A STOCHASTIC NETWORK-INTERDICTION MODEL FOR CYBER SECURITY

by

Mehmet Ertem

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

2014

Date of final oral examination: 01/03/2014

The dissertation is approved by the following members of the Final Oral Committee:

Vicki M. Bier, Professor, Industrial and Systems Engineering
Jeffrey T. Linderoth, Professor, Industrial and Systems Engineering
James R. Luedtke, Assistant Professor, Industrial and Systems Engineering
Oguzhan Alagoz, Associate Professor, Industrial and Systems Engineering
Somesh Jha, Professor, Computer Science

To my wife *Tuğba*, and my daughter, *Sena Ahsen*

ACKNOWLEDGMENTS

PhD is a long and challenging journey. Throughout my PhD studies, I have learned not only how to do research but also how to make better decisions. When I look back at my life, one part I will always be proud to have is definitely my PhD years.

First and foremost, I would like to thank my advisor, Vicki Bier, who guided me in this journey. I acknowledge that without her endless help and great support, this dissertation would not be possible. I will always remember her tolerance and positive approach to my mistakes which helped me to understand the nature of making science and not to give up when things go wrong. Also, I am thankful to her for giving me the opportunity to closely observe a wonderful scholar for many years. I also owe gratitude to my collaborators, Professors Somesh Jha and Xinming Ou from Computer Science department, for their support and time to help me to understand the nature of cyber security. Also, many thanks goes to Professors Jeff Linderoth and James Luedtke, for their great help and guidance in the modeling part of my research. I am also indebted to Professor Oguzhan Alagoz for his valuable suggestions and insights.

I believe that for a success in academic life, one needs a peaceful family life. Thankfully, I found more than that with my wonderful wife, Tuğba and lovely daughter, Sena Ahsen. I am thankful to my wife for her love, emotional support and understanding in this challenging journey. Her endless support and love will be always remembered. Sena Ahsen joined our family during the last two very busy years of my PhD studies; and in addition to making our lives happier, she inspired me to better understand the meaning of life and the process of continued learning. I will never forget her happiness and the positive energy she gave off to me when I came back to home after a long day. Also, I would like to thank my parents, Zeliha and Hasan, and my parents-in-law, Leyla and Bayram, who have always trusted me and supported my decisions.

I thank my fellow students at the University of Wisconsin-Madison. Among them, special thanks go to (in alphabetical order): Turgay Ayer, Mehmet Ayvaci, Safa Erenay, Wen-Chieh Hu, Taher Jamshidi, Mustafa Rasim Kilinc, Fuat Kosanoglu, Sinan Tas, Chen Wang and Jonathan Welburn for their motivational support. I have unforgettable memories with each of them. I also would like to extend my gratitude to my other fellow students, including (in alphabetical order): Oguz Akkas, Majid Aksari, Onur Asan, Merve Bodur, Gizem Cavuslar, Mucahit Cevik, Nan Chen, James Codella, Mehmet Ali Ergun, Mahdi Hamzeei, Mustafa Ozkaynak, Lisa Tang, Sait Tunc, Yuan Yuan, Li Zeng, and Qiang Zhou. Lastly, I would like to thank numerous wonderful friends that I have had in Madison. This section would not be sufficient to list all of their names, but they will always be remembered.

DISCARD THIS PAGE

TABLE OF CONTENTS

Page

List of Tables

List of Figures

ABSTRACT

We propose a general defender-attacker model for security of computer networks, using attack graphs to represent the possible attacker strategies and defender options. The defender's objective is to maximize the security of the network under a limited budget. In the literature, most network-interdiction models allow the attacker only one attempt; other models allow multiple attempts, but assume that any subsequent attempt begins at the point where the previous attempt failed. By contrast, in computer security, the attacker could be operating from the safety of a foreign country, and the cost of changing attack strategies may be quite low, so a new model is needed.

To capture the ability of the attacker to launch multiple attempts, we represent the attacker's success on each arc of the attack graph probabilistically, and formulate the resulting problem as a multiple-stage stochastic network-interdiction problem. In the resulting game, the defender anticipates both the attacker's strategy choices, and their probabilities of success, and chooses which arcs in the attack graph to protect in order to defend against multiple attempted attacks. The attacker then launches an optimal attack against the system, knowing which arcs have been protected. If the attacker fails at the first attempt, a second-stage optimal strategy is chosen, based on a revised attack graph showing which arcs have been successfully traversed (with success probabilities of 1), and which arc failed (assumed to have success probability 0). We solve the resulting problem using multi-stage stochastic optimization with recourse, and explore the attacker's strategies.

Chapter 1

Introduction

An attack graph is essentially "a succinct and complete representation of all possible scenarios for attacking a computer network" [63]. Using attack graphs, it is possible to develop an defender-attacker model that will help defenders to identify optimal or at least near-optimal defenses. Moreover, one can develop models to defend a system against different types of attackers. With regard to computer security, one can imagine attacker types such as hackers, spies, terrorists, corporate raiders, professional criminals, etc., who may try to optimize different objectives. For example, an opportunistic attacker is merely looking for an easy target, and may be deflected to an easier target if one target (e.g., a specific internet retailer) becomes too difficult or costly to attack successfully. By contrast, realistic levels of attack difficulty or cost are unlikely to deter a determined attacker (e.g., a hostile government) from its preferred target.

Many defender-attacker models have been developed for security of computer networks. These models mostly use game theory to identify equilibrium strategies for both the attacker and the defender. However, most models do not use attack graphs in developing their game trees. For example, Liu et al. [38] develop a conceptual game-theoretic formalization, with different game-theoretic models for different situations (e.g., correlation among attack actions, accuracy of intrusion detection). Lye and Wing [40] discuss different game strategies in network security. Viewing the interactions between an attacker and the system administrator as a two-player stochastic game, they compute Nash equilibria or best-response strategies for the two players (attacker and administrator) using nonlinear programming. Modeling the problem as a general-sum stochastic game makes it possible for Lye and Wing to identify multiple Nash equilibria, which can help the system administrator anticipate the attacker's best attack strategies.

There are also many defender-based methodologies that do not explicitly consider attacker types or strategies. For example, Sheyner et al. [63] develop an approximation to find the smallest subset of measures whose deployment would make the system "safe." They claim that even just finding a set of measures to make the system safe is NP-complete, so they use a greedy algorithm to find an approximate solution to the problem. In their algorithm, they treat finding a minimum critical set in an attack graph analogously to a minimum hitting-set problem. They then apply a greedy critical-set algorithm to find the minimum critical set that covers at least one arc from each of the possible paths on the attack graph. Their model would be appropriate for the cases in which there is no defender budget limit. However, in reality, most of the security systems are subject to defense budget limit.

There are several key questions to be answered in developing a game-theoretic model of computer security using attack trees. For instance, is it realistic to assume a non-myopic attacker, who considers his future actions in choosing an optimal multi-stage attack strategy, or is it sufficient to protect only against a myopic attacker, who may launch an optimal single-stage attack without consideration of future attack options? If planning (conservatively) to protect against a non-myopic attacker, how many steps ahead can/should the defender look in choosing his defense strategy? What is the tradeoff between near-optimality of results (by looking ahead multiple steps) and computational feasibility (by defending against a more myopic attacker)? In our work, we plan to solve the defender's problem optimally for protecting against a non-myopic attacker who can launch two or three separate attack attempts, and compare the optimal defender solution to the solution of a defender who is protecting against only a myopic attacker, to see how much penalty is incurred by assuming a myopic attacker strategy. If the payoffs of the two defender solutions are close to each other, this would suggest that the solution to the non-myopic optimization problem can be approximated by the solution to a corresponding myopic optimization problem. Otherwise, solution strategies capable of solving for optimal defenses against multi-stage attacks will be needed.

To illustrate, in Figure 1.1, assume that the solid path is the optimal strategy for a specific attacker type, and that if we block this path, the attacker will choose his second optimal path. Does it make any difference which arc on that path we block? Assume for purposes of argument that

we block the third arc on this path, the one denoted by the "X," and that the attacker would then choose the dashed arcs for his next optimal path. In this case, it might have been better to block the second arc instead of the third arc, because that would automatically eliminate the attacker's first two optimal choices on the attack graph. However, whether that is in fact optimal may depend on numerous other system characteristics, like how costly it is for the defender to block one of the dashed arcs, how much less desirable the attacker's third choice of attack path is (e.g., the bold path in Figure 1.1), etc. A fully optimal defense strategy may be computationally challenging to find, so the defender may need to decide how conservative to be in assessing attacker look-ahead capabilities in trading off optimality or near-optimality of results against computational feasibility.

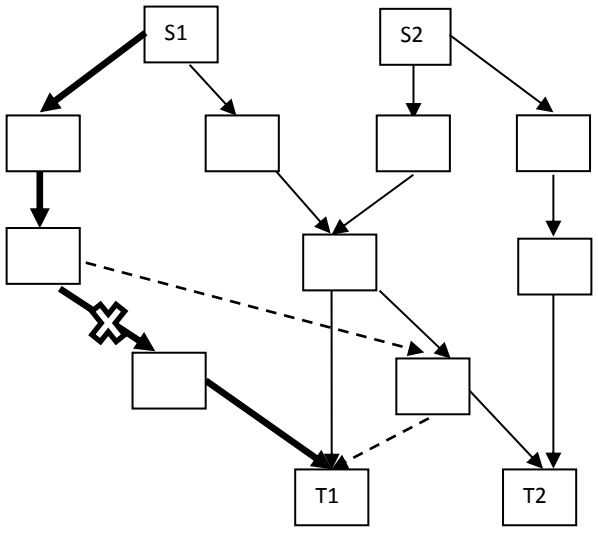

Figure 1.1: A simple attack graph

The remainder of this dissertation is organized as follows. In Chapter 2, we review the relevant literature on attack-graph based cyber security models, game-theoretic approaches to cyber security, and stochastic network-interdiction models. In Chapter 3, we formulate two-stage and three-stage stochastic network-interdiction models for the attacker problem, and a bi-level optimization model for the defender problem, and propose solution approaches for both the attacker and defender models. In Chapter 4, we show the results for the attacker problem. In Chapter 5,

we show the results for the defender problem. Finally, in Chapter 6, we summarize some of the important research findings, and discuss some possible future work and extensions.

Chapter 2

Literature Review

Computers have been in our lives for more than fifty years now. Especially with greater reliance on the internet, computer security has become a major issue. Virtually all important cyber networks have been attacked by hackers, spies, terrorists, corporate raiders, professional criminals, etc. [59]. The US-CERT (United States Computer Emergency Readiness Team) web site has announced more than 52,000 vulnerabilities, a number that increases by 500 to 600 per month. The worldwide cost of cyber attacks is around one trillion dollars [66], demonstrating the importance of research on cyber security.

Computer security has been studied by many researchers. The main tools for protecting networked systems include cryptography, "demilitarized zones" (designated places on a computer network where public services are located), virtual private networks (VPN), honey pots, vulnerability scanners, firewalls, and intrusion detection systems (IDSs) [59]. Encryption is a strong tool for securing data packets, but it is not always realistic to encrypt every packet in a network, and in any case encryption keys can often be stolen or acquired through "social engineering" even if the encryption code cannot be "cracked" by brute computational force. A VPN is a way to connect securely with the network a private network from outside; however, this connection can also provide a way for attackers to get inside the network, if they are able to obtain the privileges of an authorized user. Honey pots are decoys designed to attract the interest of attackers; however, if an attacker is able to identify the honey pots, this could provide knowledge of the network structure. Vulnerability scanners are used to scan a given network for potential weaknesses, but their findings are not always reliable. Finally, firewalls are designed to protect networks from malicious activities, and IDSs identify attack signatures and detect anomalous system behavior. However, if cyber

attackers know their general capabilities, they may be able to take advantage of security holes or gaps. Attackers can also exploit the complexity of computer networks by finding a multiplicity of ways to attack them.

When computer networks were first introduced, they were not designed to protect against potential attacks [59]. Moreover, with the exponential growth of computer and internet usage, computer networks became extremely complex. Therefore, it is difficult if not impossible to secure computer networks completely, and intelligent attackers can always take advantage of new weaknesses or combinations of existing vulnerabilities.

Moreover, operational decisions about updating network security often involve patching newly discovered security holes after an attack. In other words, once a new virus or cyber attack has been detected, firewalls, intrusion-detection algorithms, and other defense tools are updated accordingly. However, game theory could in principle facilitate a more proactive approach to computer security, by anticipating the responses of potential attackers.

Another important challenge is that there are usually numerous alternative strategies available for attacking even a small network. Schneier [59] gives an example of this multiplicity of attack strategies for the case where the goal of the attacker is to read a specific message being sent from one computer to another. Some of these attack strategies include convincing the sender and/or the recipient to reveal the message, reading the message when it is being written into the sender's computer or stored on the sender's and/or the recipient's disk, stealing the sender's or the recipient's computer and trying to recover the message, etc. Of course, for a large network, with thousands of servers and end users, security would be an even more complex problem.

Considering the complexity of cyber networks, attack graphs have been developed as a way to represent all possible attack alternatives for a given network [63]. An attack graph can show the vulnerabilities of a given cyber network in a compact way. As a result, many researchers have used attack graphs to build security models for cyber networks [35].

So far, there have been only a few studies (e.g., [68], [8]) that use both attack graphs and game theory to model the security of computer networks. However, many studies of computer networks have been conducted using either game-theoretic tools or attack graphs alone. We summarize

these studies in sections 2.1 and 2.2, respectively. In section 2.3, we discuss possible solution algorithms from the literature for the network-interdiction game that results from applying game theory to attack graphs.

2.1 Attack-Graph Based Cyber Security Models

Attack graphs are important tools for analyzing network vulnerabilities. Several studies have been conducted on how to efficiently generate attack graphs for cyber networks. An attack graph generally includes a pair of source and terminal nodes, with several arcs and intermediate nodes. On a given attack graph, each node represents one possible state of the attack process. Each path through an attack graph represents an alternative way to reach a specific goal (e.g., to compromise the database of a bank, etc.), and each arc represents a specific subattack on a given path. A successful attack must start from the source node and reach the end node before failing or being detected.

Schneier [58] discusses attack trees, which are a primitive version of attack graphs. He represents all vulnerabilities of a system in a tree, with and/or logic to represent relations between nodes. Here, the root node represents the main goal, and leaf nodes represent different ways to reach the main goal. Schneier assigns a value to each arc to represent the cost of the corresponding subattack. He gives an example involving e-mail security to illustrate how attack trees could be used to help choose defensive strategies.

In their review paper, Lippmann and Ingols [35] summarize different approaches to generating attack graphs. Sheyner et al. [63] introduce automatic attack-graph generation to facilitate the construction of complete attack graphs even for extremely large networks. They propose that once a complete attack graph has been generated, it could be used to identify all possible attacks and defend against them. In particular, Sheyner et al. use optimization to find a minimal set of security measures that can protect against all possible attacks. Noel and Jajodia [47] develop a similar model that finds the smallest number of sensors to completely cover an attack graph. Ou et al. [49] propose a more computationally efficient approach for generating and representing attack graphs. In their framework, nodes in the attack graph represent logical statements that encode the network

states, and arcs represent causality relations between the network configuration and the attacker's potential gains. They propose that logical attack graphs could be built more quickly, and represent the vulnerabilities of networks more accurately, than physical attack graphs. Saha [55] extends the work of Ou et al. [49] to include more sophisticated security policies in advanced operating systems, such as Windows XP and SELinux. Finally, [65], [13], and [26] develop approaches for building more compact attack graphs.

Sawilla and Ou [57] propose a new algorithm to identify critical assets in attack graphs. They generalize Google's PageRank algorithm to rank attack assets (AssetRank) using attack graphs. They apply their algorithm to several networks, and find that the resulting ranks are reasonably consistent with the subjectively estimated importance of particular assets to prospective attackers. Mehta et al. [42] also develop an algorithm to rank states in an attack graph, to identify the most critical portions of the graph.

Frigault and Wang [15] consider attack graphs as a special type of Bayesian network, and propose that exploiting one vulnerability may make the next vulnerability on the same attack path more susceptible to attack. Therefore, they analyze attack graphs using conditional probabilities. They also model the evolving nature of vulnerabilities using dynamic Bayesian networks, and apply their model to two case studies. However, it might not always be feasible to assess the conditional probabilities of all vulnerabilities in a large network.

Wang et al. [70] develop a method to harden a network against multiple-step intrusions using attack graphs. In their solution approach, they aim to disable the initial conditions of cyber attacks, rather then removing exploits.

Xie et al. [73] propose an approach for representing attack graphs in a simpler way. They construct a two-layer attack graph with an upper layer that shows the attacker's access privileges. The lower layer then corresponds to a traditional attack graph with one source and one target. This approach allows them to prioritize potentially vulnerable hosts, as a stepping stone to developing an efficient network-hardening strategy.

2.2 Game-Theoretic Approaches to Cyber Security

Most traditional network-security solutions lack a quantitative decision-making framework. Ideally, such a framework should consider the behavior of attackers. To this end, game theory is a good tool to use for modeling the security of cyber networks. Several researchers have emphasized the value of using game theory for the analysis of security risks. For example, Bier et al. [7] argue that traditional risk-analysis approaches can fail to correctly assess the risks from intelligent attackers. Therefore, considering the importance of game-theoretic models for cyber security, we here review several recent papers that propose game-theoretic models for network-security problems.

We can categorize games into two groups (cooperative games and non-cooperative games), based on the players' relations. Network-security games are non-cooperative games, since the attacker and the defender have opposing objectives [48].

Further, we can categorize games as static or dynamic games [48]. Static games are one-stage games, while dynamic games are multi-stage games in which players can update their actions in each stage. One special case of dynamic games is stochastic games. In a stochastic game, the players take actions and receive payoffs that depend not only on their actions and the current state of the game, but also on a set of transition probabilities. The main difference between stochastic games and other types of dynamic games is that stochastic games involve probabilistic transitions between states of the game.

Games can also be sequential or simultaneous [48]. Simultaneous games are games in which all players take their actions at the same time (or, if moving at different times, the later players do not know the actions of the earlier players). In sequential games, later players have some knowledge about the earlier player's actions. Based on this definition, sequential games are a subset of dynamic games. However, there can also be dynamic games in which, at some periods, the players move simultaneously, or other combinations of simultaneous and sequential games.

Games can be categorized as having perfect or imperfect information, and complete or incomplete information, according to what each player knows about the other player(s) [48]. In a

perfect-information game, every player knows the past moves of all other players. If the past moves of some players are not known by at least some other players, then we have a game of imperfect information. Simultaneous games can thus be considered games of imperfect information. Similarly, in a complete-information game, every player knows the possible strategies and payoffs of all other players (but not necessarily their actions). By contrast, in a game of incomplete information, at least one player does not know the possible strategies and payoffs of at least one of the other players.

It is also possible to categorize games as constant-sum or general-sum games [48]. In a constant-sum game, the total available resources are independent of player choices, so the resources of all players always sum to a constant. One special case of constant-sum games is zero-sum games, in which the total payoffs of all players always equal zero for any combination of strategies. General-sum games are games in which the resources of all players need not sum to a constant.

Jormakka and Mölsä [29] apply game theory to four static games involving information warfare. For each game, Jormakka and Mölsä define possible strategies for both the attacker and the defender, and find the resulting equilibrium strategy or strategies for each case. Carin et al. [9] develop an economic model to find efficient investment strategies against reverse-engineering attacks. Liu et al. [38] develop a conceptual game-theoretic formalization to infer attacker intent, objectives, and strategies (AIOS). In their study, they focus on inferring likely attack strategies, given a specific model of attack intent and objectives. In their case study, they infer the strategies of attackers that impose distributed denial-of-service attacks. Liu et al. [39] also study the problem of IDS location in mobile ad-hoc networks. In particular, they model the interactions between attacking and defending nodes as a two-player, non-cooperative, non-zero-sum game. They analyze the problem as both a static and a dynamic game, and consider both complete and incomplete information regarding the maliciousness of nodes.

Lye and Wing [40] develop a game-theoretic model for the security of a computer network. In their study, they focus on three attack scenarios: defacing of a website; denial of service; and

stealing of confidential data. Viewing the interactions between an attacker and the system administrator as a two-player stochastic game, they find Nash equilibria or best-response strategies for the two players (attacker and administrator) using nonlinear programming. Modeling the problem as a general-sum stochastic game makes it possible for Lye and Wing to identify cases with multiple Nash equilibria.

Xiaolin et al. [72] propose a Markov model to assess the risks of computer networks, considering the current and anticipated future security status. They use a Markov chain to model the potential actions of attackers to assess the risk, and another Markov chain to represent possible defense strategies to decrease the number of system vulnerabilities. Nguyen et al. [45] propose a stochastic game-theoretic model for security and intrusion detection in computer networks, and apply it to a small sample network. They model the security of the network using a weighted directed graph, with nodes representing combinations of security assets and vulnerabilities, and with edges representing relations among the nodes. References [2], [3], and [4] also propose similar stochastic-game models for IDSs.

2.3 Stochastic Network-Interdiction Models

One way to use attack graphs to develop defensive strategies is to find the optimal set of arcs (or, equivalently, attacker actions) for the defender to interdict. This critical set is likely to depend on factors such as interdiction costs, the defender's budget constraint, the effect of an arc interdiction on the attacker's future actions, etc. In the literature, many network-interdiction algorithms have been developed to find the optimal set of arcs to interdict on a given graph. In our work, we develop a network-interdiction model using attack graphs to find optimal game-theoretic defense strategies for cyber security. Therefore, we now review the literature on network-interdiction modeling.

Network-interdiction models have been studied in areas such as military applications and transportation security. During the Vietnam War, McMasters and Mustin [41] developed deterministic models to interdict enemy troops or material flows on a transportation network, using optimization to allocate a limited number of aircraft to best interdict the enemy's supply line. More recently,

[64], [53], and [71] have all applied deterministic network-interdiction models to military problems, or to interdiction of illegal drugs or precursor chemicals. Similarly, [16] and [19] study the problem of maximizing the shortest path of an attacker, using a deterministic network-interdiction algorithm.

When one or more aspects of the problem are not known with certainty, we have a stochastic network-interdiction problem. One of the first studies of stochastic network interdiction was by Cormican et al. [11], who proposed a stochastic network-interdiction model to minimize the expected maximum flow achievable by an adversary on a given network. In their formulation, Cormican et al. [11] represent interdiction success or failure by binary random variables, and analyze the resulting problem as a two-stage stochastic integer program. In the first stage of their problem, the defender minimizes the maximum network flow achievable by the adversary, by interdicting a set of arcs; in the second stage, the attacker then chooses the path on the network that provides the maximum flow. Cormican et al. [11] use a sequential-approximation algorithm to solve the problem, finding lower and upper bounds for the optimal value of the objective function at each iteration.

Israeli and Wood [25] study a shortest-path network-interdiction problem, where the defender interdicts a set of arcs to maximize the shortest path achievable by the attacker. Their problem is stochastic, because the defender is assumed not to know the attacker's actual origin and destination, but rather to have only a probability distribution over possible origin-destination pairs. They formulate the resulting bilevel max-min problem as a stochastic mixed-integer program. Bayrak and Bailey [5] extend the shortest-path network-interdiction problem to the case where the interdictor and the evader have different levels of knowledge about the network. They formulate their problem as a stochastic nonlinear mixed-integer program, and then solve by converting it to a stochastic linear mixed-integer program.

Pan and Morton [51] propose a stochastic network-interdiction model in which the interdictor chooses a set of arcs on which to install radiation sensors, to minimize the reliability of the evader's maximum-reliability path (e.g., maximize the probability that a nuclear smuggler would

be detected). Again, Pan and Morton [51] assume that the interdictor has only a probability distribution for the evader's origin-destination pair. They formulate the resulting problem as a stochastic mixed-integer program, and propose a decomposition method for its solution. Similarly, Dimitrov and Morton [14] develop several network-interdiction models, and apply these models to variety of problems.

Recently, Morton [44] has compared deterministic and stochastic network-interdiction models and their solution algorithms. He also summarizes the types of stochastic network-interdiction models according the defender's information level about the attacker's source-terminal pair.

