendron (2009). Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications 42*(3), 371–392.

Costa, A. M., P. M. França, and C. Lyra Filho (2011). Two-level network design with intermediate facilities: an application to electrical distribution systems. *Omega 39*(1), 3–13.

Couedelo, A. (2018). Robust design of distribution networks considering worst case interdictions. Master's thesis, Concordia University.

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

Crainic, T. G., A. Frangioni, and B. Gendron (2001). Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics 112*(1-3), 73–99.

Crainic, T. G., M. Gendreau, and B. Gendron (2021). *Network design with applications to transportation and logistics*. Springer Nature.

Crainic, T. G., M. Hewitt, F. Maggioni, and W. Rei (2021). Partial Benders decomposition: general methodology and application to stochastic network design. *Transportation Science 55*(2), 414–435.

Cui, T., Y. Ouyang, and Z.-J. M. Shen (2010). Reliable facility location design under the risk of disruptions. *Operations Research 58*(4-part-1), 998–1011.

DeNegre, S. T. and T. K. Ralphs (2009). A branch-and-cut algorithm for integer bilevel linear programs. In *Operations research and cyber-infrastructure*, pp. 65–78. Springer.

Feremans, C., M. Labbé, and G. Laporte (2003). Generalized network design problems. *European Journal of Operational Research 148*(1), 1–13.

Fischer, T. and P. Merz (2007). A memetic algorithm for the optimum communication spanning tree problem. In *International Workshop on Hybrid Metaheuristics*, pp. 170–184. Springer.

Fischetti, M., I. Ljubić, M. Monaci, and M. Sinnl (2019). Interdiction games and monotonicity, with application to knapsack problems. *INFORMS Journal on Computing 31*(2), 390–410.

Forghani, A., F. Dehghanian, M. Salari, and Y. Ghiami (2020). A bi-level model and solution methods for partial interdiction problem on capacitated hierarchical facilities. *Computers & Operations Research 114*, 104831.

Fragkos, I., J.-F. Cordeau, and R. Jans (2017). *The multi-period multi-commodity network design problem*. CIRRELT, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport.

Fragkos, I., J.-F. Cordeau, and R. Jans (2021). Decomposition methods for large-scale network expansion problems. *Transportation Research Part B: Methodological 144*, 60–80.

Frederickson, G. N. and R. Solis-Oba (1999). Increasing the weight of minimum spanning trees. *Journal of Algorithms 33*(2), 244–266.

Garg, M. and J. C. Smith (2008). Models and algorithms for the design of survivable multicommodity flow networks with general failure scenarios. *Omega 36*(6), 1057–1071.

Gendron, B., T. G. Crainic, and A. Frangioni (1999). Multicommodity capacitated network design. In *Telecommunications Network Planning*, pp. 1–19. Springer.

Ghaffarinasab, N. and R. Atayi (2018). An implicit enumeration algorithm for the hub interdiction median problem with fortification. *European Journal of Operational Research 267*(1), 23–39.

Ghaffarinasab, N. and A. Motallebzadeh (2018). Hub interdiction problem variants: Models and metaheuristic solution algorithms. *European Journal of Operational Research 267*(2), 496–512.

Ghare, P., D. C. Montgomery, and W. Turner (1971). Optimal interdiction policy for a flow network. *Naval Research Logistics Quarterly 18*(1), 37–45.

Ghorbani-Renani, N., A. D. González, K. Barker, and N. Morshedlou (2020). Protection-interdiction-restoration: Tri-level optimization for enhancing interdependent network resilience. *Reliability Engineering & System Safety 199*, 106907.

Ghosh, S. and P. Jaillet (2022). An iterative security game for computing robust and adaptive network flows. *Computers & Operations Research 138*, 105558.

Gicquel, C., S. Vanier, and A. Papadimitriou (2022). Optimal deployment of virtual network functions for securing telecommunication networks against distributed denial of service attacks: a robust optimization approach. *Computers & Operations Research 146*, 105890.

Graham, R. L. and P. Hell (1985). On the history of the minimum spanning tree problem. *Annals of the History of Computing 7*(1), 43–57.

Gudapati, N. V., E. Malaguti, and M. Monaci (2022). Network design with service requirements: Scaling-up the size of solvable problems. *INFORMS Journal on Computing 34*(5), 2571–2582.

Guo, G., G. Hackebeil, S. M. Ryan, J.-P. Watson, and D. L. Woodruff (2015). Integration of progressive hedging and dual decomposition in stochastic integer programs. *Operations Research Letters 43*(3), 311–316.

