MODELS AND ALGORITHMS FOR CRITICAL INFRASTRUCTURE PROTECTION WITH AN APPLICATION TO CYBERSECURITY PLANNING

by

Kaiyue Zheng

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Industrial and Systems Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2017

Date of final oral examination: 09/15/2017

The dissertation is to be approved by the following members of the Final Oral Committee:
Laura A. Albert, Associate Professor, Industrial and Systems Engineering
Vicki M. Bier, Professor, Industrial and Systems Engineering
James R. Luedtke, Associate Professor, Industrial and Systems Engineering
Xin Wang, Assistant Professor, Industrial and Systems Engineering
Victor Zavala, Assistant Professor, Chemical and Biological Engineering

For my parents.

Acknowledgments

This dissertation would not have been possible without the support of many people.

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Laura A. Albert, for teaching me how to become an independent researcher, giving me excellent advice on my research and writing, always showing me support and having confidence in my abilities, and giving me the freedom to explore topics I am interested in. With so many things I have learned from her, I would benefit for the rest of my professional life.

Moreover, I would like to thank Prof. James Luedtke for giving me excellent guidance and advice on my research, and offering a great course on integer programming, which is my favorite graduate course. I would also like to thank my other committee members, Prof. Vicki Bier, Prof. Xin Wang, and Prof. Victor Zavala for providing valuable feedback to my dissertation proposal.

I am also grateful for the Department of Industrial and Systems Engineering for awarding me the Richard S. and Harriet K. Fein Scholarship, the National Science Foundation for funding my dissertation research, and our collaborators from the Sandia National Laboratory for providing valuable information and feedback on our projects.

Special thanks are due to my Master degree advisor Prof. Mark S. Daskin for introducing me into operations research, giving me valuable advice on doing research, and encouraging me to pursue a PhD.

During the past four years, I was lucky to work in the most spacious student lab in the department full of great labmates. I am grateful for their friendship, support and advice during all these years. I want to thank: Hyemin Jeon, Namsuk Cho and Merve Bodur for their guidance through the first two years of my PhD, Soovin Yoon for always listening to me when I had a bad day, Eric DuBois for all the delicious cakes he made for us, Suzan Afacan for organizing so many fun events, and all my other fellow colleagues for enjoyable memories. I also want to thank all my friends in Madison and everywhere else in the world who are always there for me when I need them, especially I want to thank Yingzi Zhang for sharing so much wisdom and strength with me during the last and toughest year of my PhD. Last but not least, I want to thank my sweet kitty Mocha for constantly reminding me to rest in those late nights by climbing onto my laptop to nap.

Finally, I want to thank my parents for their unconditional love and support, who always encourage me to pursue everything I want in my life and have full confidence in me whenever I was confused. This dissertation is dedicated to them.

Contents

List of Tables

List of Figures

Abstract

Globalized supply chains bring enormous security risks to the information system, concerns of which have been widely and intensively expressed by both federal agencies and commercial organizations. In this dissertation, we investigate how to prioritize investment in mitigations to enhance the security of Information Technology (IT) infrastructure that balances cost and threat reduction. We propose an optimization framework consisting of mixed-integer programming and bilevel programming models and algorithms for protecting the critical infrastructure with a focus on cybersecurity planning and management.

First, we propose budgeted maximum multiple coverage (BMMC) models that identify a set of cost-effective mitigations to maximally reduce generic vulnerabilities in the IT infrastructure. We address the uncertainty regarding mitigation effectiveness, an inherent issue associated with cybersecurity planning, by proposing a stochastic expected-value BMMC model, denoted as EBMMC. Approximation algorithms with guaranteed approximation ratios are proposed to solve the models. Next, we extend EBMMC to three alternative models that provide solutions robust to worst case scenarios given uncertain mitigation effectiveness, including models that maximize the worst case coverage, minimize the worst case regret, and maximize the average coverage in some of the worst cases.

Furthermore, we address a more complicated and realistic problem when adversarial attackers are present. We investigate how to identify a best combination of cost-effective mitigations that maximally delay supply chain attacks when there exist multiple adversaries and uncertainty regarding mitigation effectiveness. We propose new Stackelberg game models that explicitly formulate the interaction between a defender and multiple attackers, including a deterministic interdiction model for delaying multiple adversarial projects (DIMAP) and a stochastic model variant (SIMAP) that incorporates uncertain delay times. We propose a Lagrangian heuristic that identifies near-optimal solutions efficiently. Finally, we propose a new exact algorithm for solving a particular critical infrastructure protection problem, the r-interdiction median problem with fortification (RIMF), which improves an existing algorithm in the literature and can be generalized to solve a broader range of facility interdiction and protection problems.

Chapter 1

Introduction and Motivation

1.1 The Importance of Cyber-Infrastructure Protection

Globalized supply chains bring efficiency and convenience to public and private sectors, while also introducing numerous risks to critical infrastructure. Information Technology (IT) infrastructure has enormous security risks due to its importance to national and economic security. According to a 2012 U.S. Government Accountability Office (GAO) report [79], threats to the Federal IT supply chains include, for example, installation of counterfeit hardware or software, disruption in the production or distribution of a critical product, reliance on malicious or unqualified third-party service providers, software security vulnerabilities in supplier systems, or poorly trained employees. Cybersecurity risks in federal supply chains have increased dramatically in recent years [77, 32]. According to a 2015 GAO report [78], the number of reported cyber incidents has increased $1,121\%$ between 2006 and 2014. The Director of National Intelligence [76] listed managing the "enormous vulnerabilities within the IT supply chain" as one of the U.S.'s top two cyber-security challenges [99]. Several reports from commercial organizations [91, 43] also echo these concerns. The White House proposed new policy directives for securing critical IT physical assets that reflect the awareness of the increasing role of cybersecurity in critical infrastructure [46, 47, 49], and for directing federal funding to develop mitigation approaches for global supply chain risk management [48].

IT supply chains for both government and industry rely on a complex network of third-party suppliers and vendors, contractors (sub-contractors), and affiliates. Lack of security in any component or link creates vulnerabilities that could be exploited by attackers. For example, in 2013, Target experienced one of the largest data breaches in retail history. In less than a month, over 40 million payment cards and roughly 110 customers' data were stolen due to malware insertion into its Point of Sale (POS) system. The initial intrusion to Target's main system is believed to be a third party vendor, a heating, ventilation and air conditioning (HVAC) company, where attackers exploited vulnerabilities in its remote diagnostics and stole network credentials [1]. Automated teller machine (ATM) malware attacks in recent years are other examples of supply chain attacks. In 2014, so-called Tyupkin malware affected ATMs from a major manufacturer running Microsoft Windows 32-bit operating system, which then spread to several countries including Russia, the United States, India, and China [3]. In 2015, the United States Office of Personnel Management (OPM) experienced a breach involving over 22 million federal employees. Investigation shows the attackers likely exploited vulnerabilities in an OPM contractor, a background-check provider, by stealing credentials and inserting malware [2].

Recently, the National Institute of Standards and Technology (NIST) published supply chain risk management practices for improving the security of federal information systems and organizations [73]. NIST proposes a threat scenario analysis framework that includes developing threat scenarios with exploits of vulnerabilities and identifying applicable controls. They demonstrate the use of this generic framework with examples such as telecommunications counterfeits and industrial espionage. By weighing mitigation costs to supply chain risks, this framework provides decision makers with cost-effective strategies to reduce risks from threats. Given the enormous risks in the system, NIST suggests that a more structured approach with well-defined goals and scope to represent threat scenarios should be proposed and used by decision makers, as well as a systematic way to deploy security mitigations that balances cost and threat reduction.

Preventive measures, i.e., mitigations, play an important role in protection planning. They are the key factors for improving trustworthiness of the IT infrastructure. Mitigations refer to any security controls that can be applied by subject matter experts (SMEs) to improve the difficulty of an attack exploit or reduce the consequences associated with the attack. Mitigation approaches can span process, people, and technology, since they are related to the functioning of IT supply chains [91]. Examples of mitigations include replacing physical components of the IT infrastructure that contains vulnerabilities, developing and deploying anti-virus software, establishing and implementing security policies or procedures, conducting regular security checks and training for employees. Mitigations "cover" vulnerabilities in the system by increasing the difficulty of attacks or reducing their consequences. Some mitigations may work in similar ways and may have overlapping capabilities regarding which vulnerabilities they cover. Deploying mitigations (controls) is costly, and federal or commercial organizations usually have a limited budget for selecting and deploying mitigations. Additionally, although efforts have been made towards assessing the risks contained in the system [46, 47], comprehensive security policies and mitigations have not been developed and implemented [78]. The uncertainty in mitigation effectiveness, coupled with the interrelationships between different mitigations, makes it very challenging to determine which combination of mitigations to implement to achieve maximum benefit given budget limitations.

Therefore, it is critical to develop an optimization framework that prioritizes investment in mitigations to enhance the security of the IT infrastructure that addresses i) attack modeling of threat scenarios; ii) selecting a best combination of cost-effective mitigations subject to budgetary requirements; iii) uncertainty embedded in mitigation effectiveness. Our research seeks to fill in this gap.

1.2 Literature Review

At present there are few optimization models that study how to mitigate cyber-risks through strategic planning or investment. Leskovec et al. [64] provide an optimization model for cost-effective detection of outbreaks in networks. The optimization model is general in that it can be used to locate sensors for detecting the spread of contaminants or ideas in a network, and it has application to social networks as well as cyber and physical networks. Afful-Dadzie and

Allen [4] propose a Markov decision process model for managing data-driven maintenance policies for protecting an IT network where data are scarce. Zhuo and Solak [105] propose a stochastic programming model to optimize a firm's cybersecurity budget in an investment portfolio, with a focus on addressing uncertainties in the effectiveness of the countermeasures. Nagurney et al. [72] use a game theoretic approach to address vulnerabilities in electronic transactions via the Internet. In the presence of cybersecurity vulnerabilities, the retailers aim to maximize their profits by determining the product transaction and also security levels, while the consumers adjust the product price they are willing to pay based on not only demand but also the average security in the supply chains. In a more recent paper, Nagurney and Shukla [71] investigate the value of information sharing for firms by evaluating non-cooperative and cooperative cybersercurity investments, respectively, by introducing both competitive and cooperative game theoretic models.

1.2.1 Attack Modeling

Attack modeling is an important first step of our problem. Attack modeling in the cybersecurity domain originates from the construction of attack trees or graphs [90, 65]. Attack graphs characterize attacks against the system, in which each path represents a likely attack consisting of a series of exploits to achieve attack goals. Constructing attack graphs is crucial in network vulnerability analysis. Based on the attack graph, SMEs can analyze and evaluate their defensive strategies according to their security needs [54], for example, where to locate intrusion detection systems, and how to select defensive measures. One main limitation of this analysis is that the graph size grows exponentially in the number of state variables. Industry practice usually constructs the attack trees by hand using red teams, which is tedious and error-prone. Therefore, a number of efforts in this area focus on automated attack graph generation [92]. There are a few game theoretic approaches along this line of research that explicitly formulate the interaction between the attacker and the defender. Many of these approaches are based on attack-defense trees, or attack-response trees, an extension of attack trees with countermeasures or response strategies that visualize the interaction between the two players [14, 62, 61]. Zonouc et al. [106] introduce a game theoretic intrusion response engine that identifies an instantaneous cost-effective response against adversarial attackers. They formulate the interaction between the attacker and the system administrator as a Stackelberg game and address the uncertainties in intrusion detection by converting attack-response trees into Markov Decision Process (MDP) models. Bistarelli et al. [15] study the strategic game between the attacker and the defender by structuring a game model, defining utility functions for the attacker and the defender, respectively, and studying the Nash equilibrium. For more game theoretic approaches, especially those that are based on Nash equilibrium as they are applied to network security, we refer the readers to Roy et al. [86].

Different from the real-time intrusion-detection decision analysis in the attack graph framework, this dissertation focuses on an infrastructure protection planning problem that is of interest to federal or industrial decision makers who are responsible for policies and investment in cyber infrastructure security. The very first step is to model supply chain attacks. A supply chain attack usually starts with an initial intrusion into the system's "weakest" component or link, followed by a series of exploits, and eventually it reaches its attack goal if successful. This is in similar spirit to finding a feasible "path" through a network, where nodes represent attack states and arcs correspond to transition of states fulfilled by attack activities. While an attack graph is a tool mainly used for detection and forensics by security analysts, it is a powerful tool to organize discovered vulnerabilities and visualize their dependencies (sequences), which provides a structured approach to represent attack scenarios. Therefore, the attack modeling framework in our research is developed based on the concept of attack graph with corresponding adaptation and modifications.

1.2.2 Coverage Models

Coverage models, where p facilitates are located in a manner that maximizes the coverage of elements (maximum coverage problem [25]), or minimizes the number of facilities to cover all elements (set cover problem), are widely used in many applications, for example, locating fire engines and ambulances [42, 16]. Given their flexibility, they are particularly useful in homeland security applications where accurate data might not be available and some model inputs rely on subject matter experts (SMEs). For example, coverage models have been used to identify how to optimally screen checked baggage on commercial aviation flights to reduce vulnerability in aviation security systems [51, 52]. A comprehensive study of coverage models and their applications can be found in Daskin [29].

The maximum coverage problem is NP-hard and belongs to the domain of monotone submodular maximization problems with a cardinality constraint, for which Nemhauser et al. [74] show that a greedy heuristic achieves an approximation factor of $(1 - 1/e)$. The budgeted maximum coverage problem (BMC) [57] generalizes the maximum coverage problem by replacing the cardinality constraint with a knapsack constraint. Khuller et al. [57] show that a greedy heuristic achieves an approximation factor of $(1 - 1/\sqrt{e})$, which can be improved to an optimal $(1 - 1/e)$ approximation factor by incorporating the greedy heuristic into a partial enumeration scheme. Sviridenko [97] generalizes these results to any submodular maximization problem subject to a knapsack constraint. Additionally, Fisher et al. [39] show that a greedy heuristic achieves an approximation ratio of $1/2$ when maximizing a monotone submodular set function subject to a matroid constraint. This result is improved to an optimal $(1 - 1/e)$ approximation factor by Vondrak [100] and Calinescu et al. [20] by adapting a continuous greedy algorithm and a pipage rounding procedure [5]. Two recent papers by Badanidiyuru and Vondrak [8] and Buchbinder et al. [19] propose faster algorithms with almost the same optimal approximation ratios. For the study of approximation algorithms for the general submodular maximization problem subject to multiple knapsack constraints, we refer the readers to Kulik et al. [63].

As mentioned earlier, mitigation effectiveness in cybersecurity planning is uncertain in nature. In the strategic facility location literature, uncertainty is dealt by defining alternative future scenarios. Some of the most common measures the planner attempts to optimize, assuming an underlying maximum coverage problem, include i) the expected coverage across all scenarios; ii) maximizing the minimal coverage across all scenarios, and iii) minimizing the maximal regret across all scenarios. The first measure is considered risk-neutral as all scenarios are weighted by their probabilities and summed up in the objective function. The second and third measures are more robust since they hedge against the "worst-case" scenarios. Regret is defined for each scenario as the difference between the coverage of a solution in that scenario and the coverage of the optimal solution for that single scenario. This usually involves pre-solving the problem for each individual scenario to obtain corresponding optimal solution, which can be seen as the best strategy that would have been selected if this realization of the future occurred. Therefore, regret is often interpreted as the opportunity loss for an uncertain future.

Robust methods only require as input a set of realizations of the uncertain parameters, not an explicit probability distribution as in stochastic optimization, and therefore, robust methods have a clear advantage in homeland security applications where many of the model inputs rely on the estimation from the subject matter experts (SMEs) who have limited knowledge of the problem, its inputs, and associated probability distributions. Robust optimization has been a powerful and popular tool for decision making in different areas, such as supply chain disruption planning [96] and adversarial risk analysis [66]. We refer to Bertsimas et al. [12] for a recent review on robust optimization that highlights its computational tractability and broad range of application, and Ben-Tal et al. [10] for a textbook treatment.

A recently popular risk measure in stochastic programming [6], the conditional value at risk (CVaR), can also provide risk insights for a robust decision maker CVaR is defined as the expected loss in the α worst-case tail of the loss distribution, which is initially proposed to quantify the risk for loss in finance [84, 85]. Chen et al. [24] apply CVaR to a facility location problem where they compare the model and its computationally efficiency closely to an earlier work that features a α-reliable Min-Max regret model [30], and demonstrate the advantage of this coherent measure. In a recent paper, Noyan [75] incorporates CVaR to a two-stage stochastic disaster preparedness management problem where a weighted sum of expected-value and CVaR is optimized to determine the response facility locations, and their corresponding inventory levels are determined under different types of uncertainties. Compared to the two worst-case measures discussed above, i.e., max-min coverage and min-max regret, the quantile-based CVaR measure is less pessimistic, since it provides solutions that are robust to the worst cases and also capture the magnitude of the coverage in the worst cases. Moreover, CVaR is coherent and computationally tractable through linear programming techniques.

1.2.3 Interdiction Models

Stackelberg game models are widely applied to network interdiction problems, in the literature on infrastructure protection and resilience, where they address ways to fortify critical components in the system or reduce vulnerability. Typically, the defender acts first by interdicting components of the attacker's network and the attacker reacts by selecting recourse actions. The core of these problems is usually a variant of a network flow problem, such as maximum flow [102], shortest path [50], or maximum-reliability path [70]. Stochastic network interdiction problems [28] incorporate uncertainties that often arise in adversarial attacks. The stochasticity often occurs in network parameters, such as arc capacities and interdiction attempts. We refer to [95] and [69] for a review of network interdiction models, algorithms and applications.

There is one type of network interdiction problem called the "system interdiction problem" [50], where interdictions disrupt the adversary's 'economy' by compromising its key components or activities, such as power generators or weapon production facilities. A classic example is the nuclear weapons project interdiction problem introduced by Brown et al. [18]. The defender deploys limited resources to delay the completion of a nuclear weapons project while the attacker aims to complete the project as soon as possible through a detailed project management procedure (i.e., decision critical path method). Brown et al. [17] discuss the complexity of several project interdiction problems with the defender being constrained by a budget constraint and the attacker's problem varying from a simple critical path method to that with multiple technologies and project expediting. With a linear program in the second stage, a common solution approach starts with reformulation by dualizing the second stage problem, resulting in a max-max mixed-integer programming problem that can be readily solved by a commercial solver. The model in Brown et al. [18] is more sophisticated and detailed, which contains integer variables in both stages, therefore dualization is no longer a valid choice. Instead, they propose a special decomposition algorithm that involves solving mixed-integer programs alternatively, and they incorporate solution-elimination constraints in the master problem to ensure bounds convergence.

Interdiction problems within the context of location analysis focus on identifying the most critical infrastructure in the system and designing fortification strategy to prevent them from being disrupted by natural disasters or adversarial attacks. The initial work can be traced to Church and Scaparra [27] who propose a r-interdiction medium problem (RIM) and a r-interdiction covering problem (RIC) that identify the r most critical facilities whose lost leads to the most damage to the system. Scaparra and Church [89] consider a fortification layer on top of RIM, denoted as RIMF, that explicitly considers the subset of facilities to fortify in order to minimize the damage, while the attacker can make a subsequent move to interdict unfortified facilities. They formulate the interdiction-fortification model as a bilevel static Stackelberg game, with the defender as the leader and the attacker as the follower. Aksen et al. [7] generalize the cardinality constraint for selecting facilities to fortify in RIMF to a budget constraint and incorporate a capacity expansion cost for operating facilities that take on new customers originally assigned to interdicted facilities. Moreover, O'Hanley et al. [81] propose a bilevel model for protecting ecological sites where the maximum species loss following the worst-case loss of a restricted subset of non-reserve sites is minimized. Another bilevel model based on maximum coverage problem is proposed by O'Hanley and Church [80] for locating facilities to maximize the initial coverage by p facilities and the post-interdiction coverage when r facilities are lost to interdiction.

Interdiction models are usually formulated as defender-attacker Stackelberg game which are in the form of two-stage mixed-integer program. General approaches for solving bi-level programming problem include decomposition, duality and reformulation as identified in Morton et al. [70]. Decomposition involves the use of outer-approximation cutting plane algorithms such as Bender's decomposition [11, 40], L-shaped methods [94, 101]. Duality corresponds to taking the dual of the second stage problem that enables a reformulation of the problem as a nested min-min or max-max problem which can be solved as a single-level problem. Reformulation is helpful when an equivalent model can be formulated with structure that is easier to handle. In Chapter 4, we reformulate the bilevel max-min interdiction model to a nested max-max problem via dualization, and then apply a Lagrangian heuristic that decomposes the max-max problem into a number of smaller problems and updates lower and upper bounds via subgradient optimization.

For more details about Lagrangian relaxation and subgradient method, we refer the readers to [41, 37, 38].

The interdiction problem can be particular difficult to solve when integer or binary variables appear in the second stage, which belongs to the domain of mixed integer bilevel linear programming (MIBLP) problems. In their seminal paper in 1990, Moore and Bard [68] explains the difficulty of solving such problem, and propose an implicit enumeration algorithm based on Branch and Bound for solving fairly small instances. Given the difficulty of finding the optimal solution to MIBLP problems, very few exact algorithms are proposed in the following decades until recently when a rapidly increasing interest for MIBLP emerged. DeNegre [31] proposes a branch-and-cut algorithm for solving MIBLP with pure integer variables at both levels, which is made available publicly as the MibS solver [83]. Saharidis and Ierapetritou [87] propose a Benders-like decomposition algorithm for solving general MIBLP problems. Xu and Wang [103] use a branch-and-bound framework for solving MIBLP with integer leader problem and mixed-integer programming (MIP) follower problem. Caramia and Mari [22] propose another branch-and-cut algorithm for solving pure integer leader and follower MIBLP problems. In a most recent paper, Fischetti et al. [36] propose a general-purpose algorithm for solving MIBLP that embeds several new classes of cuts and an effective bilevel-specific preprocessing procedure. They conduct an extensive computational study and demonstrate that the proposed algorithm outperforms existing methods from the literature, including the methods by DeNegre [31] and Caramia and Mari [22]. Particularly, Tang et al. [98] study a class of algorithms for solving general MIBLP zero-sum interdiction problems.

1.3 Dissertation Overview

We provide a detailed overview of the dissertation in this section. Our research is among one of first attempts to apply operations research methodologies to cybersecurity planning and management for the protection of the critical infrastructure. The main contribution of this dissertation to the literature are new maximum coverage models and interdiction models, and corresponding algorithms, as they are applied to a new area.

In Chapter 2, we study how to identify strategies for mitigating generic cyber vulnerabilities in the IT infrastructure. We propose maximum coverage models that prioritize the investment in security mitigations to maximally reduce vulnerabilities in the system. We use multiple coverage to reflect the implementation of a layered defense, and we consider the possibility of coverage failure to address the uncertainty in the effectiveness of some mitigations. Budgeted Maximum Multiple Coverage (BMMC) problems are formulated, and we demonstrate that the problems are submodular maximization problems subject to a knapsack constraint. Other variants of the problem are formulated given different possible requirements for selecting mitigations, including unit cost cardinality constraints and group cardinality constraints. We design greedy approximation algorithms for identifying near-optimal solutions to the models. We demonstrate an optimal $(1 - 1/e)$ approximation ratio for BMMC and a variation of BMMC that considers the possibility of coverage failure, and a $1/2$ approximation ratio for a variation of BMMC with a cardinality constraint and group cardinality constraints. The computational study suggests that our models yield solutions that use a layered defense and provide an effective mechanism to hedge against the risk of possible coverage failure. We also find that the approximation algorithms efficiently identify near-optimal solutions.

In Chapter 3, we extend the stochastic expected budgeted maximum multiple coverage model (EBMMC) that identifies "good" solutions on average that may be unacceptable in certain circumstances. We propose three alternative models that consider different robustness methods that hedge against worst case risks, including models that maximize the worst case coverage, minimize the worst case regret, and maximize the average coverage in the $(1 - \alpha)$ worst cases (conditional value at risk). We illustrate the solutions to the robust methods with a case study and discuss the insights their solutions provide into mitigation selection compared to an expected-value maximizer. We also report solution times of the models on a set of instances with varying size. The robust methods provide valuable tools and insights for the decision makers with different risk attitudes to manage cybersecurity risks under uncertainty.

In Chapter 4, we study how to protect critical infrastructure against adversarial attacks to reduce cyber vulnerabilities, which are sophisticated, hard to detect, and likely to have severe consequences. These dynamic, persistent risks cannot be completely eliminated, and therefore, it is important to protect IT infrastructure from these risks by delaying adversarial exploits. In this paper, we propose max-min interdiction models for critical infrastructure protection that prioritizes cost-effective security mitigations to maximally delay adversarial attacks. We consider attacks originating from multiple adversaries, each of which aims to find a "critical path" through the attack surface to complete the corresponding attack as soon as possible. Decision makers can deploy mitigations to delay attack exploits, however, mitigation effectiveness is sometimes uncertain. We propose a stochastic model variant to address this uncertainty by incorporating random delay times. The proposed models can be reformulated as a nested max-max problem using dualization. We propose a Lagrangian heuristic approach that decomposes the max-max problem into a number of smaller subproblems, and updates upper and lower bounds to the original problem via subgradient optimization. We evaluate the perfect information solution value as an alternative method for updating the upper bound to improve the Lagrangian upper bound. Computational results demonstrate that the Lagrangian heuristic identifies near-optimal solutions efficiently, which outperforms a general purpose mixed-integer programming solver on medium and large instances.

In Chapter 5, we study a particular critical infrastructure protection problem, the r-interdiction median problem with fortification (RIMF), and propose a new exact algorithm for solving it, which improves an existing method by Scaparra and Church [88] that uses interval search to iteratively improve lower and upper bounds to RIMF. The new algorithm consists of solving a greedy heuristic and a set cover problem iteratively that guarantees to find an optimal solution. More specifically, the greedy heuristic obtains a feasible solution to the problem, and the set cover problem is solved to verify optimality of the solution and provide a direction for improvement if not optimal. Computational results on benchmark and randomly generated data demonstrate that our approach is more efficient in identifying optimal solutions than Scaparra and Church's method across all instances, with up to one order of magnitude faster.

In Chapter 6, we summarize this dissertation and discuss extensions for future research, including a direct extension to the interdiction models presented in Chapter 5, and the potential of integrating a new concept in security analysis, the attack-defense tree, into our optimization framework for multi-stage sequential decision making.

Chapter 2

Budgeted Maximum Multiple Coverage Models for Cybersecurity Planning and Management

This chapter introduces the budgeted maximum multiple coverage (BMMC) models that prioritize security mitigations to mitigate generic cybersecurity risks in IT infrastructure. The attack threats are formulated generically as attack paths consisting of multiple vulnerability nodes. Mitigation controls are deployed to affect the vulnerability nodes directly. By covering each attack path multiple times, we implement a layered defense that is resilience to cyber attacks. Moreover, we incorporate uncertain mitigation effectiveness into the modeling framework to achieve an expected-value solution that performs well over uncertain future events.

In summary, this chapter makes the following contributions:

1. We propose new coverage models (e.g., BMMC and EBMMC) that address how to select the right portfolio of security mitigations that balance cost and threat reduction to yield a layered security defense of cyber-infrastructure and federal IT supply chains, while also addressing the uncertainty surrounding the effectiveness of mitigations.

2. We demonstrate that the proposed coverage models are submodular maximization problems subject to a knapsack constraint. We also formulate variations to the base BMMC and EBMMC models and demonstrate that they are submodular maximization problems subject to a cardinality or matroid constraint.

3. We propose heuristics for identifying near-optimal solutions to the models that use a polynomial number of function evaluations. We demonstrate an optimal $(1 - 1/e)$ approximation ratio for BMMC and EBMMC, an optimal $(1 - 1/e)$ approximation ratio for the particular cases of BMMC and EBMMC with uniform costs (i.e., a cardinality constraint), and a $1/2$ approximation ratio for the BMMC and EBMMC variations with a cardinality constraint and group cardinality constraints.

4. Extensive computational studies are conducted on the approximation algorithms with comparison to optimal solutions obtained from a general purpose mixed-integer solver. The greedy heuristics achieve high performance in both solution quality and computational time. We also demonstrate the practical value of the models in selecting mitigations by incorporating multiple coverage and possible mitigation coverage failure.

5. We discuss how to modify the approximation algorithms to produce solutions for varying budget levels, leading to a naturally "nested" set of solutions that can be used to help decision-makers weigh the trade-offs between cost and vulnerability reduction. A set of solutions is efficient to produce from a computational point of view and is more useful in practice than a single "best" solution.

The rest of this chapter is organized as follows. We define the problem and present its formulation in Section 2.1. In Section 2.2, we demonstrate that all the problems are monotone submodular maximization problems subject to a linear or matroid constraint, and then we introduce algorithms with constant approximation ratios. In Section 2.3, we conduct an extensive computational study to investigate the practical insights of the proposed models and test the efficiency and solution quality of the proposed algorithms. Section 2.4 concludes this chapter.

2.1 Models

In this section, we introduce several optimization models for prioritizing mitigations to protect IT supply chains by covering vulnerabilities. Cyber attacks exploit weak links or components in the supply chain network. We explicitly consider the steps taken to carry out an attack, which correspond to the vulnerabilities in the system. Accordingly, we formulate attacks as “attack paths” , each of which contains multiple nodes (vulnerabilities). We make several assumptions with the attack path. First, we do not consider the sequence of steps (nodes) on a path, rather we focus on reducing risks by protecting vulnerabilities in entire attack paths. Second, different attack paths may contain the same vulnerability node, for example, hardware may contain a vulnerability that can be exploited in multiple attacks against a computer network.

In our models, we assume the set of attacks paths is enumerated and given as a complete list. Thus, if the set of attack paths is given implicitly in an attack tree, a first step in applying our methodology is to enumerate the set of attack paths in the tree. The number of paths in an attack tree is given by the number of leaf nodes in the tree. Although in general an attack tree may become very large, many classes of IT attacks have limited access and controls, and thus the number of enumerated paths is manageable. Work with collaborators at Sandia National Laboratory (SNL) suggest that this assumption is reasonable for our application since the sizes of the attack trees are expected to be moderate. Given a ground set of vulnerability nodes, each attack path consists of a subset of nodes necessary to carry out an attack. We define a “coverage” relationship between a mitigation and a vulnerability node, since this type of input usually rely on SMEs. Mitigations “cover” vulnerability nodes, which in turn cover their associated attack paths. Multiple coverage is achieved by covering different nodes in the attack path, not from covering the same node in an attack path multiple times. This assumption is motivated by our discussion with collaborators at SNL, which corresponds to the concept of layered defense. In addition, we assume that i) coverage is non-decreasing in the number of nodes covered on an attack path, because the more vulnerability nodes covered in an attack path the better security we achieve; and ii) the marginal improvement in coverage decreases with the number of nodes covered on an attack path, because the marginal benefit for investing mitigations to cover more vulnerabilities in an attack path decreases.

The goal is to cover vulnerabilities in the attack paths by deploying security mitigations in a layered defense subject to a security budget. We present two types of models: a deterministic model that assumes the mitigation coverage is always effective, and a stochastic model that allows for uncertainty in the effectiveness of some mitigations. Each model is formulated and explained in the following subsections. We also consider other variations of the model to allow for different requirements, such as a cardinality constraint instead of a budget constraint as well as group cardinality constraints that limit decision-makers from choosing more than one mitigation from a pre-specified set due to other project requirements.

2.1.1 Deterministic Model

Our notation is defined as follows. Let S denote the set of attack paths, let N denote the set of vulnerabilities (nodes), and let $N_s \subseteq N$ denote the subset of vulnerability nodes in attack path $s \in S$, representing the steps it takes to carry out the attack. We define a critical level c_i associated with each vulnerability $i \in N$. A lower weight c_i corresponds to a lower type of vulnerability. We take the type of vulnerabilities into account when quantifying the coverage of each attack path. Let M denote the set of available mitigations M and let $M_n \in M$ denote the subset of mitigations that cover vulnerability node $n \in N$. Each mitigation $m \in M$ has an implementation cost b_m, and the total budget for selecting mitigations is B. In many situations, much of this underlying data may be collected from best practices and SMEs who have knowledge of the underlying processes, an understanding of the selected mitigation controls, and knowledge of the mitigations' possible effectiveness [67].