Stochastic network-interdiction models are more difficult to solve than similar deterministic models. As a result, many solution algorithms and heuristics have been proposed. Cormican [10] categorized the solution methods available as of 1995 as either decomposition methods or sequential-approximation algorithms. In general, decomposition algorithms iteratively improve the bounds of the objective function by solving the attacker's and the defender's problems for every possible realization of the problem (scenario). Examples of decomposition algorithms include [11], [25], [51], and [22]. However, decomposition algorithms require the objective function of the problem to be convex. Moreover, some decomposition algorithms require taking the dual of the objective function, something that is not always easy, and may not always even be possible. Also, when the number of possible scenarios is large, decomposition methods may become computationally difficult because the number of subproblems to be solved at each iteration is equal to the number of scenarios.

Sequential-approximation algorithms are computationally more efficient when the number of possible scenarios is large. Like decomposition, sequential-approximation algorithms find upper and lower bounds on the optimal value of the objective function. In sequential approximation, the true problem is approximated by sequentially refining the partitioning of the sample space to create more representative subsets, which provide tighter bounds on the optimal value of the objective function. At each iteration, sequential-approximation algorithms use decomposition to

find bounds on the optimal value of the objective function in each partition. Therefore, sequential-approximation algorithms face the same limitations as decomposition. Examples of sequential-approximation algorithms include [11], and [30].

Recently, Janjarassuk and Linderoth [27] proposed a third approach, sample-average approximation (SAA), to solve stochastic network-interdiction problems. In SAA, the true problem is solved approximately by generating samples from the state space (the set of all possible scenarios) of the problem. The SAA problem turns out to be a deterministic optimization problem once the samples have been generated, which approximates the true stochastic problem. Deterministic optimization problems can often be solved to optimality using suitable algorithms like the L-shaped decomposition algorithm ([67]). Examples of SAA methods include [60], [69], [56], [1], [62], [34].

In their paper, Janjarassuk and Linderoth [27] compare the computational performance of different methods for solving stochastic network-interdiction problems. They conclude that SAA appears to be especially effective for large problems, since a relatively small number of samples may be enough to represent a large number of possible scenarios. Therefore, we propose to use SAA for the solution of our multi-stage stochastic network-interdiction problem.

Another method to solve stochastic network-interdiction problems is dynamic programming. However, few papers in the literature have used this approach ([12], [23], [21]). Gutin [21] proposed solving interdiction games using Markov decision processes (MDP). In their problem, a project is planned by a player who schedules the tasks' start times and allocates available resources to minimize the project's completion time, while another player chooses a set of tasks to delay in order to maximize the total completion time of the project. Gutin modeled the resulting problem as a stochastic network-interdiction model and solved it using a specialized dynamic programming algorithm proposed by Creemers et al. [12], since standard MDP algorithms seemed to be computationally infeasible for even medium-sized networks. The proposed algorithm by Creemers et al. [12] solves the problem efficiently in two steps based on a partitioning of the state space. The first step consists of generating "uniformly directed cuts," which represent the sets of activities

that can be performed in parallel. In the second step, a backward stochastic dynamic programming recursion is applied to determine the optimal decision. Hindelang and Muth [23] developed a dynamic programming algorithm for the decision critical path method (DCPM). Hindelang and Muth used the standard backward induction algorithm to evaluate DCPM networksin which the nodes and arcs represent projects and their precedence relationships. Their algorithm improved the solution time of the DCPM network problem compared to previous approaches using integer programming. Also, the structure of their algorithm allows them to find the optimal solution for any desired number of project due dates with little extra computational effort.

In theory, we might be able to use MDP to solve our problem. However, given that stochastic programming is widely used to solve stochastic network-interdiction problems, and the SAA approach seems to be quite efficient, we propose to use stochastic programming instead of MDP in this study.

Chapter 3

Problem Statement

Here, we adopt a fully game-theoretic and non-myopic approach to represent attacker and defender behavior. In particular, we set up the game as a defender-attacker model, because the defender moves first in our problem. Potential objective functions for the attacker could be minimizing the probability of detection, minimizing the expected cost of all arcs attacked before an eventual successful attack (if the attacker is not budget-constraint), maximizing the success probability (if the attacker has a limited number of attack opportunities), etc. We assume that the attacker's objective is maximizing the probability of success, with limited resources; however, we could presumably also solve the problem for other attacker objectives, with slight differences in the problem formulation. We also assume that the defender wants to minimize the attacker's probability of success, with limited resources.

In our model, we use attack graphs to represent attack success probabilities. Each arc on an attack graph represents the probability of successfully traversing one step of a complete attack (as represented by a path). We assume that the outcomes of attempting to traverse different arcs are independent. However, in reality, some arcs on the attack graph may represent similar vulnerabilities, and thus may not be independent; for example, traversing one of these arcs may indicate that the attacker capabilities are sufficient to give a high chance of success in traversing other arcs. Zhang et al. [75] propose a modeling approach in which multiple dependent arcs or nodes on a single path could be grouped together into one arc (or node). Using their approach or similar approaches, it is possible to revise the attack graph to make sequential arcs on the resulting revised attack graph independent. Thus, even though our optimization model requires the arcs to be independent, it can still be used to address problems with some degree of dependence between arcs.

Our game is non-cooperative, because the defender wishes to minimize damage while the attacker attempts to maximize damage. Also, it is a stochastic game, because it is uncertain whether the attacker will be successful when attempting to traverse the network, and the defender develops his strategy anticipating this uncertainty. We can further categorize our game as a sequential game, because the defender moves first to protect the network, and than the attacker launches an attack to traverse the protected network. Further, we assume a complete-information game between the attacker and the defender, which means that both the defender and the attacker know each other's possible strategies and payoffs.

Finally, we assume a game of perfect information, where the attacker has knowledge of the defender's chosen protection strategy when choosing how to attack the network. This might be an unrealistic assumption, but it is conservative, since it gives the attacker an advantage to assume that he knows the defender's chosen strategy and therefore will not waste resources on attacks on protected arcs. This assumption can be relaxed to explore the differences for a game of imperfect information, but that relaxation is considered beyond the scope of this thesis.

In the literature, several network-interdiction algorithms have been developed, both deterministic and stochastic. For example, Israeli and Wood [25] develop a deterministic shortest-path network interdiction model and solve the problem with the help of a decomposition algorithm. Their problem is deterministic because the network topology and attack success are assumed to be fully known by both the defender and the attacker. In addition, the defender moves first; the attacker takes action only after the defender's action has been revealed to him. However, a deterministic model may not be adequate for computer security. Thus, we propose a stochastic approach to take into account the attacker's uncertainty about whether a given attack strategy will succeed.

Cormican et al. [11] develop a stochastic network-interdiction problem to minimize the expected maximum flow through a network. Their problem involves uncertainty arising from both uncertain interdiction successes and unknown arc capacities. They set up a two-stage stochastic network-interdiction problem. The first-stage decision variables define which arcs the interdictor (defender) attempts to traverse. After the uncertainty is revealed (i.e., when it becomes known whether the interdiction was successful), the adversary (attacker) finds the optimal path through

the remaining network. The interdictor (defender) can make either one attempt or several attempts to traverse the network - but if several attempts are made, they are modeled as being essentially simultaneous, so that all attempts are undertaken before the results of any attempt is revealed.

Pan and Morton [51] formulate a stochastic maximum-reliability path problem to detect an evader (attacker). In their problem, the interdictor (defender) installs sensors on a subset of the network arcs in order to detect the evader. The evader chooses a source/terminal-node pair on the network, from the available pairs. The interdictor knows only the probabilities of the evader choosing the various pairs, not which pair is chosen. The interdictor minimizes the expected maximum reliability of the evader, with limited resources. Morton and Pan model their problem as a stochastic mixed-integer program, and solve it using an enhanced decomposition method. Morton and Pan apply their model to a physical network, assuming that the attacker does not switch to another path in the case of detection or failure. However, in our problem, the attacker may switch to another path if he fails on a particular arc of the optimal path.

We here formulate the attacker's problem as a multi-stage stochastic network-interdiction problem (MSNIP). When attacking the system, the attacker's success is uncertain. Thus, the attacker may need to make several attempts. The defender should anticipate this in choosing a defensive strategy. Thus, the problem becomes a bi-level stochastic programming problem when we consider both the defender and attacker objectives. In our model, each of the attacker's attempts represents a stage of the MSNIP. For example, the attacker's first attempt represents the first stage of the MSNIP, the second attempt represents the second stage if the attacker fails at the first attempt, and the third attempt represents the third stage if the attacker fails at the second attempt, etc. For the purposes of this thesis, we formulated the attacker's problem for only the two-stage and three-stage cases. Formulating the problem for more than three stages is possible, but would be demanding.

3.1 Notation and Assumptions

We use the following notation in our model:

- $N_1(V, A_1)$ is a directed acyclic graph with source node $\psi \in V$ and sink node $\tau \in V$. Here, V represents the set of nodes or vertices, and A_1 represents the set of arcs on the attack graph.

- $\xi \in \{0,1\}^{|A_1|}$ is a random matrix where $\xi_{ij} = 1$ if the attacker can successfully traverse arc $(i,j) \in A_1$, and 0 otherwise. Here, we assume that the ξ_{ij} are independent random variables.
- $S = \{\xi_1, \xi_2, ...\xi_N\}$ is the set of all possible realizations of random matrix ξ.
- $A_d = \{(\psi, j) | j \in V\}$ represents the set of dummy arcs from the source node to all other nodes of the attack graph.
- $A_2 = A_1 \bigcup A_d$.
- p_{ij} = probability that the attacker can successfully traverse arc (i,j) if the defender does not interdict arc $(i,j) \in A_1$ ($p_{\psi j} = 1,\ \forall (\psi, j) \in A_d$).
- q_{ij} = probability that the attacker can successfully traverse arc (i,j) if the defender interdicts arc $(i,j) \in A_1$.
- $A(x,y,\xi)$ = $A_1 \setminus \{(i,j) \in A_1 \mid$ the attacker failed to traverse arc (i,j) given x, y, and $\xi\} \cup \{(\psi, j) \in A_d \mid$ node j was reachable given x, y, and $\xi\}$. In other words, $A(x,y,\xi)$ represents the union of A_1 and the set of dummy arcs from the source node to all nodes that were reached by the attacker in previous stages, for a given realization of the matrix ξ, excluding the arcs at which the attacker failed in previous stages.
- $G(V, A(x,y,\xi))$ is the corresponding (revised) attack graph based on defender action x, attacker action y and random matrix ξ.
- d_{ij} = cost of attacking arc $(i,j) \in A_2$, where $d_{\psi j} = 0\ \forall (\psi, j) \in A_d$.
- c_{ij} = cost of interdicting arc $(i,j) \in A_1$.
- L = total available budget of the attacker.
- B = total available budget of the defender.
- b_i = "supply" for node i (where the source node has a supply of 1, the terminal node has a supply of -1, and all intermediate nodes have a supply of 0).

- $x_{ij} \in \{0,1\}$ = decision variable for the defender (1 if the defender interdicts arc $(i,j) \in A_1$, and 0 otherwise).
- $y_{ij} \in \{0,1\}$ = first-stage decision variable for the attacker (1 if the attacker chooses a path that includes arc $(i,j) \in A_1$ in the first stage, and 0 otherwise).
- $w_{js} \in \{0,1\}$ = indicator variable for the attacker (1 if the attacker successfully reached node $j \in V$ in the first stage under scenario $s \in S$, and 0 otherwise).
- $v_{ijs} = w_{is}y_{ij} \in \{0,1\}$ = indicator variable for the attacker (1 if the attacker successfully reached node $i \in V$ in the first stage under scenario $s \in S$, and chose a path that includes arc (i,j), and 0 otherwise).
- $z_{ijs} \in \{0,1\}$ = second-stage decision variable for the attacker (1 if the attacker chooses a path that includes arc $(i,j) \in A_2$ in the second stage after observing scenario $s \in S$ in the first stage, and 0 otherwise).
- $o_{js} \in \{0,1\}$ = indicator variable for the attacker (1 if the attacker successfully reached node $j \in V$ under scenario $s \in S$ in stage two, and 0 otherwise).
- $u_{ijs} = o_{is}z_{ijs} \in \{0,1\}$ = indicator variable for the attacker (1 if the attacker successfully reached node $i \in V$ in the second stage under scenario $s \in S$, and chose a second-stage path that includes arc (i,j), and 0 otherwise).
- $t_{ijs} \in \{0,1\}$ = third-stage decision variable for the attacker (1 if the attacker chooses arc $(i,j) \in A_2$ in stage three after observing scenario $s \in S$ in stage two, and 0 otherwise).

3.2 Defender-Attacker Model

3.2.1 Two-stage Case

In the two-stage model, the defender first interdicts (protects) a set of arcs in the attack graph representing the available strategies for attacking the system; the attacker then launches an optimal attack against the system, based on knowledge of which arcs have been protected. If the attacker

fails at the first attempt, then he will choose a second-stage optimal attack strategy, based on the revised attack graph showing which arcs have already been successfully traversed, and which arc was impossible to traverse (leading to failure of the first-stage attack).

We have a defender-attacker problem, where the attacker maximizes a function $F(x, y, \xi)$ that describes the "penetrability" of the network if the defender chooses to interdict $x \in X$ and the attacker chooses initial action $y \in Y$, conditional on a given realization of random matrix ξ.

Let $F(x, y, \xi)$ be the probability that the attacker can successfully traverse the graph $G(V, A(x, y, \xi))$ without detection, as given by $F(x, y, \xi) = \{\prod_{(i,j)\in A(x,y,\xi)} p_{ij}^{z_{ijs}}\}$. The defender's optimization model can then be formulated as follows:

$$\min_{x\in X}\{\max_{y\in Y} \mathbb{E}[F(x, y, \xi)]\} \tag{3.1}$$

subject to

$$\sum_{(i,j)\in A_1} c_{ij}x_{ij} \leq B \tag{3.2}$$

where 3.2 represents the defender's resource constraint. Here, $\max_{y\in Y} \mathbb{E}[F(x, y, \xi)]$ is the attacker's objective, which is to maximize the expected probability of successfully traversing the attack graph in two stages (attempts). We can model the attacker's stochastic program as a deterministic optimization problem by writing the expected value $\mathbb{E}[F(x, y, \xi)]$ as the weighted sum: $\mathbb{E}[F(x, y, \xi)] = \sum_{s\in S} p_s F(x, y, \xi_s) = \sum_{s\in S} p_s\{\prod_{(i,j)\in A(x,y,\xi_s)} p_{ij}^{z_{ijs}}\}$

Unfortunately, this objective function is non-linear. In order to efficiently solve the attacker's optimization problem, we need to linearize the attacker's objective function. Pan and Morton [51] use a similar objective function in their model and formulate it as a linear program. Similar to their approach, we can reformulate the function as a mixed-integer linear program by introducing extra variables and constraints. Let's introduce a positive variable g_s for each scenarios $s \in S$, given by $g_s = \{\prod_{(i,j)\in A(x,y,\xi)} p_{ij}^{z_{ijs}}\}$, representing the attacker's success probability for that scenario. Then, we can formulate the attacker's objective function as a linear program as follows:

$$\max \mathbb{E}[F(x, y, \xi)] = \sum_{s\in S} p_s g_s \tag{3.3}$$

subject to

$$\sum_{(\psi,j)\in FS(\psi)} (\lambda_{\psi js} + \mu_{\psi js}) = 1 \qquad \forall(\psi, j) \in A_2,\ \forall s \in S \tag{3.4}$$

$$\sum_{(i,j)\in FS(i)} (\lambda_{ijs} + \mu_{ijs}) = \sum_{(j,i)\in RS(i)} (p_{jis}\lambda_{jis} + q_{ijs}\mu_{ijs}) \qquad i \in A_2 - \{\psi, \tau\},\ \forall s \in S \tag{3.5}$$

$$g_s = \sum_{(j,\tau)\in RS(\tau)} (p_{j\tau s}\lambda_{j\tau s} + q_{j\tau s}\mu_{j\tau s}) \qquad \forall(j, \tau) \in A_2,\ \forall s \in S \tag{3.6}$$

Here, λ_{ijs} is the probability that the attacker reaches node $i \in V$ and arc (i, j) is not interdicted in scenario $s \in S$, and the chosen second-stage path includes arc (i, j). Similarly, μ_{ijs} is the probability that the attacker reaches node $i \in V$ and arc (i, j) is interdicted in scenario $s \in S$, and the chosen second-stage path includes arc (i, j). Hence, λ_{ijs} and μ_{ijs} cumulate the products of success probabilities along the chosen path from the source node to node j.

For a fixed defender decision $x \in X$, the resulting deterministic optimization model for the attacker's problem is given below:

$$\max_{y,z} \sum_{s\in S} p_s g_s \tag{3.7}$$

subject to

$$\sum_{(i,j)\in A_1} d_{ij}y_{ij} + \sum_{(i,j)\in A_2} d_{ij}z_{ijs} \le L \qquad \forall s \in S \tag{3.8}$$

$$\sum_{j|(i,j)\in A_1} y_{ij} - \sum_{j|(i,j)\in A_1} y_{ji} = b_i \qquad \forall j \in V \tag{3.9}$$

$$\sum_{j|(i,j)\in A_2} z_{ijs} - \sum_{j|(i,j)\in A_2} z_{jis} = b_i \qquad \forall j \in V,\ \forall s \in S \tag{3.10}$$

$$w_{\psi s} = 1 \qquad \forall s \in S \tag{3.11}$$

$$w_{js} = \sum_{i|(i,j)\in A_1} \xi_{ijs} w_{is} y_{ij} \qquad \forall j \in V,\ \forall s \in S \tag{3.12}$$

$$z_{\psi js} \le w_{js} \qquad \forall j \in V,\ \forall s \in S \tag{3.13}$$

$$z_{ijs} \le \xi_{ijs} y_{ij} + 2 - y_{ij} - w_{is} \qquad \forall(i, j) \in A_1,\ \forall s \in S \tag{3.14}$$

$$\sum_{(\psi,j)\in FS(\psi)} (\lambda_{\psi js} + \mu_{\psi js}) = 1 \qquad \forall(\psi, j) \in A_2, \forall s \in S \tag{3.15}$$

$$\sum_{(i,j)\in FS(i)} (\lambda_{ijs} + \mu_{ijs}) = \sum_{(j,i)\in RS(i)} (p_{ji}\lambda_{jis} + q_{ij}\mu_{ijs}) \quad i \in A_2 - \{\psi, \tau\}, \forall s \in S \tag{3.16}$$

$$g_s = \sum_{(j,\tau)\in RS(\tau)} (p_{j\tau}\lambda_{j\tau s} + q_{j\tau}\mu_{j\tau s}) \quad \forall (j,\tau) \in A_2,\ \forall s \in S \tag{3.17}$$

$$\lambda_{ijs} + \mu_{ijs} \leq z_{ijs} \quad \forall (i,j) \in A_2,\ \forall s \in S \tag{3.18}$$

$$\lambda_{ijs} \leq 1 - x_{ij} \quad \forall (i,j) \in A_1 \tag{3.19}$$

Here, the attacker wants to maximize the overall expected probability of a successful attack in two stages (3.7). 3.8 is the attacker's budget constraint. Constraints 3.9 and 3.10 ensure that the attacker's chosen path goes from the source node to the end node in the first and second stages, respectively. Constraint 3.11 ensures that the source node is reachable in all scenarios in the first stage. Constraint 3.12 enforces that a node $j \in N$ is reachable in scenario $s \in S$ $(w_{js} = 1)$ only if there exists a node $i \in N$ that is reachable $(w_{is} = 1)$, and arc (i,j) was chosen to be traversed $(y_{ij} = 1)$ and was traversable $(\xi_{ijs} = 1)$, in scenario $s \in S$ in the first stage. However, constraint 3.12 is nonlinear because it includes the product of two variables. Therefore, before solving, we linearize this constraint by introducing the binary variable v_{ijs} and the following four replacement constraints:

$w_{js} = \sum_{i|(i,j)\in A_1} \xi_{ijs} v_{ijs} \quad \forall j \in V,\ \forall s \in S$

$v_{ijs} \leq w_{is} \quad \forall (i,j) \in A_1,\ \forall s \in S$

$v_{ijs} \leq y_{ij} \quad \forall (i,j) \in A_1,\ \forall s \in S$

$v_{ijs} \geq w_{is} + y_{ij} - 1 \quad \forall (i,j) \in A_1,\ \forall s \in S.$

Constraint 3.13 ensures that if node $j \in N$ was not reachable in $s \in S$ $(w_{js} = 0)$, then the attacker can not traverse any arc starting from node $j \in N$ in scenario $s \in S$ $(z_{\psi js} = 0)$. Constraint 3.14 enforces that the attacker can not attempt to traverse arc (i,j) $(z_{ijs} = 0)$, if node $i \in N$ was reachable $(w_{is} = 1)$, and arc (i,j) was chosen to be traversed $(y_{ij} = 1)$ but was not traversable $(\xi_{ijs} = 0)$, in scenario $s \in S$. In other words, the attacker can not choose a second-stage path that includes arc (i,j) if he fails on arc (i,j) in the first stage in scenario $s \in S$. For a given scenario $s \in S$, the attacker's problem finds a path maximizing g_s by forcing one unit of

flow out of ψ in 3.15, enforcing flow conservation at all intermediate nodes in 3.16, and defining the flow that reaches the terminal node as g_s (which appears in the objective function) in 3.17. Constraint 3.18 connects the second-stage decision variable z_{ijs} with the linearizing variables λ_{ijs} and μ_{ijs}. Constraint 3.19 simply ensures that $\lambda_{ijs} = 0$ if the defender interdicts arc (i, j) $(x_{ij} = 1)$.

3.2.2 Three-stage Case

Similar to the two-stage model, in the three-stage model, the defender first protects a set of arcs in the attack graph representing the available strategies for attacking the system; the attacker then launches an optimal attack against the system, based on knowledge of which arcs have been protected. If the attacker fails at the first attempt, then he will choose a second-stage optimal attack strategy, based on the revised attack graph showing which arcs have already been successfully traversed, and which arc was impossible to traverse (leading to failure of the first-stage attack). If the attacker fails at the second attempt, then he will try one more time to traverse the system based on the revised attack graph resulting from the first and second stages.

We use the same notation as in the two-stage model, with additional variables and constraints to represent the third stage. In particular, the attacker's objective is now to maximize $F(x, y, z, \xi)$ which describes the penetrability of the network if the defender chooses to interdict arcs x, and the attacker chooses initial action y and second-stage action z, conditional on a given realization of random matrix ξ representing which arcs are traversable by the attacker.