Hemmati, M. and J. C. Smith (2016). A mixed-integer bilevel programming approach for a competitive prioritized set covering problem. *Discrete Optimization 20*, 105–134.

Hemmecke, R., R. Schultz, and D. L. Woodruff (2003). Interdicting stochastic networks with binary interdiction effort. In *Network Interdiction and Stochastic Integer Programming*, pp. 69–84. Springer.

Hendricks, K. B. and V. R. Singhal (2005). An empirical analysis of the effect of supply chain disruptions on long-run stock price performance and equity risk of the firm. *Production and Operations Management 14*(1), 35–52.

Hien, L. T. K., M. Sim, and H. Xu (2020). Mitigating interdiction risk with fortification. *Operations Research 68*(2), 348–362.

Holzmann, T. and J. C. Smith (2021). The shortest path interdiction problem with randomized interdiction strategies: Complexity and algorithms. *Operations Research 69*(1), 82–99.

Hsu, L.-H., R.-H. Jan, Y.-C. Lee, C.-N. Hung, M.-S. Chern, et al. (1991). Finding the most vital edge with respect to minimum spanning tree in weighted graphs. *Information Processing Letters 39*(5), 277–281.

Hu, T. C. (1974). Optimum communication spanning trees. *SIAM Journal on Computing 3*(3), 188–195.

Hunt, K. and J. Zhuang (2024). A review of attacker-defender games: Current state and paths forward. *European Journal of Operational Research 313*(2), 401–417.

Israeli, E. and R. K. Wood (2002). Shortest-path network interdiction. *Networks 40*(2), 97–111.

Ivanov, D., A. Dolgui, B. Sokolov, and M. Ivanova (2017). Literature review on disruption recovery in the supply chain. *International Journal of Production Research 55*(20), 6158–6174.

Jabarzare, Z., H. Zolfagharinia, and M. Najafi (2020). Dynamic interdiction networks with applications in illicit supply chains. *Omega 96*, 102069.

Jabbarzadeh, A., B. Fahimnia, J.-B. Sheu, and H. S. Moghadam (2016). Designing a supply chain resilient to major disruptions and supply/demand interruptions. *Transportation Research Part B: Methodological 94*, 121–149.

Janjarassuk, U. and J. Linderoth (2008). Reformulation and sampling to solve a stochastic network interdiction problem. *Networks 52*(3), 120–132.

Jiang, J. and X. Liu (2018). Multi-objective Stackelberg game model for water supply networks against interdictions with incomplete information. *European Journal of Operational Research 266*(3), 920–933.

Jin, J. G., L. Lu, L. Sun, and J. Yin (2015). Optimal allocation of protective resources in urban rail transit networks against intentional attacks. *Transportation Research Part E: Logistics and Transportation Review 84*, 73–87.

Kerivin, H. and A. R. Mahjoub (2005). Design of survivable networks: A survey. *Networks: An International Journal 46*(1), 1–21.

Kleinert, T., M. Labbé, I. Ljubić, and M. Schmidt (2021). A survey on mixed-integer programming techniques in bilevel optimization. *EURO Journal on Computational Optimization 9*, 100007.

Klibi, W., A. Martel, and A. Guitouni (2010). The design of robust value-creating supply chain networks: a critical review. *European Journal of Operational Research 203*(2), 283–293.

Lanza, G., T. G. Crainic, W. Rei, and N. Ricciardi (2021). Scheduled service network design with quality targets and stochastic travel times. *European Journal of Operational Research 288*(1), 30–46.

Laporte, G., J. A. Mesa, and F. Perea (2010). A game theoretic framework for the robust railway transit network design problem. *Transportation Research Part B: Methodological 44*(4), 447–459.

Lei, X., S. Shen, and Y. Song (2018). Stochastic maximum flow interdiction problems under heterogeneous risk preferences. *Computers & Operations Research 90*, 97–109.

Leitner, M., I. Ljubić, M. Monaci, M. Sinnl, and K. Tanınmış (2022). An exact method for binary fortification games. *European Journal of Operational Research*.

Li, Q., B. Zeng, and A. Savachkin (2013). Reliable facility location design under disruptions. *Computers & Operations Research 40*(4), 901–909.

Li, X. and Y. Ouyang (2010). A continuum approximation approach to reliable facility location design under correlated probabilistic disruptions. *Transportation Research Part B: Methodological 44*(4), 535–548.