Decision variables

- x_m = 1 if mitigation $m \in M$ is chosen, 0 otherwise, $m \in M$,
- z_n = 1 if node $n \in N$ is covered by a selected mitigation, 0 otherwise, $n \in N$,

- y_s = the weighted number of vulnerability nodes in attack path $s \in S$ that are covered, $s \in S$.

More specifically, y_s is a function of z_n, that is, $y_s = \sum_{n \in N_s} c_n z_n$. To capture the impact of multiple coverage, we introduce a function $f_s(y_s), s \in S$, that captures the coverage of attack path $s \in S$ with respect to y_s. According to our problem assumptions, for each $s \in S$, $f_s(\cdot)$ is defined as non-decreasing and concave in y_s. Note that $f_s(\cdot)$ might not be identical across all attack paths $s \in S$ since it can reflect the the likelihood of occurrence and the consequences of the attacks.

The Budgeted Maximum Multiple Coverage problem (BMMC) is formulated as an integer programming model:

BMMC

$$\max \quad \sum_{s \in S} f_s(y_s) \tag{2.1}$$

$$\text{s.t.} \quad y_s = \sum_{n \in N_S} c_n z_n, \quad s \in S \tag{2.2}$$

$$z_n \leq \sum_{m \in M_n} x_m, \quad n \in N \tag{2.3}$$

$$\sum_{m \in M} b_m x_m \leq B \tag{2.4}$$

$$x_m \in \{0,1\}, m \in M, \quad z_n \in \{0,1\}, n \in N \tag{2.5}$$

The objective (2.1) is to maximize the total coverage over all attack paths. Multiple coverage is defined by constraint sets (2.2) and (2.3). Constraint set (2.2) defines y_s as the weighted number of nodes covered in attack path $s \in S$. Constraint (2.3) states that node $n \in N$ is covered if there exists at least one mitigation that covers it. The budget constraint is captured by (2.4). The last set of constraints (2.5) require the variables to be binary.

BMMC is NP-hard since it generalizes the Budgeted Maximum Coverage (BMC) problem [57], which is an NP-hard problem. This occurs when each attack path is composed of a single, unique node (i.e., $|N| = |S|$ with $N_s = \{s\}, s \in S$), and there is single coverage (i.e., $f_s(0) = 0$ and $f_s(1) = 1, s \in S$).

The attack data for BMMC, including attack paths and nodes, and the mitigation data, including the set of mitigations and the nodes they affect, can be obtained in conjunction with SMEs and risk analysts, as described earlier in this section. The mitigation costs can be set using data available from Federal budget offices. The coverage function is determined by decision-makers depending on their perception of the attributes and impact of each attack. For example, some attack paths might have fewer consequences or have a smaller probability of occurring, and a coverage function that encourages single coverage might be preferred in these cases so that fewer mitigation resources are directed toward covering their nodes. Additionally, weighing different attack paths can be achieved by multiplying the coverage functions by weight parameters.

2.1.2 Stochastic Model

The deterministic model makes the assumption that mitigation coverage is always effective and consistent. However, cyber threats are dynamic, persistent threats, and therefore, past information on mitigation controls is often incomplete, inaccurate, and not predictive of future mitigation efforts. One of our goals is to identify a set of mitigations that maximizes expected coverage given coverage uncertainty. To do so, we consider two states of the mitigation coverage, effective and ineffective. We define random variables $\xi_{mn}, m \in M, n \in N$ that are equal to 1 if the coverage of mitigation m is effective on node n, and 0 otherwise. The total coverage of the attack paths, denoted by $f(x, \xi)$, depends on which mitigations x are selected, and the random variable ξ. We are interested in improving the overall security of the system in the long run, therefore, it is natural to maximize the expected coverage, $\mathbb{E}_{\xi}[f(x, \xi)]$

The Expected-value Budgeted Maximum Multiple Coverage problem (EBMMC) is formulated as a stochastic integer programming model:

EBMMC

$$\max_{x} \quad \mathbb{E}_{\xi}\Big[\sum_{s\in S} f_s(y_s)\Big] \tag{2.6}$$

$$\text{s.t.} \quad \sum_{m\in M} b_m x_m \le B \tag{2.7}$$

$$\text{s.t.} \quad y_s = \sum_{n\in N_S} c_n z_n, \quad s \in S \tag{2.8}$$

$$z_n \le \sum_{m\in M_n} \xi_{mn} x_m, \quad n \in N \tag{2.9}$$

$$x_m \in \{0,1\}, m \in M \tag{2.10}$$

$$z_n \in \{0,1\}, n \in N. \tag{2.11}$$

The objective of the stochastic model (2.6) is to maximize the expected value of the total coverage across all attack paths. The random variable ξ_{mn} appears in one of the multiple coverage constraints, (2.9), stating that a node will not be covered if there is no effective mitigation covering it. EBMMC provides important insights that cannot be obtained from the deterministic model regarding how to select mitigations in the face of coverage uncertainty. EBMMC introduces an incentive to select two mitigations that cover the same node if that node is critical and if the mitigations that cover that node may not be effective. Since BMMC is a special case of EBMMC when mitigations are always effective, EBMMC is NP-hard.

There are $|M| \times |N|$ binary random variables ξ_{mn}. Therefore, there are $2^{|M|\times|N|}$ possible realizations of these random variables, which leads to exponential growth in the size of the stochastic problem. One way to address this issue is to solve an approximate stochastic problem using the well-studied Sampling Average Approximation (SAA) approach [58].

In the SAA approach, we take a set Ω of samples $\xi^{\omega}, \omega \in \Omega$ according to a given distribution of ξ, and approximate the expected value function with a sample average function:

$$\mathbb{E}_{\xi}[f(x,\xi)] \approx \sum_{\omega\in\Omega} \frac{1}{|\Omega|} f(x,\xi^{\omega}).$$

Given this approximation, we can derive a deterministic equivalent form of EBMMC. First we need to define coverage variables for each scenario $\omega \in \Omega$ as follows:

- z_n^ω = 1 if node $n \in N$ is covered by a selected mitigation in scenario $\omega \in \Omega$, 0 otherwise,
- y_s^ω = the weighted number of nodes in attack path $s \in S$ that are covered in scenario $\omega \in \Omega$.

The deterministic equivalent form of EBMMC based on SAA (SAA-EBMMC) is formulated as an integer programming model:

SAA-EBMMC

$$\max \quad \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \sum_{s \in S} f_s(y_s^\omega) \tag{2.12}$$

$$\text{s.t.} \quad y_s^\omega = \sum_{n \in N_S} c_n z_n^\omega, \quad s \in S, \omega \in \Omega \tag{2.13}$$

$$z_n^\omega \leq \sum_{m \in M_n} \xi_{mn}^\omega x_m, \quad n \in N, \omega \in \Omega \tag{2.14}$$

$$\sum_{m \in M} b_m x_m \leq B \tag{2.15}$$

$$x_m \in \{0,1\}, m \in M, \quad z_n^\omega \in \{0,1\}, n \in N, \omega \in \Omega \tag{2.16}$$

According to the SAA approach[58], an optimal solution of SAA-EBMMC provides an optimal solution of EBMMC with probability approaching 1 exponentially fast with the increase of $|\Omega|$. Therefore, in the computational study, we solve a sample average approximation EBMMC with large samples (1000 - 10,000 scenarios) to accurately approximate the EBMMC solutions. Although we conduct a computational study on the sample average approximation problem, all the theoretical results associated with the approximation algorithms in Section 2.2 apply to EBMMC.

Besides BMMC, we are also interested in the following two particular cases of EBMMC:

1. The k-cardinality constrained Expected-value Maximum Multiple Coverage Problem (kEMMC) This is when the budget constraint is replaced with a cardinality constraint, i.e., $b_m = 1,\ m \in M$, and at most $k = B$ mitigations can be selected.
2. The k-cardinality constrained Maximum Multiple Coverage Problem (kMMC). This is the deterministic analog of kEMMC without coverage uncertainty.

In all model variations, the decision is to select a subset of mitigations by setting the values of the $x_m,\ m \in M$, which in turn set the values of z_n and y_s in BMMC and kMMC or z_n^ω and y_s^ω in the deterministic equivalent forms of EBMMC and kEMMC. We note that the integer programming formulations for EBMMC and kEMMC have more variables and constraints than their deterministic counterparts BMMC and kMMC, respectively, and hence, we do not expect to solve large problem instances to optimality in a reasonable period of time. Therefore, approximation algorithms that can identify near-optimal solutions in polynomial-time are desirable. We introduce approximation algorithms with constant worst-case approximation ratios to EBMMC in Section 2.2. These algorithms can also be applied to the other three model variations.

2.1.3 Group Cardinality Constraints

In real settings, there are often requirements for selecting mitigations in addition to a budget or cardinality constraints. We consider one type of requirement here—group cardinality constraints—that leads to a fifth model variation of kEMMC. Group cardinality constraints, or multiple choice constraints are motivated by a practical concern that some mitigations have conflicting effects and cannot be implemented together. For example, a mitigation that suggests replacing an untrustworthy vendor and a mitigation that suggests improving this vendor's security procedure, cannot be selected together.

Let $M_1, M_2, ..., M_\ell$ be a partition of the mitigation set M, where $\cup_{i=1}^{\ell} M_i = M$ and $M_i \cap M_j = \emptyset, \forall i, j = 1, ..., \ell, i \neq j$. The group cardinality constraints are defined as follows:

$$\sum_{m \in M_i} x_m \leq 1, \quad i = 1, ..., \ell. \tag{2.17}$$

The k-cardinality constrained Expected-value Maximum Multiple Coverage problem with Group cardinality constraints (kEMMCG) can be modeled as an integer programming model that maximizes expected coverage (2.6) subject to (2.8) - (2.11), (2.17), and the k-cardinality constraint (2.18):

$$\sum_{m \in M} x_m \leq k, \tag{2.18}$$

We also introduce its deterministic variant, the k-cardinality Maximum Multiple Coverage problem with Group cardinality constraints (kMMCG), which can be considered as kEMMCG without coverage uncertainty (or as kMMC with group cardinality constraints). Both kEMMCG and kMMCG generalize the maximum coverage problem with a cardinality constraint and group cardinality constraints [23], which considers a standard maximum covering objective with single coverage, and no uncertainty. Chekuri and Kumar[23] propose a greedy heuristic with a constant 1/2-approximation ratio for solving this problem. In the next section, we show that this greedy heuristic achieves a 1/2-approximation ratio for kEMMCG as well as any nondecreasing submodular maximization problem with a cardinality constraint and group cardinality constraints.

2.2 Approximation Algorithms

In this section, we present approximation algorithms with a polynomial number of objective function evaluations for all models presented in the previous section, and prove constant worst-case approximation ratios associated with the algorithms. The proofs presented in Section 2.2.1 are associated with EBMMC. We also show that they can be easily adapted for the particular cases of EBMMC, i.e., BMMC, kEMMC and kMMC. In Section 2.2.2, we demonstrate results for kEMMCG and its particular case kMMCG.

2.2.1 EBMMC and Its Particular Cases

A $(1 - 1/e)$-approximation ratio can be achieved in polynomial time for any submodular maximization problems subject to a knapsack constraint [97] or a cardinality constraint [74]. In the following theorem, we show that EBMMC can be reformulated as a submodular maximization problem subject to a knapsack constraint.

A set function $g(\cdot)$ is submodular if given two sets $A_1, A_2 \subseteq T$ and $A_1 \subseteq A_2$, for any $\alpha \in T \backslash A_2$, $g(A_1 \cup \{\alpha\}) - g(A_1) \geq g(A_2 \cup \{\alpha\}) - g(A_2)$. For a set $A \subseteq M$, we define its characteristic vector $\chi^A \in \{0,1\}^{|M|}$, where its i-th element is equal to 1 if $i \in A$ and 0 otherwise. We define a set function $g(\cdot) : 2^{|M|} \to \mathbb{R}$ as follows:

$$g(A) = \mathbb{E}_\xi[f(\chi^A, \xi)] \tag{2.19}$$

Then we can reformulate EBMMC as maximizing set function $g(\cdot)$ subject to a knapsack constraint:

$$\max_{A \subseteq M} \{g(A) : \sum_{m \in A} b_m \leq B\} \tag{2.20}$$

We now present our first theorem as follows.

Theorem 1. EBMMC is a non-decreasing submodular maximization problem subject to a knapsack constraint.

Proof. To demonstrate this, we need to show $g(\cdot)$ is a non-decreasing submodular set function. Given two sets, A_1 and A_2 such that $A_1 \subseteq A_2 \subseteq M$, and an element $m \in M \setminus A_2$, we need to show $g(A_2) \geq g(A_1)$ and $g(A_1 \cup \{m\}) - g(A_1) \geq g(A_2 \cup \{m\}) - g(A_2)$.

We first show that for fixed $\bar{\xi}$, it is true that $f(\chi^{A_2}, \bar{\xi}) \geq f(\chi^{A_1}, \bar{\xi})$ and $f(\chi^{A_1 \cup \{m\}}, \bar{\xi}) - f(\chi^{A_1}, \bar{\xi}) \geq f(\chi^{A_2 \cup \{m\}}, \bar{\xi}) - f(\chi^{A_2}, \bar{\xi})$. $f(\chi^A, \bar{\xi})$ is the total coverage when a set of mitigations A is selected. Let y_s^A be the weighted number of nodes covered on attack path s when A is selected. We have $f(\chi^A, \bar{\xi}) = \sum_{s \in S} f_s(y_s^A)$. Since $f_s(\cdot)$ is nondecreasing, we have $f_s(y_s^{A_2}) - f_s(y_s^{A_1}) \geq 0$ and thus $f(\chi^{A_2}, \bar{\xi}) \geq f(\chi^{A_1}, \bar{\xi})$. Due to the same reason, we also have $f_s(y_s^{A_1 \cup \{m\}}) - f_s(y_s^{A_1}) \geq 0, f_s(y_s^{A_2 \cup \{m\}}) - f_s(y_s^{A_2}) \geq 0$. Since $f_s(\cdot)$ is concave, the marginal increase of the objective value is decreasing. Therefore, we have, $f_s(y_s^{A_1 \cup \{m\}}) - f_s(y_s^{A_1}) \geq f_s(y_s^{A_2 \cup \{m\}}) - f_s(y_s^{A_2})$. It follows that $f(\chi^{A_1 \cup \{m\}}, \bar{\xi}) - f(\chi^{A_1}, \bar{\xi}) \geq f(\chi^{A_2 \cup \{m\}}, \bar{\xi}) - f(\chi^{A_2}, \bar{\xi})$.

By definition, $g(A) = \mathbb{E}_\xi[f(\chi^A, \xi)]$. Taking the expected value with respect to ξ on both sides of the above inequalities does not change their signs. We therefore have, $g(A_2) \geq g(A_1)$ and $g(A_1 \cup \{m\}) - g(A_1) \geq g(A_2 \cup \{m\}) - g(A_2)$. □

Corollary 2. kEMMC and kMMC are submodular maximization problems subject to a cardinality constraint.

Theorem 1 allows us to apply well-known results to our model. Next, we show how to adapt the two approximation algorithms in Khuller et al. [57] to EBMMC to achieve $(1 - 1/\sqrt{e})$ and $(1 - 1/e)$ approximation ratios for EBMMC, respectively. Let $b(A)$ denote the total budget cost when $A \subseteq M$ is selected, and let $\Delta g_m(A) = g(A \cup \{m\}) - g(A), m \in M$ capture the marginal increase of the objective value when m is added into set A.

The first approximation algorithm is a modified greedy algorithm. It selects the better of a greedy solution (a set of mitigations) and the single best mitigation. Given a current solution A and a set of available mitigations T, the greedy algorithm selects a budget feasible mitigation with the highest $\Delta g_m(A)/b_m$ ratio in each iteration until no more mitigations can be selected.

Greedy(A,T):

1. Compute $\Delta g_m(A)$ for all $m \in T$, select mitigation $m^* \in \mathrm{argmax}_{m \in T}\{\Delta g_m(A)/b_m\}$.
2. If $b(A \cup \{m^*\}) \leq B$, do $A \leftarrow A \cup \{m^*\}$. Compute $T' = \{j \in T : b(A \cup \{j\}) > B\}$, do $T \leftarrow T \setminus T'$.
3. If $T = \emptyset$, return A; otherwise, go to Step 1.

Greedy achieves an optimal $(1 - 1/e)$ approximation ratio [74] for kEMMC and kMMC. This is the best possible approximation ratio unless $P = NP$ [35]. The first approximation algorithm for BMMC and EBMMC is a modified greedy algorithm as shown below:

Algorithm 1.

0. $A \leftarrow \emptyset, T \leftarrow M$.
1. $A \leftarrow Greedy(A, T)$.
2. Select mitigation $m^* \in \mathrm{argmax}_{m \in T}\{g(\{m\})\}$.
3. If $g(A) \geq g(\{m^*\})$, return A; otherwise, return $\{m^*\}$.

Algorithm 1 has an approximation ratio of $(1 - 1/\sqrt{e})$ for EBMMC and BMMC. It requires a polynomial number $O(|M|^2)$ of objective function value computations. *Greedy(A, T)* computes $\Delta g_m(A)$ for all $m \in T$, which can be a large number of computations. The computational time can be improved by reducing the number of evaluations considered in Step 1 of *Greedy(A, T)* using the "lazy evaluations" procedure proposed by Leskovec et al. [64]. This procedure exploits submodularity by computing $\Delta g_m(A)/b_m$ for the available mitigations in decreasing order based on the previous iteration that led to mitigation m^* being selected, and it terminates when a new value of $\Delta g_m(A)/b_m$ (after adding m^*) is greater than all values from the previous iteration (before adding m^*). This procedure's correctness follows from submodularity, since $\Delta g_m(A)/b_m$ can never increase as the set of selected mitigations A grows. This procedure can drastically reduce the number of computations in Step 1 of *Greedy(A,T)*, however, it does not change the worst-case time complexity.

Greedy and Algorithm 1 also provide insight on the situation when the total budget is not available all at once and instead becomes available over time [60]. First, consider the simple case with unit cost (kEMMC and kMMC). Each time the budget increases by one unit, *Greedy* selects the mitigation with the best marginal increase in objective function value. If we run *Greedy* until all mitigations are selected, we obtain a naturally nested set of solutions, each of which approximates the problem with a constant factor $(1 - 1/e)$. Therefore, it provides a prioritized list of mitigations that can be implemented by decision-makers interested in selecting mitigations with the highest priority level [59]. For any value of k, the set of k mitigations selected by *Greedy* is always a naturally nested set.

The nested solution obtained from *Greedy* is not maintained when costs are not identical as in EBMMC and BMMC, where a different set of mitigations may be selected by *Greedy* for different values of the budget. However, Algorithm 1 can be adapted to construct a set of solutions that approximate the efficient frontier between coverage and cost, thus yielding a set of "good" solutions that will help decision-makers make trade-offs between competing objectives. This can be achieved for Algorithm 1 by setting the budget to $B = \sum_{m \in M} b_m$ and saving the solution, its objective function value, and its budget value after each mitigation is added. This set of solutions is naturally nested. Moreover, it can serve as a "warm start" for finding a set of mitigations at any intermediate budget level. Specifically, if the lists of solution and budget values are: $\{A_1, A_2, ..., A_{|M|}\}$, and $\{B_1, B_2, ..., B_{|M|}\}$, respectively, when the budget level is an intermediate budget value B with $B_i < B < B_{i+1}$, we can run Algorithm 1 with $A \leftarrow A_i$, $B \leftarrow B - B_i$, and $T \leftarrow M \setminus A_i$ in step 0 rather than solve from scratch with $A \leftarrow \emptyset, T \leftarrow M$.

The second algorithm uses Greedy in a partial enumeration scheme that achieves an approximation ratio $(1 - 1/e)$ for EBMMC and BMMC. The bound of $(1 - 1/e)$ is the best possible that can be achieved for EBMMC and BMMC in polynomial time unless P=NP [35]. We describe Algorithm 2 as follows:

Algorithm 2.

```
0. $T_2 \leftarrow \emptyset$, $q = 3$.
1. For all $A \subseteq M$, such that $|A| < q$ and $b(A) \leq B$, compute $g(A)$ and assign the solution with
   the highest objective value to $T_1$.
2. For all $A \subseteq M$, such that $|A| = q$ and $b(A) \leq B$, do:
       $T \leftarrow M \setminus A$
       $W \leftarrow Greedy(A, T)$;
       if $g(W) > g(T_2)$, then $T_2 \leftarrow W$
3. If $g(T_1) > g(T_2)$, return $T_1$; otherwise, return $T_2$.
```

Algorithm 2 requires $O(|M|^5)$ objective function value evaluations, which makes it less practical than Algorithm 1. We compare the solution quality and computational times of these two approximation algorithms in Section 2.3.

2.2.2 A Greedy Heuristic for Solving Group Cardinality Constraints

In this subsection, we present a modified greedy algorithm that accounts for the additional group cardinality constraints, and we demonstrate that it achieves a $1/2$ approximation ratio for kEMMCG and kMMCG.

Using the set function $g(\cdot)$ defined for EBMMC, we can reformulate kEMMCG as follows:

$$\max_{A\subseteq M}\{g(A) : |A| \leq k, |A \cap M_i| \leq 1, i = 1, ..., \ell\}. \tag{2.21}$$

Without loss of generality, we assume that $g(\emptyset) = 0$, i.e., $g(\cdot)$ is a normalized set function, and that the number of groups is greater than the maximum number of selected mitigations, i.e., $\ell \geq k$, otherwise the group cardinality constraints can be removed from the model. kEMMCG (2.21) maximizes a non-decreasing submodular set function subject to a uniform matroid constraint and a partition matroid constraint, the combination of which is equivalent to a truncated partition matroid constraint. A truncated partition matroid is a particular case of a matroid. Fisher et al. [39] demonstrate that a greedy heuristic achieves a $1/2$-approximation ratio for maximizing a monotone submodular set function subject to a matroid constraint. Chekuri and Kumar [23] present a greedy heuristic for solving the maximum coverage problem subject to a cardinality constraint and group cardinality constraints (a particular case of kEMMCG) with a $1/2$-approximation ratio. Their analysis differs from the usual analysis for maximum coverage problem in the literature and is instead based on the multiple knapsack problem. Using a similar analysis, we show that the greedy heuristic in [23] can be adapted with the same $1/2$ approximation ratio to kEMMCG and the general monotone submodular maximization problem subject to a cardinality constraint and group cardinality constraints.

Let $\Delta g_m(A) = g(A \cup \{m\}) - g(A)$ be the marginal increase in the objective value when m is added into A. Algorithm 3 iteratively selects mitigation that maximizes the marginal improvement in the objective value; however it considers mitigations only from those groups that have not already had a mitigation selected from them.

Algorithm 3

```
0. $A \leftarrow \emptyset, T \leftarrow M$.
1. Compute $\Delta g_m(A)$ for all $m \in T$, select mitigation $i \in \text{argmax}_{m \in T} \Delta g_m(A)$.
2. If $|A| < \ell$, do $A \leftarrow A \cup \{i\}$ and delete all mitigations that belong to the same group as $i$
   from $T$, go to Step 1; else stop and return solution $A$.
```

Algorithm 3 requires $O(\ell M)$ objective function value evaluations. We present the following theorem to demonstrate that Algorithm 3 obtains a $1/2$-approximation ratio for kEMMCG and kMMCG by proving the result for the general monotone submodular maximization problem subject to a cardinality constraint and group cardinality constraints. Note that this result matches the approximation ratio in Fisher [39] for monotone submodular maximization problem subject to a matroid constraint. Let $i_1, i_2, ..., i_\ell$ be the indices of the groups from which Algorithm 3 selects mitigations. For the simplicity of the following proof, without loss of generality, we relabel above groups from 1 to ℓ. For $j = 0, ..., \ell$, let A_j be the set of mitigations that Greedy has selected until the jth iteration, and $A_0 = \emptyset$, $A = A_\ell$. Let O_j be the mitigation that the optimal solution selects from group j, and $O = \{O_j, 1 \leq j \leq \ell\}$.

Theorem 3. Algorithm 3 achieves an approximation ratio of $1/2$ for maximizing a monotone submodular set function subject to a cardinality constraint and group cardinality constraints.

Proof. In each iteration of Algorithm 3, we select a mitigation that has the largest marginal objective value, thus we have

$$g(A_j) - g(A_{j-1}) \geq g(O_j \cup A_{j-1}) - g(A_{j-1})$$

Since $A_j \subseteq A$ for any $j = 1, ..., \ell$, and $g(\cdot)$ is a non-decreasing submodular set function, it follows that

$$g(A_j) - g(A_{j-1}) \geq g(O_j \cup A) - g(A)$$

Summing both sides over all groups $j = 1, ..., \ell$ yields

$$\begin{aligned} g(A) = \sum_j (g(A_j) - g(A_{j-1})) &\geq \sum_j (g(O_j \cup A) - g(A)) \\ &\geq g(O \cup A) - g(A) \\ &\geq g(O) - g(A). \end{aligned}$$

The second inequality follows from the fact that $g(\cdot)$ is a submodular set function, i.e., the marginal increase in the objective value when adding a set O to A is smaller than the sum of the marginal increase in the objective when adding each element in O to A individually. The third inequality follows because $O \subseteq O \cup A$ and $g(\cdot)$ is non-decreasing.

Hence, $g(A) \geq \frac{1}{2} g(O)$. □

2.3 Computational Results

This section is divided into three parts, each of which focuses on one introduced model. Throughout the computational study, we examine two topics, solution analysis and the empirical performance of the approximation algorithms. Specifically, we investigate the insights the models provide into how to select mitigations by incorporating multiple coverage and the possibility of coverage failure, as compared to simpler models. Additionally, we evaluate the performance of the approximation algorithms by examining their empirical approximation ratios and computational time with comparison to a mixed integer programming (MIP) solver. The models were programmed and run with Python 2.7.10 and were solved using Gurobi 6.0 as the MIP solver. The approximation algorithms were run with PyPy 2.5.1, a fast alternative Python compiler that can significantly improve the speed of Python programs. The tests were run on an Intel Core i5-3470 CPU at 3.20 GHz with 4 GB of RAM. A time limit was set to one hour.

As we mentioned in Section 2.1, real data that can be used to test our models are scarce or inaccessible. To test the algorithms and evaluate the models, we constructed synthetic data with varying sizes in conjunction with our collaborators at SNL. Problem instances are defined by four main parts of data: 1) the set covering data, which includes the set of mitigations, the set of nodes, the covering relationship between the nodes and mitigations, the cost coefficients, and the budget value; 2) the multiple coverage data, which includes the number of attack paths, the list of nodes on each attack paths and the objective cost coefficients; and 3) the stochastic data. As we mention in Section 2.1, given the intractability of the size of EBMMC, we solve SAA-EBMMC with a set of scenarios large enough to approximate the solution of EBMMC. The stochastic data consists of the set of scenarios, and the distribution of the random variables ξ; and 4) the group data for kMMCG and kEMMCG, which includes a group of subsets of mitigations.

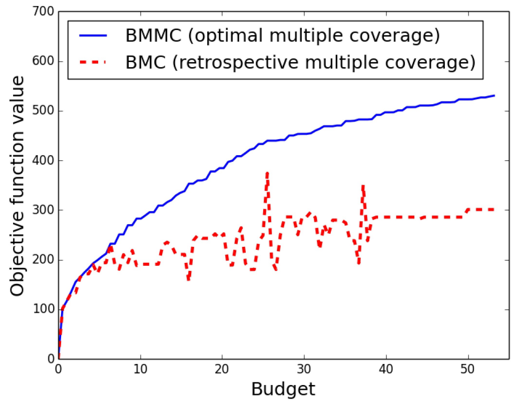
2.3.1 BMMC Solutions

We start by solving BMMC to examine the effect multiple coverage has on selecting mitigations compared to BMC. Additionally, BMMC has fewer variables and constraints compared to EBMMC, and solving it gives us an understanding of how the computational times of the approximation algorithms scale. The data was created synthetically as follows. We specify that each node can be covered by at most three different mitigations and randomly sample from the set of mitigations M with replacement to generate $M_n, n \in N$ using a pseudo-random generator. Similarly, we set the upper bound on the number of nodes in each attack path to five and randomly sample from the set of nodes N with replacement to generate a list of nodes in each attack path $N_s, s \in S$. In some applications, the number of nodes in a path may exceed five. However, feedback from our collaborators indicate that this is a reasonable upper bound since there are relatively few access points for control in the supply chain vulnerabilities under consideration. The coverage function $f_s(\cdot)$ is defined in different forms for different purposes, as is done in this and the following subsections.

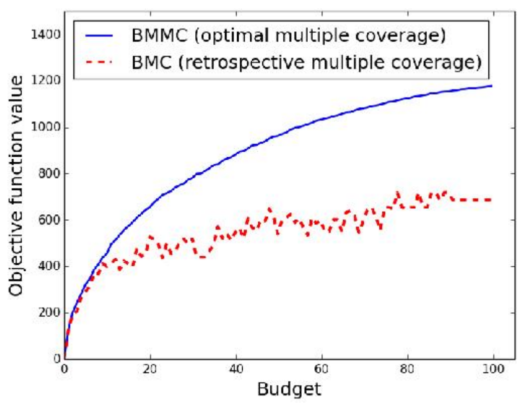
The mitigation costs $b_m, m \in M$ are generated as Uniform $(0, 1)$ random variables using a pseudo-random generator. We create eight datasets consisting of the set covering data and the multiple coverage data for the purpose of BMMC solution analysis, and testing Algorithm 1, 2, and 3 throughout the computational study. The size of the datasets ranges from $20(|M|) \times 20(|N|) \times 10(|S|)$ to $1,000(|M|) \times 1,000(|N|) \times 500(|S|)$. The size of a dataset could reflect two aspects in real settings: the scale of the planning problem, or the size of the network; and the size of the organization that undergoes the planning. Each created dataset has a unique size. We will refer to these datasets by their size in the remaining part of this section.

First, we investigate the insights provided by BMMC with multiple coverage. Attack paths in our problem represent vulnerabilities. In a problem with single coverage (e.g., BMC), there is no incentive in terms of the objective to cover a path more than once. In multiple coverage problems as proposed in this chapter (e.g., BMMC), each attack path can be covered multiple times to encourage a layered defense, where attacks can be prevented in several ways by different mitigations. This yields a more robust approach for cybersecurity planning that responds to the dynamic nature of security threats [86] and also yields different solutions than single coverage models.

We compare BMMC solutions with BMC solutions on two instances from the main datasets to illustrate the insights. Instance (a) has 50 mitigations, 50 nodes and 20 attack paths, and instance (b) has 100 mitigations, 100 nodes and 50 attack paths. Both instances for BMC and BMMC are identical aside from the objective function. Using our notation, the BMC coverage function is $f_s(y^s) = \min\{1, y_s\}$. For BMMC, we select a linear coverage function $f_s(y_s) = a_s y_s$ for simplicity. In both cases, a_s is the weight associated with attack path $s \in S$ created to differentiate the paths, which is created in a random manner. We generate $a_s, s \in S$ as Uniform $(0, 10)$ random variables using a pseudo-random generator. Solving BMMC and BMC returns the optimal multiple coverage solution and the optimal single coverage solution, respectively. While we cannot compare these two solutions directly given that they are optimizing two different objective functions, we retrospectively evaluate the BMMC multiple coverage objective function values associated with the BMC solutions.



(a) $|M| = 50, |N| = 50, |S| = 20$



(b) $|M| = 100, |N| = 100, |S| = 50$

Figure 2.1: Objective function values vs. budget when maximizing single coverage (BMC) and multiple coverage (BMMC). BMC solutions are retrospectively evaluated with multiple coverage objective function for comparison with the optimal BMMC solutions.