As before, we assume that the defender wishes to minimize the maximum penetrability achievable by the attacker in three stages. For a fixed defender decision $x \in X$, the deterministic three-stage model for the attacker's problem is similar to the two-stage model. In the three-stage model, we keep constraints 3.9 through 3.17, and constraint 3.19. However, we introduce the following new attacker objective function and constraints in the three-stage case.

$$\max_{y,z,t} \sum_{s \in S} p_s g_s \tag{3.20}$$

subject to

$$\sum_{(i,j)\in A_1} d_{ij}y_{ij} + \sum_{(i,j)\in A_2} d_{ij}z_{ijs} + \sum_{(i,j)\in A_2} d_{ij}t_{ijs} \leq L \quad \forall s \in S \tag{3.21}$$

$$\sum_{j|(i,j)\in A_2} t_{ijs} - \sum_{j|(i,j)\in A_2} t_{jis} = b_i \quad \forall j \in V,\ \forall s \in S \tag{3.22}$$

$$o_{\psi s} = 1 \quad \forall s \in S \tag{3.23}$$

$$o_{js} = \sum_{i|(i,j)\in A_2} \xi_{ijs}u_{ijs}z_{ijs} \quad \forall j \in V,\ \forall s \in S \tag{3.24}$$

$$t_{\psi js} \leq w_{js} + o_{js} \quad \forall j \in V,\ \forall s \in S \tag{3.25}$$

$$t_{ijs} \leq \xi_{ijs}y_{ij} + 2 - y_{ij} - w_{is} \quad \forall (i,j) \in A_1,\ \forall s \in S \tag{3.26}$$

$$t_{ijs} \leq \xi_{ijs}z_{ijs} + 2 - z_{ijs} - o_{is} \quad \forall (i,j) \in A_1,\ \forall s \in S \tag{3.27}$$

$$\lambda_{ijs} + \mu_{ijs} \leq t_{ijs} \quad \forall (i,j) \in A_2,\ \forall s \in S \tag{3.28}$$

$$z_{kls_a} - z_{kls_b} \leq (\xi_{ijs_a}y_{ij} + 2 - y_{ij} - w_{is_a}) + (\xi_{ijs_b}y_{ij} + 2 - y_{ij} - w_{is_b})$$
$$\forall (k,l) \in A_2,\ \forall (i,j) \in A_1,\ \forall (s_a, s_b) \in S \tag{3.29}$$

$$t_{kls_a} - t_{kls_b} \leq (\xi_{ijs_a}z_{ijs_a} + 2 - z_{ijs_a} - q_{is_a}) + (\xi_{ijs_b}z_{ijs_b} + 2 - z_{ijs_b} - q_{is_b})$$
$$\forall (k,l) \in A_2,\ \forall (i,j) \in A_1,\ \forall (s_a, s_b) \in S \tag{3.30}$$

Here, the attacker wants to maximize the overall expected probability of a successful attack in three stages (3.20). Constraint 3.21 is the attacker's three-stage budget constraint. Constraint 3.22 ensures that the attacker's chosen path goes from the source node to the end node in the third stage. Constraint 3.23 ensures that the source node is reachable in all scenarios in the second stage. Constraint 3.24 enforces that a node $j \in N$ is reachable ($o_{js} = 1$) only if there exist a node $i \in N$ that was reachable ($o_{is} = 1$), and arc (i,j) was chosen to be traversed ($z_{ijs} = 1$) and was traversable ($\xi_{ijs} = 1$), in scenario $s \in S$ in the second stage. Similar to constraint 3.12, constraint 3.24 is also nonlinear and can be linearized in the same manner. Constraint 3.25 ensures that if node $j \in N$ was not reachable in scenario $s \in S$ both in stage one ($w_{js} = 0$) and in stage two ($o_{js} = 0$), then the attacker can not start to traverse from node $j \in N$ in scenario $s \in S$ in stage three ($t_{\psi js} = 0$). Constraint 3.26 enforces that the attacker can not attempt to traverse arc

(i, j) in scenario $s \in S$ in stage three ($t_{ijs} = 0$) if node $i \in N$ was reachable ($w_{is} = 1$) in stage one, and arc (i, j) was chosen to be traversed ($y_{ij} = 1$) but was not traversable ($\xi_{ijs} = 0$), in scenario $s \in S$ in stage one. Similarly, constraint 3.27 enforces that the attacker can not attempt to traverse arc (i, j) in stage three ($t_{ijs} = 0$) if node $i \in N$ was reachable in scenario $s \in S$ ($\rho_{is} = 1$) in stage two, and arc (i, j) was chosen to be traversed ($z_{ijs} = 1$) but was not traversable ($\xi_{ijs} = 0$), in scenario $s \in S$ in stage two. Similar to constraint 3.18, constraint 3.28 connects the third-stage decision variable t_{ijs} with the linearizing variables λ_{ijs} and μ_{ijs}.

Here, 3.29 and 3.30 are *nonanticipativity* constraints for the three-stage model. In general, nonanticipativity constraints enforce that "...the value of the decision vector, chosen at stage t, may depend on the information (data)... available up to time t, but not on the results of future observations" [61]. In our case, constraint 3.29 ensures that if the attacker fails on arc (i, j) in the first stage in scenarios s_a and s_b, the attacker must choose the same second-stage path in both scenarios (since any difference could depend only on the results of future observations.) Similarly, constraint 3.30 ensures that every time the attacker fails on arc (i, j) in the second stage, the attacker should choose the same third-stage path.

The nonanticipativity constraints capture the relationship between the decisions and the resolution of uncertainty. In particular, our model is a multi-stage stochastic program with "decision-dependent uncertainty" [18], since which uncertainties get resolved depends on which arcs the attacker chooses to traverse. Goel and Grossmann [18] categorize stochastic-programming problems into two groups: those with exogenous uncertainty [28], where the optimization decisions have no effect on the stochastic process; and those with endogenous uncertainty [52], where the optimization decisions influence the related stochastic process. In our problem, the arc on which the attacker fails is not only uncertain, but also depends on the attacker's decision (i.e., the path that the attacker decides to traverse). Because the optimization decision affects the stochastic process, the uncertainty in our model is endogenous. Therefore, our nonanticipativity constraints follow the modeling techniques of Goel and Grossmann [17], [18].

In the two-stage model, we did not need nonanticipativity constraints, because the information the attacker received from the first stage (the arcs he had already traversed successfully, and the

arc at which he failed) will be the same for all scenarios in which he failed on the same arc. So, the attacker will choose the same second-stage path for those scenarios without the need for extra constraints. However, in the three-stage model, without a nonanticipativity constraint, the attacker could choose different second-stage paths for scenarios in which he failed on the same arc at the first stage, because it would be more beneficial to select different arcs in those scenarios when considering the results of future observations (in the third stage). In reality, however, the attacker should choose the same second-stage action for all scenarios in which he failed on the same arc. So, we need nonanticipativity constraints to ensure that the model works properly.

3.3 Solution Approach for the Attacker Problem: Sample Average Approximation

Unfortunately, there is no general closed-form solution method for stochastic network interdiction problems. Therefore, we need to use a scenario-based approach to solve our multi-stage stochastic network interdiction problem. In our case, a scenario describes which arcs the attacker would be able to successfully attack if an attack was attempted. If the true scenario were known ahead of time, the problem would become a deterministic network-interdiction problem, because there would be no uncertainties about attack success or failure. However, in order to find an optimal attack strategy when the true scenario is unknown, the attacker and the defender are both assumed to consider a random subset of possible scenarios.

We can formulate the general two-stage stochastic programming problem as follows:

$$\nu^* = \max_{y \in Y} \{\nu(y) := \mathbb{E}[F(y, \xi)]\} \tag{3.31}$$

Here, the function $F(y, \xi)$ is the objective function of the second-stage attacker problem for a given x. In general, however, it is virtually impossible to solve $\mathbb{E}[F(y, \xi)]$ in 3.31 either analytically or numerically because of its high dimensionality. Thus, we need to use an approximate method. A common way of solving stochastic optimization problems is to simulate possible outcomes. However, the number of possible outcomes might be large or even infinite, creating computational challenges. In such cases, a random subset of all possible scenarios could be used to solve the

problem approximately. There are two main methods used to solve stochastic-programming problems: decomposition; and the SAA method [30]. The main idea of both methods is to find both upper and lower bounds on the optimal value of the objective function. The difference between the decomposition method and SAA is the sampling technique used. For the decomposition method, sampling takes place during the course of the algorithm. By contrast, in SAA, the sampling takes place before the solution algorithm. We adopt SAA, as follows:

Suppose that we have N realizations of the random matrix ξ, $\{\xi^1, \xi^2, ...\xi^N\}$. These random realizations could be either historical data or random variables. In our case, we generate these by Monte Carlo simulation. For any given $y \in Y$, we can then estimate $\nu(y)$ by averaging the values $F(y, \xi^j)$ for $j = 1, 2, ...N$. This is essentially just an SAA of the "true" problem given in (3.31). The formulation of the SAA problem is given below:

$$\hat{\nu}_N = \max_{y \in Y} \{\hat{\nu}_N(y) := \frac{1}{N} \sum_{j=1}^{N} F(y, \xi^j)\} \tag{3.32}$$

For a given sample, we can consider the SAA problem in (3.32) to be a stochastic-programming problem with scenarios $\xi^1, \xi^2, ...\xi^N$, each occurring with probability $\frac{1}{N}$. Since we sample the ξ^j from the set S, $\hat{\nu}_N(y)$ will be an unbiased estimator of $\nu(y)$. We can conclude that

$$\mathbb{E}[\hat{\nu}_N(y)] = \nu(y) \tag{3.33}$$

SAA appears to be especially effective for large problems, since a relatively small number of samples may be enough to represent a large number of possible scenarios. Therefore, we use SAA to find both lower and upper bounds on the optimal value of the objective function. Let ν^* in 3.31 be the optimal value of the objective function. Moreover, for samples $\xi^1, \xi^2, ..., \xi^N$, let $\hat{\nu}_N$ in 3.32 be the optimal value of the SAA problem. Then, from Shapiro et al. [61], the following two results hold:

$$\mathbb{E}(\hat{\nu}_N) \geq \nu^* \tag{3.34}$$

$$\hat{\nu}_N(\hat{y}) \leq \nu^* \quad \forall \hat{y} \in Y. \tag{3.35}$$

Thus, finding $\mathbb{E}(\hat{\nu}_N)$ as in 3.33 and $\hat{\nu}_N(\hat{y})$ as in 3.32 will give upper and lower bounds on the true optimal value, ν^*, of the objective function, respectively. We can compute these bounds using the following steps:

- Step 1: Generate M independent SAA problems of size N, by sampling from the set S.
- Step 2: Solve each of the SAA problems to get $\hat{\nu}_N^j$. Then $W_{N,M} = \frac{1}{M}\sum_{j=1}^{M}\hat{\nu}_N^j$ is an unbiased estimate of $\mathbb{E}(\hat{\nu}_N)$.
- Step 3: For each of the M independent SAA problems, generate a sample of size $\bar{N}$ from set S. Then we have $\nu(\hat{y}) = \mathbb{E}[\hat{\nu}_N^j(\hat{y}) := \frac{1}{\bar{N}}\sum_{i=1}^{\bar{N}} F(\hat{y}, \xi^{i,j})]$ for $\forall \hat{y} \in Y$, and $Q_{\bar{N},M} = \frac{1}{M}\sum_{j=1}^{M}\nu_{\bar{N}}^j(\hat{y})$ is an unbiased estimate of $\nu(\hat{y})$.
- Step 4: If the upper and lower bounds obtained from steps 2 and 3, respectively, are not sufficiently close, steps 1 through 3 can be repeated with a larger value of N.

Figure 3.1 shows the effect of the parameters M, N, and $\bar{N}$ on the results obtained using SAA. Here, increasing the number of SAA problems (M) provides tighter confidence intervals on the upper-bound value $\nu(\hat{y})$. Similarly, a larger number of recourse samples $(\bar{N})$ provides tighter confidence intervals on the lower-bound value $\mathbb{E}(\hat{\nu}_N)$. Finally, a larger number of scenarios (N) in theory leads the upper and lower bounds to converge to a good estimate of the optimal value of the objective function. However, lack of convergence may occur if some of the individual SAA problems cannot be solved to optimality. The SAA method can also be applied to stochastic programs with more than two stages by the appropriate definition of scenarios.

3.4 Solution Approach for the Defender Problem: Integer L-Shaped Method

Bilevel stochastic programming problems are difficult to solve. In the literature, there are algorithms that can solve bilevel stochastic programming problems approximately under certain conditions. However, to our knowledge, all of these algorithms require the lower-level problem (the attacker's problem, in our context) to be a linear optimization problem [20]. By contrast, in our model, the attacker's problem is a stochastic mixed-integer program, because we use binary

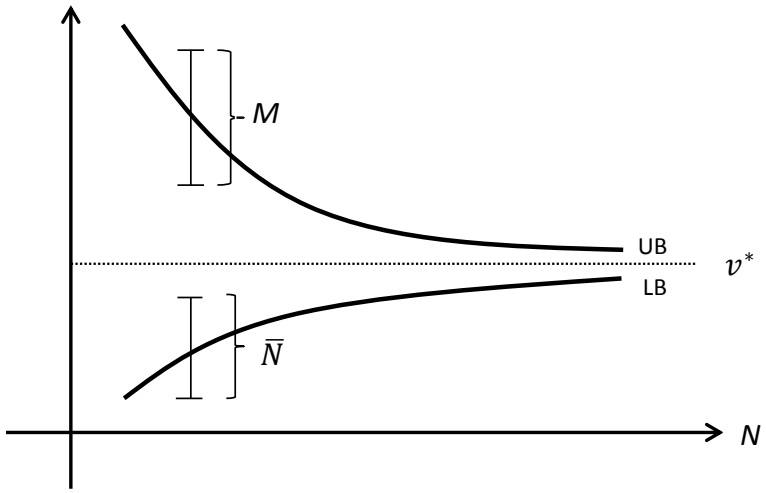

Figure 3.1: Effects of M, N and $\bar{N}$ on the MCM results (UB: Upper bound, LB: Lower bound)

decision variables to define the optimal path for the attacker. For the defender's problem, we also use binary decision variables to define which arcs are interdicted by the defender. Several studies have developed solution algorithms for specific stochastic integer programming problems [6], [31], [32], [54]. However, Laporte and Louveaux [33] introduced the integer L-shaped method, which has general applicability to stochastic integer programs. In their study, Laporte and Louveaux [33] show that the integer L-shaped method finds the optimal solution of a problem (if it exists) in a finite number of steps, if the problem has a valid set of feasibility cuts and a valid set of optimality cuts. According to Laporte and Louveaux [33], stochastic integer programs have both valid feasibility cuts and valid optimality cuts. Thus, the integer L-shaped method can be used for the solution of stochastic integer programs provided that: 1) the lower-level expectation function $F(x, y, \xi)$ (the attacker's problem, in our model) is computable for a given upper-level (defender) decision variable x; and 2) the objective value of the lower-level optimal solution has a lower bound.

We can consider our problem a bilevel stochastic integer program where the defender interdicts a set of arcs at the upper level, and then the attacker makes multiple attempts to traverse the network at the lower level. The first assumption above (computability) is satisfied if there are enough paths

for the attacker to traverse on a given attack graph. For the two-stage case, there should be at least two available paths, and for the three stage case, at least three paths should be available, because the attacker must select another path in a given attack stage if he fails at the previous attack stage. Moreover, for sufficiently large defender budget levels, the defender might be able to interdict all or most of the paths, in which case the attacker's problem $F(x, y, \xi)$ would not be computable. This situation is avoided, however, because of our assumption that if the defender interdicts arc $(i, j) \in A_1$ on the attack graph, then the probability of successfully attacking arc $(i, j) \in A_1$ will be reduced from p_{ij} to $q_{ij} > 0$. The second assumption (boundedness) holds because the objective value of the attacker's problem (the probability of succeeding in multiple attempts) is naturally bounded below by 0.

For stage K of the integer L-shaped method, we define the master problem as follows:

$$\min \ \theta \tag{3.36}$$

$$s.t. \sum_{(i,j)\in A_1} c_{ij}x_{ij} \leq B \tag{3.37}$$

$$\theta \geq \hat{\nu}_N(x_k) - \hat{\nu}_N(x_k)\Big(\sum_{(i,j)\in A_1:(x_{k(ij)}=0)} x_{ij}\Big), \quad k = 1, ..., K \tag{3.38}$$

$$\hat{\nu}_N(x_k) \geq 0, \ \theta \geq 0, \quad k = 1, ..., K \tag{3.39}$$

Here, variable θ represents an appropriate approximation of $F(x, y, \xi)$; i.e. an approximation satisfying constraint 3.38, which represents the K identified optimality cuts. Function $\hat{\nu}_N(x_k)$ represents the solution of the attacker's SAA problem with N scenarios for a given defender solution x_k. Constraint 3.37 represents the defender's budget limit. As discussed above, because our problem satisfied both computability and boundedness assumptions, the optimality cuts in 3.38 will always be valid.

We use the following steps to solve the defender's problem with the integer L-shaped algorithm:

- Step 0: Set $\vartheta = 0$. The variable θ is the lower bound for the objective function value of the defender problem, so is set equal to 0. An initial defender solution x^{ϑ} is selected, and the

corresponding attacker's problem $\hat{\nu}_N(x^\vartheta)$ is solved. Set $\overline{\psi} = \hat{\nu}_N(x^\vartheta)$, where $\overline{\psi}$ is the upper bound for the objective function value of the defender problem. If $\overline{\psi} = 0$, then go to Step 4. Otherwise, set $K = 1$, add one optimality cut to 3.38, and go to Step 1.

- Step 1: Set $\vartheta = \vartheta + 1$. Solve the master problem in 3.36 through 3.39. Let $(x^\vartheta, \theta^\vartheta)$ be an optimal solution.
- Step 2: Compute $\hat{\nu}_N(x^\vartheta)$ and $\psi^\vartheta = \hat{\nu}_N(x^\vartheta)$. If $\psi^\vartheta < \overline{\psi}$, set $\overline{\psi} = \psi^\vartheta$.
- Step 3: If $\theta^\vartheta = \overline{\psi}$, then go to Step 4. Otherwise, set $K = K + 1$, add one optimality cut to 3.38, and return to Step 1.
- Step 4: Set x^ϑ as the optimal defender solution and stop.

Chapter 4

Results for Attacker Model

In this chapter, we first solve the attacker's SAA problem, find bounds on the attacker's objective function, and illustrate the convergence of those bounds using Monte Carlo simulation. Then, we explore whether it is necessary to model a non-myopic attacker, who considers his future actions in choosing an optimal multi-stage attack strategy, or if it is sufficient to protect against only a myopic attacker, who may launch an optimal single-stage attack without consideration of future attack options. Also, we compare the two-stage and three-stage attacker cases, to explore whether the two-stage case adequately approximates higher levels of non-myopic attacker strategy. In the final section, we compare the current two-stage model with a relaxed version of the two-stage model that allows the attacker to choose a set of arcs in the first stage that does not form a complete path.

We use two simple attack graphs for our sensitivity analysis of the attacker model. The smaller attack graph on the left side of Figure 4.1 includes $|D| = 19$ nodes and $|A| = 35$ arcs, with a single source-terminal (S-T) pair. The second attack graph in Figure 4.1 is a larger instance consisting of $|D| = 55$ nodes and $|A| = 101$ arcs, again with a single S-T pair.

For each attack graph, we generate instances using the uniform distribution to determine the success probabilities p_{ij}. In particular, we select distributions with different mean arc success probabilities (0.2, 0.5, and 0.8). We are also interested in the effect of the variance of success probabilities for a given mean, so for the mean of 0.2, we allow the arc success probabilities to be distributed either Uniform(0,0.4) or Uniform(0.1,0.3). For a mean of 0.5, we use Uniform(0,1),

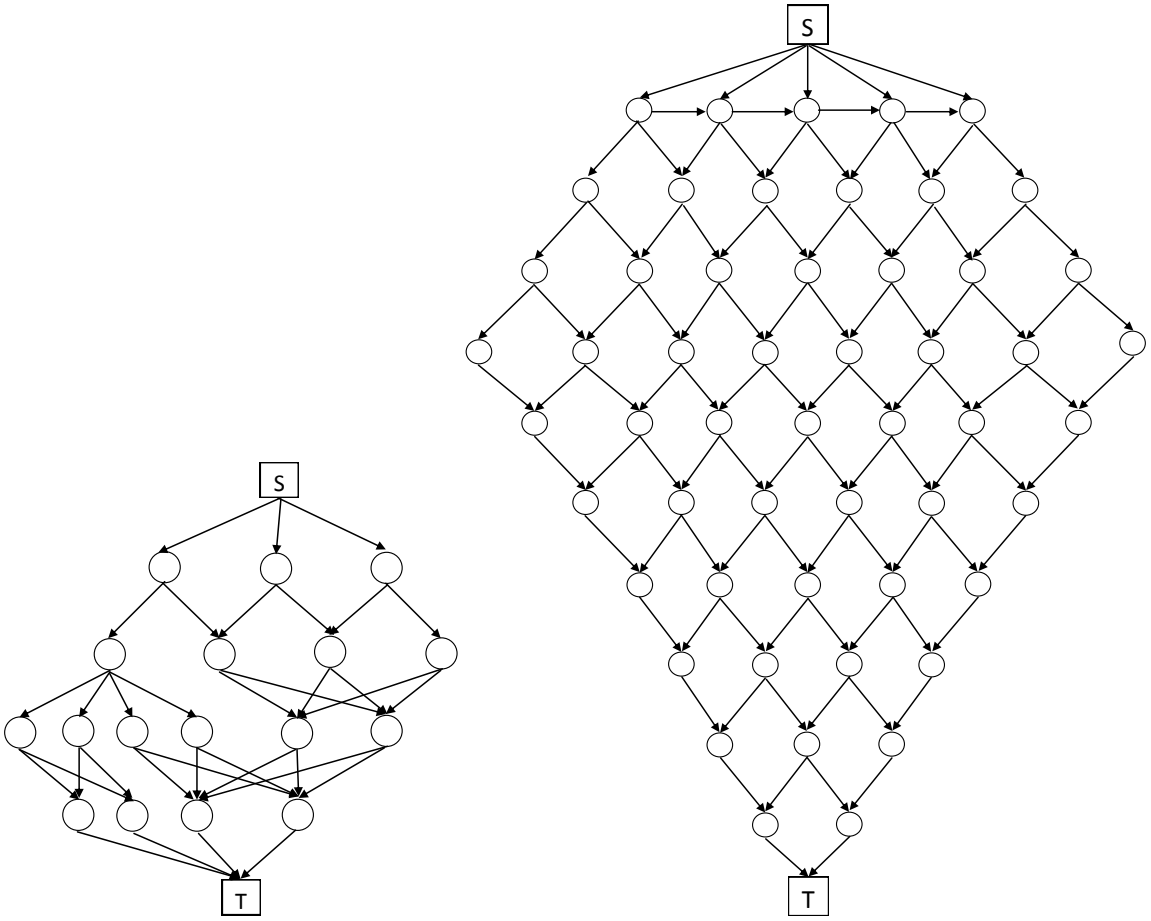

Figure 4.1: Attack graph instances

Uniform(0.3,0.7), and Uniform(0.4,0.6); and for a mean of 0.8, we use Uniform(0.6,1) and Uniform(0.7,0.9). In total, therefore, we consider seven possible distributions of arc success probabilities, for each attack graph, to test the effect of different mean success probabilities, and different variances for a given mean.

Throughout the sensitivity analysis, we assume that the attacker has an unlimited budget. This assumption can be revised in future work to explore the effect of attacker resource levels.

4.1 Convergence

In an attack graph, there are $2^{|A|}$ possible scenarios, so it is impractical to consider all of them even for the smaller attack graph with $|A| = 35$. As discussed in Chapter 3, we use M independent SAA samples each with a limited number of scenarios (N) to solve the attacker problem approximately. After finding the optimal attack strategy for the N randomly selected scenarios, we then fix the first-stage strategy to the identified optimal strategy and run the model $\bar{N}$ more times to assess the performance of that strategy in a process known as resampling. The average of the M SAA samples gives the upper bound, and the average of the $\bar{N}$ resamples gives the lower bound on the optimal value of the objective function.

In our analysis, we use the CPLEX solver, which is reasonably powerful and is able to solve our moderate-sized MIP problems to optimality. For larger instances, it is often able to find near-optimal solutions. We ran the models on a computer with an Intel(R) Core(TM)2 CPU 2.13 GHz processor and 2 GB RAM. Figures 4.2 and 4.3 show the computation time for the two-stage model for the smaller and larger attack graphs, respectively. The results indicate that the computation time typically increases more than linearly with the number of scenarios.

As the mean arc success probability increases, the required computation time also increases. Moreover, for a given mean success probability, the required computation time is also usually greater when the variance of the distribution is small, especially when the mean success probability is large.