Liberatore, F., M. Scaparra, and M. Daskin (2011). Analysis of facility protection strategies against an uncertain number of attacks: The stochastic R-interdiction median problem with fortification. *Computers & Operations Research 38*(1), 357–366.

Liberatore, F., M. P. Scaparra, and M. S. Daskin (2012). Hedging against disruptions with ripple effects in location analysis. *Omega 40*(1), 21–30.

Lim, C. and J. C. Smith (2007). Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IIE Transactions 39*(1), 15–26.

Lim, M., M. S. Daskin, A. Bassamboo, and S. Chopra (2010). A facility reliability problem: Formulation, properties, and algorithm. *Naval Research Logistics 57*(1), 58–70.

Lin, K.-C. and M.-S. Chern (1993). The most vital edges in the minimum spanning tree problem. *Information Processing Letters 45*(1), 25–31.

Lin, L. and M. Gen (2006). Node-based genetic algorithm for communication spanning tree problem. *IEICE Transactions on Communications 89*(4), 1091–1098.

Linhares, A. and C. Swamy (2017). Improved algorithms for mst and metric-tsp interdiction. *arXiv preprint arXiv:1706.00034*.

Losada, C., M. P. Scaparra, R. L. Church, and M. S. Daskin (2012). The stochastic interdiction median problem with disruption intensity levels. *Annals of Operations Research 201*(1), 345–365.

Lozano, L. and J. C. Smith (2017). A backward sampling framework for interdiction problems with fortification. *INFORMS Journal on Computing 29*(1), 123–139.

Lozano, L., J. C. Smith, and M. E. Kurz (2017). Solving the traveling salesman problem with interdiction and fortification. *Operations Research Letters 45*(3), 210–216.

Magnanti, T. L. and L. A. Wolsey (1995). Optimal trees. *Handbooks in Operations Research and Management Science 7*, 503–615.

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

Märkert, A. and R. Gollmer (2008). User's guide to ddsip–ac package for the dual decomposition of two-stage stochastic programs with mixed-integer recourse. *Department of Mathematics, University of Duisburg-Essen, Duisburg*.

McMasters, A. W. and T. M. Mustin (1970). Optimal interdiction of a supply network. *Naval Research Logistics Quarterly 17*(3), 261–268.

Medal, H. R., E. A. Pohl, and M. D. Rossetti (2014). A multi-objective integrated facility location-hardening model: Analyzing the pre-and post-disruption tradeoff. *European Journal of Operational Research 237*(1), 257–270.

Mohammadi, M., P. Jula, and R. Tavakkoli-Moghaddam (2017). Design of a reliable multi-modal multi-commodity model for hazardous materials transportation under uncertainty. *European Journal of Operational Research 257*(3), 792–809.

Morton, D. (2010). Stochastic network interdiction. *Wiley Encyclopedia of Operations Research and Management Science*.

Morton, D. P., F. Pan, and K. J. Saeger (2007). Models for nuclear smuggling interdiction. *IIE Transactions 39*(1), 3–14.

Mota, C. L. (2016). *The Optimum Communication Spanning Tree Problem: Properties, Models and Algorithms*. Ph. D. thesis, Universitat Politècnica de Catalunya (UPC).

Nandi, A. K., H. R. Medal, and S. Vadlamani (2016). Interdicting attack graphs to protect organizations from cyber attacks: A bi-level defender–attacker model. *Computers & Operations Research 75*, 118–131.

Naoum-Sawaya, J. and B. Ghaddar (2017). Cutting plane approach for the maximum flow interdiction problem. *Journal of the Operational Research Society 68*(12), 1553–1569.

Nguyen, D. H. and J. C. Smith (2022). Network interdiction with asymmetric cost uncertainty. *European Journal of Operational Research 297*(1), 239–251.

O'Hanley, J. R. and R. L. Church (2011). Designing robust coverage networks to hedge against worst-case facility losses. *European Journal of Operational Research 209*(1), 23–36.

Pan, F., W. S. Charlton, and D. P. Morton (2003). A stochastic program for interdicting smuggled nuclear material. In *Network Interdiction and Stochastic Integer Programming*, pp. 1–19. Springer.

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

Parajuli, A., O. Kuzgunkaya, and N. Vidyarthi (2017). Responsive contingency planning of capacitated supply networks under disruption risks. *Transportation Research Part E: Logistics and Transportation Review 102*, 13–37.

Parajuli, A., O. Kuzgunkaya, and N. Vidyarthi (2021). The impact of congestion on protection decisions in supply networks under disruptions. *Transportation Research Part E: Logistics and Transportation Review 145*, 102166.