Figure 2.1 shows the multiple coverage objective function values associated with the BMMC and BMC against the budget values for the two problem instances. The optimal multiple coverage objective values of BMMC solutions are shown in solid lines, while the retrospective multiple coverage objective function values of BMC solutions are shown in dashed lines. The single coverage objective function values of BMC (not shown in Figure 2.1) eventually do not increase with respect to budget, and its retrospective multiple coverage objective function value fluctuates with respect to budget because the BMC solutions change for different budget levels. BMMC (optimal multiple coverage) and BMC (retrospective multiple coverage) are very similar when budget is very low and they start to diverge for higher values of the budget. When the budget is very low, both BMC and BMMC select nearly the same set of mitigations and cover attack paths at most once. Once the budget becomes large enough to warrant covering some attack paths more than once, BMMC and BMC select different sets of mitigations. The gap between these BMMC (optimal multiple coverage) and BMC (retrospective multiple coverage) is the value added for multiple coverage.

Next, we test the performance of Algorithm 1 and 2 for solving BMMC on the eight main datasets. We set the budget to be below 5% of the total cost of all mitigations. A budget value that is too small or large could make solving the instance trivial. For example, if the budget is too small and only one or two mitigations can be selected, it is more likely that the approximation algorithms could find an optimal or very near-optimal solution; or if the budget is large, given the overlapping capacity of the mitigations, it is possible that both the approximate and optimal solutions cover all nodes in the attack paths while only using a fraction of the budget. Our experimentation suggests that a budget value that falls within 5% of the total mitigation costs given our parameters is a reasonable choice given our input parameters. We also test the algorithms for kMMC, the particular case of BMMC with a cardinality constraint, on the same instances by setting the mitigation cost to $b_m = 1, m \in M$ and selecting $B = 0.1 \times |M|$. The objective function $f_s(\cdot)$ can take various forms as long as it is concave and non-decreasing. We set $f_s(\cdot) = -a_s(y_s)^2 + 2a_s|N_s|y_s, y_s = 1, 2, ..., |N_s|$, which is zero when no nodes in $s \in S$ are covered and achieves its maximum value of $a_s|N_s|^2$ when all N_s nodes are covered.

Table 2.1 reports the approximation ratios and computation times of these two algorithms for BMMC and kMMC. The approximation ratio is calculated as the objective value returned by the approximation algorithm divided by the optimal objective value. For both BMMC and kMMC, we observe similar performance between Algorithm 1 and 2. Algorithm 1 is faster than the MIP solver in most cases, especially in the last instance where for BMMC the MIP solver Gurobi requires 925 seconds for solving BMMC while Algorithm 1 runs in less than 7 seconds, and for kMMC, Gurobi does not find an optimal solution within the one hour time limit while Algorithm 1 runs in less than 2 seconds. The observed approximation ratios for Algorithm 1 over these test instances are within 3% of optimal for both models. The computational time for Algorithm 2 increases tremendously as the problem size increases. Algorithm 2 only manages to solve the first four instances for BMMC and the first three for kMMC within the one hour time limit. Given its accuracy and computational speed, Algorithm 1 appears to be more practical than Algorithm 2. Solving BMMC optimally with the MIP solver is fast for all but the largest datasets, so we also consider it more practical than Algorithm 2. Given that the algorithms have fairly close practical performance for solving the cardinality constrained model and the more general model with a knapsack constraint, in the next subsection we test Algorithm 1 and 2 mainly on

Table 2.1: Empirical approximation ratios and computational times of Algorithm 1 and 2 for solving BMMC on eight instances, with comparison to a MIP solver

(a) BMMC

			Approx. ratio		Time (s)		
$\lvert M\rvert$	$\lvert N\rvert$	$\lvert S\rvert$	Alg. 1	Alg. 2	Alg. 1	Alg. 2	BMMC
20	20	10	100.0%	100.0%	0.00	0.01	0.01
50	50	20	97.4%	100.0%	0.04	0.65	0.07
100	100	50	98.3%	100.0%	0.09	176.98	0.15
200	200	100	99.1%	100.0%	0.14	2419.00	0.21
500	500	500	99.7%	NA	0.54	NA^1	0.23
600	600	300	99.6%	NA	1.01	NA^1	0.97
800	800	400	99.3%	NA	1.67	NA^1	0.46
1,000	1,000	500	99.9%	NA	6.83	NA^1	924.65

1: out of time

(b) kMMC

			Approx. ratio		Time (s)		
$\lvert M\rvert$	$\lvert N\rvert$	$\lvert S\rvert$	Alg. 1	Alg. 2	Alg. 1	Alg. 2	BMMC
20	20	10	100.0%	100.0%	0.00	0.01	0.02
50	50	20	100.0%	100.0%	0.03	0.15	0.03
100	100	50	98.3%	98.3%	0.07	43.21	0.07
200	200	100	100.0%	NA	0.24	NA^1	0.18
500	500	500	99.0%	NA	0.63	NA^1	0.95
600	600	300	99.7%	NA	0.89	NA^1	1.25
800	800	400	98.8%	NA	1.00	NA^1	4.86
1,000	1,000	500	NA	NA	1.98	NA^1	NA^1

1: out of time

the knapsack constrained model, i.e., SAA-EBMMC.

To further examine the performance of Algorithm 1, we test it on larger problem instances. For the set covering data, we use the test instances posted in the online OR-library [9]. We select five instances, scpnrg1-scpnrg5, from one of the hardest problem sets[21]. All instances have the same size with $|M| = 10,000$ and $|N| = 1,000$. We then create the multiple coverage data as in the instances used earlier. Table 2.2 compares the performance of Algorithm 1 with the MIP solver on these instances. None of the Algorithm 2 instances finished within the one hour

Table 2.2: Empirical approximation ratios and computational times of Algorithm 1 for solving BMMC on five large instances from the OR-library, with comparison to a MIP solver.

				Approx. ratio	Time (s)	
Set Covering instance	$\|M\|$	$\|N\|$	$\|S\|$	Alg. 1	Alg. 1	BMMC
scpnrg1	10,000	1,000	500	99.7%	2.8	1716.9
scpnrg2	10,000	1,000	500	99.8%	3.3	2819.2
scpnrg3	10,000	1,000	500	NA	3.2	NA[1]
scpnrg4	10,000	1,000	500	NA	3.1	NA[1]
scpnrg5	10,000	1,000	500	NA	3.0	NA[1]

1: out of time

time limit, and therefore, those results are omitted from Table 2.2. The MIP solver finds optimal solutions for the first two instances within time limit. Algorithm 1 completes within 4 seconds across all instances, and it finds solutions within 1% of the optimal solution values.

As mentioned in Section 2.2, Algorithm 1 can be used to identify a set of non-dominated solutions between coverage and cost, thus yielding a set of solutions that can help decision-makers make tradeoffs between competing objectives. To do so, we set $B = \sum_{m \in M} b_m$ and obtain a list of mitigations $\{m_1, m_2, .., m_{|M|}\}$ that are selected successively. An example with an instance ($|M| = 50 \times |N| = 50 \times |S| = 20$) from the main dataset is illustrated in Figure 2.2. To obtain optimal solutions, we solve the model at each budget level separately (20 times for this figure), which requires significantly more computational time compared to Algorithm 1, which obtains all points in Figure 2.2 in single pass through Algorithm 1. We can see that at most of the budget levels, Algorithm 1 coincides with the optimal solution, and the overall difference between the two objective function values is extremely small. Recall that this set of solution from Algorithm 1 is naturally nested. It is efficient to generate and gives decision-makers an overview of tradeoffs between cost and vulnerability reduction. However, it is important to note that the nestedness feature for Algorithm 1 is only maintained when the budget increases to the exact level of the solid points in Figure 2.2.

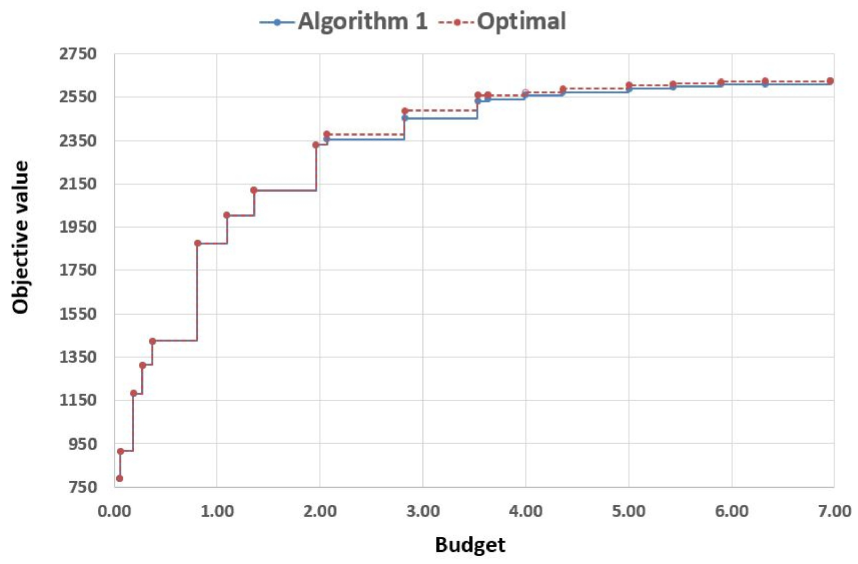


Figure 2.2: Objective values vs. budget for a list of naturally nested Algorithm 1 solutions and the optimal solutions at corresponding budget levels.

2.3.2 SAA-EBMMC Solutions

Given Algorithm 1's effectiveness in identifying near-optimal solutions to large problem instances of BMMC, we use it instead of Algorithm 2 for solving the stochastic model EBMMC. Meanwhile, we examine the impact of uncertainty on mitigation selection by comparing the single coverage objective function values and solutions associated with BMMC and SAA-EBMMC. We consider a single coverage objective function to focus on the impact that coverage failure has on solutions.

We assume that ξ are independent and identically distributed (i.i.d.) Bernoulli random variables. A sample $\xi^\omega, \omega \in \Omega$, that captures coverage effectiveness is drawn from the Bernoulli distribution with success probability $\bar{p} = 0.5$. To demonstrate the differences in the solutions when using the stochastic model (SAA-EBMMC) as compared to the deterministic model (BMMC), we create a set of new test instances with 20 mitigations and 20 attack paths with one node on each path. For each of the six instances, we distinguish the attack paths by assigning two different weights—10 or 1—to the attack paths to illustrate the difference between the stochastic and deterministic solutions. Attack paths that are assigned with the higher weight 10 are called "important" attack

Table 2.3: The differences between the solutions of BMMC and SAA-EBMMC.

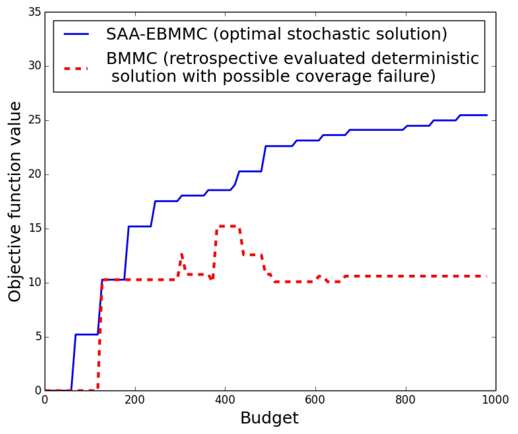
	Average number of times a node is covered		Average number of times a node on an important attack path is covered	
Instance	BMMC	SAA-EBMMC	BMMC	SAA-EBMMC
1	1.10	1.30	1.00	1.70
2	1.10	1.80	1.00	2.20
3	1.00	1.20	1.10	1.50
4	1.10	1.65	1.10	2.0
5	0.85	1.15	1.10	1.10
6	0.50	0.85	0.50	1.10

$|M| = 20, |N| = 20, |S| = 20, |\Omega| = 1000.$

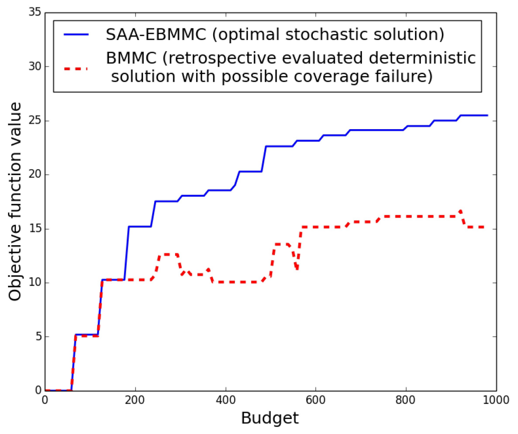
paths.

Table 2.3 reports the average number of times an attack path is covered across all attack paths (left side) and across the so-called important attack paths (right side). Since each attack path is composed of a single node, these results indicate the average number of times a node is covered. In all instances, the SAA-EBMMC solutions cover an attack path more times on average than do the BMMC solutions. This is because the SAA-EBMMC solutions tend to select mitigations that cover more nodes, whereas there is no incentive to cover a node twice in BMMC where there is no coverage failure. This issue is accentuated across the important attack paths, as shown on the right hand side of Table 2.3. Important attack paths are those with higher weights, and thus contribute more to the objective value when covered. The stochastic model solutions hedge against the risk of coverage failure by selecting mitigations that in general cover nodes in attack paths, especially the important attack paths, more than once as compared to BMMC, where covering nodes a second time does not contribute toward the objective function value.

We can further demonstrate the value the stochastic solutions contribute to the problem by examining the objective function value added for optimizing SAA-EBMMC compared to BMMC. Figure 2.3 shows the objective function values of solutions from two models plotted against the budget values on instances 1 and 3. Since the objective values of BMMC do not include the lost coverage due to random mitigation failure, we can not directly compare the objective



(a) Instance 1



(b) Instance 3

Figure 2.3: Objective function values vs. budget when maximizing the deterministic model (BMMC) and the stochastic model (SAA-EBMMC). To compare solutions of the two models, expected-value objective function values associated with the BMMC solutions are retrospectively evaluated using the same coverage effectiveness random variable sample $\xi^{\omega}, \omega \in \Omega$ generated for solving SAA-EBMMC.

values of SAA-EBMMC and BMMC. Instead, we retrospectively evaluate the expected-value total objective function values associated with BMMC solutions using the same sample of the coverage effectiveness random variables, $\xi^{\omega}, \omega \in \Omega$, generated for SAA-EBMMC. The optimal objective values of SAA-EBMMC are shown in solids lines in Figure 2.3, while BMMC (retrospective stochastic solution) are shown in dashed lines. For both sub-figures, the optimal objective values of SAA-EBMMC increases as the budget value increases. However, the BMMC retrospective expected-value objective function values do not increase monotonically because the BMMC solutions overlook the possibility of coverage failure and do not perform steadily under the scenarios. Given that the SAA-EBMMC solutions are optimal with respect to random coverage failure represented by the scenarios Ω, their objective values are higher or at least equal to objective function values of BMMC (retrospective stochastic solution). The gap between SAA-EBMMC (optimal stochastic solution) and BMMC (retrospective stochastic solution) is the value added by solving the stochastic model.

Next, we test Algorithm 1 on four SAA-EBMMC instances from the datasets with size $(20(|M|) \times$

Table 2.4: Empirical approximation ratios and computational times of Algorithm 1 for solving SAA-EBMMC with independent coverage failures, with comparison to a MIP solver

				Approx. ratio	Time (s)	
$\lvert M\rvert$	$\lvert N\rvert$	$\lvert S\rvert$	$\lvert\Omega\rvert$	Alg. 1	Alg. 1	EBMMC
20	20	10	1,000	100.0%	0.2	35.7
20	20	10	2,000	100.0%	0.2	129.5
20	20	10	3,000	100.0%	0.2	244.2
20	20	10	4,000	100.0%	0.3	433.8
20	20	10	5,000	100.0%	0.3	683.8
20	20	10	8,000	100.0%	0.4	1946.4
20	20	10	10,000	100.0%	0.5	2930.6
50	50	20	1,000	99.5%	1.0	307.1
50	50	20	2,000	99.5%	1.7	1286.7
50	50	20	3,000	100.0%	2.6	3570.5
100	100	50	1,000	100.0%	7.1	894.3
100	100	50	2,000	NA	16.3	NA[1]
100	100	50	3,000	NA	25.6	NA[1]
200	200	100	1,000	NA	73.9	NA[1]
200	200	100	2,000	NA	165.8	NA[1]

1: Gurobi out of memory

$20(|N|) \times 10(|S|)$ - $200(|M|) \times 200(|N|) \times 100(|S|))$. We generate ξ^ω using Python's SciPy package, and the sample size ranges from $1,000$ to $10,000$. We couple each instance with the stochastic data of different sample sizes, and examine the computational performance of Algorithm 1 for solving each instance. Table 2.4 reports the approximation ratios and computational times of Algorithm 1 compared to the MIP solver. Algorithm 1 runs in under 166 seconds across all problem instances considered, while the MIP solver solves only eleven problem instances within the memory limit. Algorithm 1 maintains its high performance in this case of solving SAA-EBMMC, identifying optimal solutions in nine of the eleven instances and near-optimal solutions (within 1%) for the remaining two instances.

2.3.3 Group Cardinality Constraints Solution

Finally, we test the performance of Algorithm 3 for identifying approximate solutions to kEMMCG and its deterministic variant kMMCG. Algorithm 3 achieves a $1/2$-approximation ratio for these two models. We report the empirical performance of Algorithm 3 on the deterministic model kMMCG to study its running time and accuracy.

We first compare the solutions obtained from Algorithm 3 to those obtained by the MIP solver on two medium-sized instances from the datasets ($50(|N|) \times 50(|M|) \times 20(|S|)$ and $100(|N|) \times 100(|M|) \times 50(|S|)$) with varying values of ℓ and k. Given the number of groups ℓ, we randomly sample from the set of mitigations M without replacement to generate subgroups of mitigations $M_i, i = 1, ..., \ell$. We set $k \leq \frac{1}{2}\ell$ because the cardinality constraint might be redundant sometimes when k is too large (e.g., $k \geq \ell$), and the group cardinality constraints might be redundant sometimes when k is too small (e.g., $k \leq 1$). Table 2.5 reports that Algorithm 3 identifies near-optimal solution (within 4%) across all instances. Further testing on a larger dataset (scpnrg1 with $10,000(|N|) \times 10,000(|M|) \times 500(|S|)$) is reported in Table 2.6. It demonstrates for large instances, Algorithm 3 still identifies near-optimal solution (within 3%) in less than 2 seconds while the MIP solver requires up to 124 seconds for those instances that complete within one hour time limit. Gurobi encountered memory issues (out of memory) on the last four instances when ℓ and k become larger.

2.4 Conclusions

In this chapter, we develop insights for prioritizing and deploying security mitigations in a layered defense. We do so by proposing budgeted maximum multiple coverage problems to characterize investment in cybersecurity mitigations with the objective of maximizing multiple coverage that is modeled as a submodular set function. We introduce model variations that take into consideration the possibility of coverage failure, leading to a more robust investment plan. We also consider several other variations motivated by the decision maker's different requirements for selecting security mitigations, including cardinality constraints and group

Table 2.5: Empirical approximation ratios and computational times of Algorithm 3 for solving kMMCG on two instances with varying ℓ and k, with comparison to a MIP solver.

					Approx. ratio	Time (s)	
$\lvert M\rvert$	$\lvert N\rvert$	$\lvert S\rvert$	ℓ	k	Alg. 3	Alg. 3	kMMCG
50	50	20	50	20	100.0%	0.07	0.02
50	50	20	10	5	99.6%	0.03	0.03
100	100	50	100	30	100.0%	0.10	0.04
100	100	50	34	17	99.0%	0.07	0.11
100	100	50	20	10	97.0%	0.06	0.06
100	100	50	10	5	96.7%	0.04	0.08

Table 2.6: Empirical approximation ratios and computational times of Algorithm 3 for solving kMMCG on a large instance from the OR-library with varying ℓ and k, with comparison to MIP solver

					Approx. ratio	Time (s)	
$\lvert M\rvert$	$\lvert N\rvert$	$\lvert S\rvert$	ℓ	k	Alg. 3	Alg. 3	kMMCG
10,000	1,000	500	10	5	98.8%	0.5	24.7
10,000	1,000	500	15	7	97.4%	0.8	40.2
10,000	1,000	500	20	10	98.8%	1.2	124.0
10,000	1,000	500	34	17	NA	2.2	NA[1]
10,000	1,000	500	50	25	NA	4.2	NA[1]
10,000	1,000	500	100	50	NA	11.7	NA[1]
10,000	1,000	500	200	80	NA	22.9	NA[1]

1: Gurobi out of memory

cardinality constraints.

We demonstrate that our problems can be formulated as a submodular maximization problem subject to linear or matroid constraint. We develop approximation algorithms with a polynomial number of objective function evaluations (queries) for each of the problem variation, and prove that guaranteed performance ratios are associated with the approximation algorithms. An optimal $(1-1/e)$ approximation ratio is demonstrated for solving EBMMC as well as its particular cases BMMC, kMMC, and kEMMC. A $1/2$ approximation ratio is identified for solving kMMCG and kEMMCG.

In the computational study, we observe that greedy algorithms identify near-optimal solutions to large scale problem instances, within 5% of the optimal values across all the instances we tested. It suggests that simple greedy algorithms can serve as a preferred option for solving large scale maximum coverage variants. Another benefit of the developed greedy algorithms is that they can identify a set of non-dominated solutions at no additional computational expense.

Our models can be used by decision-makers to select mitigations with an overall budget or cardinality requirement as well as multiple choice limitations. In reality, there could be additional requirements or a mixture of different types of selection requirements. These can be easily adapted into our models and algorithms by removing or adding certain constraints. However, it is not clear if a polynomial-time $(1 - 1/e)$ algorithm exists for solving submodular maximization problems subject to multiple general knapsack or linear constraints. Work is in progress to develop approximation algorithms with guaranteed performance ratios for other problem variants with special structures.

Chapter 3

Robust Methods for Mitigating Risks in Cyber Infrastructure

The expected-value budgeted maximum multiple coverage (EBMMC) model presented in the last chapter provides a solution that performs well on average in the long run, i.e., a solution that is satisfactory in most scenarios when uncertainty regarding mitigation effectiveness arises. However, an expected-value model like EBMMC does not usually provide solutions that prepare the system against worst-case scenarios. It is possible that a combination of mitigations could not prevent vulnerabilities as intended and leave the system unacceptably vulnerable to an extreme scenario. In other words, EBMMC does not model the impact of adversarial attacks. As a result, expected-value solutions might lead to actual situations that are unacceptable for the decision maker.

To address these limitation, this chapter introduces and compares three robust models that extend EBMMC (renamed as MaxExpCoverage for consistency with other models) to capture risk associated with mitigations "failing", including models that maximize the worst case coverage, minimize the worst case regret, and maximize the average coverage in the $(1-\alpha)$ worst cases (conditional value at risk). The central contribution of this chapter is to demonstrate how risk measures can be combined with BMMC models to provide insights into cybersecurity planning.

The advantages of this approach include:

1. The robust methods are more conservative to worst-case risks than an expected-value maximization model, and thereby they hedge against the risks brought by adversarial attacks or disastrous events, which is important in homeland security applications like cybersecurity where extreme events often lead to tremendous loss and damage.

2. By optimizing over the worst case performance of the supply chain, robust optimization models could provide insights into defensive response to adversarial attacks by assuming the adversary (e.g., hackers, criminal groups, nations, terrorists, etc.) is limited to select the worst case attack scenarios. The decision makers can gain practical insights quickly from the robust methods without the need to quantify the adversarial attacks in cyber-infrastructure, which can be very challenging given lack of information (e.g., attacker profiles), and to solve a two-stage interdiction models (e.g., [69, 95]), which can be computationally intensive.

3. Each robust method provides a different perspective into interpreting the worst-case response, which can be employed by the decision makers to entertain the trade-offs and select the solution that best suits their goals. The first two worst case robust methods, i.e., maximizing the worst coverage and minimizing the worst regret, do not require an explicit distribution of the uncertain parameters, which makes them a very practical tool for homeland security problems. The third robust method, maximizing the expected coverage in the $(1-\alpha)$ worst cases, can be seen as a combination of the worst-case risk measures and the expected-value measure. It provides the decision makers the flexibility to obtain a solution with their desired risk preference by adjusting α.

We proceed as follows. In Section 3.1, we first describe the MaxExpCoverage model in [104] and then introduce the robust coverage models that maximizes the worst-case coverage, minimizes the worst-case regret, and maximizes the expected coverage in the $(1-\alpha)$ worst-case, respectively. In Section 3.2, we illustrate the model solutions and insights with a case study. We provide additional computational results conducted on a variety of instances to further demonstrate the difference between proposed models and to show their computational performance, which sheds light on the types of solutions the models could yield in different settings. In Section 3.3, we summarize the discussion in this chapter.

3.1 Models

In this section, we introduce and compare the following four models:

1. a model that maximizes the expected coverage across all scenarios, denoted MaxExpCoverage;
2. a model that maximizes the worst-case coverage across all scenarios, denoted MaxMinCoverage;
3. a model that minimizes the maximal regret across all scenarios, denoted MinMaxRegret;
4. a model that maximizes the conditional expected coverage that does not exceed a pre-specified quantile level in the coverage (CVaR), denoted MaxCVaR.

Cyber attackers exploit vulnerabilities in the IT infrastructure and usually take several exploits to achieve attack goals. In this research, we use attack paths to represent supply chain attacks with multiple nodes on each of them representing the attack steps (exploits). Attack paths can be easily enumerated from an attack tree. Input from collaborators suggests that the size of attack trees for this application is anticipated to be moderate, since there are limited opportunities or access points for influence and control in the supply chains under consideration. Let S be a set of attack paths recognized by SMEs, each of which contains a subset of vulnerability nodes $N_s, s \in S$ with $\bigcup\limits_{s \in S} N_s = N$. Some attack paths may have more strategic importance due to their potential consequences if successful, and therefore, we let a_s captures the importance (weight) of attack path $s \in S$.

Let M be the set of applicable mitigations identified by SMEs, and M_n be the subset of mitigations that cover node $n \in N$. A vulnerability node is said to be protected if it is covered by at least one mitigation. A layered defense is achieved through multiple coverage of an attack path, i.e., covering different nodes in an attack path. We define a general coverage function $f_s(\cdot)$ to quantify the coverage of attack path $s \in S$ with respect to the number of nodes covered in it. We assume that it is non-decreasing and concave, since better security is achieved when more nodes are covered and the marginal benefit from covering more nodes is decreasing. Mitigation coverage may "fail"—meaning that coverage is not realized—due to the dynamic and persistent nature of cyber threats and the limited knowledge SMEs have about its effectiveness. We consider a set of realizations of mitigation effectiveness $|\Omega|$, where the corresponding random variable ξ^{ω}_{mn} is equal to 1 if the coverage of $m \in M$ on node $n \in N$ is effective in scenario ω, and 0 otherwise. We assume each scenario $\omega \in \Omega$ occurs with probability $p^{\omega} \in [0, 1], \omega \in \Omega$ with $\sum_{\omega\in\Omega} p^{\omega} = 1$. Additionally, we associate each mitigation $m \in M$ with a cost b_m that captures its deployment and implementation. Let the total budget for selecting mitigations be B.

The attack paths can be constructed with the aid of SMEs, which yields $N, S, N_s, s \in S$, and $a_s, s \in S$. Likewise, the mitigations that control each node, $M_n,\ n \in N$, can be obtained by interviewing SMEs. Information collected by experts can be used to construct a set of realizations for the potential effectiveness $\xi^{\omega}_{mn}, \omega \in \Omega$ and their associated probabilities each realization occurs $p^{\omega} \in [0, 1], \omega \in \Omega$ with $\sum_{\omega\in\Omega} p^{\omega} = 1$, potentially by sampling. The set of mitigations M, their costs $b_m,\ m \in M$ and total budget B can be obtained from federal decision makers, manager, and experts who are familiar with the mitigation options available and have estimates of their associated costs.

All models use a common set of decision variables, which are defined as follows.

- x_m = 1 if mitigation $m \in M$ is chosen, and 0 otherwise;
- z^{ω}_n = 1 if node $n \in N$ is covered by at least one selected mitigation under scenario $\omega \in \Omega$, and 0 otherwise;
- y^{ω}_s = the number of nodes in attack path $s \in S$ that are covered under scenario $\omega \in \Omega$.

The expected coverage maximization model, MaxExpCoverage, which corresponds to the SAA-EBMMC model in [104], is formulated below.

MaxExpCoverage:

$$\max \quad \sum_{\omega \in \Omega} p^{\omega} \sum_{s \in S} a_s f_s(y_s^{\omega}) \tag{3.1}$$

$$\text{s.t.} \quad y_s^{\omega} \leq \sum_{n \in N_S} z_n^{\omega}, \quad s \in S, \omega \in \Omega \tag{3.2}$$

$$z_n^{\omega} \leq \sum_{m \in M_n} \xi_{mn}^{\omega} x_m, \quad n \in N, \omega \in \Omega \tag{3.3}$$

$$\sum_{m \in M} b_m x_m \leq B \tag{3.4}$$

$$x_m \in \{0,1\}, \quad m \in M \tag{3.5}$$

$$z_n^{\omega} \in \{0,1\}, n \in N, \omega \in \Omega \tag{3.6}$$

The objective function in (3.1) is the expected value of the total coverage of all attack paths across all scenarios. This nonlinear function can be easily linearized by adding new variables and constraints, see [104] for details. Constraint set (3.2) computes the number of nodes covered in each attack path, and constraint set (3.3) states that a node is covered if there exists at least one selected mitigation that covers it. Constraint (3.4) is the budget constraint. Constraint sets (3.5) and (3.6) enforce $\mathbf{x}$ and $\mathbf{z}$ variables to be binary.

MaxExpCoverage returns a solution that performs well on expected, i.e., acceptable in most scenarios, in the presence of uncertainty. However its actual performance can be unacceptable to decision-makers for some realizations of ξ as it might compromise the coverage in a few scenarios to achieve a better expected coverage. Therefore, we are motivated to identify robust solutions that avoid worst-case performance. The following robust models address the uncertainty from different perspectives and identify solutions that plan for different risk situations.

In the first robust model, we aim to identify a solution that has the best worst-case performance across all scenarios. Denote variable u as the minimal coverage across all scenarios. We present the following model, MaxMinCoverage, that maximizes the worst-case coverage:

MaxMinCoverage:

$$\begin{aligned}
\max \quad & u && (3.7)\\
\text{s.t.} \quad & u \leq \sum_{s\in S} a_s f_s(y_s^\omega), \quad \forall \omega \in \Omega && (3.8)\\
& (3.2) - (3.6)
\end{aligned}$$

The minimal coverage u across all scenarios, as defined by constraints (3.8), is maximized in the objective (3.7). This measure is often considered to be overly pessimistic by evaluating only the most extreme scenario, regardless of the coverage in other scenarios. We list two examples when MaxMinCoverage is overly pessimistic. First, if the worst-case scenario occurs with an extremely small probability but requires an expensive mitigation to cover, MaxMinCoverage would suggest selecting this mitigation even when the coverage in most scenarios is compromised. Second, considering the case when there are several equivalent worst-case scenarios that employ different sets of mitigations, if the total budget is not enough to select all required mitigations, the resulting minimal coverage is not improved after exhausting the entire budget. Meanwhile, coverage in most scenarios is neglected in the decision process.

While MaxMinCoverage allocates mitigations to improve the worst case scenarios, these scenarios might not "demand" the most defensive resources. Considering a case when the worst case coverage is only improved by a small amount in a MaxMinCoverage solution, it is likely that other scenarios can benefit more from the same amount of budget. This is the motivation for the following robust model that aims to prioritize mitigations towards the scenarios with the greatest regret. Regret is defined for each scenario as the difference between the coverage of a solution in that scenario and the coverage for that single scenario in the optimal solution. To calculate the regret, we need to pre-solve the optimal solution for each scenario. The subproblem is a deterministic MaxCoverage problem that can be solved quickly by a general purpose solver [104]. Let $g^*(\omega)$ capture the optimal subproblem objective function value when scenario $\omega \in \Omega$ is realized, and let r capture the maximal regret across all scenarios. This new model, MinMaxRegret, minimizes the maximal regret across all scenarios:

MinMaxRegret:

$$\begin{aligned} \min \quad & r && (3.9)\\ \text{s.t.} \quad & g^*(\omega) - \sum_{s \in S} a_s f_s(y_s^\omega) \leq r, \quad \forall \omega \in \Omega && (3.10)\\ & (3.2) - (3.6) \end{aligned}$$

The maximal regret, defined in constraints (3.10), is minimized in the objective (3.9). The left hand size of constraint set (3.10) computes the regret for each scenario, i.e., the difference between the coverage of a solution in that scenario and the coverage of the optimal solution for that single scenario, which is different from MaxMinCoverage where mitigations are allocated to improve the coverage of the worst case scenarios, MinMaxRegret allocate mitigations to the scenarios with the greatest regret, i.e., the scenarios whose coverage can be improved the most given the same budget. MinMaxRegret is related to MaxMinCoverage in that they are both worst-case risk models. They neglect the tail distribution of the uncertain events and the magnitude of the worst-case scenarios, both of which may be important to decision makers under certain circumstances. To address this issue, we introduce the conditional value at risk (CVaR) measure. CVaR considers the worst case events, which is overlooked by MaxExpCoverage, and also assess the coverage value in the tail distribution of the uncertainty, which is overlooked by MaxMinCoverage and MinMaxRegret.