For the two-stage attacker problem, we used 30 independent SAA samples $(M = 30)$, set the resampling size to $\bar{N} = 300$, and then varied the number of scenarios to explore the convergence of the lower and upper bounds for each instance of both the smaller and larger attack graphs. Figures 4.4, 4.5 and 4.6 show the lower and upper bounds for the smaller attack graph instances with mean 0.2, 0.5, and 0.8, respectively. Likewise, Figures 4.7, 4.8 and 4.9 show the lower and upper bounds for the larger attack graph instances with mean 0.2, 0.5, and 0.8, respectively. It is easy to see from these figures that as we increase the number of scenarios (N), the upper and lower bounds converge reasonably well for most instances. Based on the results, it appears

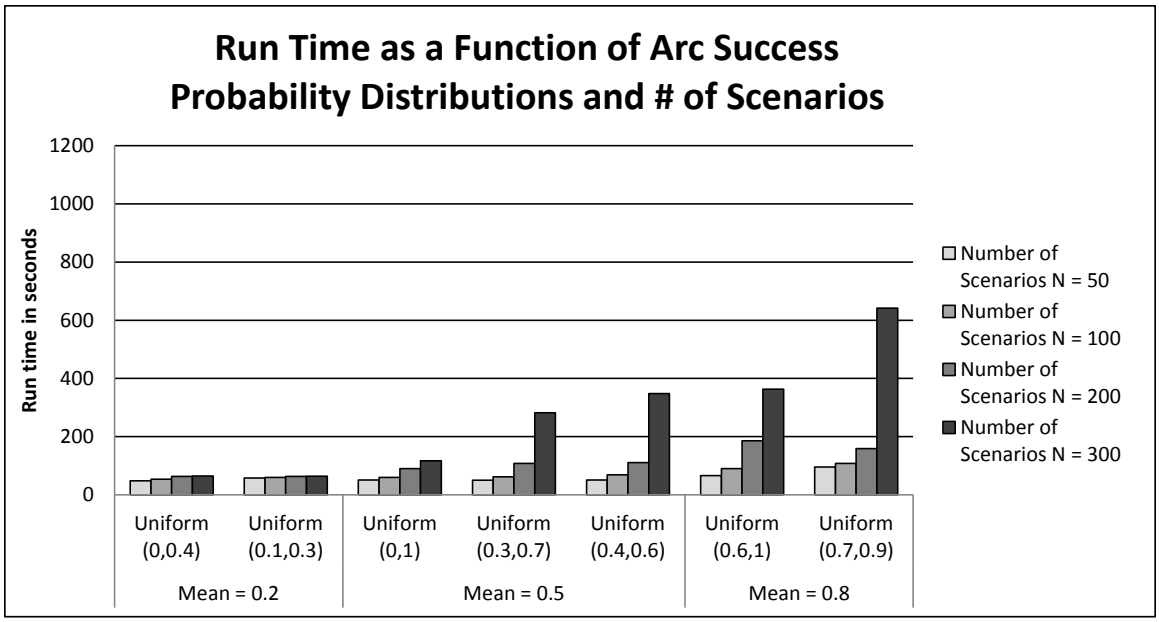

Figure 4.2: For the two-stage model, required run times for the smaller attack graph instances for different number of scenarios (N).

that we can adequately approximate the optimal strategies for the small attack graph with only $N = 300$ scenarios even though there are 2^{35} possible scenarios in total. Similarly, we can find reasonable bounds on the objective function value of the larger attack graph with around $N = 200$ scenarios even though there are 2^{101} possible scenarios in total. However, for small mean success probabilities (Figures 4.4 and 4.7), a larger number of scenarios appears to be required for the bounds to converge well. Of course, we could achieve better convergence for an even larger number of scenarios, but because the computation time grows quickly in some of these instances, as shown in Figures 4.2 and 4.3, we limited the number of scenarios to $N = 300$ for the smaller attack graph, and $N = 200$ for the larger attack graph.

Based on these results, we can conclude that for a given mean success probability, greater variability of the success probabilities from one arc to another generally yields a higher overall success probability. This is reasonable, because when the variance of the distribution is large, some paths on the attack graph would have much higher overall success probabilities than others.

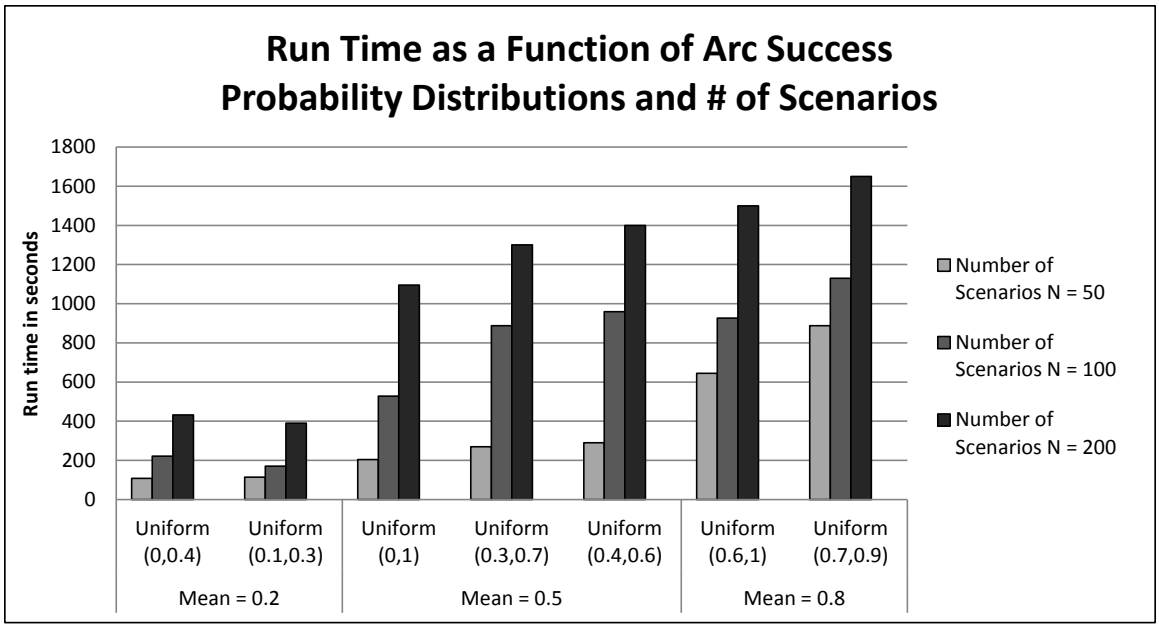

Figure 4.3: For the two-stage model, required run times for the larger attack graph instances for different number of scenarios (N).

Another observation is that the confidence intervals of the bounds are generally wider when the variance of the distribution is large, as expected. Finally, when the mean success probabilities are close to 0.5 (Figures 4.5 and 4.8), a larger number of scenarios appears to be required for the bounds to converge well, especially for larger attack graphs.

For the three-stage model, the required computational effort is much more than for the two-stage model, because the problem size grows rapidly with the increase of number of stages [61]. Using SAA, we were able to find bounds on the objective function value of the three-stage attacker problem, but we were able to solve the smaller attack graph problem for only 40 scenarios, which might not be enough for good convergence. For more than 40 scenarios, the optimization problem become computationally complex. For the smaller attack graph, we used 20 independent SAA samples $(M = 20)$, set the resampling size to $\bar{N} = 50$, and varied the number of scenarios to explore the convergence of the lower and upper bounds on the optimal value of the attacker objective function. Figures 4.10, 4.11, and 4.12 show the lower and upper bounds for the smaller

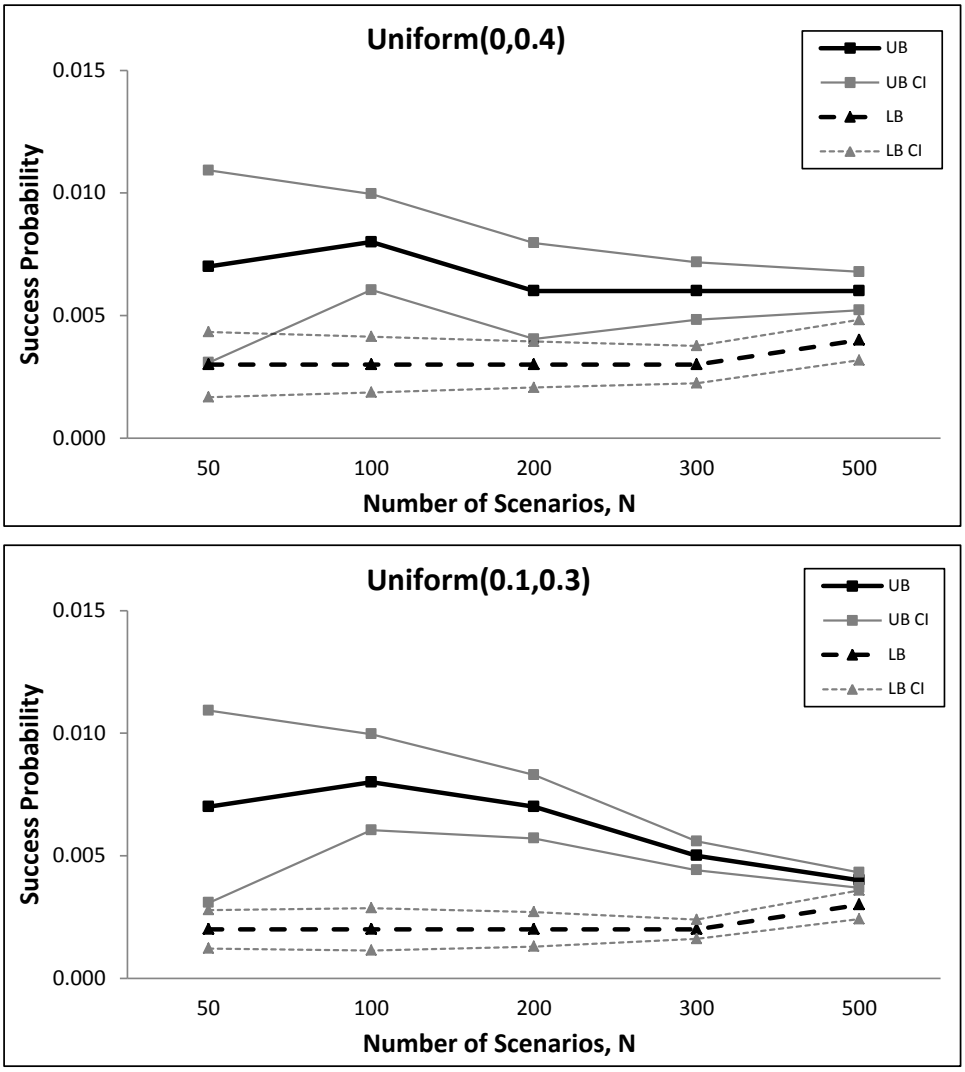

Figure 4.4: For the two-stage model, upper and lower bounds of the objective-function value of the smaller attack graph instances with success probabilities uniformly distributed with mean 0.2, for different number of scenarios (N). (UB: Upper bound, UB CI: Upper bound confidence intervals, LB: Lower bound, LB CI: Lower bound confidence intervals)

attack graph. The results show reasonable convergence of the upper and lower bounds, at least for 40 scenarios. However, the numerical values of the bounds are decreasing in the number of scenarios. This indicates that results for significantly less than 40 scenarios are not reliable.

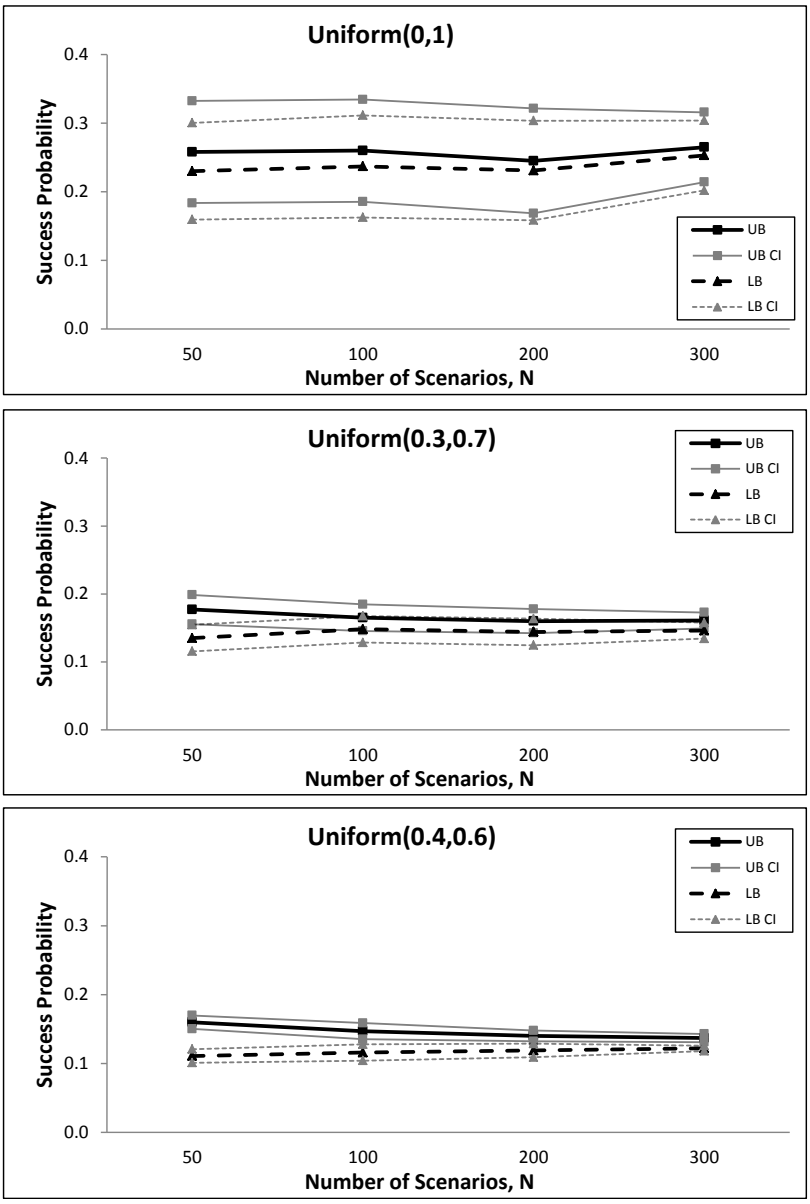

Figure 4.5: For the two-stage model, upper and lower bounds of the objective-function value of the smaller attack graph instances with success probabilities uniformly distributed with mean 0.5, for different number of scenarios (N). (UB: Upper bound, UB CI: Upper bound confidence intervals, LB: Lower bound, LB CI: Lower bound confidence intervals)

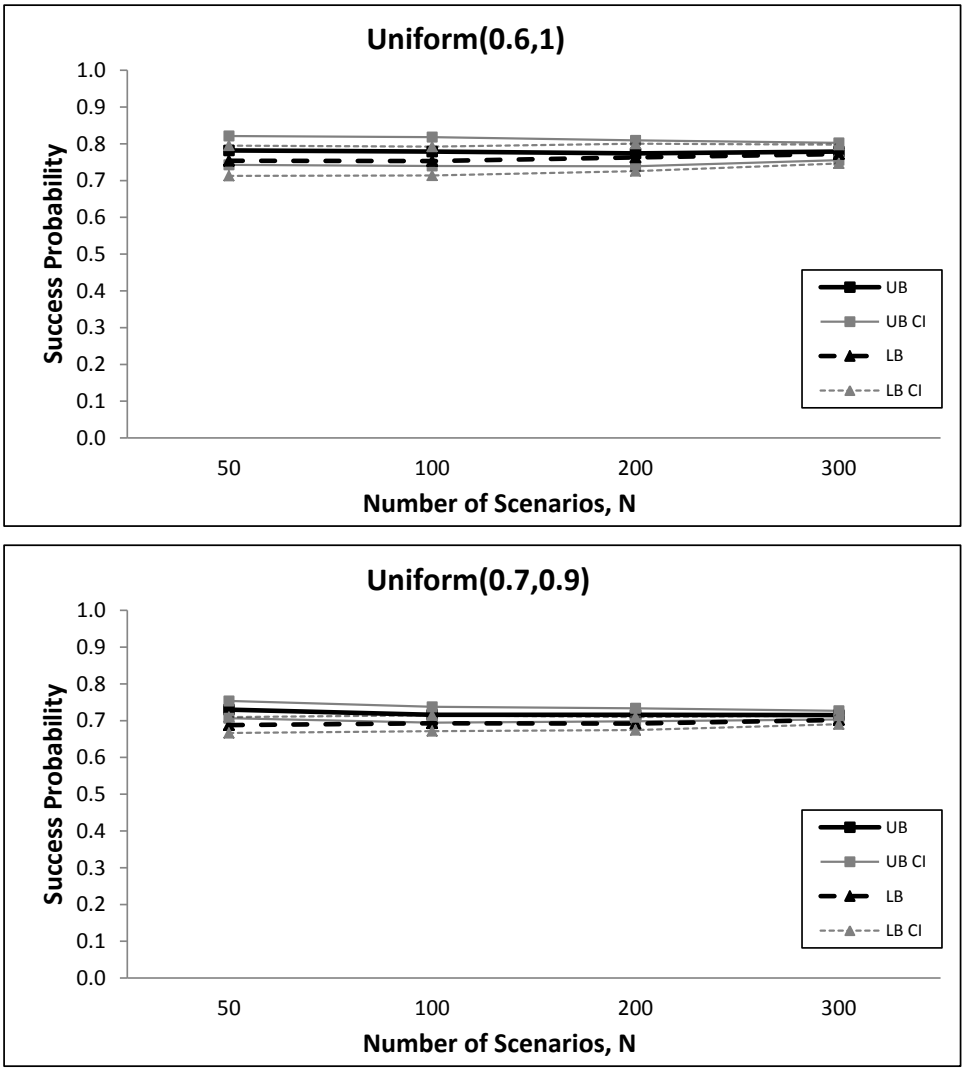

Figure 4.6: For the two-stage model, upper and lower bounds of the objective-function value of the smaller attack graph instances with success probabilities uniformly distributed with mean 0.8, for different number of scenarios (N). (UB: Upper bound, UB CI: Upper bound confidence intervals, LB: Lower bound, LB CI: Lower bound confidence intervals)

Thus, overall, we believe that we can get reliable results for the two-stage model, especially for the smaller attack graph. For the larger attack graph, a large number of scenarios is required to get good convergence on the bounds, especially when the mean of the arc success probabilities is close to 0.5, making it difficult to get reliable results. Also, due to the difficulty of solving the

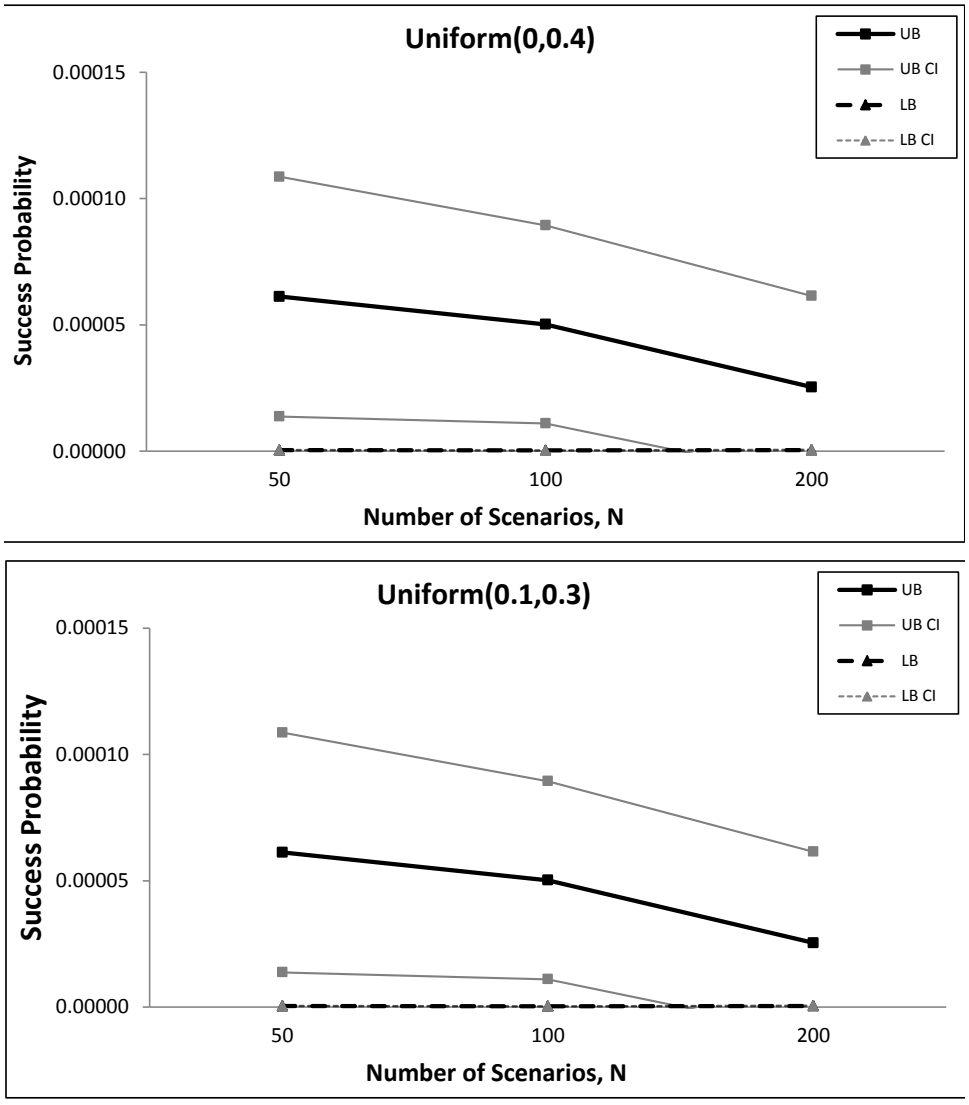

Figure 4.7: For the two-stage model, upper and lower bounds of the objective-function value of the larger attack graph instances with success probabilities uniformly distributed with mean 0.2, for different number of scenarios (N). (UB: Upper bound, UB CI: Upper bound confidence intervals, LB: Lower bound, LB CI: Lower bound confidence intervals)

larger attack graph even for a two-stage model, in our future analysis, we will not do sensitivity analysis for the larger attack graph for the three-stage model. Therefore, for arc success probability

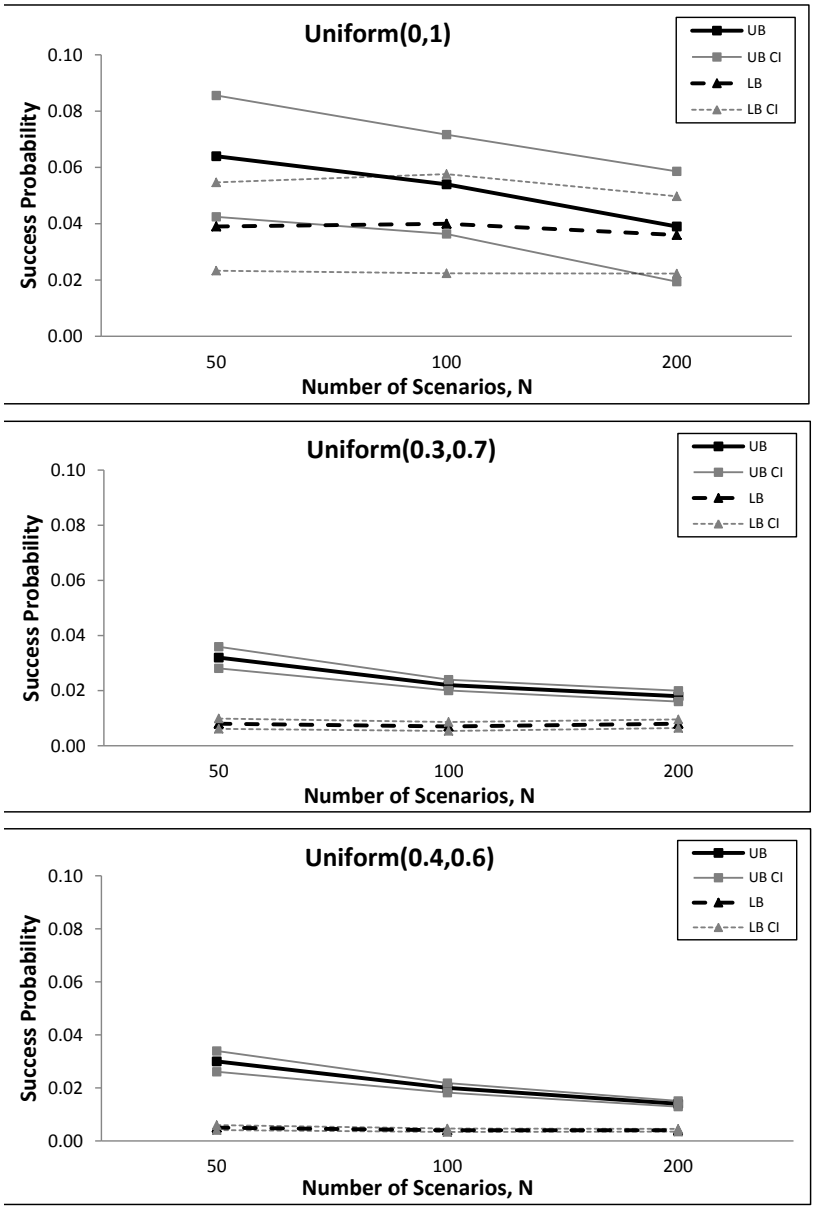

Figure 4.8: For the two-stage model, upper and lower bounds of the objective-function value of the larger attack graph instances with success probabilities uniformly distributed with mean 0.5, for different number of scenarios (N). (UB: Upper bound, UB CI: Upper bound confidence intervals, LB: Lower bound, LB CI: Lower bound confidence intervals)

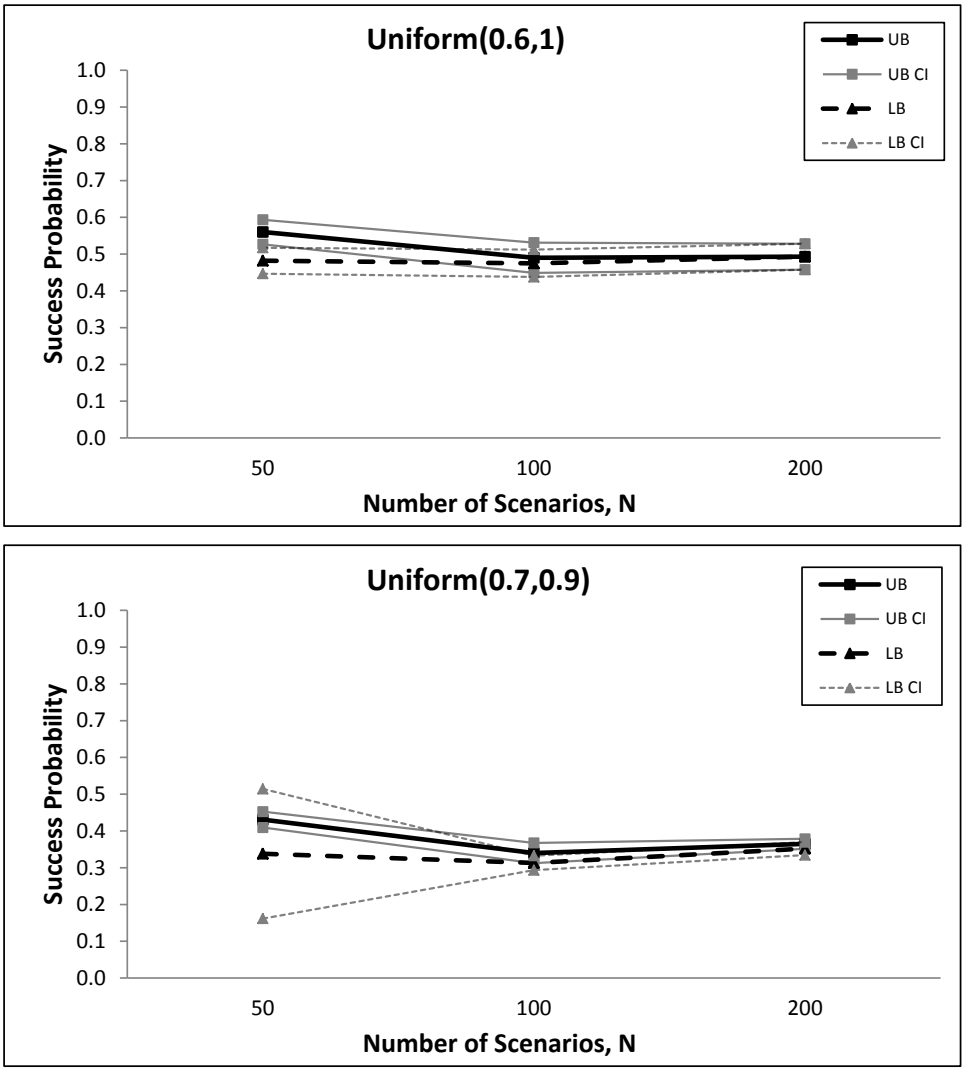

Figure 4.9: For the two-stage model, upper and lower bounds of the objective-function value of the larger attack graph instances with success probabilities uniformly distributed with mean 0.8, for different number of scenarios (N). (UB: Upper bound, UB CI: Upper bound confidence intervals, LB: Lower bound, LB CI: Lower bound confidence intervals)

distributions with large variances, we will use more SAA samples and recourse samples to find narrower confidence intervals on the results, for the two-stage model.