Paraskevopoulos, D. C., T. Bektaş, T. G. Crainic, and C. N. Potts (2016). A cycle-based evolutionary algorithm for the fixed-charge capacitated multi-commodity network design problem. *European Journal of Operational Research 253*(2), 265–279.

Parvaresh, F., S. H. Golpayegany, S. M. Husseini, and B. Karimi (2013). Solving the p-hub median problem under intentional disruptions using simulated annealing. *Networks and Spatial Economics 13*(4), 445–470.

Parvaresh, F., S. M. Husseini, S. H. Golpayegany, and B. Karimi (2014). Hub network design problem in the presence of disruptions. *Journal of Intelligent Manufacturing 25*(4), 755–774.

Pay, B. S., J. R. Merrick, and Y. Song (2019). Stochastic network interdiction with incomplete preference. *Networks 73*(1), 3–22.

Peng, P., L. V. Snyder, A. Lim, and Z. Liu (2011). Reliable logistics networks design with facility disruptions. *Transportation Research Part B: Methodological 45*(8), 1190–1211.

Prim, R. C. (1957). Shortest connection networks and some generalizations. *The Bell System Technical Journal 36*(6), 1389–1401.

Rad, M. A. and H. T. Kakhki (2013). Maximum dynamic network flow interdiction problem: New formulation and solution procedures. *Computers & Industrial Engineering 65*(4), 531–536.

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

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

Ramamoorthy, P., S. Jayaswal, A. Sinha, and N. Vidyarthi (2018). Multiple allocation hub interdiction and protection problems: Model formulations and solution approaches. *European Journal of Operational Research 270*(1), 230–245.

Ratliff, H. D., G. T. Sicilia, and S. Lubore (1975). Finding the n most vital links in flow networks. *Management Science 21*(5), 531–539.

Rothlauf, F. (2009). On optimal solutions for the optimal communication spanning tree problem. *Operations Research 57*(2), 413–425.

Sadati, M. E. H., D. Aksen, and N. Aras (2020). A trilevel r-interdiction selective multi-depot vehicle routing problem with depot protection. *Computers & Operations Research 123*, 104996.

Salmeron, J., K. Wood, and R. Baldick (2004). Analysis of electric grid security under terrorist threat. *IEEE Transactions on power systems 19*(2), 905–912.

Salmeron, J., K. Wood, and R. Baldick (2009). Worst-case interdiction analysis of large-scale electric power grids. *IEEE Transactions on Power Systems 24*(1), 96–104.

Sarayloo, F., T. G. Crainic, and W. Rei (2021). A learning-based matheuristic for stochastic multi-commodity network design. *INFORMS Journal on Computing 33*(2), 643–656.

Scaparra, M. P. and R. L. Church (2008a). A bilevel mixed-integer program for critical infrastructure protection planning. *Computers & Operations Research 35*(6), 1905–1923.

Scaparra, M. P. and R. L. Church (2008b). An exact solution approach for the interdiction median problem with fortification. *European Journal of Operational Research 189*(1), 76–92.

Shen, S. and J. C. Smith (2012). Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs. *Networks 60*(2), 103–119.

Shen, S., J. C. Smith, and R. Goli (2012). Exact interdiction models and algorithms for disconnecting networks via node deletions. *Discrete Optimization 9*(3), 172–188.

Sherman, E. (2020). 94% of the fortune 1000 are seeing coronavirus supply chain disruptions: Report.

Smith, J. C., C. Lim, and F. Sudargho (2007). Survivable network design under optimal and heuristic interdiction scenarios. *Journal of Global Optimization 38*(2), 181–199.

Smith, J. C. and Y. Song (2020). A survey of network interdiction models and algorithms. *European Journal of Operational Research 283*(3), 797–811.

Snyder, L. V., Z. Atan, P. Peng, Y. Rong, A. J. Schmitt, and B. Sinsoysal (2016). OR/MS models for supply chain disruptions: A review. *IIE Transactions 48*(2), 89–109.

Snyder, L. V. and M. S. Daskin (2005). Reliability models for facility location: the expected failure cost case. *Transportation Science 39*(3), 400–416.

Song, Y. and S. Shen (2016). Risk-averse shortest path interdiction. *INFORMS Journal on Computing 28*(3), 527–539.

Stackpole, B. (2022). Ripple effects from Russia-Ukraine war test global economies.