Value at risk (VaR) and CVaR are risk measures of loss functions widely used in the finance and insurance industries. In a traditional minimization context, VaR$_\alpha$ is the α-quantile of the cost distribution and CVaR$_\alpha$ is the conditional expected cost exceeding VaR$_\alpha$, where α is a quantile specified by the decision maker. As the financial market fluctuates very often and can be highly unpredictable, VaR and CVaR help decision makers manage the likelihood of loss caused by certain types of risks. This makes VaR and CVaR an useful tool for managing cyber risks because cyber threats evolves constantly and are also hard to predict. VaR helps to mark the boundary between normal and extreme events, in addition to which CVaR assesses the conditional coverage in the tail distribution and provides a less conservative robust solution.

Given the maximization context in this research, we redefine VaR_α as the $(1-\alpha)$-quantile of the coverage distribution and CVaR_α as the expected coverage that does not exceed VaR_α. Their formal definitions are presented as follows.

$$\begin{aligned}
\text{VaR}_\alpha[g(\mathbf{x},\xi)] &= \sup\Big\{\eta|\mathbb{P}(g(\mathbf{x},\xi)\geq\eta)\geq\alpha\Big\}\\
\text{CVaR}_\alpha[g(\mathbf{x},\xi)] &= \mathbb{E}[g(\mathbf{x},\xi)|g(\mathbf{x},\xi)\leq \text{VaR}_\alpha(g(\mathbf{x},\xi))]\\
&= \sup_\eta\Big\{\eta-\frac{1}{1-\alpha}\mathbb{E}[(\eta-g(\mathbf{x},\xi))_+]\Big\}
\end{aligned}$$

CVaR has many advantages compared to VaR. First, CVaR quantifies risks beyond VaR, and is coherent. More importantly, CVaR can be linearized and readily solved by a linear programming solver, which popularizes its application in operations research applications. We refer the readers to two seminal papers on CVaR optimization for more details [84, 85]. We define additional variables $\nu^\omega, \omega\in\Omega$ and η that help define and linearize CVaR in the following model, MaxCVaR, that maximizes CVaR_α of the coverage across all scenarios.

MaxCVaR:

$$\max \quad \eta-\frac{1}{1-\alpha}\sum_{\omega\in\Omega}p^\omega\nu^\omega \tag{3.11}$$

$$\text{s.t.} \quad \nu^\omega\geq\eta-\sum_{s\in S}a_sf_s(y_s^\omega),\quad \forall\omega\in\Omega \tag{3.12}$$

$$\nu^\omega\geq 0,\quad \forall\omega\in\Omega \tag{3.13}$$

$$(3.2)-(3.6)$$

The objective function in (3.11) is the linearized CVaR, which is defined by constraints (3.12) and (3.13). α is an user defined parameter. A larger α implies a less risk-neutral solution. Notice that MaxCVaR is equivalent to MaxMinCoverage when $\alpha=1$, and it is equivalent to MaxExpCoverage when $\alpha = 0$. The decision makers can adjust α according to their risk preferences to obtain solutions with different robustness. MaxCVaR offers more flexibility to the decision process as compared to the previous three models.

3.2 Illustrative Examples

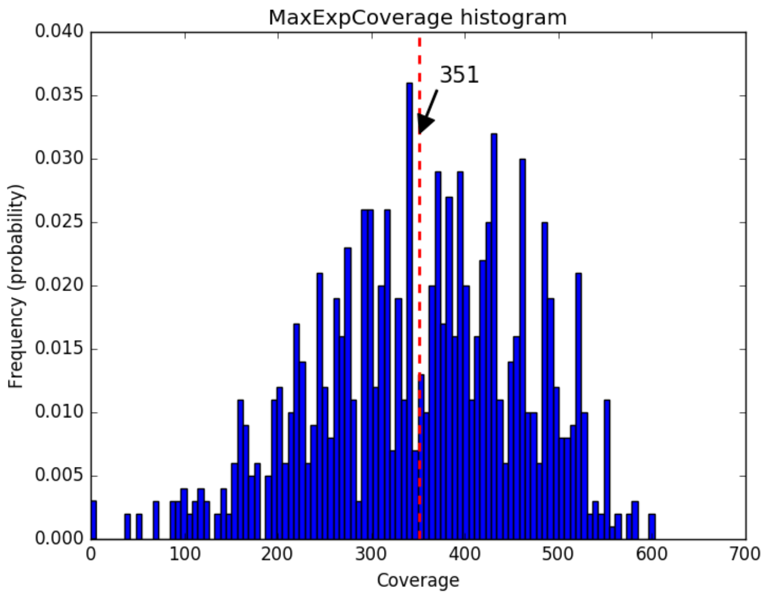
In this section, we compare and illustrate models insights with several examples. In the first subsection, we conceptually compare the four models visually with histograms that display the empirical distributions over all scenarios associated with the optimal solution of the four models. Next, we present a case study based on realistic data to analyze the model solutions. All models were programmed in Python 2.7.10 and solved with Gurobi 6.0. The data instances were run on an Intel Core i5-3470 CPU at 3.20 GHz with 4 GB of RAM. A time limit was set to one hour.

3.2.1 Visual Illustration

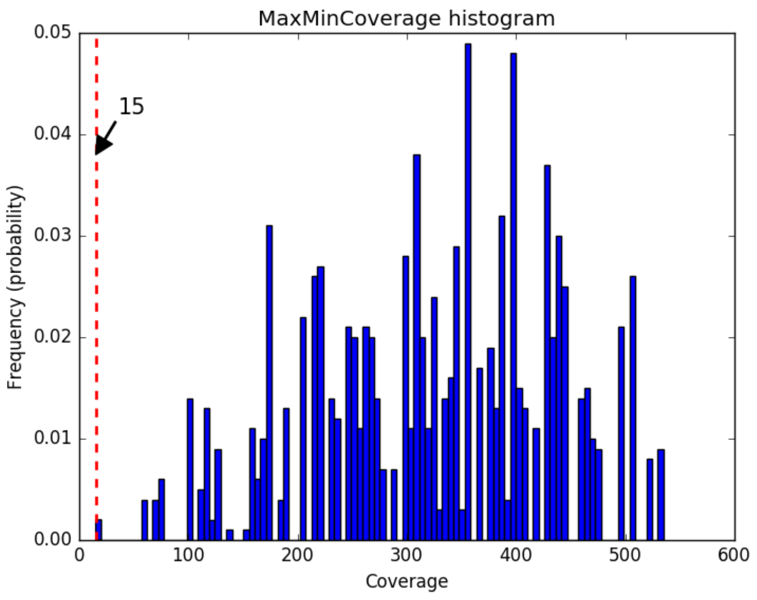
Figure 3.1 provides a visual demonstration of the model solutions by illustrating the empirical histograms over the scenarios for each model. We construct a data instance for demonstrative purposes, with $|M| = 20, |N| = 20, |S| = 10, |\Omega| = 1,000$. We specify that each mitigation can cover up to three nodes and use a pseudo uniform random number generator to determine the list of nodes covered by each mitigation. We set the upper bound on the number of nodes in each attack path to five and randomly generate a list of nodes in each attack path. Parameters $a_s, s \in S$ and $b_m, m \in M$ are both generated using a pseudo uniform random number generator within ranges $(0, 10)$ and $(0, 1)$, respectively. The budget B is set to 5% of the total cost of all mitigations. The coverage function $f_s(\cdot)$ for attack path $s \in S$ is set to $f_s(\cdot) = -(y_s^\omega)^2 + 2|N_s|y_s^\omega$ as a function of y_s, which is non-decreasing and concave as required. A sample $\xi_{mn}^\omega, m \in M, n \in N, \omega \in \Omega$ with $|\Omega| = 1,000$, which captures coverage effectiveness, is drawn from the Bernoulli distribution with success probability 0.5.

The MaxExpCoverage (a) and MaxMinCoverage (b) histograms illustrate the empirical distribution of the attack path coverage, $\sum_{s \in S} a_s f_s(\mathbf{y}^\omega)$, over $|\Omega| = 1,000$ scenarios, that are associated

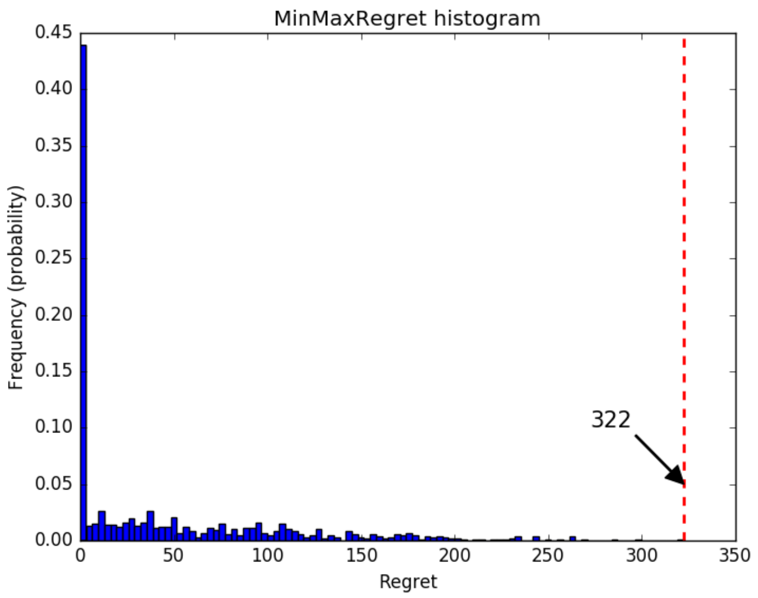
Figure 3.1: Empirical distribution functions (histograms) associated with the optimal solution of MaxExpCoverage, MaxMinCoverage, MinMaxRegret and MaxCVaR, respectively.



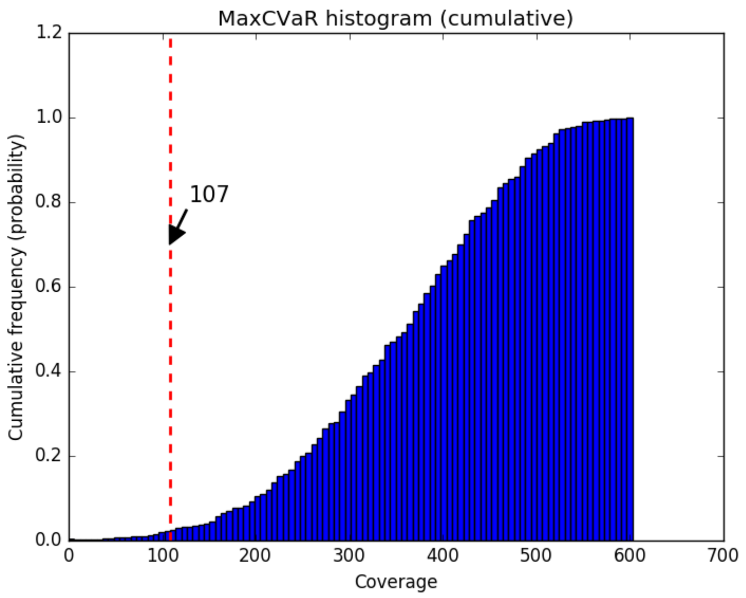
(a) Histogram of coverage associated with the optimal MaxExpCoverage solution across all scenarios. The optimal expected coverage is 351.



(b) Histogram of coverage associated with the optimal MaxMinCoverage solution across all scenarios. The optimal worst-case coverage is 15.



(c) Histogram of regret associated with the optimal MinMaxRegret solution across all scenarios. The optimal maximal regret is 322.



(d) Cumulative histogram of coverage associated with the optimal MaxCVaR solution across all scenarios with $\alpha = 0.95$. The optimal CVaR value is 107.

with their optimal solutions, respectively. Comparing these two histograms, we can see the trade-off clearly: MaxExpCoverage(a) is more aggregated overall and achieves the best expected coverage 351, however the worst case coverage associated with its optimal solution is 0. MaxMin-Coverage (b), on the other hand, has a smaller expected coverage value, 325, associated with its optimal solution, and has a more scattered empirical histogram. However, its worst case coverage, 15, is better than that of MaxExpCoverage. In figure (c), we plot the empirical distribution of the regret, $g^*(\omega) - \sum_{s \in S} a_s f_s(\mathbf{y}^\omega)$, over all scenarios, that are associated with the optimal MinMaxRegret solution. The tallest bar in the leftmost indicates that about 44% of the scenarios have zero regret, which means the coverage achieved in those scenarios is the best they can achieve with the current set of available mitigations M. Most scenarios have regret smaller than 100. The worst case regret is 322, which is the best maximal regret achieved with the optimal MinMaxRegret solution. Note that the scenario that achieves the worst regret in (c) is different from the scenario that achieves the worst coverage in (b). More evident results are presented in the next subsection. Finally, we plot the cumulative histogram of the coverage, $\sum_{s \in S} a_s f_s(\mathbf{y}^\omega)$, over all scenarios associated with optimal MaxCVaR solution, in (d). We set $\alpha = 0.95$ in this example. Different from the previous two worst case risk models, MaxCVaR maximizes the expected coverage in the 5% tail of the cumulative density function, i.e., $\mathbb{E}[g(\mathbf{x}, \xi) | g(\mathbf{x}, \xi) \leq \text{VaR}_{0.95}(g(\mathbf{x}, \xi))]$. The optimal CVaR value is 107 as marked in the plot. We can see that the tail is relatively shorter and "thinner" compared to the other part of the distribution, indicating that MaxCVaR works on improving the tail performance of the solution.

3.2.2 Case Study

In this subsection, we compare the robust methods and illustrate the insights of their solutions with a case study. We base our study on two main datasets, each of which contains three instances, that consist of attack paths and mitigation controls. The first set of data is from the book by Shostack on attack modeling [93], where in the appendix the author presents 15 STRIDE attack trees, such as spoofing a client or tampering with a data flow, as well as mitigation approaches associated with each attack. The second set of data is randomly generated, following the same rule as presented in the last subsection for generating the example instance. As public data of attack tree and mitigation for Federal IT supply chain attacks are scarce, we believe these two sets of data are sufficient to demonstrate the insights that the robust methods bring into mitigation selection compared to an expected-value model.

We first describe the construction of the first dataset using the attack tree examples introduced by Shostack [93]. The sizes of the 15 attack trees are relatively small, with roughly $10-20$ nodes on each tree, feasible attack paths can be easily generated from each tree, which gives us S, N, and $Ns, s \in S$. Shostack [93] also provides applicable security controls believed by the author that protects the vulnerabilities in each tree, giving us the data for M and $M_n, n \in N$. We note that these mitigations have overlapping capacities over some nodes, which is consistent with our model assumption. Shostack [93] does not provide information on the weights of attack path ($a_s, s \in S$), mitigation costs ($b_m, m \in M$), and budget (B), so we generate these parameters in a random manner similar to that in Section 3.2.1. The listed mitigations in this dataset only affect the end node (leaf) of the attack path. Therefore, there is only one node “active” on each path. Our problem is reduced to a single coverage case with this dataset. Accordingly, we set the coverage function to $f_s(y_s^\omega) = a_s y_s^\omega$, a linear function commonly used in maximum coverage models in the literature. This dataset contains 52 mitigations, 103 nodes, and 103 attack paths in total. We create three instances with different stochastic data, i.e., three set of samples ξ_{mn}^ω drawn independently from Bernoulli distribution with success probability 0.5 and size $|\Omega| = 100$. A larger number of scenarios is not preferred here as the worst case might be too extreme to provide useful insights into decision making.

We randomly create a second set of instances to assess multiple coverage and additional instances. To generate these instances, we follow the same rule introduced in Section 3.2.1 to create three different instances with size $|M| = 50, |N| = 150, |S| = 50, |\Omega| = 100$, and use the same concave coverage function.

We compare the models by retrospectively evaluating all objective function values, i.e., the expected coverage, the minimal coverage, the maximal regret, and the CVaR, associated with their optimal solutions, respectively. To achieve this, we first solve the four models to obtain their the optimal solutions, and then calculate each measure associated with the solutions, respectively. We choose three different quantiles for the MaxCVaR model, $0.95, 0.90, 0.85$. Table 3.1 and Table 3.2 show results for three instances in dataset one and the other three instances in dataset two, respectively. Rows in the table represent different models and Columns correspond to different measure evaluations. The bold values on the diagonal from top-left to bottom right report the optimal objective function values, and the other values report the other performance measures associated with the models. For example, in Table 3.1a, the MaxExpCoverage solution has the largest expected coverage value 136.0 compared to that of the other model solutions in that column, which vary from 124.2 to 132.9.

Next, we examine the trade-offs between different measures and contrast their results to shed light on the insights they provide for decision-making. First, we compare the expected-value model with the two worst-case robust models. We notice that MaxExpCoverage solutions usually have poor worst-case coverage due to the fact that the model does not optimize over any worst case scenarios. Similarly, MaxMinCoverage solutions usually have a relatively low expected-coverage as the model overlooks average performance over all scenarios. For example, in Table 3.1a, the minimal coverage for the MaxExpCoverage solution is 76.8, which is the lowest among all model solutions, at the same time expected coverage of the MaxMinCoverage solution is 124.2, which is also the lowest while all the other model solutions have relatively close value towards the optimal solution. The MaxMinCoverage solution uses all mitigation resources on improving the minimal coverage, which would be beneficial if the decision maker is extremely risk averse. The other worst case robust method, MinMaxRegret, optimizes over the worst case regret. The difference between the two measures are demonstrated in the results. The maximal regret of the MaxMinCoverage solution is often the worst among all models across the instances in the two tables. Similarly, the minimal coverage of the MinMaxRegret solutions are also non-satisfactory. This implies the scenario with the worst coverage is usually not the scenario with the worst regret. Additionally, we observe some resemblance between the solution performance of MaxExpCoverage and MinMaxRegret. Their solutions have close performance when evaluated with each other's measure. For example, in Table 3.1c, the MinMaxRegret solution achieves an expected coverage of 142.9, close to the optimal coverage of 145.3, and MaxExpCoverage solutions has a maximal regret 34.8, which is closest to the optimal maximal regret compared to the other model solutions.

Decision makers responsible for investment planning usually require flexibility from the solutions. They may have multiple goals to achieve and have to decide on trade-offs. The first three models lack such flexibility, which is fulfilled by the introduction of the MaxCVaR model, which balances expected-value and worst-case optimization through a quantile parameter α. We can see clear evidence from the results by comparing the solution performance of three MaxCVaR models with different α values, and with the other models. We use the results in Table 3.2c as evidence. $\text{MaxCVaR}_{0.95}$ maximizes the expected coverage in the 5% worst-case scenarios, i.e., the five worst-case scenarios given the total 100 scenarios. Its solution is identical to that of MaxMinCoverage which maximizes the one worst-case scenario. When α decreases to 0.90, we obtain a slightly more risk-neutral MaxCVaR solution, the expected coverage of which increases by 34.2, and the maximal regret of which is improved by 56.4. However, its minimal coverage is compromised by decreasing to 58.3. As α further decreases to 0.85, the MaxCVaR solution is identical to that of MaxExpCoverage, indicating that maximizing the expected coverage in the 15 worst-case scenarios requires the same portfolio of mitigations as maximizing the expected coverage over all scenarios. In general, when α is close to 1, MaxCVaR solution is more risk conservative and behaves more closely to the MaxMinCoverage solution. As α decreases, the MaxCVaR solutions become more risk-neutral and exhibit more resemblance towards MaxExpCoverage solutions.

Finally, we demonstrate how to achieve solutions with a desired level of security by adjusting the parameter α in the MaxCVaR model, since α is a user-defined parameter that reflects the risk attitudes of the decision maker. We investigate the insights on mitigation selection strategy that the MaxCVaR solution provides by varying the values of α. Figure 3.2 demonstrates how the choice of α shapes the empirical cumulative distribution functions of the coverage across the scenarios. We solve a new medium-sized data instance with $|M| = 50, |N| = 50, |S| = 20, |\Omega| = 200$ with three different α values, $0.99, 0.95, 0.90$. We then compute the coverage of all

Table 3.1: Cross-comparison of four model solutions on cyber attack data for three instances in dataset one

	Exp. cov.	Min. cov.	Max. reg.	$\text{CVaR}_{0.95}$	$\text{CVaR}_{0.90}$	$\text{CVaR}_{0.85}$
MaxExpCoverage	**136.0**	76.8	34.1	87.1	96.2	101.4
MaxMinCoverage	124.2	**85.0**	51.9	88.8	93.5	97.2
MinMaxRegret	131.7	82.4	**30.4**	91.0	95.8	100.2
$\text{MaxCVaR}_{0.95}$	131.3	84.0	41.8	**92.1**	97.2	100.8
$\text{MaxCVaR}_{0.90}$	130.7	79.2	36.8	89.4	**97.5**	101.2
$\text{MaxCVaR}_{0.85}$	132.9	79.6	35.8	88.6	97.2	**101.9**

(a) $|M| = 52, |N| = 103, |S| = 103, |\Omega| = 100$

	Exp. cov.	Min. cov.	Max. reg.	$\text{CVaR}_{0.95}$	$\text{CVaR}_{0.90}$	$\text{CVaR}_{0.85}$
MaxExpCoverage	**140.3**	95.2	39.0	101.1	105.8	109.1
MaxMinCoverage	134.5	**99.0**	47.8	100.5	103.1	104.9
MinMaxRegret	139.5	89.4	**32.6**	99.9	104.5	108.7
$\text{MaxCVaR}_{0.95}$	139.1	95.0	39.3	**104.3**	107.6	10.3
$\text{MaxCVaR}_{0.90}$	139.1	95.0	39.3	104.3	**107.6**	110.3
$\text{MaxCVaR}_{0.85}$	139.1	95.0	39.3	104.3	107.6	**110.3**

(b) $|M| = 52, |N| = 103, |S| = 103, |\Omega| = 100$

	Exp. cov.	Min. cov.	Max. reg.	$\text{CVaR}_{0.95}$	$\text{CVaR}_{0.90}$	$\text{CVaR}_{0.85}$
MaxExpCoverage	**145.3**	98.8	34.8	105.2	108.5	111.1
MaxMinCoverage	134.8	**102.8**	41.0	104.5	105.9	107.5
MinMaxRegret	142.9	101.2	**27.1**	104.5	109.0	112.3
$\text{MaxCVaR}_{0.95}$	138.4	99.8	43.8	**109.2**	112.0	113.6
$\text{MaxCVaR}_{0.90}$	143.2	100.4	37.1	108.0	**112.5**	115.0
$\text{MaxCVaR}_{0.85}$	143.2	100.4	37.1	108.0	112.5	**115.0**

(c) $|M| = 52, |N| = 103, |S| = 103, |\Omega| = 100$

attack paths, $\sum_{s\in S} a_s f_s(\mathbf{y}^\omega)$, associated with the optimal solution corresponding to each α value, respectively. The cumulative histograms of the coverage across all scenarios are then plotted for each α value as shown in Figure 3.2. As mentioned earlier, a smaller value of α in MaxCVaR leads to a more risk-neutral policy, which is demonstrated in the figure. In general, these plots are useful for decision makers to adjust the tail distribution by varying theα value until a desired level of security is achieved.

Table 3.2: Cross-comparison of four model solutions on randomly generated data for three instances in dataset two

	Exp. cov.	Min. cov.	Max. reg.	$\text{CVaR}_{0.95}$	$\text{CVaR}_{0.90}$	$\text{CVaR}_{0.85}$
MaxExpCoverage	**1673.0**	991.5	949.2	1091.7	1167.8	1230.3
MaxMinCoverage	1595.8	**1011.1**	1268.9	1095.3	1136.0	1176.3
MinMaxRegret	1673.0	991.5	**949.2**	1091.7	1167.8	1230.3
$\text{MaxCVaR}_{0.95}$	1595.8	1011.1	1268.9	**1095.3**	1136.0	1176.3
$\text{MaxCVaR}_{0.90}$	1656.8	664.5	1276.1	1064.3	**1176.7**	1229.7
$\text{MaxCVaR}_{0.85}$	1673.0	991.5	949.2	1091.7	1167.8	**1230.3**

(a) $|M| = 50, |N| = 150, |S| = 50, |\Omega| = 100$

	Exp. cov.	Min. cov.	Max. reg.	$\text{CVaR}_{0.95}$	$\text{CVaR}_{0.90}$	$\text{CVaR}_{0.85}$
MaxExpCoverage	**2131.3**	1475.7	270.0	1569.2	1638.3	1694.8
MaxMinCoverage	2084.3	**1524.7**	334.4	1586.0	1642.6	1680.3
MinMaxRegret	2131.3	1475.7	**270.0**	1569.2	1638.3	1694.8
$\text{MaxCVaR}_{0.95}$	2084.3	1524.7	334.4	**1586.0**	1642.6	1680.3
$\text{MaxCVaR}_{0.90}$	2096.7	1501.9	434.4	1583.0	**1670.4**	1711.1
$\text{MaxCVaR}_{0.85}$	2096.7	1501.9	434.4	1583.0	1670.4	**1711.1**

(b) $|M| = 50, |N| = 150, |S| = 50, |\Omega| = 100$

	Exp. cov.	Min. cov.	Max. reg.	$\text{CVaR}_{0.95}$	$\text{CVaR}_{0.90}$	$\text{CVaR}_{0.85}$
MaxExpCoverage	**1296.4**	722.3	289.0	862.1	930.3	973.2
MaxMinCoverage	1252.3	**819.9**	393.6	888.8	921.8	942.4
MinMaxRegret	1296.4	722.3	**289.0**	862.1	930.3	973.2
$\text{MaxCVaR}_{0.95}$	1252.3	819.9	393.6	**888.8**	921.8	942.4
$\text{MaxCVaR}_{0.90}$	1286.5	761.6	337.2	886.9	**938.5**	966.4
$\text{MaxCVaR}_{0.85}$	1296.4	722.3	289.0	862.1	930.3	**973.2**

(c) $|M| = 50, |N| = 150, |S| = 50, |\Omega| = 100$

3.3 Conclusions

Uncertainties are embedded in cybersecurity planning given the dynamic feature of cyber attacks and the limited knowledge SMEs have regarding attacker profiles and mitigation approaches. In this chapter, we address one of the main uncertainties associated with mitigation effectiveness. We propose three robust methods as an extension to the expected-value budgeted maximum multiple coverage (MaxExpCoverage) model introduced in Zheng et al. [104] to investigate robust solutions that hedge against worst case risks. An expected-value solution performs well

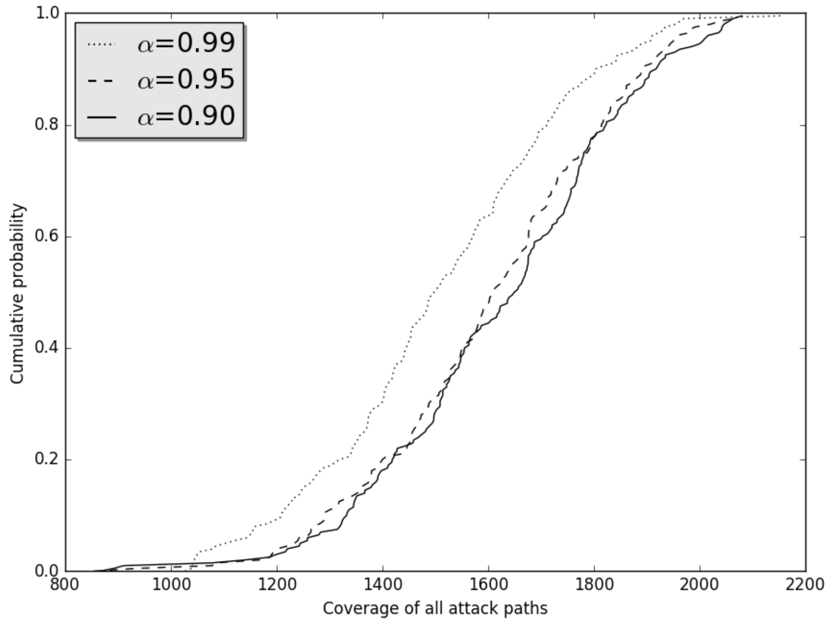


Figure 3.2: Empirical cumulative distribution function of the coverage associated with the optimal MaxCVaR solution for different α values

on average across all scenarios but might be unacceptable in certain circumstances. We first introduce two worst-case robust optimization models, MaxMinCoverage and MinMaxRegret, that optimize the worst-case coverage and the worst-case regret, respectively. Next, we introduce a more flexible model that maximizes the expected coverage in the $(1 - \alpha)$ worst-case scenarios, combining expected-value maximization and worst-case optimization. By varying α, the decision maker with different risk preferences can achieve solutions that are either more risk conservative (with α closer to 1) or more risk neutral (with α closer to 0).

We compare the four models, MaxExpCoverage, MaxMinCoverage, MinMaxRegret, and Max-CVaR, and investigate their insights through a case study that consists of real cyber attack data and randomly generated data. We demonstrate the trade-offs between different models by retrospectively evaluating their solutions with different measures. We also demonstrate how MaxCVaR can be used to achieve solutions with different robustness by varying α. Our study provides decision makers in cybersecurity planning with analytical tools for managing risks in IT supply chains. They can use this robust and stochastic optimization framework to obtain solutions that meet with their budget requirement and the level of security they aim to achieve.

Modeling the decision framework against adaptive adversaries in IT supply chains is an important next step given the drastic increase in number, scale and impact of adversarial attacks. The robust methods proposed in this chapter can be interpreted as modeling an adversary who chooses the minimal coverage attack (MaxMinCoverage), or the maximal regret attack (MinMaxRegret), or an attack in the $(1-\alpha)$ worst-cases. An important extension to this framework explicitly models the interaction between decision makers and adversarial attackers with bi-level programming methodology.

Chapter 4

Interdiction Models for Delaying Adversarial Attacks Against Critical Infrastructure

Chapter 2 and 3 propose an optimization framework based on maximum multiple coverage models that prioritizes security investment in mitigations to reduce generic cyber vulnerabilities in critical infrastructure. They formulate threats as attack paths, each of which contains a series of exploits. They quantify the coverage of an attack path based on the number of vulnerabilities (nodes) on it that are "covered" by mitigations and argue that an improved security of the system and a layered defense is achieved by covering an attack path multiple times. However, they do not explicitly consider risks from adaptive adversaries. We build upon these works to explicitly formulate the interactions between a defender and multiple attackers using Stackelberg game models.

In this chapter, we propose new bilevel interdiction models that investigate how to prioritize security investment to maximally compromise (delay) adversarial attacks. More specifically, We address the following salient issues associated with adversarial attacks and mitigation controls: i) we propose the use of a structured attack graph to represent threat scenarios in response to

NIST recommendations; ii) we consider multiple adversaries instead of one adversary in our decision analysis framework, allowing for a defensive strategy across multiple simultaneous attacks; iii) we address the uncertainty in mitigation effectiveness by incorporating random delay times for attack exploits "covered" by selected mitigations.