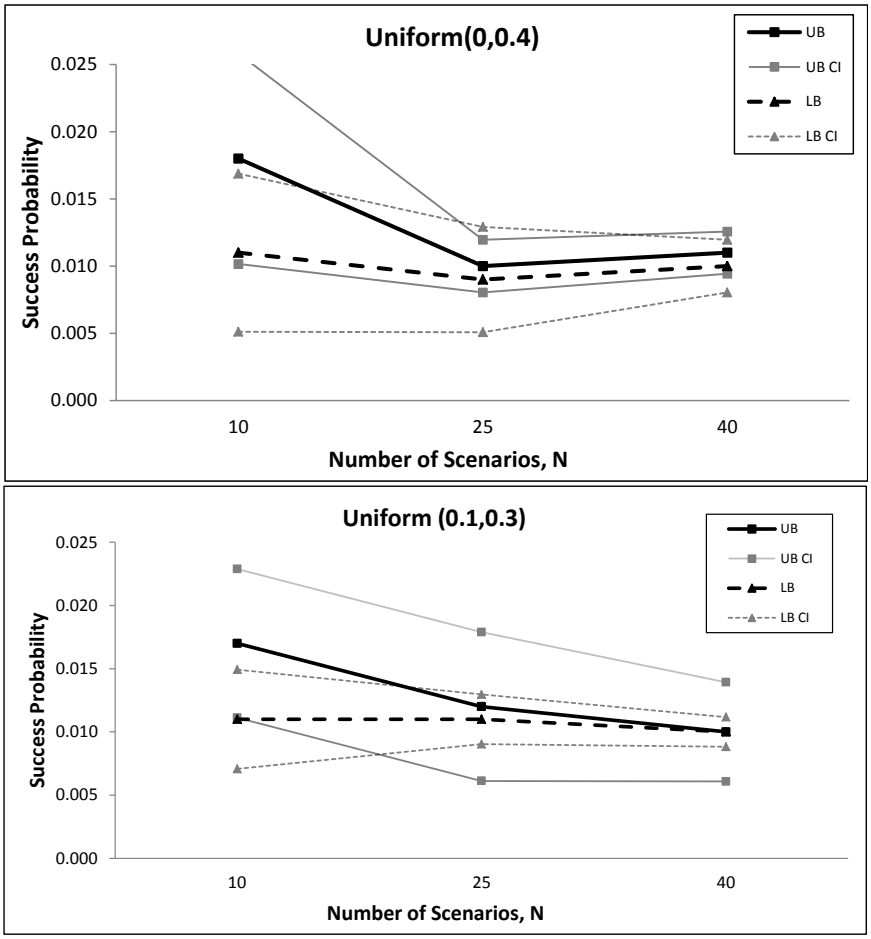

Figure 4.10: For the three-stage model, upper and lower bounds of the objective-function value of the smaller attack graph instances with success probabilities uniformly distributed with mean 0.2, for different number of scenarios (N). (UB: Upper bound, UB CI: Upper bound confidence intervals, LB: Lower bound, LB CI: Lower bound confidence intervals)

4.2 Comparison of Myopic and Non-myopic Attacker Cases

In the context of cyber security, it is important to anticipate the attacker's likely strategy and take proactive actions to defend the system. It is not unrealistic to assume an intelligent attacker who plans his attack strategy carefully before attempting to attack a network. Thus, in our model, we assume an intelligent and non-myopic attacker who considers his future actions when choosing

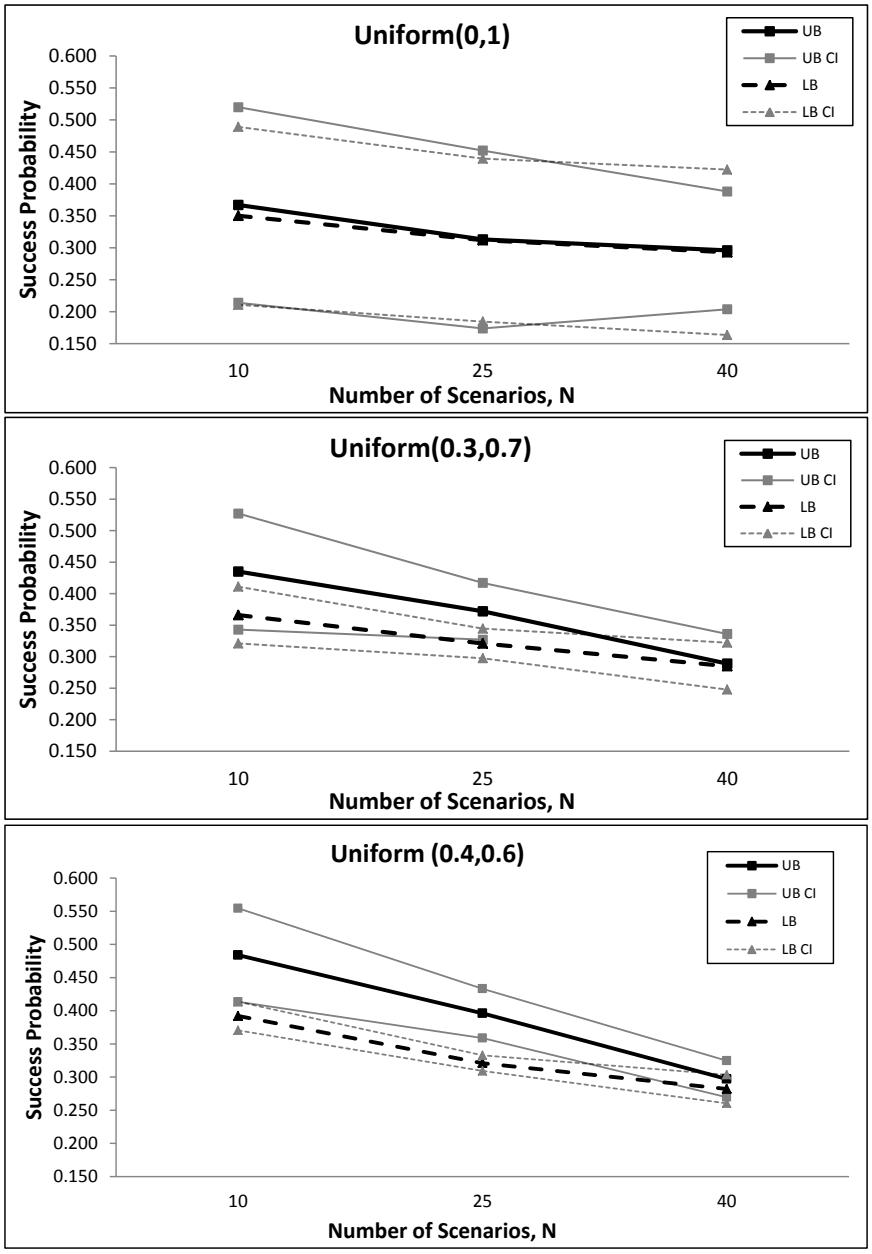

Figure 4.11: For the three-stage model, upper and lower bounds of the objective-function value of the smaller attack graph instances with success probabilities uniformly distributed with mean 0.5, for different number of scenarios (N). (UB: Upper bound, UB CI: Upper bound confidence intervals, LB: Lower bound, LB CI: Lower bound confidence intervals)

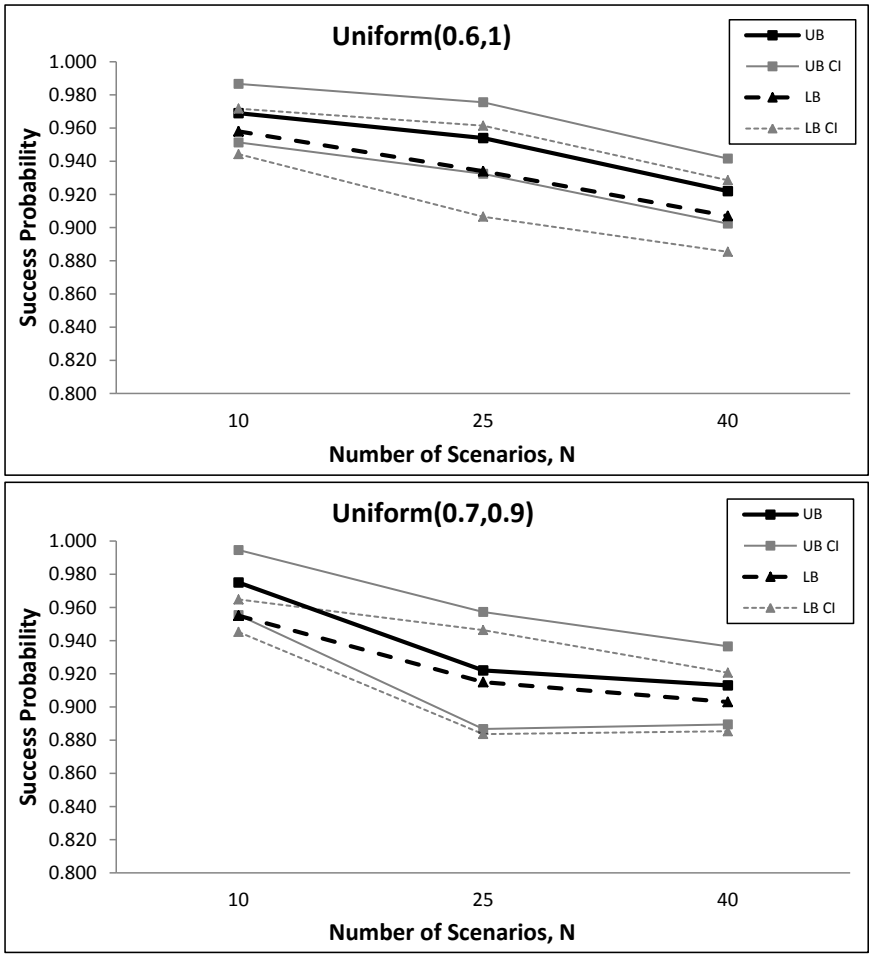

Figure 4.12: For the three-stage model, upper and lower bounds of the objective-function value of the smaller attack graph instances with success probabilities uniformly distributed with mean 0.8, for different number of scenarios (N). (UB: Upper bound, UB CI: Upper bound confidence intervals, LB: Lower bound, LB CI: Lower bound confidence intervals)

his current attack strategy. However, because modeling a non-myopic attacker is computationally demanding, it is interesting to determine whether it is sufficient to model only a myopic attacker, who at each stage may launch an optimal single-stage attack without consideration of future attack options. To answer this question, we compared the myopic and non-myopic attacker cases on both the smaller and the larger attack graph instances. In the comparison, for each attack graph instance, we first solved the attacker problem deterministically (without considering future attacks) to find

the optimal first-stage attack path for the myopic attacker, and then fixed the first-stage strategy to the identified optimal myopic strategy and ran the second-stage model $\bar{N}$ more times to assess the performance of that strategy. Similarly, for the non-myopic attacker, we solved the two-stage attacker problem to find the optimal first-stage attack path, and then we fix the first-stage strategy to the identified optimal strategy and run the second-stage model $\bar{N}$ more times to assess the performance of that strategy.

For each instance of the smaller attack graph, we used $M = 200$ samples (runs), $N = 300$ scenarios, and $\bar{N} = 400$ recourse samples. Figure 4.13 shows the resulting ratios of the expected success probability for the non-myopic attacker to the expected success probability for the myopic attacker case. Similarly, for the larger attack graph, we used $M = 100$ samples (runs), $N = 200$ scenarios, and $\bar{N} = 300$ recourse samples. Figure 4.14 again shows the ratio of the expected success probabilities for the non-myopic attacker and myopic attackers. Here, to find the upper confidence interval limit of a given ratio, we use the ratio of the upper and lower limits of the confidence intervals of the expected success probability for the non-myopic and myopic attackers at the 95% confidence level, respectively. Similarly, for the lower confidence interval limit, we find the ratio of the lower and upper limits of the confidence intervals of the expected success probability for the non-myopic and myopic attackers, respectively.

The results indicate as expected that non-myopic attackers perform better than myopic attackers. This difference is especially great when the success probabilities are low, perhaps because learning about non-promising paths is more important in that case. Moreover, for a given mean success probability, the non-myopic attacker has a greater advantage over the myopic attacker when the variance of the distribution is large. This is reasonable, because with wider distributions, some paths are much better than others, so taking future actions into account may be especially beneficial in that case.

4.3 Comparison of Two-stage and Three-stage Attack Strategies

Another important research question in a multi-stage defender-attacker model is how many attack stages need to be modeled; since modeling even two stages of attack by a non-myopic

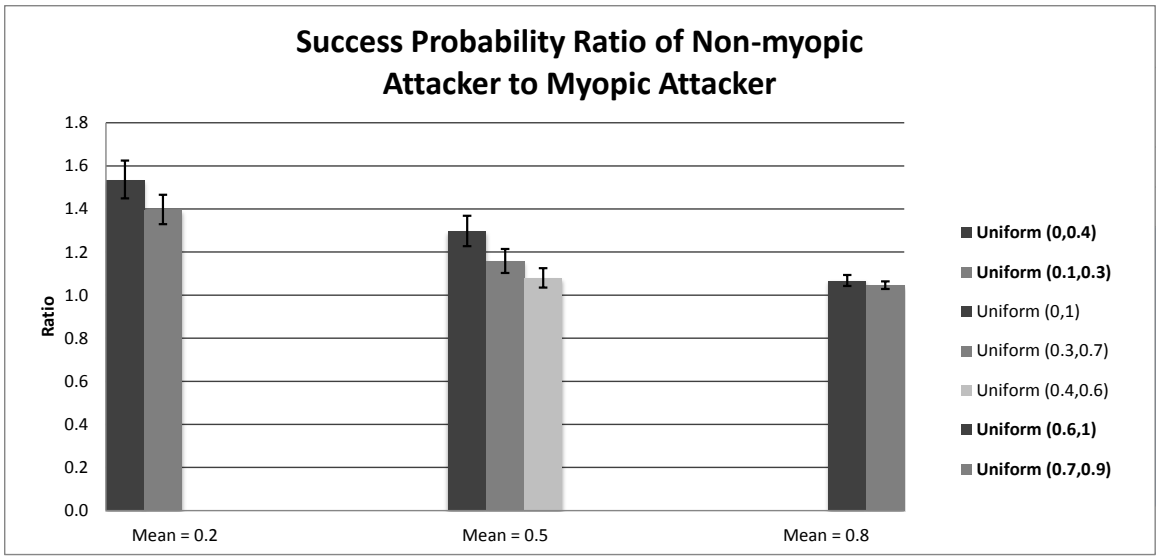

Figure 4.13: Comparison results of the myopic and non-myopic attacker cases for the smaller attack graph instances.

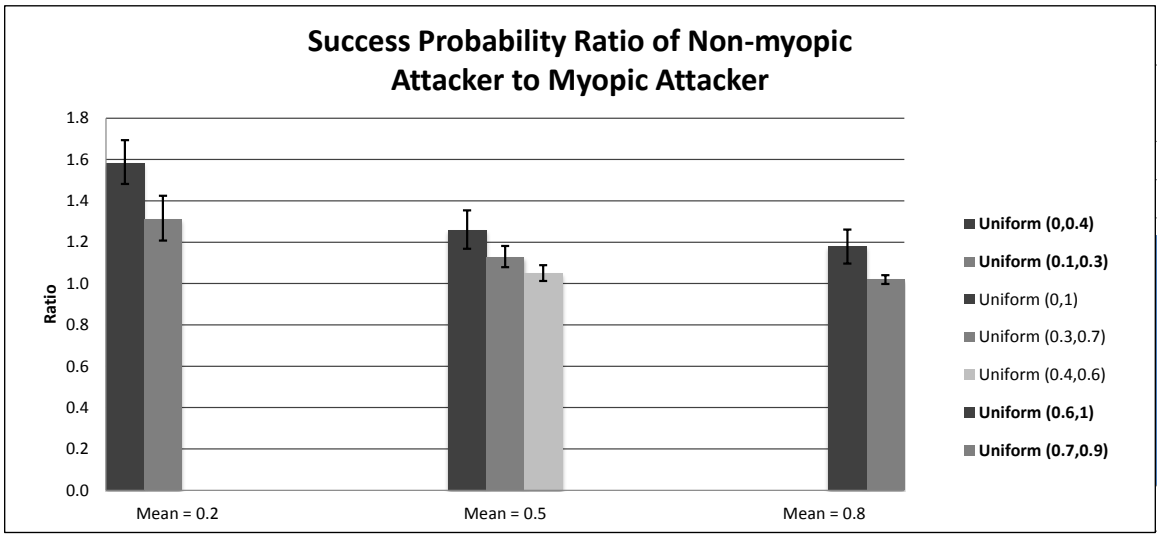

Figure 4.14: Comparison results of the myopic and non-myopic attacker cases for the larger attack graph instances.

attacker is demanding. To answer this question, we compare an attacker who can make only two attempts to one who can make three attempts. If we can adequately approximate the results of

the three-stage case with those of the two-stage case, then we may not need to model the problem for more than two stages. In the comparison, we use only the smaller attack graph, because we were able to solve the three-stage model only for the smaller attack graph. For each instance of the smaller attack graph, we solved the two-stage model to find the optimal first-stage attack path, and then fixed the first-stage strategy to the identified optimal strategy and ran the second and third stage model $\bar{N}$ more times to assess the performance of that strategy. Similarly, we solved the three-stage model to find the optimal first-stage attack path, and then fixed the first-stage strategy to the identified optimal strategy and ran the second and third stage model $\bar{N}$ more times to assess the performance of that strategy.

For each attack graph instance, we use $M = 200$ samples (runs), $N = 300$ scenarios, and $\bar{N} = 400$ recourse samples to solve the two-stage case. However, for the three-stage case, we were able to use only $M = 100$ samples (runs), $N = 40$ scenarios, and $\bar{N} = 50$ recourse samples for each instance. Figure 4.15 shows the ratios of the expected success probabilities of the three-stage and two-stage cases for the smaller attack graph. The confidence intervals on the ratios are again calculated conservatively, as in section 4.2.

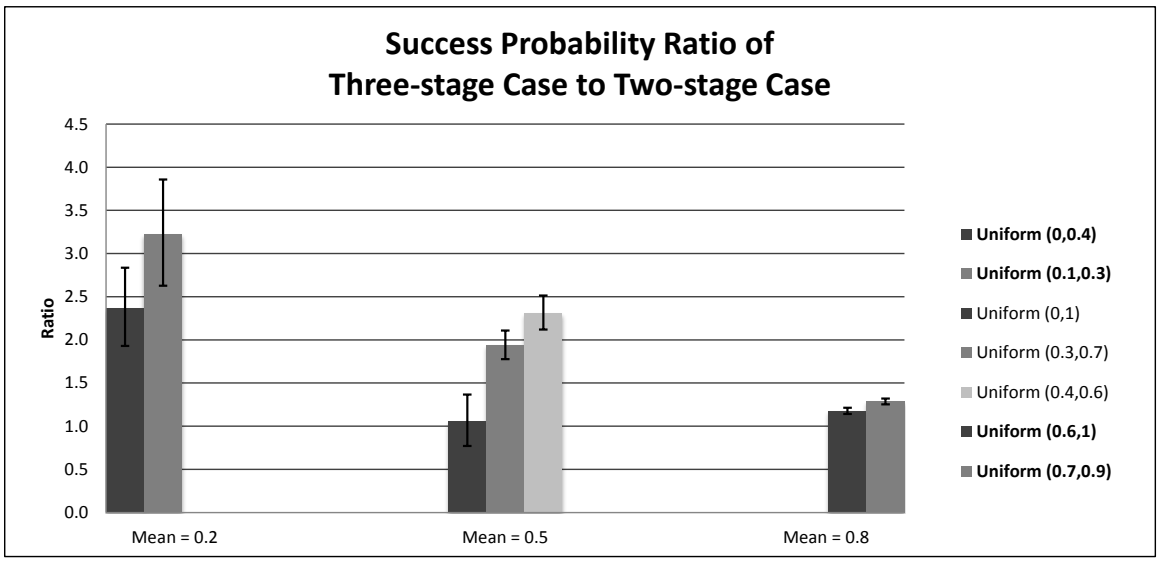

Figure 4.15: Comparison results of the two-stage and the three-stage cases

The results indicate that for a given mean success probability, the advantage of having three attack stages is greater when the distribution is narrower. This is reasonable, because with wider distributions, some paths are much better than other paths, so the best first-stage path is likely to be much better than a possible third-stage path. However, for narrower distributions, the first, second, and third-stage paths are all comparable, so a third attack stage gives more benefit.

Also, when the mean of the arc success probabilities is large, the three-stage model appears to be less beneficial. This might be because the attacker is likely to already succeed on the first or second stage. Thus, a third attack stage is more beneficial to the attacker when the mean of the arc success probabilities are small.