Taherkhani, G., S. A. Alumur, and M. Hosseini (2020). Benders decomposition for the profit maximizing capacitated hub location problem with multiple demand classes. *Transportation Science 54*(6), 1446–1470.

Tahernejad, S., T. K. Ralphs, and S. T. DeNegre (2020). A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation. *Mathematical Programming Computation 12*(4), 529–568.

Tanergüçlü, T., O. E. Karaşan, I. Akgün, and E. Karaşan (2019). Radio communications interdiction problem under deterministic and probabilistic jamming. *Computers & Operations Research 107*, 200–217.

Tilk, C. and S. Irnich (2018). Combined column-and-row-generation for the optimal communication spanning tree problem. *Computers & Operations Research 93*, 113–122.

Towle, E. and J. Luedtke (2018). New solution approaches for the maximum-reliability stochastic network interdiction problem. *Computational Management Science 15*(3), 455–477.

Ullmert, T., S. Ruzika, and A. Schöbel (2020). On the p-hub interdiction problem. *Computers & Operations Research 124*, 105056.

Wang, X., T. G. Crainic, and S. W. Wallace (2019). Stochastic network design for planning scheduled transportation services: The value of deterministic solutions. *INFORMS Journal on Computing 31*(1), 153–170.

Washburn, A. and K. Wood (1995). Two-person zero-sum games for network interdiction. *Operations research 43*(2), 243–251.

Wei, N. and J. L. Walteros (2022). Integer programming methods for solving binary interdiction games. *European Journal of Operational Research*.

Wei, N., J. L. Walteros, and F. M. Pajouh (2021). Integer programming formulations for minimum spanning tree interdiction. *INFORMS Journal on Computing 33*(4), 1461–1480.

Wood, R. K. (1993). Deterministic network interdiction. *Mathematical and Computer Modelling 17*(2), 1–18.

Wood, R. K. (2010). Bilevel network interdiction models: Formulations and solutions. *Wiley Encyclopedia of Operations Research and Management Science*.

Wu, X. and A. J. Conejo (2016). An efficient tri-level optimization model for electric grid defense planning. *IEEE Transactions on Power Systems 32*(4), 2984–2994.

Wu, Y., Z. Chen, J. Dang, Y. Chen, X. Zhao, and L. Zha (2022). Allocation of defensive and restorative resources in electric power system against consecutive multi-target attacks. *Reliability Engineering & System Safety 219*, 108199.

Wu, Y., Z. Chen, H. Gong, Q. Feng, Y. Chen, and H. Tang (2021). Defender–attacker–operator: Tri-level game-theoretic interdiction analysis of urban water distribution networks. *Reliability Engineering & System Safety 214*, 107703.

Xu, S., X. Zhang, L. Feng, and W. Yang (2020). Disruption risks in supply chain management: a literature review based on bibliometric analysis. *International Journal of Production Research 58*(11), 3508–3526.

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

Yu, G., W. B. Haskell, and Y. Liu (2017). Resilient facility location against the risk of disruptions. *Transportation Research Part B: Methodological 104*, 82–105.

Zare, M. H., J. S. Borrero, B. Zeng, and O. A. Prokopyev (2019). A note on linearized reformulations for a class of bilevel linear integer problems. *Annals of Operations Research 272*, 99–117.

Zenklusen, R. (2010). Matching interdiction. *Discrete Applied Mathematics 158*(15), 1676–1690.

Zenklusen, R. (2015). An o (1)-approximation for minimum spanning tree interdiction. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pp. 709–728. IEEE.

Zetina, C. A., I. Contreras, and J.-F. Cordeau (2019). Exact algorithms based on Benders decomposition for multicommodity uncapacitated fixed-charge network design. *Computers & Operations Research 111*, 311–324.

Zetina, C. A., I. Contreras, E. Fernandez, and C. Luna-Mota (2019). Solving the optimum communication spanning tree problem. *European Journal of Operational Research 273*(1), 108–117.

Zetina, C. A., I. Contreras, and S. Jayaswal (2021). An Exact Algorithm for Large-scale Non-convex Quadratic Facility Location. *arXiv preprint arXiv:2107.09746*.

Zhang, C. and J. E. Ramirez-Marquez (2013). Protecting critical infrastructures against intentional attacks: A two-stage game with incomplete information. *IIE Transactions 45*(3), 244–258.

Zhang, P. and N. Fan (2017). Analysis of budget for interdiction on multicommodity network flows. *Journal of Global Optimization 67*(3), 495–525.