A supply chain attack usually starts with an initial intrusion into the system's "weakest" component or link, followed by a series of exploits, and eventually it reaches its attack goal if successful. This is in similar spirit to finding a feasible "path" through a network. In classic network vulnerability analysis, security analysts use an attack tree or graph [90, 65] to characterize attacks against the system, where nodes represent attack states and arcs correspond to transition of states fulfilled by attack activities. A path from root to leaf represents a likely attack through the system. While an attack graph is a tool mainly used for detection and forensics by security analysts, it is a powerful tool to organize discovered vulnerabilities and visualize their dependencies (sequences), which provides a structured approach to represent attack scenarios. Therefore, we use an attack graph structure to represent supply chain attacks, with corresponding adaptation and modifications.

In a graph structure, arcs represent exploits of vulnerabilities that require to be carried out to complete an attack. We assign a completion time to each arc, representing the effort or difficulty of an exploit. Time is an important issue in this application, since it could take months or years for a vulnerability to be discovered and exploited [73]. The exploits follow precedence relationships, i.e., an exploit can start if all its precedent exploits have been completed. Within this context, we define an attacker's objective as completing an attack as soon as possible (i.e., with the minimum effort), which corresponds to finding a "critical path" in the attack graph. We can thus view an attack as a project with a graph representation, and the "critical path" of this project defines the fastest completion time of an attack through the system. IT infrastructure, especially those supporting federal organizations, often face threats from various adversaries. We incorporate this issue into our framework by formulating multiple adversaries, each of which possesses an independent attacker's problem. Given that different adversaries might have different knowledge about the supply chain vulnerabilities and different capacities, we create attack graphs with topology and parameters specific to each attacker.

Mitigations interact with attack graphs by delaying the completion of individual exploits (arcs), i.e., increasing the difficulty of exploits. We define the defender's objective as delaying the total weighted completion time of all adversarial attacks as much as possible by deploying a set of cost-effective mitigations. Delaying adversarial attacks reduces adversarial risk by increasing the time until there is a successful attack, which allows decision makers to examine vulnerabilities in the supply chain and develop new defenses against evolving cyber risks. Selecting mitigations is often subject to certain organizational requirements, such as an overall budget constraint. We consider an additional constraints that require the selection of at most one mitigation from a pre-defined subgroup of mitigations, called multiple-choice constraints. This is motivated by a practical concern that some mitigations have conflicting effects and cannot be implemented together. For example, a mitigation that suggests replacing an untrustworthy vendor and a mitigation that suggests improving this vendor's security procedure, cannot be selected together. Finally, uncertainty regarding mitigation effectiveness is an inherent issue in cybersecurity planning. We represent the uncertain mitigation effectiveness as a continuous random variable that captures the uncertain amount of time an exploit can be delayed by a mitigation that covers it. In a stochastic model variant, the expected value of the total weighted completion time over all attacks is maximized, yielding a solution that performs well on average when mitigation effectiveness is uncertain to decision makers.

In summary, this chapter makes the following contributions to the literature:

1. We study a critical challenge faced by decision makers regarding how to protect IT infrastructure from adversarial attacks, and we propose optimization models that investigate how to select a set of cost-effective mitigations to maximally delay these attacks, when there exist multiple adversaries and uncertainty regarding mitigation effectiveness.

2. We propose new Stackelberg game models for delaying adversarial attacks against IT infrastructure, including a deterministic max-min bilevel programming model for delaying

multiple attack projects and a stochastic max-min model variant that incorporates uncertain delay times.

3. We propose a Lagrangian heuristic that identifies near-optimal solutions in a computationally efficient manner. The models are reformulated as nested max-max problems that can be decomposed into several smaller subproblems by relaxing some of the linking constraints. We calculate the lower bound with a feasible solution recovering procedure and update the lower bound along with the upper bound via subgradient optimization. We also employ an alternative method based on the perfect information solution for calculating the upper bound that improves the Lagrangian upper bound.

4. Computational results demonstrate that the Lagrangian heuristic is efficient in identifying "good" feasible solutions. Additionally, it outperforms a general-purpose solver on medium and large instances by finding better feasible solutions in significantly shorter amount of time.

We proceed as follows. In Section 4.1, we describe the problem and present a deterministic and a stochastic max-min interdiction model for delaying multiple adversarial projects. We also show the max-max reformulation of the models based on dualization. In Section 4.2, we describe a Lagrangian heuristic that identifies lower and upper bounds to the proposed models. In Section 4.3, we demonstrate the computational effectiveness and efficiency of the Lagrangian heuristic as compared to a general mixed-integer programming solver. Section 4.4 summarizes this chapter.

4.1 Models

In this section, we present the max-min formulation of the deterministic and stochastic interdiction models for delaying adversarial attacks. We also present the corresponding max-max formulation of the models, based upon which the solution approach is developed. As discussed in the previous section, we characterize IT supply chain attacks as projects carried out by adversaries and formulate attacks based on the critical path method [56]. For consistency, we adopt project management notation for the attackers' problems. In this chapter, we use attacker and adversary interchangeably.

The IT infrastructure faces threats from various attackers. The attackers could be individuals, criminal groups, nations, or terrorists. They have their differences in their knowledge and perception of supply chain vulnerabilities, goals, and capacities. Therefore, it is important to represent each attacker with an attack graph of topology and parameters specific to its profile. We assume there is a set of projects P carried out by different adversaries that represents a set of attacks. We use a classic "activity-on-arc" graph to represent attack project for each adversary $p \in P$. For each $p \in P$, we define a directed acyclic graph (DAG) $G_p = (N_p, A_p)$, with node set N_p and arc set A_p. Each arc $(i, j) \in A_p$ corresponds to an exploit with a fixed completion time t_{ijp} which must be completed in order to finish the project. Each node $i \in N_p$ represents a "milestone event," or an intermediate project objective, that occurs when all its predecessor exploits $(k, i) \in A_p, k \in N_p$ are complete. We define two special events, the project-start event $start$ and the project-end event end. We assume the start time of $start$ as zero. The start time of end defines the completion time of the entire project. The objective of an individual attacker is to complete its attack project as soon as possible by finding the "critical path." Additionally, we define weight parameter $c_p, p \in P$ to weigh the importance or likelihood of attacks carried out by different adversaries.

Mitigations can delay exploits by extending their effort in time. We use the term "cover" if an exploit is delayed by a mitigation, and one mitigation may cover multiple attack exploits. For example, if an attacker plans to exploit vulnerabilities in a third-party vendor, a mitigation that improves this vendor's security would affect many (or all) corresponding exploits. Let M denote a set of applicable mitigations identified by SMEs, and let $M_{ijp} \in M$ be the subset of mitigations that cover exploit $(i, j) \in A_p$ in project $p \in P$. The completion time of an exploit $(i, j) \in A_p, p \in P$ is delayed by a constant amount d_{ijp} if covered by at least one selected mitigation. Additionally, we assume that each mitigation $m \in M$ is associated with a cost b_m, which captures the financial or human resources that implementing this mitigation requires. The total budget for selecting mitigations is B. We partition the set of mitigations M into ℓ mutually exclusive subsets $H_1, H_2, ..., H_\ell$, each of which contains mitigations that have conflicting effects and cannot be selected together. The objective of the defender is to delay the total weighted completion time of all attacks by deploying a set of mitigations subject to selection requirements.

Decision variables

- x_m = 1 if mitigation $m \in M$ is chosen, 0 otherwise;
- z_{ijp} = 1 if exploit $(i, j) \in A_p$ in project $p \in P$ is covered by at least one selected mitigation, 0 otherwise;
- s_{ip} = start time of event $i \in N$ in project $p \in P$.

The defender maximizes the total weighted completion time of adversarial attacks by selecting a subset of mitigations to compromise adversarial exploits, while each attacker launches an attack as soon as possible after certain exploits are delayed by the defender. Let $s_p(\mathbf{x}, \mathbf{z})$ denote the completion time of attack project $p \in P$, which is a function of the defender's decision variables **x** and **z**. The Deterministic Interdiction model for delaying Multiple Adversarial Projects (DIMAP) is formulated as follows:

DIMAP

$$\max \quad \sum_{p \in P} c_p s_p(\mathbf{x}, \mathbf{z}) \tag{4.1}$$

$$\text{s.t.} \quad \sum_{m \in M} b_m x_m \leq B \tag{4.2}$$

$$\sum_{m \in H_k} x_m \leq 1, \quad \forall k = 1, ..., \ell \tag{4.3}$$

$$z_{ijp} \leq \sum_{m \in M_{ijp}} x_m, \quad \forall (i, j) \in A_p, p \in P \tag{4.4}$$

$$x_m \in \{0, 1\}, \quad \forall m \in M \tag{4.5}$$

$$z_{ijp} \in \{0, 1\}, \quad \forall (i, j) \in A_p, p \in P \tag{4.6}$$

For each adversary $p \in P$, the time of its fastest project completion time after being compromised by the defender, $s_p(\mathbf{x}, \mathbf{z})$, is computed as follows:

$$
\begin{aligned}
s_p(\mathbf{x}, \mathbf{z}) = \min \quad & s_{end,p} && (4.7)\\
\text{s.t.} \quad & s_{jp} - s_{ip} \geq t_{ijp} + d_{ijp} z_{ijp}, \quad \forall (i,j) \in A_p && (4.8)\\
& s_{start,p} = 0 && (4.9)\\
& s_{ip} \geq 0, \quad \forall i \in N_p && (4.10)
\end{aligned}
$$

This is a max-min problem with a first stage defender who maximizes the total weighted completion time of all projects (4.1), and each attacker in the second stage minimizes the completion time of its own project after being delayed by the defender (4.7). The defender in the first stage has a limited budget constrained by (4.2) and an additional selection requirement enforced by multiple choice constraints as defined in (4.3). Constraints (4.4) state that an arc (exploit) is covered if there exists at least one selected mitigation that covers it. Constraints (4.5) and (4.6) require all first stage variables to be binary. There are $|P|$ second stage problems in total, each of which is a variant of the linear programming formulation of the critical path method. Precedence constraints (4.8) define the start time of each event. If an exploit $(i, j) \in A_p, p \in P$ is covered by at least one selected mitigation, its completion is delayed by a constant amount d_{ijp}, even if it is covered more than once. Note that we assume covering an exploit one time and multiple times have the same benefit in this model. The case when an exploit is delayed multiple times is a particular case of what is considered in this model, which can be transformed by simply extending an arc to a path with multiple arcs on it (head to tail) and assuming each new arc can be covered by at most one mitigation. Constraint (4.9) ensures the project starts at time zero, and constraint set (4.10) requires the start time of any event to be nonnegative.

The effectiveness of mitigations on compromising adversarial exploits is uncertain in nature. This is inherent in the fact that cyber threats have a dynamic nature and SMEs only have a limited knowledge about attack profiles and mitigation controls. To address this uncertainty in decision making, we consider random delay times that are only known to the SMEs through a probability mass function $\phi = \mathbf{P}\{d_{ijp} = d^{\omega}_{ijp}, (i,j) \in A_p, \omega \in \Omega, p \in P\}$, where Ω is a finite probability space. We define additional decision variables s^{ω}_{ip} as the start time of event $i \in N_p$ in project $p \in P$ under scenario $\omega \in \Omega$. Let $s^{\omega}_{p}(\mathbf{x}, \mathbf{z})$ be the completion time of attack project $p \in P$ under scenario $\omega \in \Omega$. DIMAP is extended to a stochastic model variant (SIMAP) that incorporates random delay times as follows:

SIMAP

$$\begin{aligned} \max \quad & \sum_{p \in P} c_p \Big(\sum_{\omega \in \Omega} \phi^{\omega} s^{\omega}_{p}(\mathbf{x}, \mathbf{z}) \Big) && (4.11) \\ \text{s.t.} \quad & (4.2) - (4.6) \end{aligned}$$

For each adversarial project $p \in P$, the completion time of its fastest attack under scenario $\omega \in \Omega$ after being compromised by the defender, s^{ω}_{p}, is computed as follows:

$$\begin{aligned} s^{\omega}_{p}(\mathbf{x}, \mathbf{z}) = \min \quad & s^{\omega}_{end,p} && (4.12) \\ \text{s.t.} \quad & s^{\omega}_{jp} - s^{\omega}_{ip} \geq t_{ijp} + d^{\omega}_{ijp} z_{ijp}, \quad \forall (i,j) \in A_p && (4.13) \\ & s^{\omega}_{start,p} = 0 && (4.14) \\ & s^{\omega}_{ip} \geq 0, \quad \forall i \in N_p && (4.15) \end{aligned}$$

The defender's objective is to maximize the expected total weighted completion time of all adversarial attacks (4.11). There are $|P||\Omega|$ second stage problems in total. Each adversary $p \in P$, under realization $\omega \in \Omega$, aims to carry out an attack as quickly as possible. DIMAP is a particular case of SIMAP when $|\Omega| = 1$. In the remaining part of the chapter, we center our discussion around SIMAP.

The second stage problems of SIMAP are linear programs. Hence, fixing the values of the first stage variables $\mathbf{x}$ and $\mathbf{z}$, taking the dual of the second stage problem, and releasing the first stage variables yields to the following max-max model. Denote $y^{\omega}_{ijp}, (i,j) \in A_p$ as the dual variables associated with the precedence constraints (4.13) for $\omega \in \Omega, p \in P$.

$$\max \quad \sum_{p \in P} c_p \sum_{\omega \in \Omega} \phi^\omega \Big(\max \sum_{(i,j) \in A_p} (t_{ijp} + d^\omega_{ijp} z_{ijp}) y^\omega_{ijp} \Big) \tag{4.16}$$

$$\text{s.t.} \quad (4.2) - (4.6)$$

$$\sum_{j:(i,j) \in A_p} y^\omega_{ijp} - \sum_{j:(j,i) \in A_p} y^\omega_{jip} = \begin{cases} 1, & \text{if } i = start \\ 0, & \text{if } i \in N \setminus \{start, end\}, \quad i \in N_p, \omega \in \Omega, p \in P \\ -1, & \text{if } i = end. \end{cases} \tag{4.17}$$

$$y^\omega_{ijp} \geq 0, \quad \forall (i,j) \in A_p, \omega \in \Omega, p \in P \tag{4.18}$$

SIMAP is a max-max problem, and the nonlinear term in the objective function (4.16) can be easily linearized [17]. For each project $p \in P$ under scenario $\omega \in \Omega$, we replace each arc (i,j) by a pair of arcs (i,j) and $(i,j)'$ with lengths $t_{ijp} + d^\omega_{ijp}$ and t_{ijp}, respectively. We associate flow variable y^ω_{ijp} with arc (i,j), and create an additional variable y'^ω_{ijp} that is associated with arc $(i,j)'$. The linearized reformulated SIMAP problem (LR-SIMAP) is as follows:

LR-SIMAP

$$\max \quad \sum_{p\in P} c_p \sum_{\omega\in\Omega} \phi^\omega \Big(\sum_{(i,j)\in A_p} (t_{ijp} + d^\omega_{ijp}) y^\omega_{ijp} + \sum_{(i,j)\in A_p} t_{ijp} y'^\omega_{ijp} \Big) \tag{4.19}$$

s.t. (4.2), (4.3), (4.5)

$$y^\omega_{ijp} \le \sum_{m\in M_{ijp}} x_m, \quad \forall (i,j)\in A_p, \omega\in\Omega, p\in P \tag{4.20}$$

$$\sum_{j:(i,j)\in A_p} (y^\omega_{ijp} + y'^\omega_{ijp}) - \sum_{j:(j,i)\in A_p} (y^\omega_{jip} + y'^\omega_{jip}) = \begin{cases} 1, & \text{if } i = start \\ 0, & \text{if } i \in N_p \setminus \{start, end\} \\ -1, & \text{if } i = end. \end{cases} \quad , \omega\in\Omega, p\in P \tag{4.21}$$

$$0 \le y^\omega_{ijp}, y'^\omega_{ijp} \le 1, \quad \forall (i,j)\in A_p, \omega\in\Omega, p\in P \tag{4.22}$$

The objective (4.19) maximizes the expected total weighted longest path length across all attacks and all scenarios. Constraints (4.2), (4.3) and (4.5) are carried over from the defender's problem. The new coverage constraints (4.20) state that y^ω_{ijp} is equal to one if exploit $(i,j)\in A_p, \omega\in\Omega, p\in P$ is covered by at least one selected mitigation. Constraints (4.21) are the flow constraints. One unit of flow leaves out of the source node $start$ and eventually arrives at the sink node end through a path. For each node on the path, the amount of flow comes in is equal to the amount of flow goes out. Moreover, arcs (i,j) and $(i,j)'$ cannot be selected together at the same time. Constraints (4.22) require the flow to be nonnegative and less than one.

LR-SIMAP can be readily solved by a general purpose mixed-integer programming (MIP) solver. However, given the number of variables and constraints in this model, MIP solvers are likely to experience time or memory issues when the problem size increases. In the following section, we propose a heuristic approach based on Lagrangian relaxation to decompose LR-SIMAP into a number of smaller subproblems that are easier to solve. This heuristic iteratively updates a lower and an upper bound to the objective value of LR-SIMAP using subgradient optimization method to efficiently identify optimal solutions.

4.2 Lagrangian Heuristic

In this section, we present a Lagrangian heuristic, denoted as ALG_{Lag}, that decomposes LR-SIMAP into a number of subproblems and provides a lower bound and an upper bound upon termination. Let UB and LB denote the upper and lower bound of LR-SIMAP. We start by relaxing the linking constraints (4.20) and adding them to the objective function. Let $\lambda_{ijp}^{\omega}, (i,j) \in A_p, \omega \in \Omega, p \in P$ be the Lagrangian multipliers associated with constraints (4.20). The corresponding Lagrangian relaxation problem, denoted as $LR(\lambda)$, is formulated as follows:

$$\begin{aligned} z(\lambda) = \max & \sum_{p\in P} c_p \sum_{\omega\in\Omega} \phi^{\omega} \Big(\sum_{(i,j)\in A_p} (t_{ijp} + d_{ijp}^{\omega}) y_{ijp}^{\omega} + \sum_{(i,j)\in A_p} t_{ijp} y_{ijp}^{'\omega} \Big) \\ & + \sum_{p\in P} \sum_{\omega\in\Omega} \sum_{(i,j)\in A_p} \lambda_{ijp}^{\omega} \Big(\sum_{m\in M_{ijp}} x_m - y_{ijp}^{\omega} \Big) \qquad (4.23) \\ \text{s.t.} \quad & (4.2), (4.3), (4.5), (4.21), (4.22) \end{aligned}$$

We reorganize the Lagrangian objective function (4.23) to separate the $\mathbf{x}$ and $\mathbf{y}$ variables into two terms, yielding the following formulation that facilitates decomposition:

$$\begin{aligned} z(\lambda) = \max \quad & \sum_{p\in P} \sum_{\omega\in\Omega} \Big(\sum_{(i,j)\in A_p} (c_p \phi^{\omega} t_{ijp} + c_p \phi^{\omega} d_{ijp}^{\omega} - \lambda_{ijp}^{\omega}) y_{ijp}^{\omega} + \sum_{(i,j)\in A_p} c_p \phi^{\omega} t_{ijp} y_{ijp}^{'\omega} \Big) \\ & + \sum_{m\in M} \Big(\sum_{p\in P} \sum_{\omega\in\Omega} \sum_{(i,j)\in A_{mp}} \lambda_{ijp}^{\omega} \Big) x_m \qquad (4.24) \end{aligned}$$

where A_{mp} is the set of arcs in project p that are covered by mitigation $m \in M$. We can thus decompose $LR(\lambda)$ into the two subproblems, $LR_1(\lambda)$ and $LR_2(\lambda)$, as follows:

$$\begin{aligned} z(\lambda) = \max \Big\{ & \sum_{p\in P} \sum_{\omega\in\Omega} \Big(\sum_{(i,j)\in A_p} (c_p \phi^{\omega} t_{ijp} + c_p \phi^{\omega} d_{ijp}^{\omega} - \lambda_{ijp}^{\omega}) y_{ijp}^{\omega} + \sum_{(i,j)\in A_p} c_p \phi^{\omega} t_{ijp} y_{ijp}^{'\omega} \Big) \Big| (4.21)(4.22) \Big\} \\ & \qquad (\mathbf{LR_1}(\lambda)) \\ + \max \Big\{ & \sum_{m\in M} \Big(\sum_{p\in P} \sum_{\omega\in\Omega} \sum_{(i,j)\in A_{mp}} \lambda_{ijp}^{\omega} \Big) x_m \Big| (4.2)(4.3)(4.5) \Big\} \qquad (\mathbf{LR_2}(\lambda)) \end{aligned}$$

4.2.1 Solution Methods for the Subproblems

$LR_1(\lambda)$ is an aggregation of $|P||\Omega|$ longest path problems (LPP) across all projects and scenarios. We use dynamic programming (DP) to solve each LPP. Define $f_p^\omega(i)$ as the longest path length from source node $start$ to node $i, i \in N_p, \omega \in \Omega, p \in P$. Since $G_p, p \in P$ are DAGs, without loss of generality, we assume that $N_p, p \in P$ are topologically sorted. Let S be the set containing the selected arcs in the recursive steps of the dynamic program. The DP algorithm, ALG_{LR_1}, for solving $LR_1(\lambda)$ is presented as follows:

$\mathbf{ALG_{LR_1}}$:

For $p \in P$ and $\omega \in \Omega$:

Initial settings: $f_p^\omega(start) = 0, S = \emptyset$.

Recursive step: for $i \in N_p$ in linearized order, do:

$$f_p^\omega(i) = \max_{j:(j,i)\in A_p} \Big\{ \max\{f_p^\omega(j) + c_p\phi^\omega t_{ijp} + c_p\phi^\omega d_{ijp}^\omega - \lambda_{ijp}^\omega, f_p^\omega(j) + c_p\phi^\omega t_{ijp}\} \Big\} \tag{4.25}$$

Add the selected arc (j^*, i) into set S

For each $p \in P, \omega \in \Omega$, the longest path length of G_p is $f_p^\omega(end)$, which can be computed with time and space complexity $O(|N_p| + |A_p|)$. The actual path associated with this length can be recovered by tracing reversely from end node with the selected arc in the recursive step, which is contained in the set S. More specifically, in our problem, there is a pair of arcs (j,i) and $(j,i)'$ between the same pair of nodes j and i. The outer maximization in (4.25) determines the precedent node j of the current node i in the longest path, and the inner maximization in (4.25) determines which arc of (j,i) and $(j,i)'$ is selected. The first term in the inner maximization, i.e., $f_p^\omega(j) + c_p\phi^\omega t_{ijp} + c_p\phi^\omega d_{ijp}^\omega - \lambda_{ijp}^\omega$ corresponds to the arc (j,i), and its selection leads to $y_{jip}^\omega = 1$ and $y_{jip}^{'\omega} = 0$. If the second term is selected, then $y_{jip}^\omega = 0$ and $y_{jip}^{'\omega} = 1$. Finally, the objective function value of $LR_1(\lambda)$ is $\sum_{p\in P}\sum_{\omega\in\Omega} f_p^\omega(end)$.

$LR_2(\lambda)$ is a multiple-choice knapsack problem (MCKP). Traditionally, the multiple choice constraints are equalities instead of inequalities as in our model. We can transform them into standard form by adding a dummy node m with zero cost (i.e., $b_m = 0$) and zero profit (i.e., objective coefficient of x_m in $LR_2(\lambda)$, $\sum_{p\in P}\sum_{\omega\in\Omega}\sum_{(i,j)\in A_{mp}}\lambda_{ijp}^{\omega} = 0$) to each group. After solving a standard MCKP, a feasible solution to the original $LR_2(\lambda)$ can be recovered by simply deleting dummy node solutions.

Numerous techniques and algorithms have been developed for solving MCKP, including dominance relations and class and state reduction, as they are applied to branch and bound and dynamic programming. For a complete review of various techniques for solving MCKP, we refer the readers to the book by [55]. Performance of three different types of existing algorithms are compared in [55] on a variety of randomly generated problem instances, including a branch and bound algorithm [33], a hybrid dynamic programming and branch and bound algorithm [34], and an expanding-core algorithm [82]. Among them, the expanding-core algorithm by [82] appears to have the best overall performance. Therefore, in the computational study we select the expanding-core algorithm for solving the MCKP subproblems, denoted as ALG_{LR_2}. We implement the default algorithm as described in [82]. This algorithm first solves the linear MCKP (LMCKP) based on a simple partitioning algorithm in linear time. The optimal solution to LMCKP contains at most one group with two fractional variables. The "core" initially contains the fractional group in the linear solution, and then expands if a promising group is found. To determine which group is added into "core," a positive and a negative "gradient" are calculated for each group. The set of positive gradients is sorted in non-increasing order while the set of negative gradients is sorted in non-decreasing order. Each time, we add an unselected group corresponding to the next gradient alternately from the two sets. DP is used to update the solution every time a new group is added into "core." Variable reduction and state reduction procedures are implemented to minimize the problem size. When an optimal solution is found, this algorithm ensures a minimal number of groups is enumerated.

4.2.2 Upper Bound

The objective value of the Lagrangian dual problem, $z^{LD} = \min\{z(\lambda) : \lambda \geq 0\}$, provides an upper bound to the original problem LR-SIMAP. We use the subgradient method [45] to improve z^{LD}. In iteration k, given the current multiplier vector $\lambda(k)$, a step is taken along a subgradient of $z(\lambda(k))$. Let $\mathbf{x}(k)$ and $\mathbf{y}(k)$ be the optimal solution vectors of $LR_1(\lambda(k))$ and $LR_2(\lambda(k))$ respectively. The updating formula for the Lagrangian multipliers is:

$$\lambda^{\omega}_{ijp}(k+1) = \max\Big\{0, \lambda^{\omega}_{ijp}(k) - h_k\Big(\sum_{m\in M_{ijp}} x_m(k) - y^{\omega}_{ijp}(k)\Big)\Big\}, \quad \forall (i,j) \in A_p, \omega \in \Omega, p \in P \tag{4.26}$$

where h_k is a positive scalar step size. A natural choice for the initial value of the multipliers is zero. The step size h_k is computed in the following standard form:

$$h_k = \frac{\alpha_k\Big(z(\lambda(k)) - z^*\Big)}{\sum_{p\in P}\sum_{\omega\in\Omega}\sum_{(i,j)\in A_p}\Big(\sum_{m\in M_{ijp}} x_m(k) - y^{\omega}_{ijp}(k)\Big)^2} \tag{4.27}$$

where z^* is the optimal objective value. We replace z^* with the objective value of the best known feasible solution, i.e., the best lower bound (LB) obtained until iteration k. The scalar α_k is a scalar satisfying $0 \leq \alpha_k \leq 2$. We set $\alpha_0 = 2$ initially and reduce it by a factor of two whenever $z(\lambda(k))$ fails to decrease after τ_{max} iterations. Feasible solutions to the original problem might be found during the subgradient iterations, however, it is rare in practice [38]. Since there is no way of proving optimality in the subgradient method, we terminate it either when it reaches k_{max} iterations or the optimality gap $\epsilon = (UB - LB)/LB$ is below ϵ_{min}. The selection of Lagrangian parameters τ_{max}, k_{max}, and ϵ_{min} is discussed in the computational results section. When the subgradient method terminates, we obtain an upper bound (UB) to LR-SIMAP.

4.2.3 Lower Bound

The solutions to the Lagrangian subproblems obtained during the subgradient method are often not feasible to the original problem. We introduce a technique that can quickly identify a feasible solution and accordingly update the lower bound of LR-SIMAP iteratively via subgradient optimization. Infeasibility comes from violations to the linking constraints (4.20). We can obtain a feasible solution by fixing $\mathbf{x}(k)$ and recovering $\mathbf{y}(k)$ using the following procedure:

$\mathbf{Feasibility}(\mathbf{x}(\mathbf{k})$:

0. Let $\mathbf{y}_{feas}(k) = \{y^{\omega}_{ijp} = y'^{\omega}_{ijp} = 0, (i,j) \in A_p, \omega \in \Omega, p \in P\}$, and assume there are no delays, i.e., arc length is t_{ijp} for $(i,j) \in A_p, p \in P$.

1. For each arc $(i,j) \in N_p, p \in P$, if it is covered by at least one selected mitigation in $\mathbf{x}(k)$, its length is extended by d^{ω}_{ijp} under scenario $\omega \in \Omega$. Denote this new arc length as t'^{ω}_{ijp}.

2. Calculate the longest path length of $G_p, \omega \in \Omega, p \in P$ using the DP algorithm ALG_{LR_1} presented for solving $LR_1(\lambda)$ with a new updating equation:

$$f^{\omega}_p(i) = \max_{j:(j,i)\in A_p} \{f^{\omega}_p(j) + t'^{\omega}_{ijp}\}$$

and obtain the longest path solution $Path^{\omega}_p$ by tracing back from the end node the best choice in the last step.

3. For each arc (i,j) on $Path^{\omega}_p$, let $y'^{\omega}_{ijp} = 1$ if $t'^{\omega}_{ijp} = t_{ijp}$ and $y^{\omega}_{ijp} = 1$ if $t'^{\omega}_{ijp} > t_{ijp}$. Return feasible solution $(\mathbf{x}(k), \mathbf{y}_{feas}(k))$, and lower bound $LB = \sum_{p\in P}\sum_{\omega\in\Omega} c_p \phi^{\omega} f^{\omega}_p(end)$.

Given $\mathbf{x}(k)$, we recover $\mathbf{y}_{feas}(k)$ by tracking the exploits that are covered by $\mathbf{x}(k)$. The lower bound LB is calculated as the expected-value of total weighted longest path lengths of all projects after certain exploits being delayed by $\mathbf{x}(k)$.

In summary, the Lagrangian heuristic, ALG_{Lag}, operates as follows.

Lagrangian heuristic

0. Initialization: $k = 0, \tau = 1, \alpha_0 = 0, \lambda(k) = \mathbf{0}, UB^* = \infty, LB^* = 0$.

1. Solve Lagrangian subproblems $LR_1(\lambda(k))$ and $LR_2(\lambda(k))$ with algorithms ALG_{LR_1} and ALG_{LR_2}, respectively. Let $z_1(\lambda(k))$ and $z_2(\lambda(k))$ be their optimal objective values, and $\mathbf{x}(k)$

and $\mathbf{y}(k)$ be their optimal solutions, respectively.
If $z_1(\lambda(k)) + z_2(\lambda(k)) \geq UB^*$, $\tau = \tau + 1$. If $\tau \geq \tau_{max}$, $\alpha_k = \frac{1}{2}\alpha_k$ and $\tau = 1$.
If $UB^* > z_1(\lambda(k)) + z_2(\lambda(k))$, update $UB^* = z_1(\lambda(k)) + z_2(\lambda(k))$.

2. Apply the procedure $Feasibility(\mathbf{x}(k))$ to obtain a feasible solution $(\mathbf{x}(k), \mathbf{y}_{feas}(k))$ with objective function value LB.
   If $LB^* < LB$, update $LB^* = LB$.

3. Compute step size h_k with (4.27) and Lagrangian multipliers $\lambda(k+1)$ with (4.26).
   Update $k = k + 1$, $\epsilon = (UB^* - LB^*)/LB^*$.

4. If $k \geq k_{max}$ or $\epsilon \leq \epsilon_{min}$, stop and return the best feasible solution $(\mathbf{x}(k), \mathbf{y}_{feas}(k))$, lower bound LB^* and upper bound UB^*; else go to Step 1.

4.3 Computational Results

In this section, we test the efficiency and effectiveness of the Lagrangian heuristic ALG_{Lag} compared to a MIP solver, and we discuss the SIMAP model insights by evaluating mitigation selection and the value of stochastic solutions. We compare the performance of ALG_{Lag} with mixed-integer programming solver Gurobi for solving LR-SIMAP on problem instances of varying sizes. We expect ALG_{Lag} to be more efficient in solving large scale instances as it decomposes LR-SIMAP into $(|P||\Omega| + 1)$ subproblems, each of which can be solved efficiently. Additionally, ALG_{Lag} provides a feasible and possibly optimal solution with a known optimality gap as both upper bound and lower bound are updated by subgradient optimization. ALG_{Lag} was programmed and run with Python 2.7.12. The model was solved using Gurobi 7.0.1 with a Python interface. The tests were run on an Intel Core i5 CPU at 2 GHz with 8 GB of RAM with a time limit of one hour. If ALG_{Lag} or Gurobi did not terminate within an hour, we report the objective value of the best feasible solution.