Note also that due to the ability to learn from earlier attack stages, the benefit of increasing the number of attacks by 1.5 (from 2 to 3) may yield much more than a factor of 1.5 increase in the overall probability of successful attack, especially when the mean success probability is small. this suggests that it is likely to be important to model more than two attack stages, especially for well-defended systems. Fortunately, though, run times are also much less when the arcs success probabilities are small, so this may not be a problem.

Unfortunately, however, the above findings are not highly reliable, because we were not able to solve the three-stage model with a reasonable number of scenarios. We need solve the three-stage case with many more than 40 scenarios to get reliable results, but this is not possible using the current solution approach because of its computational complexity.

4.4 Two-Stage Attacker Problem without First-Stage Path Constraint

For the attacker's problem, it is of course possible to allow a wider variety of attack options by relaxing the current model. One interesting extension is to allow the attacker to choose a set of arcs to attack in the first stage that does not form a complete path. With this relaxation, the attacker could conduct more general explorations in the first stage, to see which arcs are traversable.

In order to assess whether limiting the first-stage attack to a valid attack path is a good strategy, we compare the results of the current two-stage model and the relaxed model on the smaller attack graph, using the Uniform(0,1) distribution to generate arc success probabilities. For the attacker

budget, we use budget levels of B=5, B=10, B=15 for the number of arcs that can be targeted in the first-stage attack. For each budget level, we solve the problem with $M = 20$ samples (runs) and $N = 300$ scenarios, and compare the results with and without the first-stage path constraint. Table 4.1 summarizes the average percentage improvement on the attacker's expected success probability when there is no path constraint. Each column of this table shows the improvement in each of 20 runs for a given budget level.

The smaller attack graph has five arcs on every path. So, a budget level of five allows the attacker to try attacking the number of arcs that would be on a path. For this budget level, the attacker without a first-stage path constraint sees on average only a 3.6% improvement over the attacker with a path constraint. We also tried higher budget levels, to assess how the results change when the attacker can attempt to traverse more arcs. The results indicate that having a first-stage path constraint is a reasonable assumption when the attacker's budget is limited, because the improvement is small (only 3.6%). However, when the attacker has enough budget to traverse more than one path, requiring the first-stage attack to form a path becomes quite limiting relative to more general exploration.

Figure 4.16 illustrates the attacker's first-stage attack strategy both for the current two-stage model and for the relaxed model with different budget levels. It is easy to see from the figure that the attacker explores more paths when he has enough budget to traverse more than one path in the first stage. However, we believe that the assumption of a first-stage path constraint is not overly limiting if the attacker has only modest ability to explore in the first stage.

Table 4.1: Performance comparison of the two-stage attacker model with and without first-stage path constraint for different attacker budgets

	% improvement on the attacker's success probability		
Run	**B=5**	**B=10**	**B=15**
1	10	32	42
2	2.1	38	56
3	0.92	41	67
4	0.81	26	41
5	2.5	11	17
6	0	16	18
7	6.1	24	33
8	4.2	32	49
9	3.1	21	33
10	7.1	22	27
11	4.3	45	62
12	6.1	40	58
13	0	6	8
14	0	18	28
15	5.3	23	29
16	2.1	25	36
17	2.1	16	21
18	0	48	86
19	8.4	35	40
20	5.9	34	48
Average	**3.6**	**28**	**40**
Std. Error	**0.68**	**2.5**	**4.3**

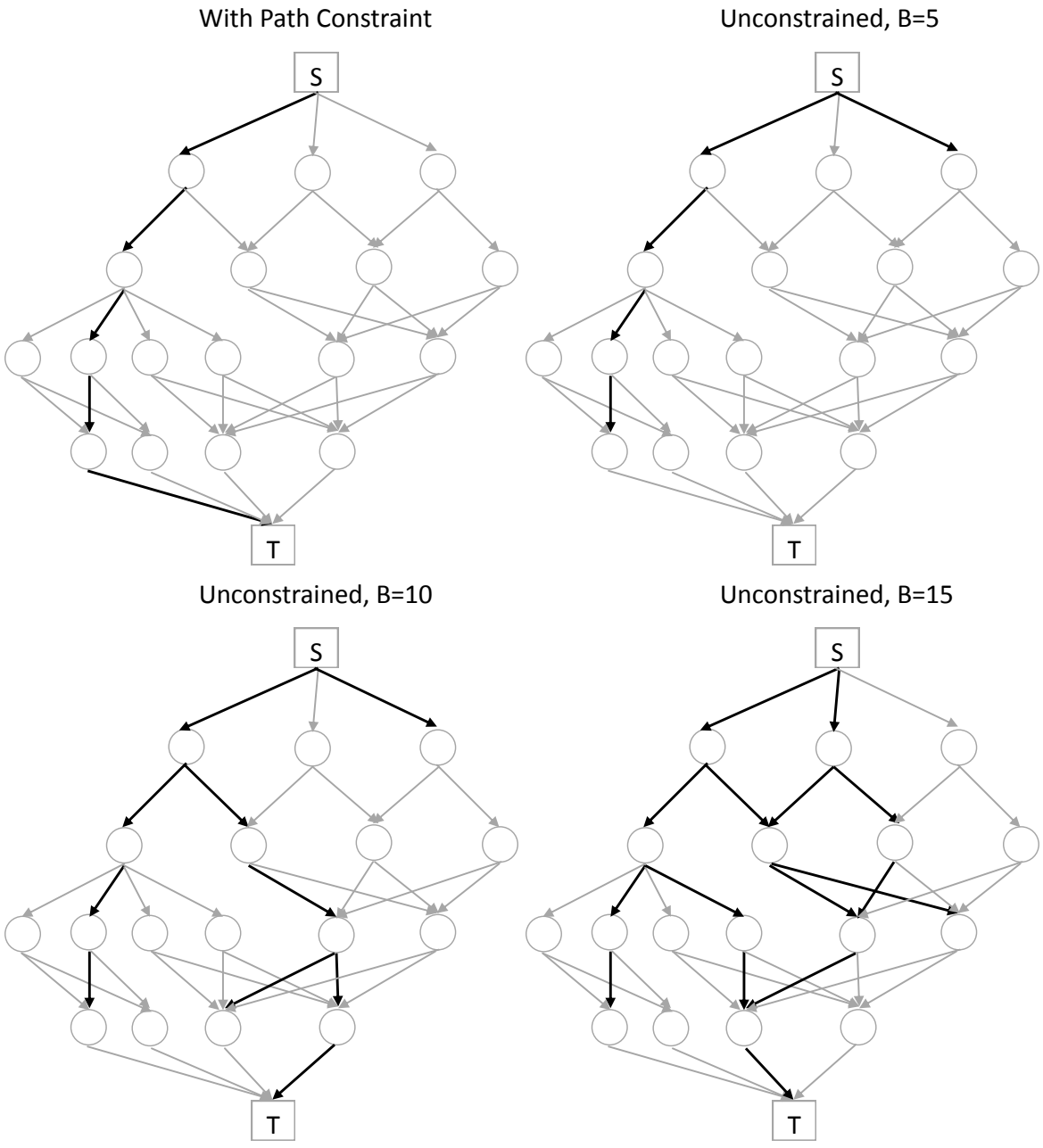

Figure 4.16: The attacker's optimal first-stage strategy (as shown by the bold arcs for the current two-stage model (top left), and the relaxed model with budget levels of 5 (top right), 10 (bottom left), and 15 (bottom right)).

Chapter 5

Results for Defender Model

In this chapter, we explore the effect of the attack graph size and structure, attack success probabilities, attacker type, and defense level on the optimal solution of the defender problem. As discussed in Chapter 3, we will use the integer L-shaped method for to solve the defender problem.

5.1 Sensitivity Analysis Setup

In our sensitivity analysis, we use three realistic network topologies (from literature [36], [37]) and two levels of vulnerability densities per host. Also, we include the smaller attack graph given on the left side of Figure 4.1. For the network topology selection, we consulted with Prof. Ou from Kansas State University. When selecting network topologies, our aim was to select common and realistic topologies and use our model to obtain some insight into attacker behaviors and optimal defense policies. Therefore, we selected networks similar to the fully-connected, and star-type networks for use in our sensitivity analysis. In a fully-connected network, each of the nodes (hosts) is connected to every other node, and in a star topology, every node is directly connected to the central node.

At the beginning, we selected four networks topologies for the sensitivity analysis. However, the corresponding attack graph of one network topology was not directed, that is not solvable by our current model. Thus, we did not include that network topology in our analysis. Therefore, in total, we have six attack graphs (three topologies, with two vulnerability densities for each topology). The topologies and corresponding attack graphs for both one and two vulnerabilities

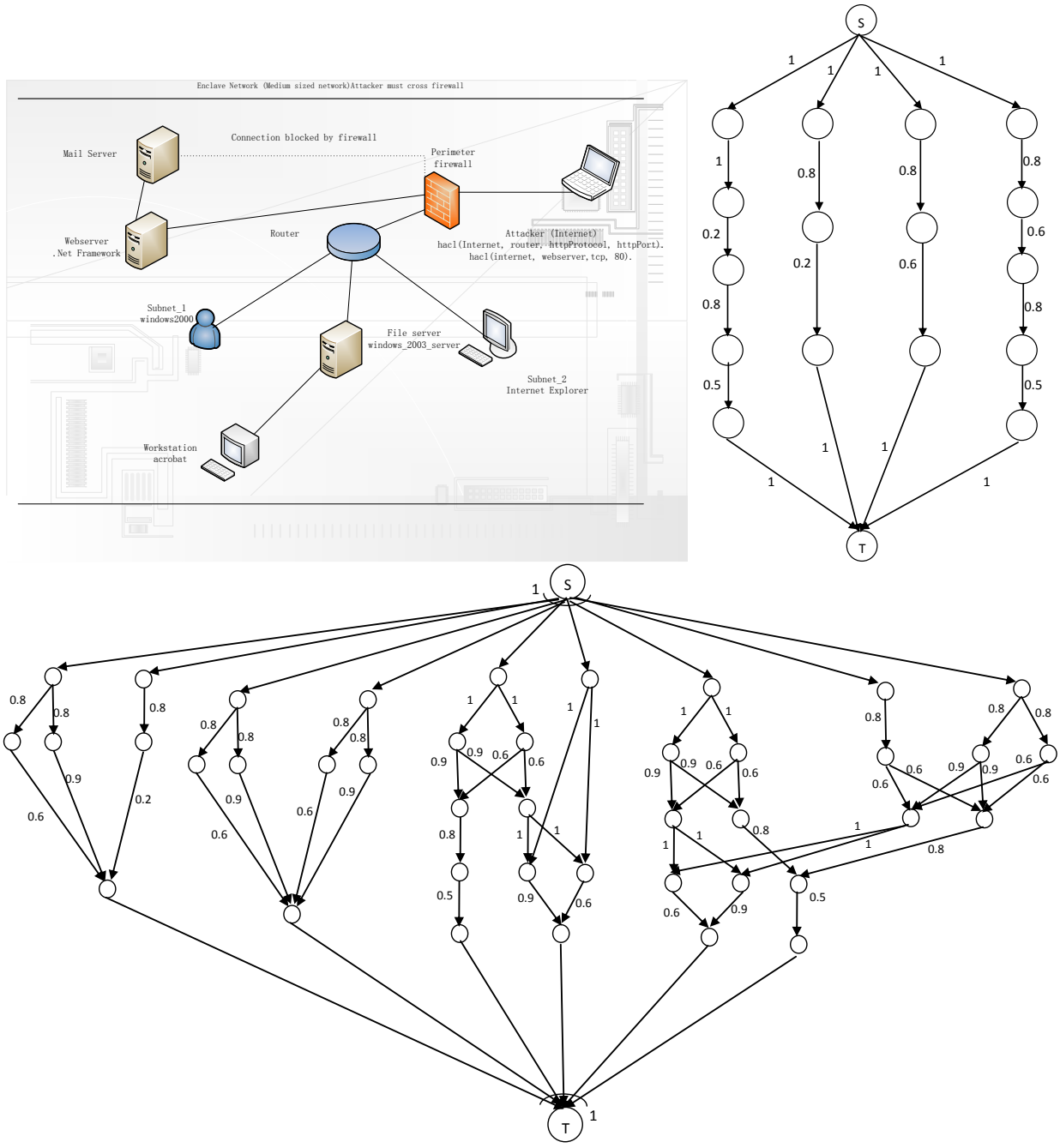

Figure 5.1: Case 1: Larger star-type network, and corresponding attack graphs with one vulnerability per host (top left), and two vulnerabilities per host (bottom)

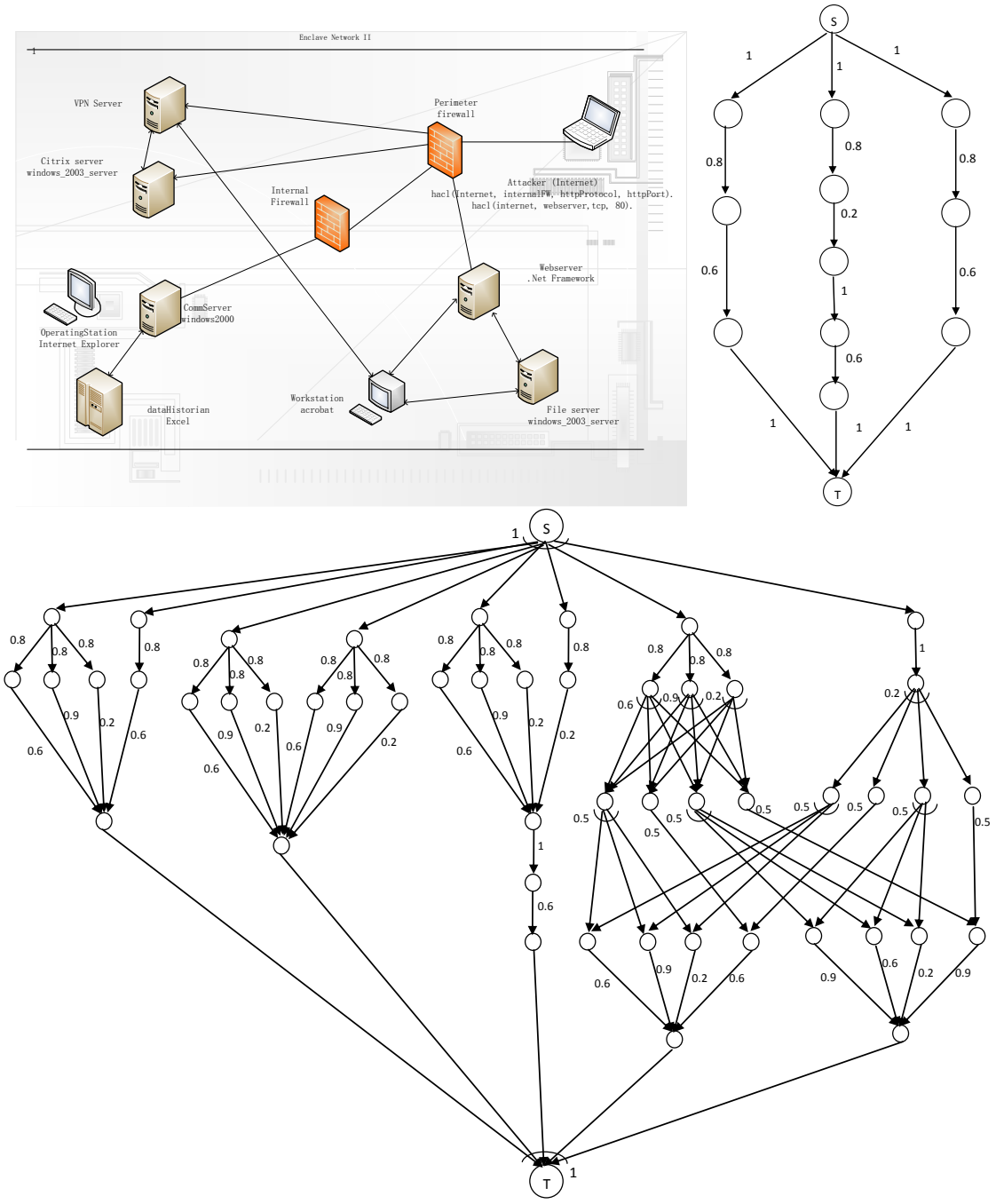

Figure 5.2: Case 2: Heavily-connected network, and corresponding attack graphs with one vulnerability per host (top left), and two vulnerabilities per host (bottom)

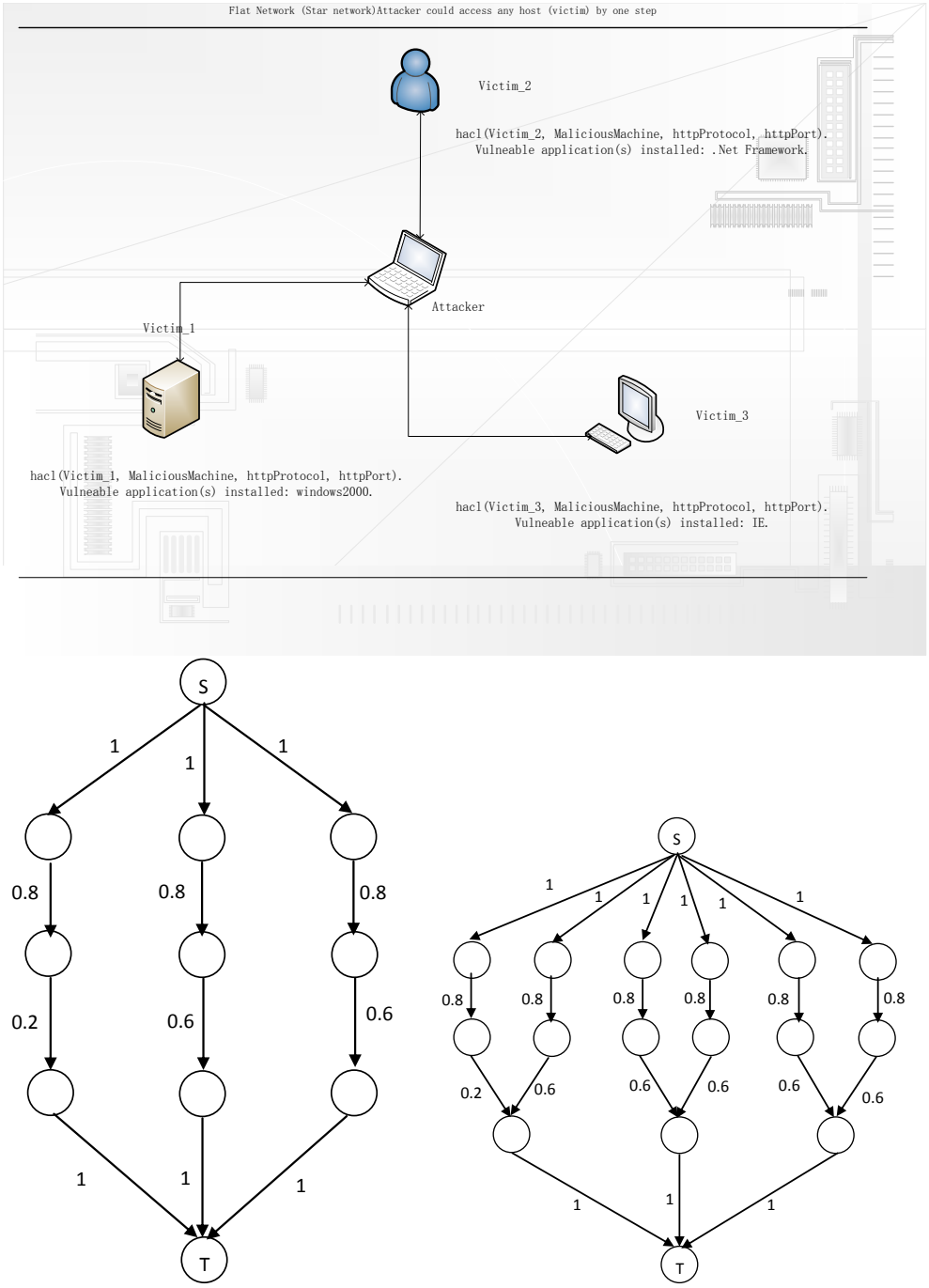

Figure 5.3: Case 3: Small star network, and corresponding attack graphs with one vulnerability per host (bottom left) and two vulnerabilities per host (bottom right)

per host are given in Figures 5.1 through 5.3. For each topology, we begin the analysis with one vulnerability per host, and then add another appropriate vulnerability to each of the nodes in the network. Prof. Ou advised us to add a remote exploitable vulnerability to all hosts, because under many situations, other types of vulnerabilities may not be exploitable.

The attack graphs shown here were produced using the MulVal software [50]. However, the resulting attack graphs include some types of nodes that are not necessary for our assessment. Therefore, in order to simplify the attack graphs, we deleted those nodes. Also, for each attack graph, we added dummy source and terminal nodes (since our model assumes a single source node and terminal node). We connect the source node to the initial nodes of the attack graph with arcs that have success probabilities of 1, and similarly for the terminal node and the final nodes of the attack graph.

We start with a base case that assigns initial success probabilities to the arcs in each attack graph. Initial success probabilities are assigned using the Common Vulnerability Scoring System (CVSS), which is an industry standard for assessing the severity of computer security vulnerabilities [43]. The National Institute of Standards and Technology tracks all emerging cyber attacks and their corresponding CVSS scores in a National Vulnerability Database (NVD) [46]. In our case, the MulVal software automatically assigns success probabilities using the NVD database.

In order to explore the impact of success probability changes on the optimal defense strategy, we also vary the success probabilities of arcs on the attack graphs. We could explore the effects of increasing the success probabilities of arcs, but making arcs more vulnerable does not seem realistic in the context of computer security. Thus, we decreased the success probabilities of arcs on all attack graphs by 50% in our sensitivity analysis.

In addition to changes in the success probabilities of arcs, we also explore the effect of the defense level. We begin by solving each attacker problem with no defenses, and then increased the number of arcs that the defender can interdict, until enough arcs have been interdicted that the attacker can no longer reach the target value.

We consider both myopic and non-myopic attackers, and also vary the number of attack stages. In particular, we consider a one-stage model (for which there is no difference between myopic

and non-myopic attackers), as well as two-stage and three-stage models for both myopic and non-myopic attackers. However, we did not apply the three-stage model to cases 1 and 2 (given in Figures 5.1 and 5.2, respectively) with two vulnerabilities per host, because the attack graphs for those cases were relatively large and computationally demanding.

5.2 Integer L-Shaped Method

To solve the defender's optimization problem, we used the integer L-shaped method. As discussed in Section 3.4, the integer L-shaped method finds the optimal solution of the defender problem by adding optimality cuts to the master problem at each iteration. The number of iterations needed to solve the defender problem is an important measure for assessing the performance of this solution approach, because of the need to solve the attacker problem at each iteration.

For a one-stage attack, we first solve the attacker problem to find the optimal attack path for a given initial defender solution, find a candidate solution of the defender problem using the optimal result of the attacker problem, and then repeat these steps at each iteration until we find the optimal defender solution. For the two-stage myopic model, we first solve the attacker problem to find the optimal one-stage attack path for a given initial defender solution, then run the second-stage recourse model $\bar{N}$ more times to assess the performance of that strategy, solve the master problem using the results of the recourse problem to find a candidate defender solution, and again repeat these steps at each iteration until we find the optimal defender solution. For the two-stage non-myopic model, we first solve the two-stage attacker problem to find the optimal first-stage attack path, then run the second-stage recourse model $\bar{N}$ more times to assess the performance of that strategy, solve the defender problem using the results of the recourse problem to find a candidate defender solution, and then repeat these steps at each iteration until we find the optimal defender solution. The three-stage myopic and and non-myopic attacker problems are solved in a similar manner.

Based on the results of case 1 shown in Table 5.1, the number of iterations needed to solve the defender problem seems to depend on the defender's budget level, and the selected initial defender solution (even though the table does not show the selected initial defender solution). The

results are similar for the other cases. For smaller defender budgets, the integer L-shaped method performed no better then explicit enumeration, since it tried all possible defender solutions. For larger budget levels, the method performed significantly better than explicit enumeration. However, the sensitivity runs for the defender problem were still computationally demanding, especially for the larger attack graphs.

Table 5.1: Number of iterations needed to solve the defender problem using the integer L-shaped method versus using explicit enumeration

	Case 1 - 1 vul/host (12 arcs)		Case 1 - 2 vul/host (52 arcs)	
Budget	L-Shaped Method	Explicit Enumeration	L-Shaped Method	Explicit Enumeration
B=1	12	12	52	52
B=2	144	144	2704	2704
B=3	220	220	2704	22,100
B=4	N/A	N/A	2704	270,725
B=5			22,100	2,598,960
B=6			22,100	20,358,520
B=7			2704	133,784,560
B=8			2704	752,538,150
B=9			2704	3,679,075,400
B=10			2704	15,820,024,220
B=11			2704	60,403,728,840
B=12			52	206,379,406,870
B=13			N/A	N/A

5.3 Sensitivity Analysis Results

As stated at the beginning of this chapter, we use three realistic attack graphs with two vulnerability levels each to assess the sensitivity of the defender solution to attack graph type and size,

attacker type (myopic or not), number of attack stages, and defense level. Figures 5.4 through 5.9 show the results.

Based on the results, we see that the attacker's success probability is generally increasing in the number of attack stages, especially when few arcs are interdicted. As expected, smaller arc success probabilities yield lower overall attack success probabilities, as does an increase in the defender budget level, in general. However, when multiple paths in an attack graph have the same success probability, interdiction of an arc on one of those paths will not reduce the overall probability of success, because the attacker could select an equally good path.

In cases 1 and 2 with one vulnerability per host (Figures 5.4 and 5.6, respectively), and case 3 (Figures 5.8 and 5.9), the myopic attacker performs just as well as the non-myopic attacker. This is because in the corresponding attack graphs, the paths do not involve any common arcs, so there is no benefit to the attacker from thinking ahead.

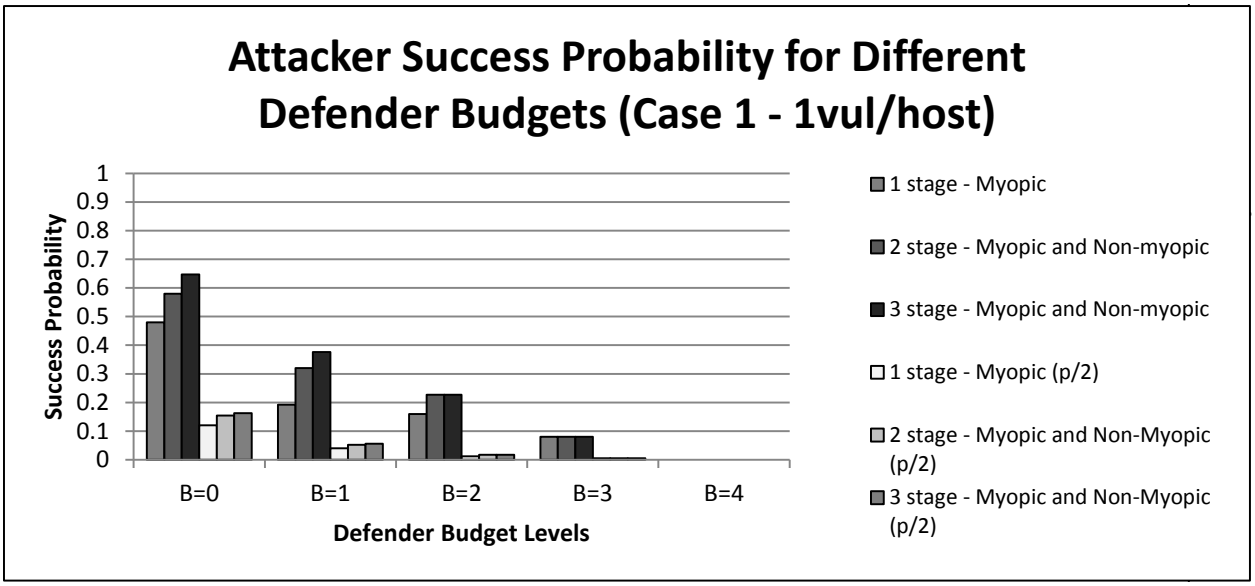

Figure 5.4: Sensitivity analysis results of the defender problem for Case 1 with one vulnerability per host for both with the base and 50% decreased arc success probabilities

In cases 1 and 2 with two vulnerabilities per host (Figures 5.5 and 5.7, respectively), the non-myopic attacker does slightly better than the myopic attacker at some budget levels, because in

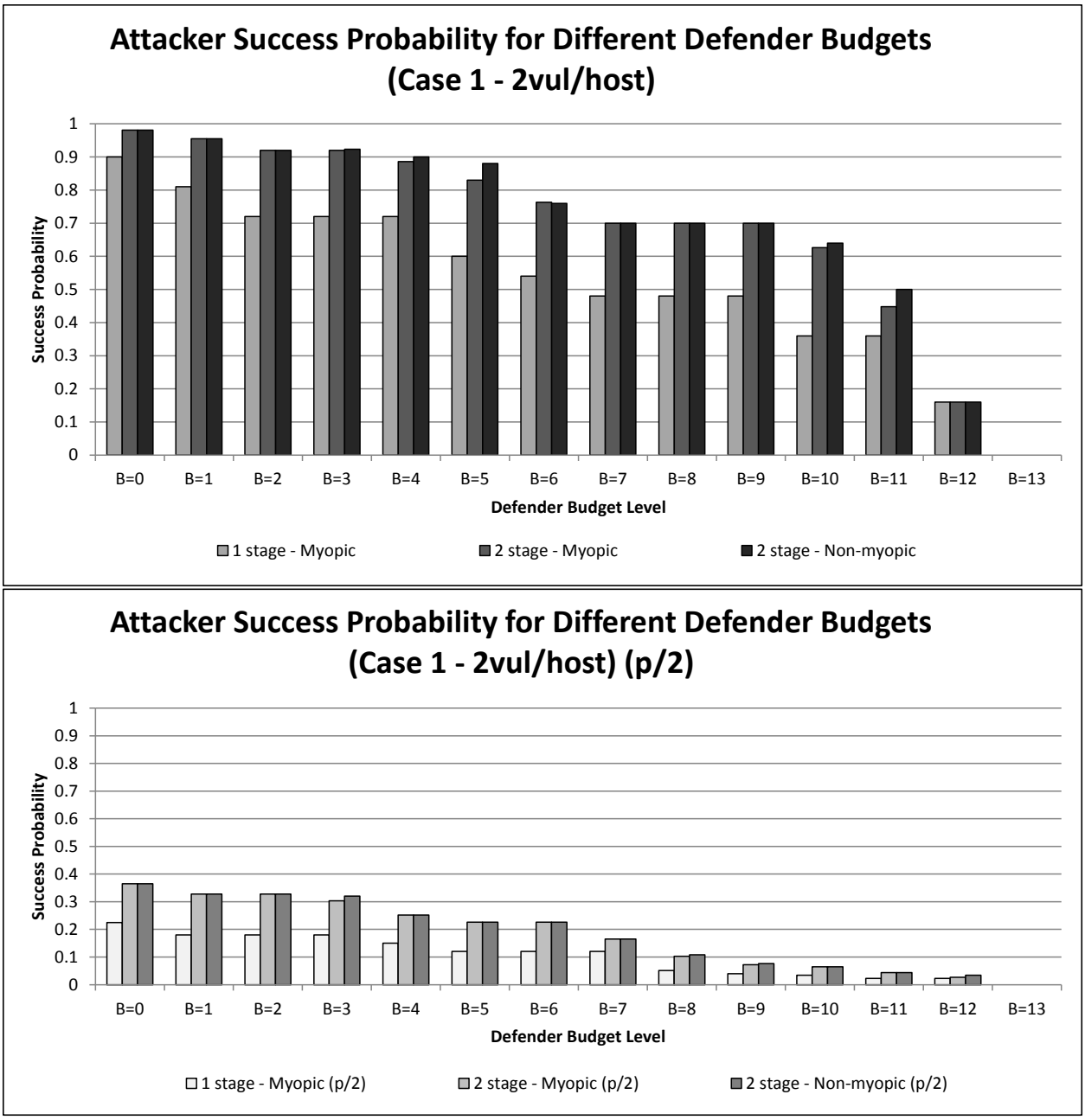

Figure 5.5: Sensitivity analysis results of the defender problem for Case 1 with two vulnerabilities per host; base arc success probabilities (top chart); arc success probabilities decreased by 50% (bottom chart)

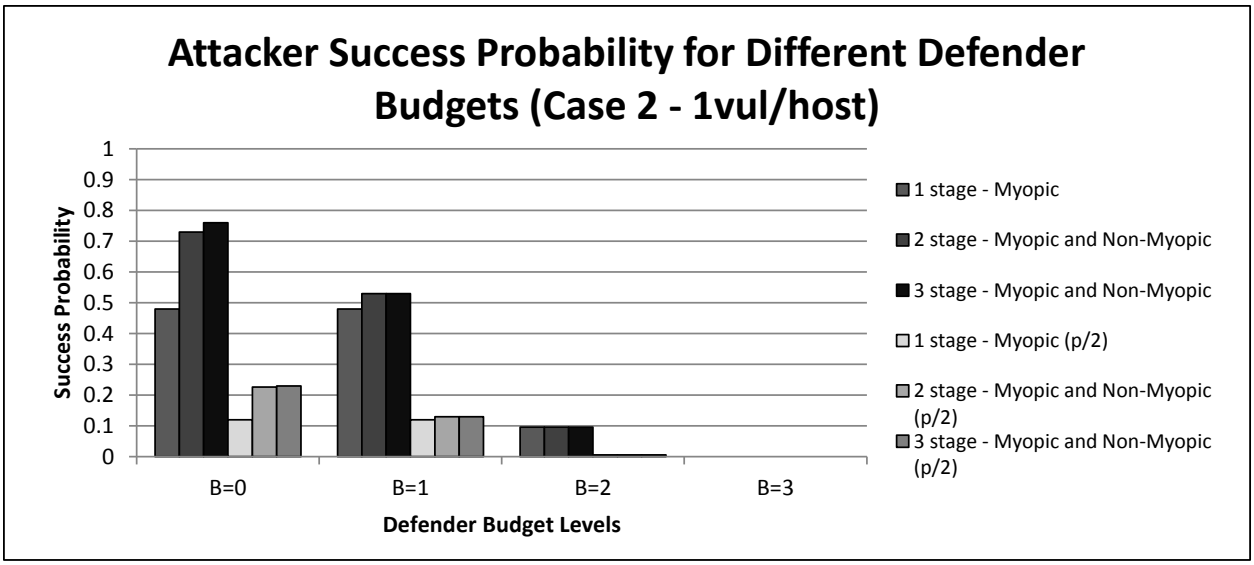

Figure 5.6: Sensitivity analysis results of the defender problem for Case 2 with one vulnerability per host for both the base and 50% decreased arc success probabilities

these attack graphs at least some paths involve common arcs. However, even when the non-myopic attacker performs better than the myopic attacker, the defender's optimal interdiction strategy is the same for both attacker types. Thus, the added computational difficulty of solving the non-myopic attacker problem does not benefit the defender.

To see whether it is sometimes important to consider a non-myopic attacker, we also performed sensitivity analysis on the smaller attack graph from Chapter 4 (given in Figure 4.1). We selected that attack graph because it has a larger number of paths that share common arcs than the other cases considered in this chapter, while also being small enough to complete the sensitivity analysis in a reasonable amount of time.

Figure 5.10 shows the results for this attack graph with arc success probabilities distributed Uniform(0,0.4). We selected this distribution because we observed in Chapter 4 that the non-myopic attacker has a greater advantage over the myopic attacker when the arc success probabilities are relatively small. Based on the results in Figure 5.10, we see that consideration of a non-myopic attacker provides an advantage to the defender when he can protect one arc, since the defense

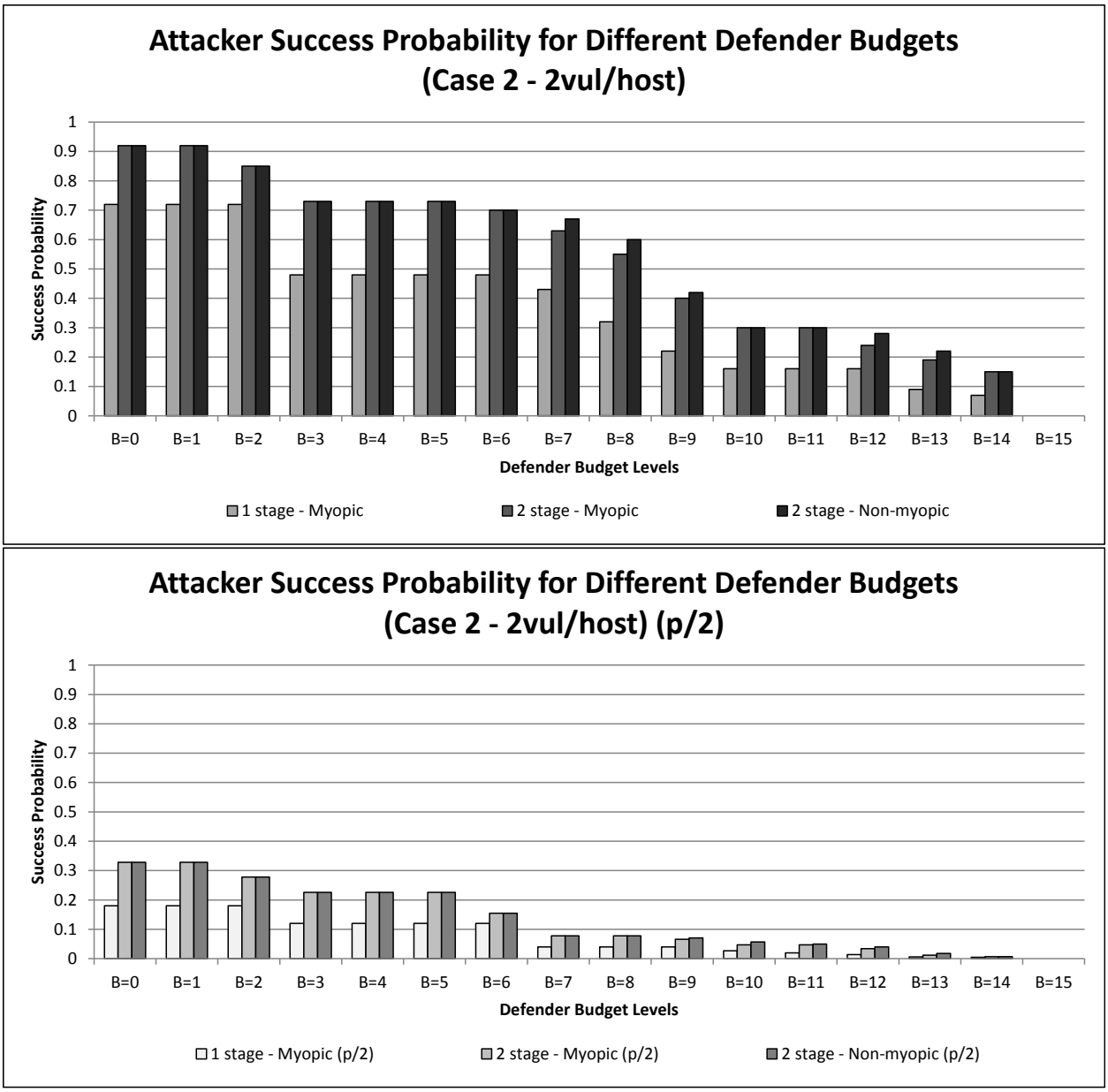

Figure 5.7: Sensitivity analysis results of the defender problem for Case 2 with two vulnerabilities per host; base arc success probabilities (top chart); arc success probabilities decreased by 50% (bottom chart)

strategies are different for the myopic and non-myopic attackers in that case, and relying on the defense against a myopic attacker leads to almost a factor of 2 increase in success probability for

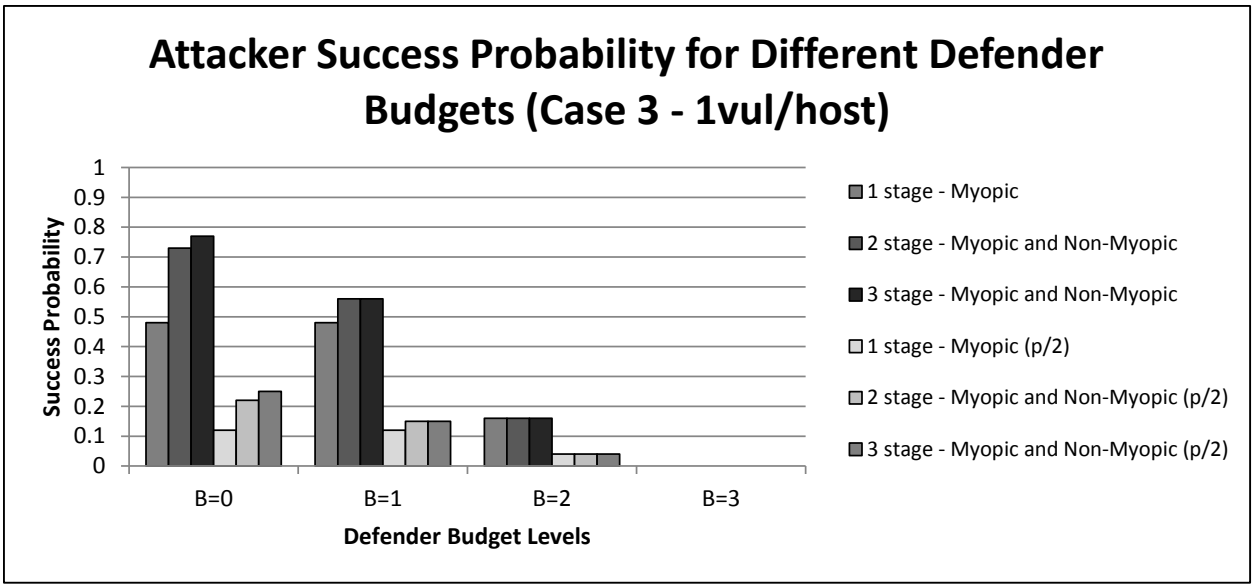

Figure 5.8: Sensitivity analysis results of the defender problem for Case 3 with one vulnerability per host for both the base and 50% decreased arc success probabilities

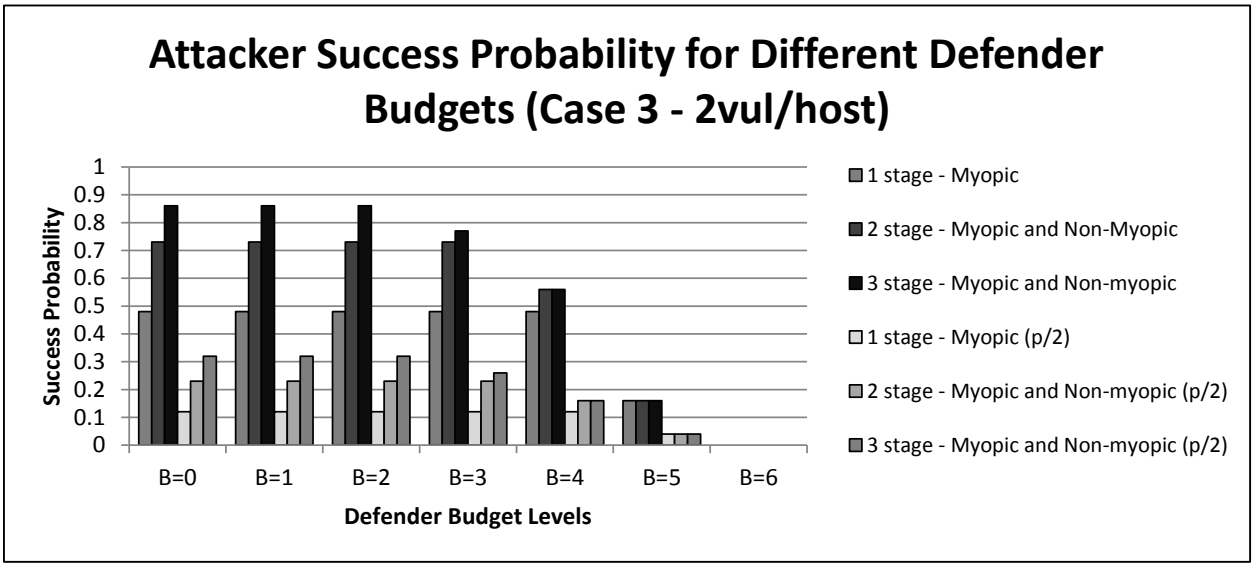

Figure 5.9: Sensitivity analysis results of the defender problem for Case 3 with two vulnerability per host for both the base and 50% decreased arc success probabilities

the attacker. However, even in this case, the advantage of considering a non-myopic defender is not large. Based on these results, we observe that anticipating the attacker's future actions is especially important when the available attack paths share more common arcs.

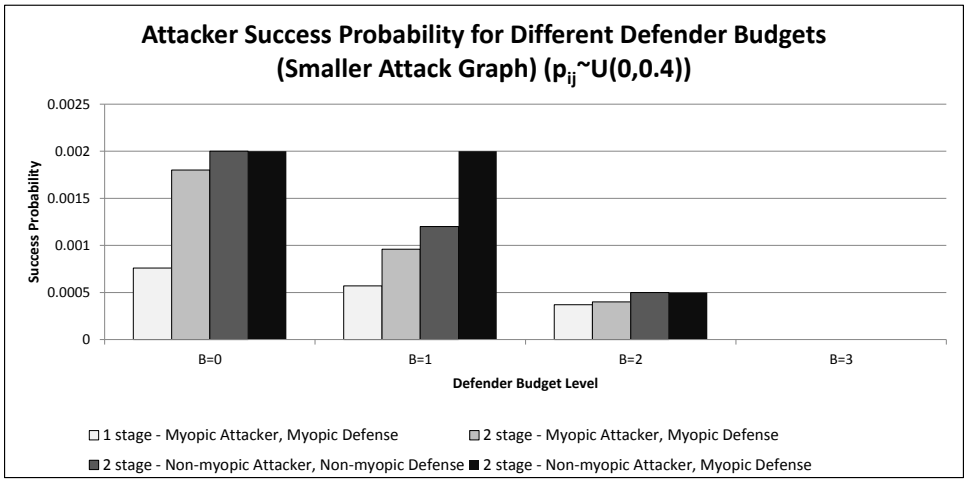

Figure 5.10: Sensitivity analysis results of the defender problem for the smaller attack graph for the arc success probabilities distributed Uniform(0,0.4)

Finally, we observed that the optimal defense is usually to protect arcs near the source node or end node of the attack graph. In our defender sensitivity analysis, we solved 91 defender problems on 7 different attack graphs. In all of these runs, the interdicted arcs were next to either the source node or the end node, never at intermediate levels of the attack graph. For example, Figures 5.11, 5.12 and 5.13 illustrate the interdicted arcs for cases 1 and 2 with two vulnerabilities for a defender budget of 10 arcs, and for the smaller attack graph from Chapter 4 (given in Figure 4.1) with a defender budget of 2 arcs, respectively.