Standardized mitigation practices for supply chain attacks are either unavailable or inaccessible [73]. Therefore, we create synthetic data in the computational study to investigate algorithm performance as well as solution analysis. The size of problem instance, i.e., number of mitigations and adversaries, and size of attack graphs, may reflect several aspects in real settings, such as the scope of the planning problem, the size of the organization and its supply chains, and the adversaries' capacities. We created data instances with various sizes to test different hypothetical cases as well as the algorithm's performance under different cases.

The data was created as follows. The number of available mitigations ranges from 10 to 100, and the number of adversaries (projects) ranges from 2 to 20. The number of scenarios, $|\Omega|$, is set to 100. The cost coefficients $b_m, m \in M$ are generated as discrete Uniform $(1, 10)$ random variables using a pseudo-random generator. The total budget B is set to be within 10% of the total costs of all mitigations, since preliminary experiments show that a larger budget may lead to trivial results. We specify the number of groups ℓ, to be between $\frac{1}{10}|M|$ and $\frac{1}{2}|M|$, since a value of ℓ that is too small or too large could make the budget constraint or the multiple-choice constraints redundant, respectively. Given the number of groups, we randomly sample from M without replacement to generate subgroups of mitigations $M_i, i = 1, ..., \ell$. Moreover, the project coefficients $c_p, p \in P$ are randomly generated as Uniform $(1, 10)$ random variables.

Next, we describe how to generate attack graphs. We specify the number of nodes $|N_p|, p \in P$ to range from 10 to 40. Without loss of generality, we assume that nodes in $N_p, p \in P$ are sorted in topological order. While some interdiction problems where arcs in the underlying graph (network) are generated based on some geographical relationship between node locations [28, 53], we create arcs in a similar manner to those in project interdiction problems [18, 44], where an arc can be created between any two nodes (events) that have a precedence relationship. More specifically, in a topologically ordered set of nodes, we create an arc $(i, j) \in A_p, p \in P$ between a node $i \in N_p$ and at most three other nodes that have larger indices, i.e., nodes in set $\{j \in N_p | j > i\}$. For each arc $(i, j) \in A_p, p \in P$, we randomly generate its completion time t_{ijp} as a Uniform $(0, 10)$ random variable. In this computational study, we assume the delay times are Uniform random variables that can be as low as zero, i.e., the exploit is not effected by mitigations, and can be as high as three times of the original completion time. Therefore, based on the generated $t_{ijp}, (i, j) \in A_p, p \in P$ values, we randomly generate d^{ω}_{ijp} as a Uniform $(0, 3t_{ijp})$

random variable for each arc $(i, j) \in A_p, p \in P$ under scenario $\omega \in \Omega$. We specify that each arc (exploit) can be covered by at most three different mitigations and randomly sample from M with replacement to generate $M_{ijp}, (i, j) \in A_p, p \in P$.

4.3.1 Algorithm Performance

We first test the performance of ALG_{Lag} and compare it with the MIP solver Gurobi on various test instances. We set the maximum number of iterations for the subgradient method to $k_{max} = 200$, since we observe the upper and lower bounds rarely improve after 200 iterations. The optimality gap for termination is set to $\epsilon_{min} = 10^{-5}$. The remaining subgradient method parameters are updated using common practice: the coefficient α_k of the step size h_k reduces by a factor of two if the Lagrangian objective value $z(\lambda(k))$ fails to decrease after $\tau_{max} = 10$ iterations.

Table 4.1 - 4.3 report computational results of ALG_{Lag} as compared to Gurobi on small, medium and large instances, respectively. The results include i) the best upper bound UB^* and the best lower bound LB^* returned by ALG_{Lag} upon termination, as well as the objective value of the optimal solution, z^*, or that of the best feasible solution, z^*_{feas}, returned by Gurobi within time limit; ii) the relative deviation of LB^* from UB^* and z^* (or z^*_{feas}), i.e., $\frac{UB^*-LB^*}{LB^*} \times 100\%$ and $\frac{z^*-LB^*}{LB^*} \times 100\%$ (or $\frac{z^*_{feas}-LB^*}{LB^*} \times 100\%$), respectively; iii) the amount of time, t_{Lag}, that ALG_{Lag} takes until termination, the amount of time, t_{LB^*}, when the best lower bound LB^* was found in ALG_{Lag}, as well as the solution time, t_{z^*} (or $t_{z^*_{feas}}$), returned by Gurobi. Note that if any of these times exceed the time limit of 3600 seconds, the corresponding values in the tables are replaced with dashes.

Table 4.1 shows the results on two sets of small instances with different sizes. Sets s1 - s10 have 10 mitigations and 5 projects with 10 nodes on each project. Sets s11 - s20 are slightly larger with 20 mitigations and 10 projects with 20 nodes on each project. Gurobi is able to solve all these instances to optimality, which allows us to assess the quality of the feasible solutions identified by ALG_{Lag} by comparing them to the optimal solutions. ALG_{Lag} finds optimal solutions for 19 of the 20 instances upon termination. For only one instance, s14, Gurobi achieves a better solution (i.e., $LB^* < z^*$), however the relative deviation of LB^* from z^* is only 1%. The total solution time of ALG_{Lag}, t_{Lag}, is greater than that of Gurobi, t_{z^*}, for all instances. However, the "actual" solution time of ALG_{Lag}, i.e., the time when the best lower bound LB^* was obtained, t_{LB^*}, is smaller than t_{z^*} for nine out of ten smaller instances s1 - 10, and is significantly smaller than t_{z^*} for all larger instances s11 - s20. Additionally, ALG_{Lag} provides an upper bound when it terminates. For s1 - s10, the best upper bound UB^* is within 0.4% deviation from LB^*, while for the larger set of instances, s11 - s20, the best upper bound has a larger deviation from LB^*, ranging from 1.8% to 6.0%. In summary, the results in Table 4.1 demonstrate that ALG_{Lag} identifies optimal or near-optimal solutions efficiently.

Table 4.1: Computational results on 20 small instances: ALG_{Lag} vs. Gurobi

Test instances							Objective value			Gap (%)		Time (s)		
Filename	$\|M\|$	$\|P\|$	$\|N_p\|$	$\|A_p\|$	$\|\Omega\|$	ℓ	UB^*	LB^*	z^*	$\frac{(UB^*-LB^*)}{LB^*}$	$\frac{(z^*-LB^*)}{LB^*}$	t_{Lag}	t_{LB^*}	t_{z^*}
s1	10	5	10	24	100	3	2411.1	2402.9	2402.9	0.3%	0.0%	90.0	0.9	5.5
s2	10	5	10	25	100	3	2263.7	2259.8	2259.8	0.2%	0.0%	91.1	1.4	4.5
s3	10	5	10	23	100	3	1802.6	1798.6	1798.6	0.2%	0.0%	89.4	4.3	8.1
s4	10	5	10	22	100	3	2195.0	2191.1	2191.1	0.2%	0.0%	89.5	3.2	4.6
s5	10	5	10	20	100	3	1675.8	1670.3	1670.3	0.3%	0.0%	85.2	1.7	3.7
s6	10	5	10	22	100	3	2801.0	2791.3	2791.3	0.4%	0.0%	90.4	3.3	2.0
s7	10	5	10	20	100	3	1911.5	1905.4	1905.4	0.3%	0.0%	98.9	2.1	3.3
s8	10	5	10	24	100	3	976.6	972.6	972.6	0.4%	0.0%	86.6	3.1	4.0
s9	10	5	10	21	100	3	1083.9	1082.3	1082.3	0.2%	0.0%	91.4	1.5	4.2
s10	10	5	10	22	100	3	2085.8	2083.7	2083.7	0.1%	0.0%	88.4	2.0	3.1
s11	20	10	20	51	100	4	6666.4	6299.3	6299.3	5.8%	0.0%	782.9	8.9	375.3
s12	20	10	20	59	100	4	9727.4	9492.7	9492.7	2.5%	0.0%	801.6	8.5	232.5
s13	20	10	20	53	100	4	9375.0	9165.7	9165.7	2.3%	0.0%	775.9	15.8	214.0
s14	20	10	20	48	100	4	9005.7	8496.9	8584.2	6.0%	1.0%	778.3	74.4	450.6
s15	20	10	20	57	100	4	5434.3	5211.9	5211.9	4.3%	0.0%	778.0	19.4	353.2
s16	20	10	20	48	100	4	9636.1	9469.5	9469.5	1.8%	0.0%	758.5	8.2	495.2
s17	20	10	20	59	100	4	7507.5	7201.8	7201.8	4.2%	0.0%	757.9	28.7	221.3
s18	20	5	20	58	100	4	8438.9	8173.1	8173.1	3.3%	0.0%	812.4	26.2	267.8
s19	20	10	20	55	100	4	8871.4	8461.6	8461.6	4.8%	0.0%	673.1	57.0	433.0
s20	20	10	20	53	100	4	8157.8	7727.7	7727.7	5.6%	0.0%	759.1	15.0	693.3

We further compare the performance of ALG_{Lag} and Gurobi on larger instances as shown in Tables 4.2 and 4.3. Table 4.2 contains medium sized instances m1 - m20, for which Gurobi does not find optimal solutions within the one hour time limit, while Table 4.3 contains the largest set of instances l1 - l10 in our computational study, for which Gurobi does not find a feasible solution within one hour. In Table 4.2, we report the objective value of the best feasible solution, z^*_{feas}, found by Gurobi in one hour. Across the 20 instances, the best lower bound LB^* found by ALG_{Lag} is better than z^*_{feas} in 17 instances, equal to z^*_{feas} in one instance, and is slightly worse than z^*_{feas} in two instances with a deviation less than 0.9%. More importantly, the computational time of ALG_{Lag}, t_{Lag}, is less than 900 seconds in every instance, which is significantly less than one hour. We note that the relative difference between t_{LB^*} and t_{Lag} is smaller than those in Table 4.1, likely due to the fact that an improved lower bound might be harder to find as the problem size becomes larger. Table 4.3 mainly presents results of ALG_{Lag} as Gurobi is in the root simplex stage when it runs out of time and does not return a feasible solution. The relative deviation of LB^* from UB^* ranges from 4.27% to 8.81% for instances in Table 4.2 and Table 4.3, which might not be ideal for evaluating solution quality. The upper bounds are not close to their corresponding lower bounds, especially for medium to large instances, mainly due to the number of relaxed constraints (i.e., the number of multipliers to update) in LR-SIMAP is too large. Based on results in these two tables, we can conclude that the Lagrangian heuristic outperforms Gurobi on medium to large instances by identifying better feasible solutions in significantly shorter times. Combining our analysis for all test instances, we consider ALG_{Lag} to be a practical algorithm and more efficient than Gurobi regarding identifying near-optimal feasible solutions for LR-SIMAP.

Table 4.2: Computational results on 20 medium instances: ALG_{Lag} vs. Gurobi

Test instances							Objective value			Gap (%)		Time (s)		
Filename	$\|M\|$	$\|P\|$	$\|N_p\|$	$\|A_p\|$	$\|\Omega\|$	ℓ	UB^*	LB^*	z^*_{feas}	$\frac{(UB^*-LB^*)}{LB^*}$	$\frac{(z^*_{feas}-LB^*)}{LB^*}$	t_{Lag}	t_{LB^*}	$t_{z^*_{feas}}$
m1	50	10	40	113	100	5	17067.7	16170.9	16078.8	5.55%	−0.57%	862.4	489.9	–
m2	50	10	40	111	100	5	17656.1	17656.1	16589.0	6.41%	−0.02%	897.8	562.7	–
m3	50	10	40	107	100	5	16902.7	15903.6	15683.3	6.28%	−1.39%	869.6	194.4	–
m4	50	10	40	111	100	5	14334.0	13576.2	13386.4	5.58%	−1.40%	857.4	518.2	–
m5	50	10	40	121	100	5	13534.8	12650.8	12756.4	6.99%	0.83%	860.6	371.85	–
m6	50	10	40	111	100	5	16790.5	15772.9	15686.8	6.45%	−0.55%	851.5	409.9	–
m7	50	10	40	120	100	5	16344.8	15403.2	15403.2	6.11%	0.00%	854.8	701.9	–
m8	50	10	40	122	100	5	11169.0	10385.4	10295.0	7.55%	−0.87%	873.2	34.6	–
m9	50	10	40	113	100	5	16667.9	15778.0	15640.8	5.64%	−0.87%	881.2	307.2	–
m10	50	10	40	107	100	5	15399.9	14422.5	14067.9	6.78%	−2.46%	880.0	445.2	–
m11	100	10	40	108	100	10	16874.1	15932.7	15591.3	5.91%	−2.14%	869.8	680.6	–
m12	100	10	40	119	100	10	18900.4	18007.9	17905.1	4.96%	−0.57%	862.5	418.3	–
m13	100	10	40	106	100	10	13421.7	12745.0	12626.9	5.31%	−0.93%	855.5	343.7	–
m14	100	10	40	117	100	10	13642.3	13083.1	12860.7	4.27%	−1.70%	876.1	782.9	–
m15	100	10	40	101	100	10	15712.1	14821.5	14768.7	6.01%	−0.36%	836.9	790.2	–
m16	100	10	40	109	100	10	15587.0	14672.3	14340.8	6.23%	−2.26%	874.6	381.9	–
m17	100	10	40	107	100	10	14257.3	13274.6	13347.2	7.40%	0.55%	861.4	764.2	–
m18	100	10	40	111	100	10	15901.9	15182.1	15069.3	4.74%	−0.74%	841.5	825.1	–
m19	100	10	40	109	100	10	19342.1	18281.7	18194.8	5.80%	−0.48%	870.5	59.1	–
m20	100	10	40	115	100	10	17607.1	16756.3	16751.1	5.08%	−0.03%	856.5	577.7	–

Table 4.3: Computational results on 10 large instances: ALG_{Lag} vs. Gurobi

Test instances							Objective value			Gap (%)		Time (s)		
Filename	$\|M\|$	$\|P\|$	$\|N_p\|$	$\|A_p\|$	$\|\Omega\|$	ℓ	UB^*	LB^*	z^*_{feas}	$\frac{(UB^*-LB^*)}{LB^*}$	$\frac{(z^*_{feas}-LB^*)}{LB^*}$	t_{Lag}	t_{LB^*}	$t_{z^*_{feas}}$
l1	100	20	40	124	100	10	28895.1	26681.1	–	8.30%	–	1836.0	1545.0	–
l2	100	20	40	116	100	10	28919.7	27191.0	–	6.36%	–	1819.1	453.0	–
l3	100	20	40	113	100	10	35152.7	32500.5	–	8.16%	–	2487.5	1035.2	–
l4	100	20	40	118	100	10	28391.1	26222.2	–	8.27%	–	1775.2	1554.8	–
l5	100	20	40	111	100	10	38805.0	35968.4	–	7.89%	–	1793.9	1385.9	–
l6	100	20	40	118	100	10	31280.7	29646.5	–	5.51%	–	1831.6	1527.6	–
l7	100	20	40	113	100	10	28005.6	25736.9	–	8.81%	–	1839.2	1659.9	–
l8	100	20	40	106	100	10	29501.3	27436.7	–	7.53%	–	1945.3	1857.2	–
l9	100	20	40	118	100	10	29879.1	27954.8	–	6.88%	–	1857.4	1511.4	–
l10	100	20	40	122	100	10	35133.2	32776.6	–	7.19%	–	1855.3	1378.0	–

4.3.2 Improving the Upper Bound

This subsection provides an alternative method using the perfect information solution value to demonstrate that the Lagrangian lower bounds LB^* are close to the optimal objective value z^* by introducing an improved upper bound. The perfect information solution, or the "wait and see" solution, assumes that we have accurate and perfect information about the future scenario. By computing the perfect information solution value under each scenario and taking the expected value across all scenarios, we obtain a natural upper bound on z^*. More details are discussed later.

Computational results in the last subsection show that the Lagrangian heuristic identifies optimal solutions efficiently in 19 of the 20 small instances, and identifies better feasible solutions than Gurobi in 18 of the 20 medium instances with shorter computational time. While these results demonstrate that the Lagrangian heuristic is efficient in identifying near-optimal solutions as compared to Gurobi, it relies on Gurobi identifying a feasible solution for assessing the quality of these solutions. For the 10 large instances or any new test instances where this is not the case, to evaluate the quality of the Lagrangian heuristic feasible solution we rely solely on the upper bound returned by the Lagrangian heuristic. As we can see from results in Tables 4.1 - 4.3, the Lagrangian upper bound UB^* is not always close to LB^*, with relative deviation ranging from 0.2% to 8.81%. UB^* is the upper bound of the optimal Lagrangian dual objective value z^{LD}, and z^{LD} is the upper bound of z^*, the optimal objective value of the original problem LR-SIMAP. In summary, we have $UB^* \geq z^{LD} \geq z^* \geq LB^*$. To improve the upper bound UB^*, we need to understand how much of the gap between UB^* and z^* comes from the gap between UB^* and z^{LD}. The gap between z^{LD} and z^* is fixed as long as the same set of constraints is relaxed in the Lagrangian relaxation framework. With a more efficient method for optimizing the Lagrangian dual, we might achieve a better upper bound that is close to z^{LD}. First, we need to check the gap between UB^* and z^{LD}. If it is relatively small, then we should look for an alternative method for obtaining an upper bound of z^*, instead of focusing on improving the method for solving the Lagrangian dual.

It is likely that there is a relatively large gap between z^{LD} and z^* given the fact that a large number of constraints (i.e., $\sum_{p \in P} \sum_{\omega \in \Omega} |A_p|$) in LR-SIMAP are relaxed, which implies the gap between UB^* and z^{LD} is relatively small. To verify our hypothesis, we illustrate the best upper bound UB^* and the best lower bound LB^* obtained by the kth iteration using the subgradient method, respectively, in Figure 4.1 for four instances from our dataset (s11, s14, m2 and m5). For all four subfigures, we can see that the lower bound LB^* improves only by a small amount in the first dozen iterations and then stabilizes. On the other hand, UB^* improves rapidly in the first 50 iterations and then very slowly afterwards. The gap between LB^* and UB^* stays unchanged after 100 iterations for all instances, which suggests that UB^* may already be close to z^{LD} and the gap between LB^* and UB^* may be mostly the gap between z^{LD} and z^*. To further verify this hypothesis, we set the subgradient method to run for a significantly long time of 24 hours and check the upper and lower bounds returned by the heuristic. More specifically, we reset the maximal number of iterations k_{max} to infinity, and we add a termination criteria to the subgradient method as a time limit of 24 hours. The final upper and lower bounds, UB^* and LB^*, obtained after 24 hours, as compared to those obtained after 200 iterations, are presented in Table 4.4. We can see that the gap between UB^* and LB^* improves by at most 0.2% after the subgradient method runs for 24 hours as compared to the gap after 200 iterations, over the four instances. These results suggest that the gap between UB^* and z^{LD} is relatively small and the gap between z^{LD} and z^* is relatively large. As long as the same set of constraints are relaxed within the Lagrangian heuristic framework, UB^* cannot be significantly improved. Therefore, we demonstrate that LB^* is close to z^* by introducing an improved upper bound that is closer to z^* that does not rely on Lagrangian relaxation.

The perfect information (wait and see) solution value, denoted z_{PI}, provides a natural upper bound on the optimal objective value z^*. It can be computed as follows. We first solve $|\Omega|$ deterministic LR-DIMAP problems, the delay times $d_{ijp}, (i,j) \in A_p, p \in P$ for each instance under $\omega \in \Omega$ are set to d^{ω}_{ijp}. Let z^{ω} be the optimal LR-DIMAP objective value, solved with Gurobi, when scenario $\omega \in \Omega$ is realized and the "perfect information" is known. Then, z_{PI} is computed as the expected value of all perfect information solution values, i.e., $z_{PI} = \sum_{\omega \in \Omega} \phi^{\omega} z^{\omega}$.

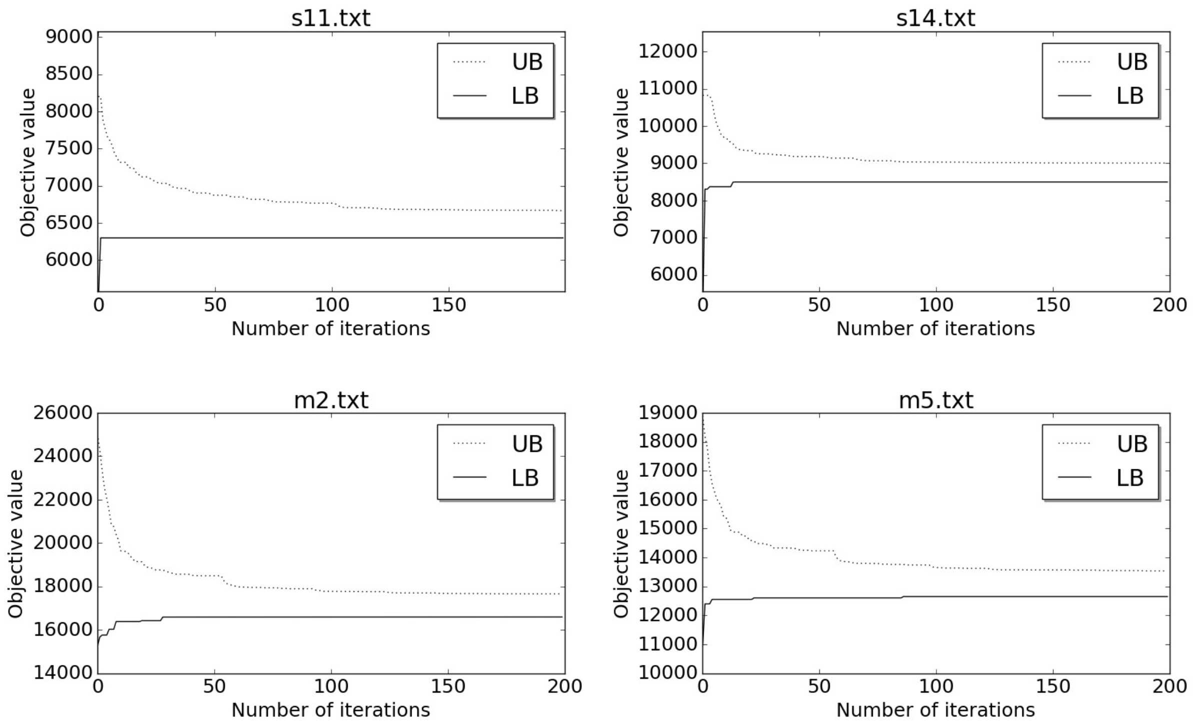


Figure 4.1: Lagrangian heuristic convergence performance: UB^* and LB^* vs. the number of subgradient iterations, respectively.

Table 4.4: Lagrangian heuristic upper and lower bounds obtained after running the subgradient method for 24 hours.

	Terminating after 200 iterations			Terminating after 24 hours		
Filename	UB^*	LB^*	$\frac{(UB^*-LB^*)}{LB^*}$	UB^*	LB^*	$\frac{(UB^*-LB^*)}{LB^*}$
s11	6666.4	6299.3	5.83%	6660.1	6299.3	5.73%
s14	9005.7	8496.9	5.99%	9002.8	8496.9	5.95%
m2	17656.1	16592.6	6.41%	17643.7	16592.6	6.33%
m5	13534.8	12650.8	6.99%	13522.3	12663.0	6.79%

It is an upper bound on the optimal LR-SIMAP objective value z^*. Table 4.5 lists the perfect information solution values and their corresponding computational times for all 50 test instances, as compared to the results associated with the Lagrangian heuristic upper bound UB^* that are originally presented in Table 4.1 - 4.3. In 46 out of the 50 instances, z_{PI} provides a better (smaller) upper bound than UB^*. More specifically, z_{PI} is better than UB^* for all 20 medium instances and all 10 large instances. To measure how much z_{PI} improves UB^* in those cases, we compare their deviations from LB^*, $\frac{(z_{PI}-LB^*)}{LB^*}$ and $\frac{(UB^*-LB^*)}{LB^*}$, as shown in the fourth and fifth columns. The z_{PI} deviation is 0% in three small instances, implying LB^* is optimal in these cases. For the remaining sets of instances, s11 - s20, m1 - m20, and l1 - l10, the z_{PI} deviation is better than the UB^* deviation by an average of 2.91%, 2.23%, and 3.35%, respectively. Additionally, the computational time for calculating z_{PI}, t_{PI}, is significantly shorter than t_{Lag}, for all small and medium instances. More specifically, for the small instances, t_{PI} is smaller than t_{Lag} by one order of magnitude; for the medium instances, t_{PI} is approximately five times faster than t_{Lag}. For the large instances, t_{PI} is slightly larger than t_{Lag} in 5 of the 10 instances. In summary, the z_{PI} upper bound improves the Lagrangian upper bound UB^* and can be computed efficiently.

Table 4.5: Comparison: perfect information solution value z_{PI} vs. Lagrangian heuristic upper bound UB^*, on 50 test instances

	Upper bound		Gap (%)		Time (s)	
Filename	z_{PI}	UB^*	$\frac{(z_{PI}-LB^*)}{LB^*}$	$\frac{(UB^*-LB^*)}{LB^*}$	t_{PI}	t_{Lag}
s1	2428.4	2411.1	1.06%	0.34%	7.4	90.0
s2	2273.3	2263.7	0.60%	0.17%	6.4	91.1
s3	1813.2	1802.6	0.81%	0.22%	7.4	89.4
s4	2191.6	2195.0	0.02%	0.18%	6.3	89.5
s5	1671.3	1675.8	0.06%	0.33%	6.1	85.2
s6	1202.3	1198.8	0.76%	0.46%	6.1	90.4
s7	1905.4	1911.5	0.00%	0.32%	5.5	98.9
s8	972.6	976.6	0.00%	0.42%	5.9	86.6
s9	1083.1	1083.9	0.08%	0.15%	6.0	91.4
s10	2083.7	2085.8	0.00%	0.10%	6.0	88.4
s11	6418.1	6666.4	1.89%	5.83%	27.1	782.9

Table 4.5 (Cont'd)

	Upper bound		Gap (%)		Time (s)	
Filename	z_{PI}	UB^*	$\frac{(z_{PI}-LB^*)}{LB^*}$	$\frac{(UB^*-LB^*)}{LB^*}$	t_{PI}	t_{Lag}
s12	9513.1	9727.4	0.22%	2.47%	26.0	801.6
s13	9168.4	9375.0	0.03%	2.28%	25.7	775.9
s14	8721.3	9005.7	2.64%	5.99%	27.5	778.3
s15	5281.9	5434.3	1.34%	4.27%	23.4	778.0
s16	9547.8	9636.1	0.83%	1.76%	26.5	758.5
s17	7249.1	7507.5	0.66%	4.24%	29.1	757.9
s18	8229.0	8438.9	0.68%	3.25%	29.9	812.4
s19	8564.1	8871.4	1.21%	4.84%	28.4	673.1
s20	7873.3	8157.8	1.88%	5.57%	30.4	759.1
m1	16510.1	17067.7	2.10%	5.55%	133.5	862.4
m2	17150.0	17656.1	3.36%	6.41%	109.6	897.8
m3	16328.9	16902.7	2.67%	6.28%	132.5	869.6
m4	13998.5	14334.0	3.11%	5.58%	111.4	857.4
m5	13048.7	13534.8	3.15%	6.99%	107.3	860.6
m6	16249.1	16790.5	3.02%	6.45%	112.8	851.5
m7	15864.3	16344.8	2.99%	6.11%	102.1	854.8
m8	10779.2	11169.0	3.79%	7.55%	112.5	873.2
m9	16257.2	16667.9	3.04%	5.64%	106.6	881.2
m10	14794.9	15399.9	2.58%	6.78%	136.7	880.0
m11	16689.7	16874.1	4.75%	5.91%	178.2	869.8
m12	18714.1	18900.4	3.92%	4.96%	197.0	862.5
m13	13302.0	13421.7	4.37%	5.31%	194.1	855.5
m14	13471.4	13642.3	2.97%	4.27%	177.7	876.1
m15	15556.1	15712.1	4.96%	6.01%	134.5	836.9
m16	15418.7	15587.0	5.09%	6.23%	188.7	874.6
m17	13999.3	14257.3	5.46%	7.40%	233.7	861.4
m18	15828.8	15901.9	4.26%	4.74%	235.6	841.5
m19	19104.5	19342.1	4.50%	5.80%	195.3	870.5
m20	17475.2	17607.1	4.29%	5.08%	199.1	856.5
l1	28065.4	28895.1	5.19%	8.30%	2423.8	1836.0
l2	28065.4	28919.7	3.22%	6.36%	2516.7	1819.1
l3	33985.3	35152.7	4.57%	8.16%	1312.0	2487.5

Table 4.5 (Cont'd)

Filename	Upper bound		Gap (%)		Time (s)	
	z_{PI}	UB^*	$\frac{(z_{PI}-LB^*)}{LB^*}$	$\frac{(UB^*-LB^*)}{LB^*}$	t_{PI}	t_{Lag}
l4	27476.2	28391.1	4.78%	8.27%	2002.5	1775.2
l5	37505.8	38805.0	4.27%	7.89%	2452.8	1793.9
l6	30342.0	31280.7	2.35%	5.51%	1404.2	1831.6
l7	27130.6	28005.6	5.42%	8.81%	1612.8	1839.2
l8	28570.8	29501.3	4.13%	7.53%	2165.8	1945.3
l9	28921.5	29879.1	3.46%	6.88%	1822.1	1857.4
l10	34096.0	35133.2	4.03%	7.19%	1773.2	1855.3

4.3.3 Solution Analysis

In this subsection, we investigate SIMAP model solutions by summarizing the mitigation selection decision and corresponding critical paths. We conduct the analysis on small instance s11, which is solved to optimality using Gurobi across nine different budget levels.

Figure 4.2 illustrates the objective function value versus budget. The objective value is non-decreasing in the budget and is not convex with respect to the budget. The shape of the objective function value curve can be explained by the solutions reported in Table 4.6. Table 4.6 reports the optimal objective value z^*, the mitigation selection solution, and the optimal critical path for attacker $p = 1$ under scenario $\omega = 1$ for different budget levels B. Note each attacker $p \in P$ optimizes a critical path under each scenario $\omega \in \Omega$, in the last column we only report the optimal critical path under one scenario as a demonstration. When $B = 0$, no mitigation is selected, the objective value z^* is equal to the total expected project completion time with no delays. When B increases from 0 to 3, mitigations 11 and 15 are selected, which delay the total project completion time by 1862.1 to 5023.9. When B increases from 3 to 5, mitigation 6 is included in the solution set, which further delays the project completion time by 801.8. The objective value does not always increase as budget increases, such as the case when B increases from 5 to 7. Additionally, the mitigation selection solutions are not necessarily nested, for example when budget increases

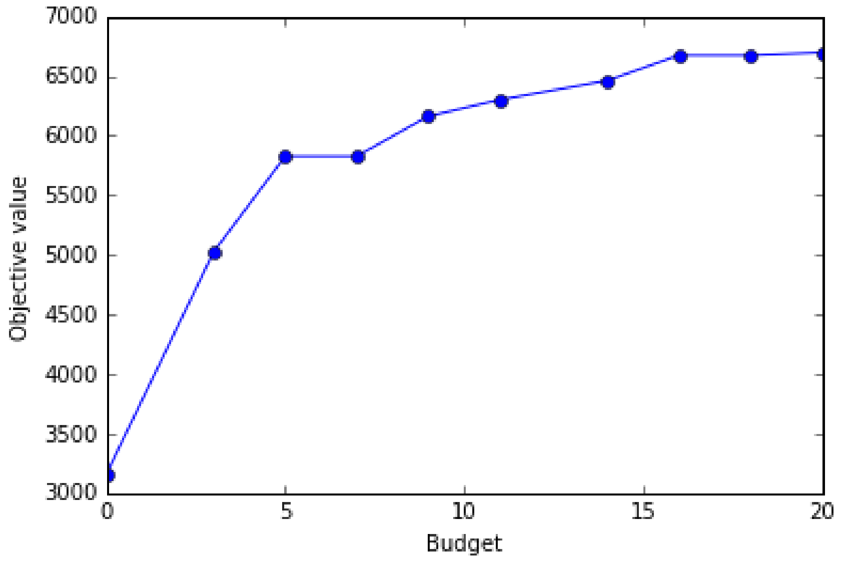


Figure 4.2: SIMAP objective value vs. budget

from 9 to 11, mitigation 15 is replaced by mitigation 17 while the remaining selection stays the same.

Table 4.6 indicates that the critical paths change as the budget level increases. There are $|P||\Omega|$ critical paths in total associated with each solution. As the budget increases, we select additional mitigations to delay arcs, and many and possibly all of the critical paths change. For example, the last column reports the optimal critical path under one situation $p = 1, \omega = 1$. The critical path changes several times as B increases, such as when B changes from 9 to 11, from 11 to 14, and from 18 to 20. These changes explain why the curve in Figure 4.2 is not convex.

4.3.4 Value of Stochastic Model Solutions

The stochastic model SIMAP takes random delay times under consideration in a stochastic programming approach. In this subsection, we demonstrate the benefit of solving the stochastic SIMAP as compared to the deterministic DIMAP by assessing the value of stochastic solution (VSS) associated with SIMAP. VSS is a common measure in stochastic programming that helps

Table 4.6: Optimal SIMAP objective function values and solutions with varying budgets on instance s11

B	z^*	Mitigation selection	Critical path for $p = 1, \omega = 1$
0	3161.8	$\{\}$	$0 \to 3 \to 4 \to 5 \to 7 \to 8 \to 11 \to 12 \to 13 \to 15 \to 16 \to 17 \to 18 \to 19$
3	5023.9	$\{11, 15\}$	$0 \to 3 \to 4 \to 5 \to 7 \to 8 \to 11 \to 12 \to 13 \to 15 \to 16 \to 17 \to 18 \to 19$
5	5825.7	$\{6, 11, 15\}$	$0 \to 3 \to 4 \to 5 \to 7 \to 8 \to 11 \to 12 \to 13 \to 15 \to 16 \to 17 \to 18 \to 19$
7	5825.7	$\{6, 11, 15\}$	$0 \to 3 \to 4 \to 5 \to 7 \to 8 \to 11 \to 12 \to 13 \to 15 \to 16 \to 17 \to 18 \to 19$
9	6162.0	$\{3, 6, 11, 15\}$	$0 \to 3 \to 4 \to 5 \to 7 \to 8 \to 11 \to 12 \to 13 \to 15 \to 16 \to 17 \to 18 \to 19$
11	6299.3	$\{3, 6, 11, 17\}$	$0 \to 3 \to 4 \to 6 \to 9 \to 10 \to 11 \to 12 \to 13 \to 15 \to 16 \to 18 \to 19$
14	6460.2	$\{1, 6, 11, 15\}$	$0 \to 3 \to 4 \to 5 \to 7 \to 8 \to 11 \to 12 \to 13 \to 15 \to 16 \to 17 \to 18 \to 19$
16	6672.2	$\{1, 6, 14, 15\}$	$0 \to 3 \to 4 \to 5 \to 7 \to 8 \to 11 \to 12 \to 13 \to 15 \to 16 \to 17 \to 18 \to 19$
18	6672.2	$\{1, 6, 14, 15\}$	$0 \to 3 \to 4 \to 5 \to 7 \to 8 \to 11 \to 12 \to 13 \to 15 \to 16 \to 17 \to 18 \to 19$
20	6696.8	$\{1, 6, 14, 17\}$	$0 \to 3 \to 4 \to 5 \to 7 \to 8 \to 11 \to 12 \to 13 \to 15 \to 16 \to 18 \to 19$

evaluate the value of implementing a mitigation strategy based on a "stochastic solution" instead of a "mean-value solution" [13]. To compute VSS, we first solve DIMAP with mean delay times $d_{ijp} = \sum_{\omega \in \Omega} \phi^\omega d^\omega_{ijp}, (i,j) \in A_p, p \in P$ and obtain a "mean-value solution" $\bar{\mathbf{x}}$. Then, given a set of realizations Ω for each project $p \in P$, we compute the expected-value of the total weighted completion time of all projects associated with this solution $\bar{\mathbf{x}}$, denoted as z_{mean}. Let z_{stoch} denote the objective function value of SIMAP with the same set of scenarios. VSS is the difference between z_{stoch} and z_{mean}, i.e., $VSS = z_{stoch} - z_{mean}$. All models are solved to optimality using Gurobi.

Table 4.7 reports VSS associated with SIMAP on 10 instances, vss1 - vss10. We increase the number of scenarios from 100 as in previous instances to 300, because a larger number of scenarios better illustrates the impact. Otherwise, the scenario-based data $d^\omega_{ijp}, (i,j) \in A_p, p \in P, \omega \in \Omega$ are generated using the same techniques described earlier. VSS for these solutions is reported in the second column from the right in Table 4.7. In addition to VSS, we also compute the ratio VSS/z_{stoch}, which is reported in the last column of Table 4.7 to facilitate comparison across instances. This ratio varies from 0.97% to 2.43% over the 10 instances, which can be significant, given the potential severe impact or consequence of adversarial supply chain attacks. We expect this ratio to increase with the size of the problem instance, since mitigation selection decisions in a stochastic solution have more ways to deviate from a mean-value solution. Therefore, despite the increase of solution time to solve SIMAP as compared to DIMAP, the stochastic solution adds significant value to the objective function as compared to a mean-value solution through its consideration of random delay times in a stochastic programming approach.

Table 4.7: Value of Stochastic Solution associated with SIMAP on 10 instances

Filename	$\|M\|$	$\|P\|$	$\|N_p\|$	$\|A_p\|$	$\|\Omega\|$	ℓ	z_{mean}	z_{stoch}	VSS	VSS/z_{stoch}
vss1	100	5	10	25	300	40	2002.3	2032.4	30.1	1.48%
vss2	100	5	10	22	300	40	2639.3	2691.8	52.6	1.95%
vss3	100	5	10	24	300	40	1908.1	1930.4	22.3	1.16%
vss4	100	5	10	20	300	40	2447.3	2501.0	53.6	2.14%
vss5	100	5	10	21	300	40	2159.5	2180.6	21.1	0.97%
vss6	100	5	10	25	300	40	1360.7	1393.3	32.7	2.34%
vss7	100	5	10	28	300	40	2972.2	3004.5	32.3	1.08%
vss8	100	5	10	22	300	40	1567.9	1606.9	39.1	2.43%
vss9	100	5	10	26	300	40	3010.5	3051.5	40.9	1.34%
vss10	100	5	10	22	300	40	2093.1	2113.6	20.5	0.97%

4.4 Conclusions

In this chapter, we address a critical challenge faced by federal and industry decision makers for selecting cost-effective mitigations to reduce risks from supply chain attacks, especially those from adversarial attacks that are sophisticated, hard to detect and likely to have severe consequences. We propose a max-min interdiction model that prioritizes security mitigations to maximally delay the completion of adversarial attacks. We consider various limitations for selecting mitigations arising from practical settings, including overlapping capacities, a budget constraint and multiple-choice constraints. We also consider attacks originated from multiple adversaries, each of which possesses a specific attack graph that defines exploits and their dependencies. Moreover, we address uncertain mitigation effectiveness by proposing an expected-value stochastic model variant that incorporates random delay times.

We formulate a deterministic model DIMAP and a stochastic model variant SIMAP for delaying multiple adversarial projects in IT infrastructure. We reformulate the models as a nested max-

max problem by dualizing the second stage linear program. Given the number of variables and constraints in the proposed models, especially the stochastic model, the max-max problem can be difficult to solve by a general purpose solver. We propose a Lagrangian heuristic for identifying near-optimal solutions in a computationally efficient manner. It decomposes the max-max problem into a number of easier-to-solve subproblems by relaxing some linking constraints. Next, in a subgradient optimization procedure, we calculate the Lagrangian relaxation objective function value as an upper bound, and we embed a procedure for quickly recovering feasible solutions and producing a lower bound. The subgradient method terminates when the optimality gap is below a pre-specified threshold or when it reaches a pre-specified maximum number of iterations.

In the computational study, we compare the solution quality and time of the Lagrangian heuristic with a mixed-integer programming solver Gurobi on 50 randomly generated instances and demonstrate that the Lagrangian heuristic is efficient in identifying optimal or near-optimal solutions. More specifically, the heuristic identifies optimal solutions in 19 of the 20 small instances. It outperforms Gurobi in 18 of the 20 medium instances and in all 10 large instances, since it identifies a feasible solution better than that of Gurobi in a significantly shorter amount of time within the time limit. Additionally, we present an alternative procedure for updating the upper bound by computing the objective value associated with the perfect information (or the wait-and-see) solution, which improves the Lagrangian heuristic upper bound in 46 of the 50 test instances with shorter computational times in most cases. Finally, we summarize mitigation selection and critical paths in solutions to one of the SIMAP model instance across various budget values. We demonstrate the value of these solutions by comparing SIMAP solutions to deterministic model solutions.

Our models are among the first interdiction models for deploying cost-effective mitigations for protecting against adversarial IT supply chain attacks. We propose a straightforward second stage problem for formulating adversarial attacks in IT supply chains. Our models offer the flexibility to be extended when more information becomes available regarding attack profiles, such as a budget constraint for selecting the adversaries' expediting efforts.

Chapter 5

An Exact Algorithm for Solving the Bilevel Facility Interdiction and Fortification Problem

In previous chapters, we investigate how to optimally allocate scarce resource to mitigate risks in cyber infrastructure. Cyber infrastructure protection, as an emerging and increasingly important topic, falls within the context of critical infrastructure protection. A stream of work in this broad area focuses on facility interdiction and fortification that identifies a subset of open facilities to fortify in order to minimize the interdiction cost, when extreme events (terrorist attacks, natural disasters, etc.) occurs that causes certain facilities out-of-service. In this chapter, we study a particular facility protection problem that interests us, the r-interdiction median problem with fortification (RIMF) [88], which fortifies q facilities with limited resource in order to minimize the post-interdiction cost when r facilities are interdicted.

In this chapter, we propose a new exact algorithm for solving RIMF [89], and show that it can be adapted to solve different variants of facility interdiction and fortification problems. RIMF has binary variables in both stages and is difficult to solve. Two main algorithms are proposed in the literature for solving it:

1. Scaparra and Church [88] reformulate the min-max problem RIMF into a maximum coverage problem with precedence constraints by enumerating all lower level solutions. They calculate lower and upper bounds to reduce the size of the reformulated problem, which is eventually solved with a general-purpose solver. The bounds are calculated as follows. They first sort the lower level solutions according to their objective function values in non-increasing order. Each lower level solution is called an "interdiction pattern," which is said to be "covered" if at least one facility in it is selected by the leader to fortify. The new objective is to select a set of facilities to fortify so that the most number of consecutive interdiction patterns (starting from top of the sorted list) is covered. They solve a greedy heuristic to obtain a lower bound, and then improve the lower bound and update an upper bound using an interval search procedure over the set of "uncovered" interdiction patterns.

2. Scaparra and Church [89] propose an alternate implicit enumeration approach for solving RIMF without full enumeration. The implicit enumeration approach is based on a tree search procedure, where an interdiction problem is solved at each node. The "branching" decision is based on a simple observation that the optimal fortification decision must include at least one facility from the "covered" interdiction patterns.

The proposed algorithm is most related to the first listed approach by Scaparra and Church [88]. Our approach also requires the complete enumeration of the lower level solutions and implements a similar greedy heuristic. Our approach differs in a key aspect that instead of obtaining a lower and an upper bound from the interval search procedure as in [88], we obtain an optimal solution from a newly-designed procedure consisting of implementing a modified greedy heuristic and solving a set cover problem iteratively. More specifically, in every iteration of Scaparra and Church's search procedure, they use linear interpolation to select a number k of interdiction patterns, and then solve a set cover problem to determine if the first k patterns can be "covered" by the defender within budget. If so, the result is used to update the lower bound and returned to the interpolation search for a larger k; otherwise, the upper bound is updated and a smaller k is selected by the interpolation search. After a pre-specified number of iterations, this procedure terminates with a lower and an upper bound. The interval search procedure might be inefficient given that the selection of k does not take full information from the previous solution. Our approach instead accounts for previous solution information every time we update lower and upper bounds, and thus is guaranteed to find an optimal solution upon termination. The central contribution of this chapter is a new exact algorithm for the solution of a general bilevel facility interdiction and fortification problem, which improves an existing algorithm [88] in the literature.

In summary, we propose a new exact algorithm for solving the r-interdiction median problem with fortification, which improves an interval search based existing method proposed by Scaparra and Church [88] by directing the search using greedy solutions that eventually obtains an optimal solution upon termination. Computational results show that our algorithm is more efficient in identifying optimal solutions than Scaparra and Church's algorithm across all test instances, and is up to one order of magnitude faster.

We process as follows. In Section 5.1, we describe the new exact algorithm as compared to Scaparra and Church's algorithm [88]. Then we demonstrate that it is guaranteed to find an optimal solution upon termination. In Section 5.2, we compare the performance of the two algorithms on benchmark and randomly generated test instances to demonstrate the efficiency of the proposed algorithm. Section 5.3 concludes the chapter.

5.1 Algorithm

In this section, we present the exact approach for solving the r-interdiction median problem with fortification (RIMF). We demonstrate that this algorithm identifies an optimal solution to RIMF upon termination. Moreover, we show that this algorithm can be adapted and modified to solve a broader range of facility interdiction and fortification problems.

We first present the formulation of the RIMF problem to facilitate further discussion. A min-max formulation of RIMF is presented in Scaparra and Church [89], where the defender in the upper level fortifies q facilities minimizes the total weighted transportation cost, while the attacker in the lower level interdicts r unfortified facilities to maximize the post-interdiction cost. Given a set of open facilities F and a set of demand nodes N with demand a_i at node $i \in N$ and distance d_{ij} between node $i \in N$ and facility $j \in F$. Three types of decision variables are defined:

- $x_j = 1$ if facility $j \in F$ is fortified, and 0 otherwise.
- $z_j = 1$ if facility $j \in F$ is interdicted, and 0 otherwise.
- $y_{ij} = 1$ if node $i \in N$ is assigned to facility $j \in F$ after interdiction.

The formulation of the r-interdiction median problem with fortification (RIMF) [89] is presented as follows.

$$\min \quad g(u) \tag{5.1}$$

$$\textbf{s.t.} \quad \sum_{j \in F} u_j = q \tag{5.2}$$

$$u_j \in \{0,1\}, \quad \forall j \in F \tag{5.3}$$

where

$$g(u) = \max \quad \sum_{i \in N} \sum_{j \in F} a_i d_{ij} y_{ij} \tag{5.4}$$

$$\textbf{s.t.} \quad \sum_{j \in F} y_{ij} = 1, \quad \forall i \in N \tag{5.5}$$

$$\sum_{j \in F} z_j = r \tag{5.6}$$

$$\sum_{k \in T_{ij}} y_{ik} \leq z_j, \quad \forall i \in N, j \in F \tag{5.7}$$

$$z_j \leq 1 - u_j, \quad \forall j \in F \tag{5.8}$$

$$y_{ij} \in \{0,1\}, \quad \forall i \in N, j \in F \tag{5.9}$$

$$z_j \in \{0,1\}, \quad j \in F \tag{5.10}$$

RIMF is a min-max bilevel programming problem with binary variables appearing in both levels. The defender in the upper level fortifies q facilities in (5.2) to minimize the maximal post-interdiction cost $g(u)$ (5.1), which is computed as the total demand weighted distances between demand nodes and non-interdicted facilities. The attacker in the lower level maximizes the post-interdiction cost (5.4) by interdicting r facilities in (5.6). A fortified facility can not be interdicted according to (5.8). Each demand node is assigned to its nearest non-interdicted facility, as enforced by (5.5) and (5.7).

Scaparra and Church's algorithm [88] does not directly incorporate the bilevel programming structure. They first enumerate all $\binom{p}{r}$ second stage interdiction patterns, where $p = |F|$, assuming there is no fortification. Each pattern corresponds to a post-interdiction objective value, $\sum_{i \in N} \sum_{j \in F} a_i d_{ij} y_{ij}$, that can be evaluated in polynomial time. The interdiction patterns are then sorted according to their objective values in non-increasing order, indexed by $H = \{1, 2, ..., |\binom{p}{r}|\}$. Let I_h be the set of interdicted facilities corresponding to each pattern $h \in H$, and let f_h be its corresponding post-interdiction cost. The defender aims to minimize the post-interdiction cost, therefore the defender wants to fortify a set of facilities that can make as many consecutive interdiction patterns (starting from $h = 1$) infeasible as possible. An interdiction pattern $h \in H$ is said to be infeasible if at least one facility in I_h is selected by the defender to fortify. Assume the optimal defender's solution x^* can cover at most h^* patterns, then the optimal attacker's solution is I_{h^*+1} and the optimal objective value is f_{h^*+1}.

Scaparra and Church's approach [88] tries to find a pattern close to h^* by performing an interval search. They first solve a greedy heuristic $Greedy$ that covers k patterns, in each iteration of which a facility that covers the most number of uncovered consecutive patterns is selected. Starting from there, they use an iterative search procedure to improve k. Let lower bound $LB = 0$ and upper bound $UB = f_{|\binom{p}{r}|}$. First, they solve a set cover problem to find out the minimum number of facilities required to cover the first k patterns, denoted as q^{sc}. If $q^{sc} \leq q$, then they update the lower bound $LB = f_{k+1}$, use a linear interpolation technique to find a new pattern k' that is larger than k, and go back to solve the set cover problem. Otherwise, if $q^{sc} > q$, then they update the upper bound $UB = f_{k+1}$, use a linear interpolation to find a new pattern k' that is smaller than k, and go back to solve the set cover problem. They terminate the interval search after a pre-specified maximum number of iterations or when the minimum tolerated optimality gap is reached. If the procedure terminates because of the former, then they solve a maximum coverage problem (using a general purpose solver) over the set H to find the optimal h^* using the found UB and LB to reduce the problem size. If the procedure terminates when the gap between UB^* and LB^* is smaller than the minimum tolerance, an optimal solution is claimed to be found.

The interval search procedure by Scaparra and Church [88] does not guarantee to find an optimal solution. They resort to solving a maximum coverage model reformulation of RIMF to reduce the gap between UB and LB returned by the interval search. Our algorithm improves their interval search procedure with a more guided search method consisting of iteratively employing a modified greedy heuristic to improve the lower bound and solving a modified set cover problem to verify optimality of the current solution and provide a direction for improvement. Our algorithm terminates naturally when an optimal solution is found. Moreover, with a guided search procedure, we expect to solve few set cover subproblems, which helps reduce the computational time.

We describe our algorithm in details as follows. First, we present the modified greedy heuristic $Greedy(M,k)$ used in our approach, where M is a set of facilities selected by the defender, and k is the index of the current pattern. $Greedy(M,k)$ is a variant of $Greedy$ as they both explore the same idea that in each iteration the facility that covers the most number of uncovered consecutive interdiction patterns is selected. While $Greedy$ is merely a one time implementation, the modified $Greedy(M,k)$ is repeatedly employed in the iterative procedure with M and k updated based on the solution of a modified set cover problem. Let $F = \{1, 2, ..., p\}$ be the set of operating facilities. In the first iteration, we have $M = \emptyset$ and $k = 0$. The details of $Greedy(M,k)$ is described as follows.

Greedy (M, k):

1. For each facility $m \in F \setminus M$, compute the consecutive number of interdiction patterns $h = k+1, k+2, ...$ covered by $M \cup \{m\}$, denoted as N_m. Select the facility $m^* =$

argmax$_{m\in F\setminus M}\{N_m\}$, and update $M = M \cup \{m^*\}, k = k + N_{m^*}$.

2. If $|M| \geq q$, stop and return M and k; else, go to Step 1.

Next, we solve a set cover problem to check if the greedy solution is optimal, i.e., if M is the optimal upper level solution. The set cover problem, denoted as $SC(M, k)$, is formulated as follows.

$\mathbf{SC(M, k)}$

$$\min \quad \sum_{m\in F} x_m \tag{5.11}$$

$$\sum_{m\in I_h} x_m \geq 1, \quad h = 1, ..., k \tag{5.12}$$

$$\sum_{m\in M} x_m \leq q - 1 \tag{5.13}$$

$$x_m \in \{0, 1\}, \quad m \in F \tag{5.14}$$

The objective (5.11) minimizes the total number of facilities required to cover the first k patterns $I_1, ..., I_k$. Constraint set (5.12) requires that at least one facility in each of the first k patterns is protected by the defender. Constraint (5.13) excludes the current greedy solution M from the feasible solution sets. Constraints (5.14) enforces decision variables to be binary. Let obj^{sc} be the optimal objective value of $SC(M, k)$, and X^{sc} be its optimal solution. Note that $obj^{sc} = |X^{sc}|$.

The optimal objective value obj^{sc} can be used to verify the optimality of the greedy solution M. Constraint (5.13) plays an important role in this phase as demonstrated below. If $obj^{sc} > q$, meaning the minimum number of facilities required to cover the first k patterns is greater than q when excluding M, then M is the optimal upper level solution since it is only set of facilities with size q that can cover the first k interdiction patterns. Next, if $obj^{sc} = q$, it means there exists at least one set of facilities with size q, besides M, that cover the first k patterns, i.e., X^{sc} is feasible to the original problem RIMF. It is possible that X^{sc} may cover more patterns in addition to k. It is also possible that $SC(M, k)$ has multiple optimal solutions besides X^{sc}. Let $\{X^{sc}_1, ..., X^{sc}_l\}$ be the set of all optimal solutions of $SC(M,k)$. We will discuss in the next section the technique for obtaining the complete set. For each solution in $\{X_1^{sc},...,X_l^{sc}\}$, we check how many additional consecutive patterns $I_{k+1}, I_{k+2}, ...$ that it covers. The solution that covers the most is the optimal upper level solution. Finally, if a smaller set of facilities is found that can cover the first k patterns, i.e., $obj^{sc} < q$, then M is not the best upper level feasible solution. We resort to the greedy heuristic $Greedy(M,k)$ to improve it by setting $M = X^{sc}$. Note k is updated through the greedy procedure. $Greedy(M,k)$ completes the current set of solution X^{sc} by greedily selecting $q - obj^{sc}$ additional facilities. Then, we enter into the next iteration by starting with solving the set cover problem again. The complete exact algorithm is described as follows.

Algorithm 1

1. Enumerate all the lower level solutions (interdiction patterns), denoted as $I_h, h \in H$, where $H = \{1, 2, ..., \binom{p}{r}\}$. Evaluate their corresponding post-interdiction objective values when there is no fortification, and sort the patterns according to their values in a non-increasing order. Relabel the interdiction patterns according to this order.

2. Initialization: $M \leftarrow \emptyset, k \leftarrow 0$.

3. $M, k \leftarrow Greedy(M,k)$.

4. Solve $SC(M,k)$, and let obj^{sc} and X^{sc} be its optimal objective value and solution, respectively.

   a) If $obj^{sc} > q$, stop and return optimal objective value f_{k+1}, optimal upper level solution M and lower level solution I_{k+1}.

   b) If $obj^{sc} = q$, identify all optimal solutions of $SC(M,k)$, $X_1^{sc}, ..., X_l^{sc}$, and examine how many additional consecutive patterns $I_{k+1}, I_{k+2}, ...$ each of them covers, denoted as $h_1, ..., h_l$. Let $i = \text{argmax}\{h_1, ..., h_l\}$. Stop and return the optimal objective value f_{k+h_i+1}, optimal upper level solution X_i^{sc} and lower level solution I_{k+h_i+1}.

   c) If $obj^{sc} < q$, set $M \leftarrow X^{sc}$ and go back to Step 3.

A main difference between Algorithm 1 and the interval search procedure in Scaparra and Church [88] is that we use a modified greedy heuristic $Greedy(M, k)$ to constantly improve k, the number of covered patterns, in each iteration using the set cover solution. Solving the set cover problem not only helps verify the optimality of the current best solution, but more importantly, it serves as a warm start for the greedy heuristic to find a better solution. In the following theorem, we demonstrate that Algorithm 1 always finds an optimal solution when it terminates.

Theorem 1. Algorithm 1 optimally solves RIMF.

Proof. We prove that Algorithm 1 guarantees to find an optimal solution upon termination.

Algorithm 1 terminates when it reaches Step 4 (a) or (b). We investigate the optimality of the solution under these two situations respectively.

When the algorithm reaches Step 4 (a), it indicates that M is the smallest set of facilities that covers the first k patterns. Given that $|M| = q$, it is a feasible upper level solution that covers the most number of consecutive interdiction patterns. Therefore, M is the optimal upper level solution, and correspondingly, I_{k+1} is the optimal lower level solution.

When the algorithm reaches Step 4 (b), more than one upper level solution with size q that can cover the first k patterns exists. As mentioned prior, all optimal solutions to $SC(M, k)$ can be found by solving $SC(M, k)$ repeatedly with additional constraints that eliminate previously found solutions. Let $S = \{M, X_1^{sc}, ..., X_l^{sc}\}$ be the set containing all feasible upper level solutions that can cover the first k patterns. If there exists a feasible solution that can cover the first k patterns with less than q facilities, it would have arrived at Step 4 (c) instead of Step 4 (b). Therefore, the optimal upper level solution must be contained in S. We check the number of additional consecutive patterns after k that each solution in S covers. The solution that covers the most is the optimal upper level solution. The set of interdicted facilities corresponding to first uncovered interdiction pattern is the optimal lower level solution.

□

5.1.1 Application to Facility Interdiction and Fortification Problems

Algorithm 1 has the flexibility to be adapted for solving different variants of facility interdiction and fortification problems. We discuss two examples in this subsection as a demonstration.

1. A generalized budget constraint in the defender's problem

We consider the case when the defender's cardinality constraint for selecting facilities, i.e., constraint (5.2), is generalized to a budget constraint with cost coefficients $b_j, j \in F$ and total budget B:

$$\sum_{j \in F} b_j u_j \leq B \tag{5.15}$$

To solve RIMF with this generalized budget constraint, we need to modify Step 3 and 4 of Algorithm 1. First, we modify Step 3 the $Greedy(M, k)$ algorithm to incorporate fortification cost associated with each facility. The generalized $Greedy(M, k)$, denoted as $Greedy_{general}(M, k)$, is described as follows:

$\mathbf{Greedy_{general}(M, k)}$:

1. For each facility $m \in F \setminus M$, compute the consecutive number of interdiction patterns $h = k+1, k+2, ...$ that are covered by $M \cup \{m\}$, denoted as N_m. Let $m^* = \text{argmax}_{m \in F \setminus M} \{N_m / b_m\}$.
2. If $\sum_{m \in M} b_m + b_{m^*} \geq B$, stop and return M and k; else update $M = M \cup \{m^*\}, k = k + N_{m^*}$, and go to Step 1.

The main difference between the two greedy algorithms is that in each iteration, $Greedy_{general}(M, k)$ selects the facility with the highest "profit-to-cost" ratio (i.e., N_m/b_m), while $Greedy(M, k)$ selects the one with the highest "profit" (i.e., N_m).

Step 4 in Algorithm 1 also undergoes several changes. First, instead of solving the set cover problem $SC(M, k)$ defined in (5.11) - (5.14), we solve the following weighed set cover (WSC) problem:

$\mathbf{WSC(M, k)}$:

$$\min \quad \sum_{m \in F} b_m x_m \tag{5.16}$$

$$\sum_{m \in I_h} x_m \geq 1, \quad h = 1, ..., k \tag{5.17}$$

$$\sum_{m \in M} b_m x_m \leq \sum_{m \in M} b_m - \epsilon \tag{5.18}$$

$$x_m \in \{0, 1\}, \quad m \in F \tag{5.19}$$

where ϵ is a minimal tolerance by which constraint (5.18) can be violated, for example, if $b_m, m \in M$ are integer, then $\epsilon = 1$. The objective of the new $WSC(M, k)$ is to find a minimal weight set of facilities, excluding the current greedy solution M, that covers the first k interdiction patterns. Let the objective value and solution of $WSC(M, k)$ be obj^{wsc} and X^{wsc}, respectively.

Algorithm 1 Step 4 (a), (b) and (c) criteria also need to change. Essentially, instead of comparing obj^{sc} with q, we compare obj^{wsc} with $\sum_{m \in M} b_m$. We now present the generalized Algorithm 1 for solving RIMF with a budget constraint in the first stage:

Generalized Algorithm 1

1. Same as Algorithm 1.
2. Same as Algorithm 1.
3. $M, k \leftarrow Greedy_{general}(M, k)$.
4. Solve $WSC(M, k)$, and let obj^{wsc} and X^{wsc} be its optimal objective value and solution, respectively.
    a) If $obj^{wsc} > \sum_{m \in M} b_m$, stop and return optimal objective value f_{k+1}, optimal upper level solution M and lower level solution I_{k+1}.
    b) If $obj^{wsc} = \sum_{m \in M} b_m$, identify all optimal solutions of $WSC(M, k)$, $X_1^{wsc}, ..., X_l^{wsc}$, and examine how many additional consecutive patterns $I_{k+1}, I_{k+2}, ...$ each of them

covers, denoted as $h_1, ..., h_l$. Let $i = \text{argmax}\{h_1, ..., h_l\}$. Stop and return the optimal objective value f_{k+h_i+1}, optimal upper level solution X_i^{wsc} and lower level solution I_{k+h_i+1}.

c) If $obj^{wsc} < \sum_{m \in M} b_m$, set $M \leftarrow X^{wsc}$ and go back to Step 3.

It can be easily shown that the Generalized Algorithm 1 solves the generalized RIMF optimally.

2. Different operations in the attacker's problem:

The attacker's problem can also vary depending on the operations. For example, the base model can be a maximal covering location problem [26] instead of a p-median location problem as in RIMF. We describe this model variant and its corresponding Algorithm 1 adaptation conceptually here, since only a minimal change needs to be made on Algorithm 1. Consider a typical maximal covering problem setting, the objective of the defender is to maximize the demand coverage by fortifying p facilities, while the attacker in the second stage aims to minimize the demand coverage by interdicting r facilities. A demand node is covered if there exists at least one non-interdicted facility that original covers it. Similar to RIMF, we also require that a fortified facility can not be interdicted.

To solve this maximal covering variant of RIMF, we can apply Algorithm 1 directly. The only difference for solving this problem variant using Algorithm 1 as compared to that of RIMF is that in Step 1 we evaluate the post-interdiction coverage function associated with the maximal covering problem instead of a cost function associated with a p-median location problem. The rest of Algorithm 1 applies naturally.

5.2 Computational Results

In this section, we present computational results to demonstrate the efficiency of our exact approach in identifying optimal RIMF solution as compared to Scaparra and Church's approach [88]. Both algorithms were programmed and run with Python 2.7.12. The models were solved using Gurobi 7.0.1 with a Python interface. The tests were run on an Intel Core i5 CPU at 2 GHz with 8 GB of RAM with a time limit of one hour.

We compare the solution time of the two algorithms on two datasets. The first dataset is generated based on the p-median instances posted in the online OR library [9], and the second dataset is created in a random manner. We describe how to generate instances in each dataset as follows. Let N denote the set of nodes where a facility can be located, and p denote the number of located facilities. First, we select four instances, pmed4 ($|N| = 100, p = 20$), pmed5 ($|N| = 100, p = 33$), pmed8 ($|N| = 200, p = 20$), and pmed13 ($|N| = 300, p = 30$), from the OR library, and solve each of them to obtain the optimal set of located facilities, respectively. Then based on each instance, we create six RIMF test instances with varying q and r values, i.e., $q = 3 - 5$ and $r = 3 - 7$. In total, we generate 24 RIMF test instances for the first dataset. Next, we randomly generated three p-median test instances with $|N| = 100, p = 10$, $|N| = 200, p = 20$, and $|N| = 300, p = 30$. Distances between each pair of demand nodes, and demand at each node, are generated as Uniform $(0, 10)$ random variables using a pseudo-random generator, respectively. In total, we generate 18 random RIMF test instances with different combination of q and r values, i.e., $q = 3-5$ and $r = 4 - 7$.