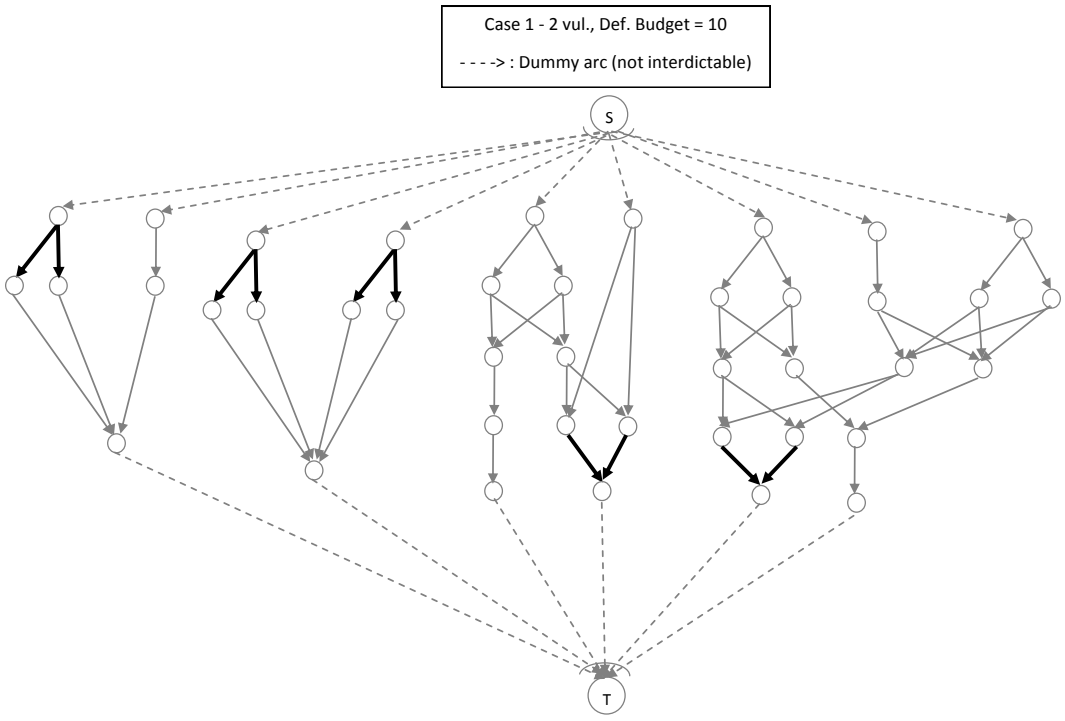

Figure 5.11: Illustration of the optimal defense strategy for case 1 with 2 vulnerabilities for a defender budget of 10 arcs

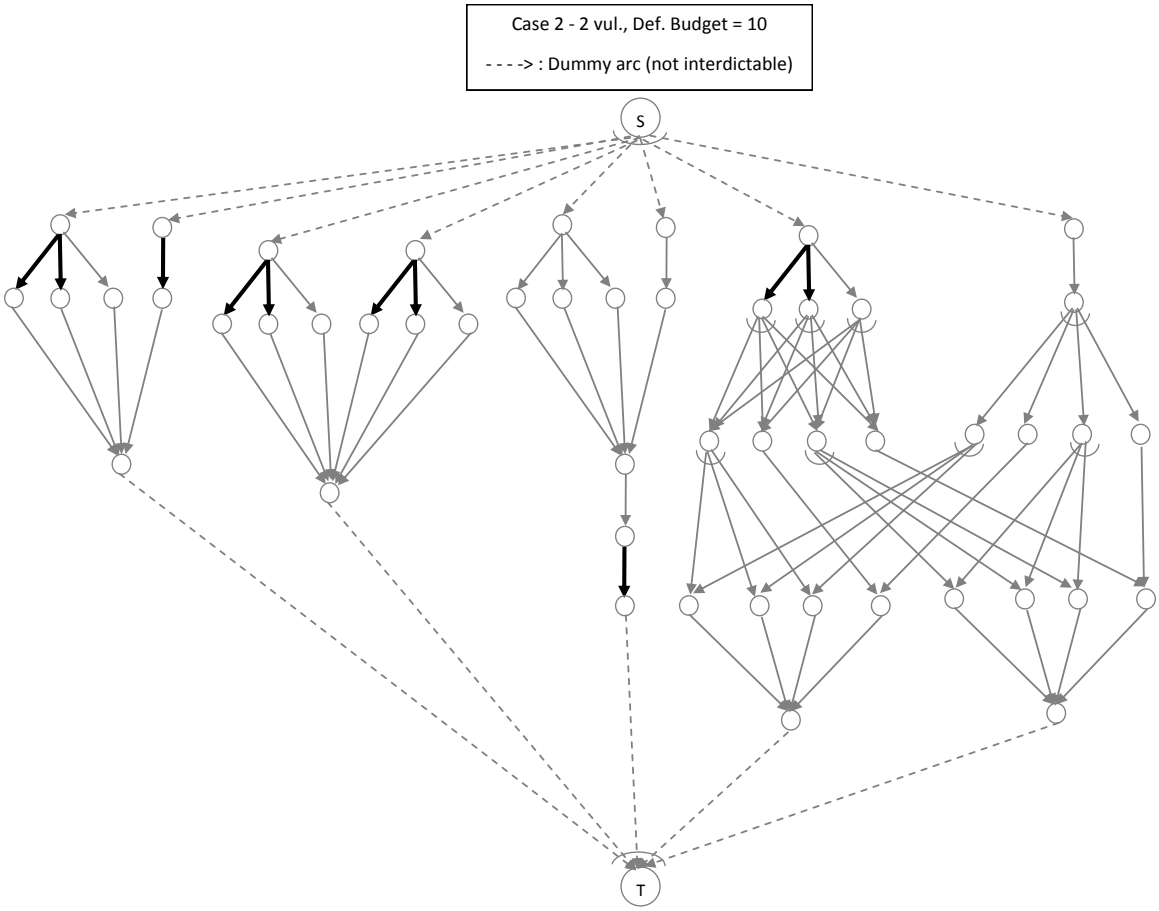

Figure 5.12: Illustration of the optimal defense strategy for case 1 with 2 vulnerabilities for a defender budget of 10 arcs

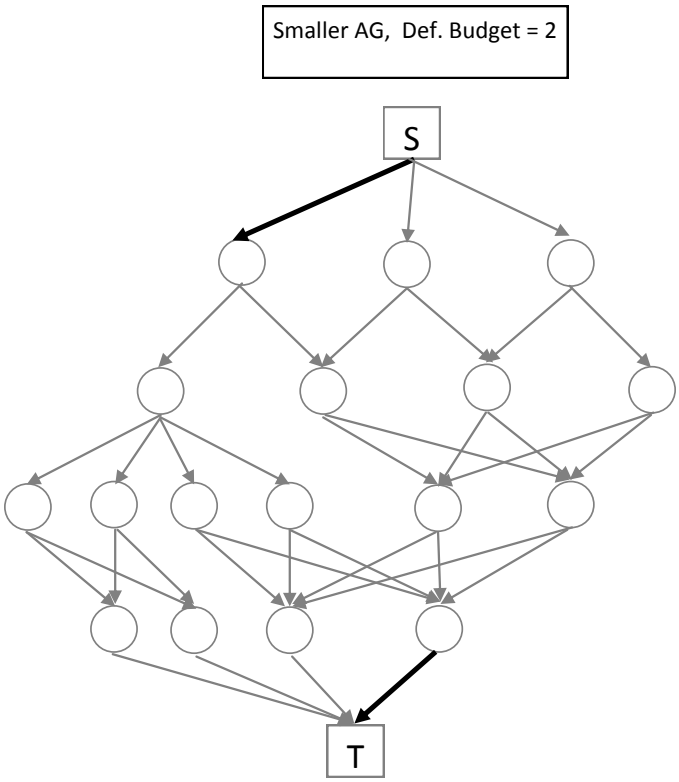

Figure 5.13: Illustration of the optimal defense strategy for the smaller attack graph for a defender budget of 2 arcs

Chapter 6

Conclusion and Future Work

In this research, we developed a general game-theoretic defender-attacker model for security of computer networks, using attack graphs to represent the possible attacker strategies and defender options. To represent the ability of the attacker to launch multiple attempts, we consider the attacker's success or failure on any arc of the attack graph to be probabilistic, and describe the resulting security problem as a multi-stage stochastic network-interdiction problem. In our problem, the defender interdicts a set of arcs, anticipating the attacker's likely actions, and then the attacker can make multiple attempts to traverse the network.

We formulated the resulting problem as a stochastic bilevel mixed-integer program with a "min-max" objective. Here, the defender attempts to minimize the attacker's success probability, and the attacker maximizes the probability of traversing the network successfully in multiple attempts.

Many defender-attacker models have been developed for security of computer networks. These models mostly use game theory to identify equilibrium strategies for both the attacker and the defender. In the literature, there are also models using attack graphs to assess the security of computer networks, but most of them are not game-theoretic. Our model is novel in the sense that it is both game-theoretic (the defender makes his decision, anticipating the attacker's likely actions) and uses attack graphs to represent the vulnerabilities of a computer network in a compact form [63].

We were able to solve the attacker's problem for both two and three attack stages, using the SAA approach. Of course, in the real world, an intelligent attacker may be able to launch many more than three attempted attacks. In principle, stochastic programming allows us to formulate models with multiple stages of uncertainty resolution. However, solution of problems with large

numbers of stages is difficult. In our study, even the solution of the three-stage attacker problem was computationally demanding. This limits the applicability of our method in practice.

However, from the literature ([24], [74]), we know that it is sometimes possible to represent a large network with a relatively small attack graph by representing similar types of hosts with similar or identical vulnerabilities as if they were a single host. In that case, it might be possible to solve the attacker problem efficiently for large networks. Also, as discussed in Chapter 2, dynamic programming could be used to solve the attacker problem, and may be more effective than stochastic programming when the number of attack stages is large. This is because adding more stages to the problem increases the size of a dynamic program less than in a stochastic program.

We compared the attacker's and defender's optimal strategies for two versus three attack stages. The results indicate that due to the ability to learn from earlier attack stages, the benefit of increasing the number of attacks by a factor of 1.5 (from 2 to 3) can yield much more than a factor of 1.5 increase in the overall probability of a successful attack, especially when the mean success probability is small. This suggests that it is likely to be important to model more than three attack stages, especially for well-defended systems. Fortunately, though, run times are also much less when the arc success probabilities are small, so this may not be a problem. When the mean of the arc success probabilities is large, the three-stage model appears to be less beneficial. This might be because the attacker is already likely to succeed on the first or second stage. Thus, a third attack stage is more beneficial to the attacker when the mean of the arc success probabilities is small. For a given mean success probability, the advantage of having three attack stages is greater when the variability of the success probabilities from one arc to another is narrower. This is reasonable, because when the variability is great, some paths are much better than others, so the best first-stage path is likely to be much better than a possible third-stage path. However, for narrower distributions, the first, second, and third-stage paths may all be roughly comparable, so that a third attack stage would give more benefit.

We also compared the myopic and non-myopic attacker cases. The results indicate, as expected, that non-myopic attackers perform better than myopic attackers. This difference is especially great when the success probabilities are low, perhaps because learning about non-promising paths is

more important in that case. Moreover, for a given mean success probability, the non-myopic attacker has a greater advantage over the myopic attacker when the variability of the arc success probabilities is large. This is reasonable, because with wider distributions, some paths are much better than others, so taking future actions into account (e.g., by exploration to discover promising paths) may be especially beneficial in that case.

For the defender problem, we found, as expected, that the attacker's overall success probability is decreasing in the level of defense. More significantly, we found that for the relatively small attack graphs studied in our sensitivity analysis, the optimal defense against a myopic attacker often the same as against a non-myopic attacker, and in any case rarely does too much worse. However, preliminary results suggest that this might not be the case when the available attack paths share numerous common arcs.

Overall, we were able to determine the optimal defense against a non-myopic attacker. Of course, our method may not be directly applicable in practice, because the optimal defense is computationally intensive to find, and in many cases, is not significantly better than the defense against a myopic attacker. Cases where the benefit of defense against a non-myopic attacker is large may tend to involve large networks, and may therefore be even more difficult to solve. Thus, while the goal of defending against a non-myopic attacker is significant, the methods we have used so far limit the applicability of our model. Thus, in future work, we plan to look for modeling and solution approaches that would help to solve our problem more efficiently even for large networks, such as dynamic programming.

Despite these limitations, however, our results nonetheless yielded some recommendations of practical significance for defense of cyber networks. First, for networks that are not well defended (i.e., highly vulnerable), defending against a myopic attacker may be adequate; the advantage of assuming a non-myopic attacker is greatest for well-defended networks. Similarly, defending against a myopic attacker may be adequate when the available attack paths share few common arcs, and when the success probabilities of attacks on different arcs vary widely. Thus, while the problem of defending against non-myopic attackers is difficult, it is encouraging to know that defense against a myopic attacker will frequently give results that are equally or almost equally

good. Finally, our results suggest that the optimal defense will usually be to protect arcs near the source node or end node of the attack graph, rather than at intermediate layers of the attack graph.

Bibliography

[1] Ahmed, S., A. Shapiro, E. Shapiro. 2002. The sample average approximation method for stochastic programs with integer recourse. *Preprint available at www.optimization-online.org*.

[2] Alazzawe, A., A. Nawaz, M. M. Bayaraktar. 2006. Game theory and intrusion detection systems. http://www.qatar.cmu.edu/iliano/courses/06s-gmu-isa767/project/papers/alazzawe-mehmet-nawaz.pdf .

[3] Alpcan, T., T. Basar. 2003. A game theoretic approach to decision and analysis in network intrusion detection. *Proceedings of the 42nd IEEE Conference on Decision and Control*, vol. 3. 2595–2600.

[4] Alpcan, T., T. Basar. 2006. An intrusion detection game with limited observations. *12th Int. Symp. on Dynamic Games and Applications*. Sophia Antipolis, France.

[5] Bayrak, H., M.D. Bailey. 2008. Shortest path network interdiction with asymmetric information. *Networks* **52**(3) 133–140.

[6] Bertsimas, D.J., P. Jaillet, A.R. Odoni. 1990. A priori optimization. *Operations Research* **38**(6) 1019–1033.

[7] Bier, V.M., L.A. Cox Jr., M.N. Azaiez. 2009. chap. 1. Why both game theory and reliability theory are important in defending infrasutructure against intelligent attacks. In V.M. Bier and M.N. Azaiez, editors, *Game Theoretic Risk Analysis of Security Threats*. Springer, 1–11.

[8] Bursztein, E., J. Mitchell. 2011. Using strategy objectives for network security analysis. *Proceedings of the 4th International Conferences on Information Security and Cryptology (INSCRYPT), Beijing, China*. Springer, 337–349.

[9] Carin, L., G. Cybenko, J. Hughes. 2008. Cybersecurity strategies: The queries methodology. *IEEE Computer* **41**(8) 20–26.

[10] Cormican, K.J. 1995. Computational methods for deterministic and stochastic network interdiction problems. Master's thesis, Naval Postgraduate School, Monterey, CA.

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

[12] Creemers, S., R. Leus, M. Lambrecht. 2010. Scheduling markovian PERT networks to maximize the net present value. *Operations Research Letters* **38**(1) 51–56.

[13] Danforth, M. 2008. Scalable patch management using evolutionary analysis of attack graphs. *Seventh International Conference on Machine Learning and Applications, 2008. ICMLA'08*. IEEE, 300–307.

[14] Dimitrov, N.B., D.P. Morton. 2013. Interdiction models and applications. *Handbook of Operations Research for Homeland Security, Herrmann, J.W., Editor*. Springer, 73–103.

[15] Frigault, M., L. Wang. 2008. Measuring network security using bayesian network-based attack graphs. *32nd Annual IEEE International Computer Software and Applications, 2008. COMPSAC'08*. IEEE, 698–703.

[16] Fulkerson, D.R., G.C. Harding. 1977. Maximizing the minimum source-sink path subject to a budget constraint. *Mathematical Programming* **13**(1) 116–118.

[17] Goel, V., I.E. Grossmann. 2004. A stochastic programming approach to planning of offshore gas field developments under uncertainty in reserves. *Computers & Chemical Engineering* **28**(8) 1409–1429.

[18] Goel, V., I.E. Grossmann. 2006. A class of stochastic programs with decision dependent uncertainty. *Mathematical programming* **108**(2-3) 355–394.

[19] Golden, B. 1978. A problem in network interdiction. *Naval Research Logistics Quarterly* **25**(4) 711–713.

[20] Gümüş, Z.H., C.A. Floudas. 2005. Global optimization of mixed-integer bilevel programming problems. *Computational Management Science* **2**(3) 181–212.

[21] Gutin, E. 2012. Dynamic interdiction games on PERT networks. Master's thesis, Imperial College.

[22] Held, H., R. Hemmecke, D.L. Woodruff. 2005. A decomposition algorithm applied to planning the interdiction of stochastic networks. *Naval Research Logistics* **52**(4) 321–328.

[23] Hindelang, T.J., J.F. Muth. 1979. A dynamic programming algorithm for decision cpm networks. *Operations Research* **27**(2) 225–241.

[24] Homer, J., A. Varikuti, X. Ou, M.A. McQueen. 2008. Improving attack graph visualization through data reduction and attack grouping. *In The 5th International Workshop on Visualization for Cyber Security (VizSEC)*. Springer, 68–79.

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

[26] Jajodia, S., S. Noel, B. O'Berry. 2005. chap. 5. Topological analysis of network attack vulnerability. In V. Kumar, J. Srivastava, and A. Lazarevic, editors, *Managing Cyber Threats: Issues, Approaches and Challanges*. Springer, 247–266.

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

[28] Jonsbråten, T.W., R.J.B. Wets, D.L. Woodruff. 1998. A class of stochastic programs with decision dependent random elements. *Annals of Operations Research* **82** 83–106.

[29] Jormakka, J., J.V.E. Mölsä. 2005. Modelling information warfare as a game. *Journal of Information Warfare* **4**(2) 12–25.

[30] Kall, P., S. Wallace. 1994. *Stochastic Programming*. Wiley.

[31] Lageweg, B.J., J.K. Lenstra, A.H.G. Rinnooy Kan, L. Stougie. 1985. Stochastic integer programming by dynamic programming. *Statistica Neerlandica* **39**(2) 97–113.

[32] Laporte, G., F. Louveaux, H. Mercure. 1992. The vehicle routing problem with stochastic travel times. *Transportation Science* **26**(3) 161–170.

[33] Laporte, G., F.V. Louveaux. 1993. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters* **13**(3) 133–142.

[34] Linderoth, J., A. Shapiro, S. Wright. 2006. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research* **142**(1) 215–241.

[35] Lippmann, R.P., K.W. Ingols. 2005. An annotated review of past papers on attack graphs. Tech. Rep. No. PR-IA-1, MIT Lincoln Lab, Lexington, MA.

[36] Lippmann, R.P., K.W. Ingols, C. Scott, K. Piwowarski, K.J. Kratkiewicz, M. Artz, R.K. Cunningham. 2005. Evaluating and strengthening enterprise network security using attack graphs. Tech. Rep. ESC-TR-2005-064, MIT Lincoln Laboratory, Lexington, MA.

[37] Lippmann, R.P., K.W. Ingols, C. Scott, K. Piwowarski, K.J. Kratkiewicz, M. Artz, R.K. Cunningham. 2006. Validating and restoring defense in depth using attack graphs. *Military Communications Conference, MILCOM 2006*. IEEE, 1–10.

[38] Liu, P., W. Zang, M. Yu. 2005. Incentive-based modeling and inference of attacker intent, objectives, and strategies. *ACM Transactions on Information and System Security (TISSEC)* **8**(1) 78–118.

[39] Liu, Y., C. Comaniciu, H. Man. 2006. A Bayesian game approach for intrusion detection in wireless ad hoc networks. *Proceedings of the Workshop on Game Theory for Networks (GameNets), Pisa, Italy, 2006*. ACM.

[40] Lye, K., J.M. Wing. 2005. Game strategies in network security. *International Journal of Information Security* **4**(1) 71–86.

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

[42] Mehta, V., C. Bartzis, H. Zhu, E. Clarke, J. Wing. 2006. Ranking attack graphs. *Proceedings of the 9th International Conference on Recent Advances in Intrusion Detection*. Springer, 127–144.

[43] Mell, P., K. Scarfone, S. Romanosky. 2006. Common vulnerability scoring system. *Security & Privacy, IEEE* **4**(6) 85–89.

[44] Morton, D.P. 2011. Stochastic network interdiction. *Wiley Encyclopedia of Operations Research and Management Science, Cochran, J. J., Editor. John Wiley & Sons, Inc* .

[45] Nguyen, K.C., T. Alpcan, T. Basar. 2009. Stochastic games for security in networks with interdependent nodes. *International Conference on Game Theory for Networks, 2009*. IEEE, 697–703.

[46] NIST. 2013. National Vulneability Database. URL http://nvd.nist.gov/.

[47] Noel, S., S. Jajodia. 2008. Optimal IDS sensor placement and alert prioritization using attack graphs. *Journal of Network and Systems Management* **16**(3) 259–275.

[48] Osborne, M.J., A. Rubinstein. 1994. *A Course in Game Theory*. MIT Press.

[49] Ou, X., W.F. Boyer, M.A. McQueen. 2006. A scalable approach to attack graph generation. *Proceedings of the 13th ACM Conference on Computer and Communications Security*. ACM, 336–345.

[50] Ou, X., S. Govindavajhala, A.W. Appel. 2005. Mulval: A logic-based network security analyzer. *14th USENIX Security Symposium*. 1–16.

[51] Pan, F., D.P. Morton. 2008. Minimizing a stochastic maximum-reliability path. *Networks* **52**(3) 111–119.

[52] Pflug, G.C. 1990. Online optimization of simulated Markovian processes. *Mathematics of Operations Research* **15**(3) 381–395.

[53] Phillips, C.A. 1992. The network destruction problem. Tech. Rep. No. SAND-92-0186C, Sandia National Labs., Albuquerque, NM.

[54] Psaraftis, H.N. 1984. On the practical importance of asymptotic optimality in certain heuristic algorithms. *Networks* **14**(4) 587–596.

[55] Saha, D. 2008. Extending logical attack graphs for efficient vulnerability analysis. *Proceedings of the 15th ACM Conference on Computer and Communications Security*. ACM, 63–74.

[56] Santoso, T., S. Ahmed, M. Goetschalckx, A. Shapiro. 2005. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research* **167**(1) 96–115.

[57] Sawilla, R., X. Ou. 2008. Identifying critical attack assets in dependency attack graphs. *Proceedings of the 13th European Symposium on Research in Computer Security (ESORICS), Malaga, Spain*.

[58] Schneier, B. 1999. Attack trees. *Dr. Dobbs Journal* **24**(12) 21–29.

[59] Schneier, B. 2000. *Secrets & Lies: Digital Security in a Networked World*. 2nd ed. Wiley.

[60] Shapiro, A. 2003. Monte Carlo sampling methods. *Handbook in Operations Research and Management Science, Shapiro, A. and Ruszczynski, A., editors., North-Holland, Amsterdam, 2003* **10** 353–425.

[61] Shapiro, A., D. Dentcheva, A.P. Ruszczyński. 2009. *Lectures on Stochastic Programming: Modeling and Theory*, vol. 9. Society for Industrial Mathematics.

[62] Shapiro, A., H. Xu. 2008. Stochastic mathematical programs with equilibrium constraints, modelling and sample average approximation. *Optimization* **57**(3) 395–418.

[63] Sheyner, O., J. Haines, S. Jha, R. Lippmann, J.M. Wing. 2002. Automated generation and analysis of attack graphs. *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 273–284.

[64] Steinrauf, R.L. 1991. Network interdiction models. Master's thesis, Naval Postgraduate School, Monterey, CA.

[65] Swiler, L.P., C. Phillips, D. Ellis, S. Chakerian. 2001. Computer-attack graph generation tool. *Proceedings of the DARPA Information Survivability Conference and Exposition*, vol. 2. IEEE, 307–321.

[66] US-CERT. 2012. US-Computer Emergency Readiness Team. URL http://www.us-cert.gov/.

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

[68] Vejandla, P., D. Dasgupta, A. Kaushal, F. Nino. 2010. Evolving gaming strategies for attacker-defender in a simulated network environment. *2010 IEEE International Conference on Privacy, Security, Risk and Trust*. 889–896.

[69] Verweij, B., S. Ahmed, A.J. Kleywegt, G. Nemhauser, A. Shapiro. 2003. The sample average approximation method applied to stochastic routing problems: A computational study. *Computational Optimization and Applications* **24**(2) 289–333.

[70] Wang, L., S. Noel, S. Jajodia. 2006. Minimum-cost network hardening using attack graphs. *Computer Communications* **29**(18) 3812–3824.

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

[72] Xiaolin, C., T. Xiaobin, Z. Yong, X. Hongsheng. 2008. A Markov game theory-based risk assessment model for network information system. *International Conference on Computer Science and Software Engineering*, vol. 3. IEEE, 1057–1061.

[73] Xie, A., Z. Cai, C. Tang, J. Hu, Z. Chen. 2009. Evaluating network security with two-layer attack graphs. *Annual Computer Security Applications Conference, 2009. ACSAC'09*. IEEE, 127–136.

[74] Zhang, S., X. Ou, J. Homer. 2011. Effective network vulnerability assessment through model abstraction. *Proceeding of the 8th International Conference on Detection of Intrusions and Malware & Vulnerability Assessment*. Springer, 17–34.

[75] Zhang, S., X. Ou, A. Singhal, J. Homer. 2011. An empirical study of a vulnerability metric aggregation method. *International Conference on Security and Management (SAM'11), special track on Mission Assurance and Critical Infrastructure Protection (STMACIP'11), Las Vegas, USA*.