Table 5.1 and 5.2 report the computational results of Algorithm 1 (ALG_1) and Scaparra and Church's algorithm ($ALG_{S\&C}$) on two datasets, respectively, including i) the p-median filename, the p, q and r values, and the total number of enumerated interdiction patterns $|H|$; ii) the optimal RIMF objective value; ii) the time spent on enumerating and sorting the interdiction patterns, the computational time of ALG_1 and $ALG_{S\&C}$. Note that the time of $ALG_{S\&C}$ includes the time of the interval search procedure and the solution time of the reduced-sized maximum coverage problem after the interval search when a minimum tolerance optimality gap is not met. We use the sort algorithm in Python's Numpy package to quickly sort the objective values associated with interdiction patterns. Moreover, while implementing ALG_1 Step 4(b), we use a Gurobi solver feature called solution pool to identify multiple optimal solutions of $SC(M, k)$ if more than one exists. Finally, if a procedure or algorithm exceeds the one hour time limit, its time is replaced with NA^1, and the corresponding values on that row are replaced with dash.

Of the 42 test instances, ALG_1 finds the optimal solution for 38 instances, and $ALG_{S\&C}$ finds the optimal solution for 36 instances, within time limit. There are two instances, indexed as 23 and 41, for which ALG_1 solves to optimality in two minutes while $ALG_{S\&C}$ runs out of time. For the 36 instances where both algorithms find the optimal solution, ALG_1 is always faster than $ALG_{S\&C}$. In 14 of these cases, ALG_1 is faster than $ALG_{S\&C}$ by approximately one order of magnitude. These results demonstrate that ALG_1 is more efficient than $ALG_{S\&C}$ in identifying optimal solutions to RIMF.

Both our algorithm and Scaparra and Church's algorithm require the complete enumeration of the lower level solutions. One of the disadvantage with this approach is that the enumeration might be computationally expensive when the number of facilities p and the attacker's capacity r become large, as you can see in instances $18, 21, 23, 24, 39, 41, 42$ in Table 5.1 - 5.2 that the computational time of enumeration and sort is significantly larger than the computational time of both algorithms. We argue that this might be rare in practice as a simultaneous interdiction on a large number of facilities is very unlikely to occur [88]. Therefore, we consider our approach practical for handling problem instances with reasonable sizes.

5.3 Conclusions

In this chapter, we present a new exact algorithm for solving the r-interdiction median problem with fortification (RIMF). The proposed algorithm improves an existing approach for solving RIMF proposed by Scaparra and Church [88] by guaranteeing to find an optimal solution upon termination. It consists of implementing two main steps iteratively: i) solving a greedy heuristic to obtain an improved feasible solution; ii) solving a set cover problem to verify optimality of this solution and to provide an direction for solution improvement. We also demonstrate that this approach can be applied to different variants of facility interdiction and fortification problems on two examples. Finally, we compare the proposed algorithm with Scaparra and Church's method on a public dataset and a randomly generated dataset and demonstrate that our algorithm is more efficient in identifying optimal solutions with faster solution times up to one magnitude

Table 5.1: Computational results of Algorithm 1 vs. Scaparra and Church's algorithm on OR library data

	p-median						Time (s)		
Index	filename	p	q	r	$\lvert H\rvert$	obj. value	enum. and sort	ALG_1	$ALG_{S\&C}$
1	pmed4	20	3	4	4845	3961.0	5.72	0.21	0.57
2	pmed4	20	3	5	15504	4171.0	18.72	1.06	2.39
3	pmed4	20	3	6	38706	4388.0	47.34	2.91	7.87
4	pmed4	20	5	5	15504	4028.0	18.48	3.85	18.82
5	pmed4	20	5	6	38706	4220.0	46.62	73.78	317.35
6	pmed4	20	5	7	77520	4415.0	103.70	211.21	2296.63
7	pmed8	20	3	4	4845	5394.0	11.57	0.39	0.66
8	pmed8	20	3	5	15504	5613.0	36.75	1.06	2.49
9	pmed8	20	3	6	38706	5838.0	92.75	2.84	7.99
10	pmed8	20	5	5	15504	5489.0	36.55	3.58	17.63
11	pmed8	20	5	6	38706	5694.0	97.19	79.67	283.01
12	pmed8	20	5	7	77520	5903.0	189.65	298.86	2108.96
13	pmed5	33	3	4	40920	1864.0	61.37	2.46	3.95
14	pmed5	33	3	5	237336	1968.0	384.90	18.31	37.39
15	pmed5	33	3	6	1107568	2073.0	1811.29	135.98	239.17
16	pmed5	33	5	5	237336	1889.0	367.88	31.39	49.81
17	pmed5	33	5	6	1107568	1984.0	1732.11	166.60	395.21
18	pmed5	33	5	7	4272048	-	NA[1]	-	-
19	pmed13	30	3	4	27405	4905.0	124.26	1.47	2.69
20	pmed13	30	3	5	142506	5026.0	602.94	11.33	20.99
21	pmed13	30	3	6	593775	5150.0	2548.03	54.67	122.69
22	pmed13	30	5	5	142506	4980.0	597.45	14.24	28.36
23	pmed13	30	5	6	593775	5098.0	2697.09	81.30	NA[1]
24	pmed13	30	5	7	2035800	-	NA[1]	-	-

1: out of time

Table 5.2: Computational results of Algorithm 1 vs. Scaparra and Church's algorithm on randomly generated data

	p-median						Time (s)		
Index	filename	p	q	r	$\|H\|$	Obj. value	enumeration and sort	ALG_1	$ALG_{S\&C}$
25	test1	20	3	4	4845	277.2	6.3	0.4	0.7
26	test1	20	3	5	15504	300.4	21.3	1.0	2.5
27	test1	20	3	6	38706	315.3	48.8	3.1	8.9
28	test1	20	5	5	15504	289.1	19.2	2.9	17.8
29	test1	20	5	6	38706	301.3	49.1	41.6	278.3
30	test1	20	5	7	77520	315.5	95.5	1534.2	3395.0
31	test2	25	3	4	12650	534.2	34.3	0.9	1.3
32	test2	25	3	5	53130	553.0	164.7	4.5	8.0
33	test2	25	3	6	177100	573.3	477.3	13.5	34.7
34	test2	25	5	5	53130	541.6	145.6	4.6	39.5
35	test2	25	5	6	177100	559.9	483.3	86.0	382.2
36	test2	25	5	7	480700	-	1296.2	NA[1]	NA[1]
37	test3	30	3	4	27405	502.3	134.9	1.0	2.7
38	test3	30	3	5	142506	520.2	669.6	9.6	19.2
39	test3	30	3	6	593775	539.3	2656.9	53.9	129.0
40	test3	30	5	5	142506	510.7	673.2	19.0	30.7
41	test3	30	5	6	593775	528.1	2835.1	75.9	NA[1]
42	test3	30	5	7	2035800	-	NA[1]	-	-

1: out of time

across all instances.

Chapter 6

Summary

Protecting the cyber infrastructure is becoming increasingly important as the information systems are deeply embedded in the critical infrastructure systems in the United States, and globally. This dissertation demonstrates the applicability of operations research in cybersecurity planning that helps decision maker prioritize mitigation investment to enhance the security of IT infrastructure, and more generally the critical infrastructure. It also shows the potential of applying operations research to more areas of interest in the domain of cybersecurity, especially in cyber physical systems security.

One of the critical challenges faced by operations researchers in this area is understanding the vulnerabilities and attack profiles in the supply chain given its complex and layered structure, and lack of real data. Models and algorithms in this dissertation provide a flexible foundation for future research, especially in the area of infrastructure protection against adversarial attacks.

6.1 Extensions

Cybersecurity planning in the presence of adversarial attackers is an important topic given the severe damage attacks bring and the complexity of the corresponding problems. The interdiction models we propose in this dissertation can be extended to include more dimensionality in the realistic setting. Section 6.1.1 shows one direct way to extend DIMAP and SIMAP that incorporate task expediting and limited expediting budget in the attacker's problems. Section 6.1.2 discusses the potential of developing mixed-integer or bilevel programming models based on a new and promising concept in system security, the attack-defense tree.

6.1.1 Attacker's Problem with Task Expediting

We extend the interdiction models DIMAP and SIMAP by considering an attacker's problem with expediting. More specifically, we assume that the attackers have certain amount of resources to expedite some of their tasks with the same objective of aiming to complete the project as soon as possible. Let d_{ijp} be the expediting cost per unit for arc (task) $(i,j) \in A_p, p \in P$, and D_p be the total budget or resource for task expediting of project $p \in P$. We introduce the following additional variables:

- e_{ij}: amount of time that arc (task) $(i,j) \in A_p, p \in P$ is expedited.

The SIMAP model with Expediting, SIMAPE, is formulated as follows:

SIMAPE

$$\max \quad \sum_{p \in P} c_p \Big(\sum_{\omega \in \Omega_p} \phi_p^{\omega} s_p^{\omega}(\mathbf{x}, \mathbf{z}) \Big) \tag{6.1}$$

$$\text{s.t.} \quad \sum_{m \in M} b_m x_m \leq B \tag{6.2}$$

$$\sum_{m \in H_k} x_m \leq 1, \quad \forall k = 1, ..., \ell \tag{6.3}$$

$$z_{ijp} \leq \sum_{m \in M_{ijp}} x_m, \quad \forall (i,j) \in A_p, p \in P \tag{6.4}$$

$$x_m \in \{0,1\}, \quad \forall m \in M \tag{6.5}$$

$$z_{ijp} \in \{0,1\}, \quad \forall (i,j) \in A_p, p \in P \tag{6.6}$$

For each adversarial project $p \in P$, the completion time of its fastest attack under scenario $\omega \in \Omega_p$ after being compromised by the defender, s_p^ω, is computed as follows:

$$s_p^\omega(\mathbf{x}, \mathbf{z}) = \min \quad s_{end,p}^\omega \tag{6.7}$$

$$\text{s.t.} \quad s_{jp}^\omega - s_{ip}^\omega \geq t_{ijp} - e_{ijp} + d_{ijp}^\omega z_{ijp}, \quad \forall (i,j) \in A_p \tag{6.8}$$

$$\sum_{(i,j)\in A_p} d_{ijp} e_{ijp} \leq D_p \tag{6.9}$$

$$e_{ijp} \leq \bar{e}_{ijp}, \quad (i,j) \in A_p \tag{6.10}$$

$$s_{start,p}^\omega = 0 \tag{6.11}$$

$$s_{ip}^\omega \geq 0, \quad \forall i \in N_p \tag{6.12}$$

The defender's problem (6.1) - (6.6) in SIMAPE is identical to that in SIMAP presented in Chapter 4, while each attacker's problem now incorporates task expediting. More specifically, in the precedence constraints (6.8), the start time of event j for attacker p is expedited by an amount e_{ijp}. The budget constraint (6.9) states that the total cost for task expediting can not succeed attacker p's budget D_p. Constraints (6.10) enforce an upper bound on each expediting amount e_{ijp}.

SIMAPE can be reformulated by dualizing the second stage linear programs and linearized non-linear objective terms using similar techniques as in previous models. We introduce an additional dual variable u_p associated with the attacker's budget constraint (6.9) for each attacker $p \in P$.

The linearized reformulated SIMAPE, LR-SIMAPE, is formulated as below.

LR-SIMAPE

$$\max \quad \sum_{p\in P} c_p \sum_{\omega\in\Omega_p} \phi_p^\omega \Big(\sum_{(i,j)\in A_p} (t_{ijp} + d_{ijp}^\omega) y_{ijp}^\omega + \sum_{(i,j)\in A_p} t_{ijp} y_{ijp}^{'\omega} \Big) - \sum_{p\in P} \Big(D_p - \sum_{(i,j)\in A_p} d_p \bar{e}_{ijp} \Big) u_p \tag{6.13}$$

$$\text{s.t.} \quad (6.2), (6.3), (6.5)$$

$$y_{ijp}^\omega \le \sum_{m\in M_{ijp}} x_m, \quad \forall (i,j) \in A_p, \omega \in \Omega_p, p \in P \tag{6.14}$$

$$\sum_{j:(i,j)\in A_p} (y_{ijp}^\omega + y_{ijp}^{'\omega}) - \sum_{j:(j,i)\in A_p} (y_{jip}^\omega + y_{jip}^{'\omega}) = \begin{cases} 1, & \text{if } i = start \\ 0, & \text{if } i \in N \setminus \{start, end\} \\ -1, & \text{if } i = end. \end{cases}$$

$$i \in N_p, \omega \in \Omega_p, p \in P \tag{6.15}$$

$$y_{ijp}^\omega + y_{ijp}^{'\omega} - d_{ijp} u_p \ge 0, \quad (i,j) \in A_p, \omega \in \Omega_p, p \in P \tag{6.16}$$

$$0 \le y_{ijp}^\omega, y_{ijp}^{'\omega} \le 1, \quad \forall (i,j) \in A_p, \omega \in \Omega_p, p \in P \tag{6.17}$$

The Lagrangian heuristic proposed for solving LR-SIMAP cannot be directly used to solve the new model LR-SIMAPE. Specialized algorithm that facilitates decomposition should be developed to solve this problem.

6.1.2 Attack-Defense Tree

Attack-defense tree (ADTree) is an extension of attack tree [65] with countermeasures or response strategies that visualizes the interaction between the attacker and the defender. Its formal definition can be found in Kordy et al. [62]. An example of ADTree taken from their paper is shown in Figure 6.1. Red circled nodes represent attack goals of the attacker, and the green circled nodes represent the countermeasures implementable by the defender. The child node of an attack goal can be its sub-goal, or its countermeasure, which is missing from the structure of an attack tree. Likewise, the child node of a countermeasure can either be a sub-countermeasure

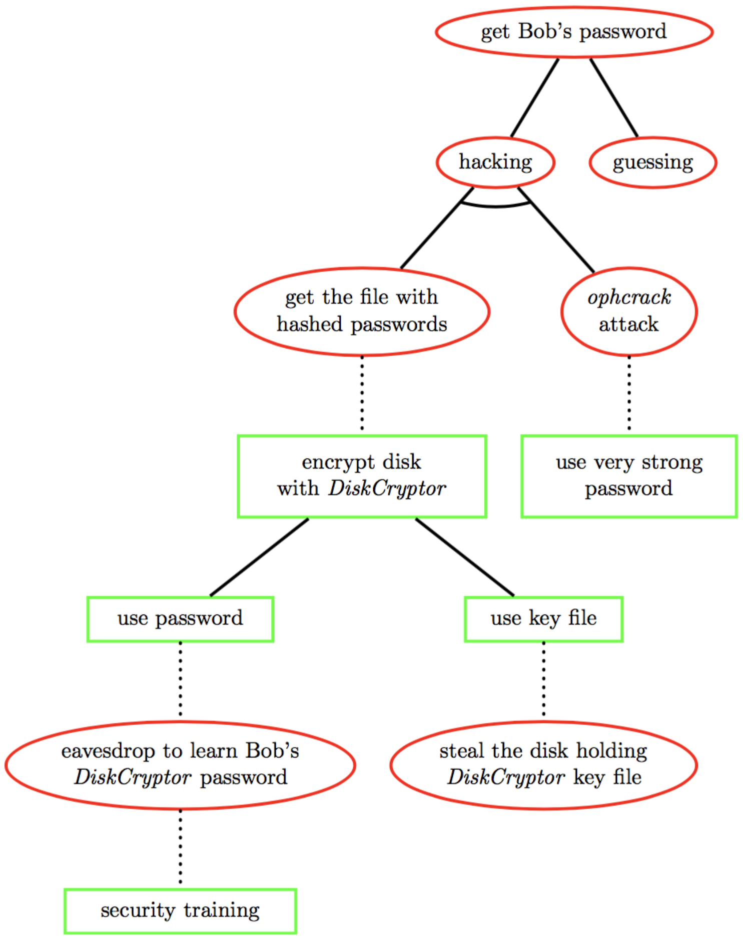

get Bob's password
hacking
guessing
get the file with hashed passwords
ophcrack attack
encrypt disk with *DiskCryptor*
use very strong password
use password
use key file
eavesdrop to learn Bob's *DiskCryptor* password
steal the disk holding *DiskCryptor* key file
security training

Figure 6.1: ADTree for getting a password

or a counter attack goal. We refer to their paper for more detailed construction assumptions of an ADTree. In summary, ADTree enhances the formalism of attack tree, which is widely recognized and used in industry practice, by explicitly integrating countermeasures (mitigations) to the model. Quantitative analysis based on ADTree can be used to analyze the effect and consequences of deploying a countermeasure or a set of countermeasures.

We see the potential of applying ADTree to enhance our attack modeling in the optimization framework, especially for adversarial attacks. In their recent paper, Kordy et al. [61] propose the use of integer programming technique to determine the set of mitigations subject to a limited budget to maximize coverage of attack strategies based on ADTree. What differs their research from ours is that they extract information from ADTree regarding the attack strategies and the corresponding sets of countermeasures, while we take this information as inputs. One of the key contribution of their work is the extraction algorithm and the demonstration of integrating mixed-integer programming techniques into ADTree decision making.

The modeling power of ADTree and its increasing application in the security community makes it appealing for the extension of our models. The interdiction models we propose exploit a two-stage sequential decision making framework, that is, a defender selects a set of countermeasures to fortify the system, and then the attacker reacts by selecting corresponding attack strategies. But the ADTree integrates “multiple-stage” decisions where the defender and the attack can counter each other’s goals multiple times. For example, in Figure 6.1, mitigation “encrypt disk with DiskCryptor” counters attack goal “get the file with hashed passwords”, its sub-measure “use password” also has a counter attack goal “eavesdrop to learn Bob’s DiskCryptor password’, furthermore, this attack goal has a countermeasure “security training”. It would be interesting to integrate ADTree into our attack modeling so as to extend our interdiction models to a multi-stage sequential decision making framework.

Bibliography

[1] Target hackers broke in via hvac company - Krebs on Security. `https://krebsonsecurity.com/2014/02/target-hackers-broke-in-via-hvac-company/`. Accessed: 2014-02-14.

[2] Exclusive: The OPM breach details you haven't seen. `https://fcw.com/articles/2015/08/21/opm-breach-timeline.aspx`. Accessed: 2015-08-21.

[3] Tyupkin: manipulating ATM machines with malware. `https://securelist.com/blog/research/66988/tyupkin-manipulating-atm-machines-with-malware/`. Accessed: 2014-10-07.

[4] A. Afful-Dadzie and T. Allen. Data-driven cyber-vulnerability maintenance policies. *Journal of Quality Technology*, 46(3):234–250, 2014.

[5] A. A. Ageev and M. Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3): 307–328, 2004.

[6] S. Ahmed. Convexity and decomposition of mean-risk stochastic programs. *Mathematical Programming*, 106(3):433–46, 2006.

[7] D. Aksen, N. Piyade, and N. Aras. The budget constrained r-interdiction median problem with capacity expansion. *Central European Journal of Operations Research*, 18(3):269–91, 2010.

[8] A. Badanidiyuru and J. Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA

’14, pages 1497–1514, Philadelphia, PA, USA, 2014. Society for Industrial and Applied Mathematics.

[9] J. E. Beasley. A Lagrangian heuristic for set-covering problems. *Naval Research Logistics*, 37:151–164, 1990.

[10] A. Ben-Tal, L. El Ghaoui, and N. Nemirovski. *Robust Optimization*. Princeton University Press, Princeton, NJ, 2009.

[11] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numebrische Mathematik*, 4:238–252, 1962.

[12] D. Bertsimas, D. B. Brown, and C. Caramanis. Theory and applications of robust optimization. Technical report, Massachusetts Institute of Technology, Cambridge, MA, 2010.

[13] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer, 2011.

[14] S. Bistarelli, F. Fioravanti, and P. Peretti. Defense trees for economic evaluation of security investments. In *First International Conference on Availability, Reliability and Security (ARES’06)*, 2006.

[15] S. Bistarelli, M. Dall’Aglio, and P. Peretti. *Strategic Games on Defense Trees*, pages 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[16] L. Brotcorne, G. Laporte, and F. Semet. Ambulance location and relocation models. *European Journal of Operational Research*, 147(3):451 – 463, 2003.

[17] G. G. Brown, W. M. Carlyle, J. Royset, and R. K. Wood. *On the complexity of delaying an adversary’s project*, pages 3–17. Springer US, Boston, MA, 2005.

[18] G. G. Brown, W. M. Carlyle, R. C. Harney, E. M. Skroch, and R. K. Wood. Interdicting a nuclear weapons project. *Operations Research*, 57(4):866–877, 2009.

[19] N. Buchbinder, M. Feldman, and R. Schwartz. Comparing apples and oranges: Query tradeoff in submodular maximization. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 1149–1168, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics.

[20] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[21] A. Caprara, P. Toth, and M. Fischetti. Algorithms for the set covering problem. *Annals of Operations Research*, 98:353–371, 2000.

[22] M. Caramia and R. Mari. Enhanced exact algorithms for discrete bilevel linear problems. *Optimization Letters*, 9:1447–1468, 2015.

[23] C. Chekuri and A. Kumar. Maximum coverage problem with group budget constraints and applications. In *Proceedings of APPROX-RANDOM, Lecture Notes in Computer Science*, 3122, pages 72–83, 2004.

[24] G. Chen, M. S. Daskin, Z. J. Shen, and S. Uryasev. The α-reliable mean-excess regret model for stochastic facility location modeling. *Naval Research Logistics*, 53(7):617–626, 2006.

[25] R. Church and C. ReVelle. The maximal covering location problem. *Papers in regional science*, 32(1):101–118, 1974.

[26] R. Church and C. ReVelle. The maximal covering location problem. *Papers of the Regional Science Association*, 32:101–118, 1974.

[27] R. L. Church and M. P. Scaparra. Protecting critical assets: the r-interdiction median problem with fortification. *Geographical Analysis*, 39:129–146, 2006.

[28] K. J. Cormican, D. P. Morton, and R. K. Wood. Stochastic network interdiction. *Operations Research*, 46(2):184–197, 1998.

[29] M. S. Daskin. *Network and Discrete Location: Models, Algorithms, and Applications, 2nd Edition*. John Wiley & Sons, New York, 2013.

[30] M. S. Daskin, S. M. Hesse, and C. S. ReVelle. α-reliable p-minimax regret: a new model for strategic facility location modeling. *Location Science*, 5(4):227–246, 1997.

[31] S. DeNegre. Interdiction and discrete bilevel linear programming. *PhD Thesis*, Lehigh University, 2011.

[32] Director of National Intelligence. Worldwide threat assessment of the U. S. intelligence community. Unclassified statement for the record, Senate Select Committee on Intelligence, Washington, D.C., 2015.

[33] M. E. Dyer, N. Kayal, and J. Walker. A branch and bound algorithm for solving the multiple choice knapsack problem. *Journal of Computational and Applied Mathematics*, 11:231–249, 1984.

[34] M. E. Dyer, W. O. Riha, and J. Walker. A hybrid dynamic programming/branch-and-bound algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research*, 58:43–54, 1995.

[35] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[36] M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl. A new general-purpose algorithm for mixed-integer bilevel linear programs. *Operations Research*, to appear, 2017.

[37] M. L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

[38] M. L. Fisher. An applications oriented guide to lagrangian relaxation. *Interfaces*, 15(2): 10–21, 1985.

[39] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. *An analysis of approximations for maximizing submodular set functions - II*, pages 73–87. Springer Berlin Heidelberg, Berlin, Heidelberg, 1978.

[40] A. M. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 1972.

[41] A. M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

[42] J. Goldberg. Operations research models for the deployment of emergency services vehicles. *EMS Management Journal*, 1(1):20–39, 2004.

[43] Booz Allen Hamilton. Vetting the global supply chain. Technical report, Booz Allen Hamilton, 2015.

[44] R. C. Harney, G. G. Brown, W. M. Carlyle, E. M. Skroch, and R. K. Wood. Anatomy of a project to produce a first nuclear weapon. *Science and Global Security*, 14:163–182, 2006.

[45] M. Held, P. Wolfe, and H. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 9(6):62–88, 1974.

[46] The White House. Executive order: Improving critical infrastructure cybersecurity. Executive order, Office of the Press Secretary, Washington, D.C., 2013.

[47] The White House. Presidential policy directive: Critical infrastructure security and resilience. Presidential policy directive, Office of the Press Secretary, Washington, D.C., 2013.

[48] The White House. The comprehensive national cybersecurity initiative. Technical report, Washington, D.C., 2015.

[49] The White House. Commission on enhancing national cybersecurity. Report on securing and growing the digital economy, Washington, D.C., 2016.

[50] E. Israeli and R. K. Wood. Shortest-path network interdiction. *Networks*, 40:97–111, 2002.

[51] S. H. Jacobson, J. E. Virta, J. M. Bowman, J. E. Kobza, and J. J. Nestor. Modeling aviation baggage screening security systems: A case study. *IIE Transactions*, 35(3):259–269, 2001.

[52] S. H. Jacobson, L. A. McLay, J. E. Kobza, and J. M. Bowman. Modeling and analyzing multiple station baggage screening security system performance. *Naval Research Logistics*, 52(1):30–45, 2005.

[53] U. Janjarassuk and J. T. Linderoth. Reformulation and sampling to solve a stochastic interdiction problem. *Networks*, 52:120–132, 2008.

[54] S. Jha, O. Sheyner, and J. Wing. Two formal analyses of attack graphs. In *Computer Security Foundations Workshop, 2002. Proceedings. 15th IEEE*, pages 49–63. IEEE, 2002.

[55] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Berlin, 2004.

[56] J. E. Kelley. Critical-path planning and scheduling: Mathematical basis. *Operations Research*, 9(3):296–320, 1961.

[57] S. Khuller, A. Moss, and J. S. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[58] A. J. Kleywegt, A. Shapiro, and T. Homem de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2001.

[59] A. Koç and D. P. Morton. Prioritization via stochastic optimization. *Management Science*, 61(3):586–603, 2014.

[60] A. Koc, D. P. Morton, E. Popova, S. M. Hess, E. Kee, and D. Richards. Prioritizing project selection. *The Engineering Economist*, 54(4):267–297, 2009.

[61] B. Kordy and W. Widel. How well can i secure my system? In *13th International Conference on integrated Formal Methods (iFM 2017)*, 2017.

[62] B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer. *Foundations of Attack–Defense Trees*, pages 80–95. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[63] A. Kulik, H. Shachinai, and T. Tamir. Approximations for monotone and nonmonotone submodular maximization with knapsack constraints. *Mathematics of Operations Research*, 38(4):729–739, 2013.

[64] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 420–429, 2007.

[65] S. Mauw and M. Oostdijk. *Foundations of attack trees*, pages 186–198. Springer, Berlin, Heidelberg, 2006.

[66] L. A. McLay, C. Rothschild, and S. Guikema. Robust adversarial risk analysis: A level-k approach. *Decision Analysis*, 9(1):41–54, 2012.

[67] Microsoft. Towards a trusted supply chain: a risk based approach to managing software integrity. Technical report, Trustworthy Computing, 2014.

[68] J. T. Moore and J. F. Bard. The mixed integer linear bilevel programming problem. *Operations Research*, 28:911–921, 1990.

[69] D. P. Morton. Stochastic network interdiction. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[70] D. P. Morton, F. Pan, and K. J. Saeger. Models for nuclear smuggling interdiction. *IIE Transactions*, 39:3–14, 2007.

[71] A. Nagurney and S. Shukla. Multifirm models of cybersecurity investment competition vs. cooperation and network vulnerability. *European Journal of Operational Research*, (2):588 – 600, 2017.

[72] A. Nagurney, L. S. Nagurney, and S. Shukla. A supply chain game theory framework for cybersecurity investments under network vulnerability. *Computations, Cryptography, and Network Security*, pages 381–398, 2015.

[73] National Institute of Standards and Technology. Supply chain risk management practices for federal information systems and organizations. Technical report, NIST Special Publication 800-161, Washington, D.C., 2015.

[74] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 14(1):265–294, 1978.

[75] N. Noyan. Risk-averse two-stage stochastic programming with an application to disaster management. *Computers & Operations Research*, 39:541–559, 2012.

[76] Director of National Intelligence. Worldwide threat assessment of the U. S. intelligence community. Unclassified statement for the record, Senate Select Committee on Intelligence, Washington, D.C., 2012.

[77] U. S. Government Accountability Office. National strategy, roles, and responsibilities need to be better defined and more effectively implemented. Technical report, GAO Publication No. 13-187, Washington, D.C., 2013.

[78] U. S. Government Accountability Office. Cybersecurity actions needed to address challenges facing federal systems. Technical report, GAO Publication No. 15-573T, Washington, D.C., 2015.

[79] U.S. Government Accountability Office. National security-related agencies need to better address risks. Gao-12-361, Government Accountability Office, 2012.

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

[81] J. R. O'Hanley, R. L. Church, and J. K. Gilless. Locating and protecting critical reserve sites to minimize expected and worst-case losses. *Biological Conservation*, 134:130–141, 2007.

[82] D. Pisinger. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research*, 83(2):394–410, 1995.

[83] T. K. Ralphs. MibS. `https://github.com/tkralphs/MibS`, 2015.

[84] R. Rockafellar and S. Uryasev. Optimization of conditional value at risk. *Journal of Risk*, 2: 21–42, 2000.

[85] R. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking and Finance*, 26:1443–1471, 2002.

[86] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu. A survey of game theory as applied to network security. In *Proceedings of the 43rd Hawaii International Conference on System Sciences*, pages 1–10, Honolulu, 2010.

[87] G. K. Saharidis and M. G. Ierapetritou. Resolution method for mixed integer bi-level linear problems based on decomposition technique. *Journal of Global Optimization*, 44(1):29–51, 2009.

[88] M. P. Scaparra and R. L. Church. An exact solution approach for the interdiction median problem with fortification. *European Journal of Operational Research*, 189:76–92, 2008.

[89] M. P. Scaparra and R. L. Church. A bilevel mixed-integer program for critical infrastructure protection planning. *Computers and Operations Research*, 35:1905–23, 2008.

[90] B. Schneier. Attack trees: Modeling security threats. *Dr. Dobb's Journal of Software Tools*, 24 (12):21–29, 1999.

[91] D. Shackleford. Combatting cyber risks in the supply chain. Technical report, SANS Institute, 2015.

[92] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing. Automated generation and analysis of attack graphs. In *Security and privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 273–284. IEEE, 2002.

[93] A. Shostack. *Threat Modeling: Designing for Security*. Wiley Publishing, 2014.

[94] R. M. Van Slyke and R. J. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17:638–663, 1969.

[95] C. J. Smith, M. Prince, and J. Geunes. Modern network interdiction problems and algorithms. *Handbook of Combinatorial Optimization*, pages 1949–1987, 2013.

[96] L. V. Snyder, M. P. Scaparra, M. S. Daskin, and R. L. Church. Planning for disruptions in supply chain networks. *INFORMS Tutorials in Operations Research*, 2006.

[97] M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

[98] Y. Tang, J-P. P. Richard, and J. C. Smith. A class of algorithms for mixed-integer bilevel min-max optimization. *Journal of Global Optimization*, 66:225–262, 2016.

[99] United States Senate Hearing. Current and projected national security threats to the United States,. Hearing before the select committee on intelligence of the United States Senate, hearing 112-481, Senate Select Committee on Intelligence, Washington, D.C., 2012.

[100] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of 40th Annual ACM Symposium on Theory Computing*, pages 67–74, New York, 2008.

[101] R. D. Wollmer. Two-stage linear programming under uncertainty with 0-1 integer first stage variables. *Mathematical Programming*, 19:279–288, 1980.

[102] R. K. Wood. Deterministic network interdiction. *Mathematical and Computer Modeling*, 17 (2):1–18, 1993.

[103] P. Xu and L. Wang. An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions. *Computers & Operations Research*, 41:309–318, 2014.

[104] K. Zheng, L. Albert McLay, and J. Luedtke. A budgeted maximum multiple coverage model for cybersecurity planning and management. *Under review*, 2016.

[105] Y. Zhuo and S. Solak. Measuring and optimizing cybersecurity investments: A quantitative portfolio approach. In *Proceedings of the 2014 Industrial and Systems Engineering Research Conference*, Montreal, Canada, 2014.

[106] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley. RRE: A game-theoretic intrusion response and recovery engine. *IEEE Transactions on Parallel and Distributed Systems*, 25(2):395–406, 2014.