Network interdiction models and algorithms for information security

By

Apurba Kumer Nandi

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Industrial and Systems Engineering
in the Department of Industrial and Systems Engineering

Mississippi State, Mississippi

December 2016
Network interdiction models and algorithms for information security

By

Apurba Kumer Nandi

Approved:

Hugh R. Medal  Mohammad Sepehrifar
(Major Professor)  (Minor Professor)

Merril Warkentin
(Committee Member)

Mahantesh Halappanavar
(Committee Member)

Sandra D. Eksioglu
(Committee Member)

Linkan Bian
(Committee Member)

Stanley F. Bullington
(Graduate Coordinator)

Jason M. Keith
Dean
Bagley College of Engineering
Major cyber attacks against the cyber networks of organizations has become a common phenomenon nowadays. Cyber attacks are carried out both through the spread of malware and also through multi-stage attacks known as hacking. A cyber network can be represented directly as a simple directed or undirected network (graph) of nodes and arcs. It can also be represented by a transformed network such as the attack graph which uses information about network topology, attacker profile, and existing vulnerabilities to represent all the potential attack paths from readily accessible vulnerabilities to valuable target nodes. Then, interdicting or hardening a subset of arcs in the network naturally maps into deploying security countermeasures on the associated devices or connections. In this dissertation, we develop network interdiction models and algorithms to optimally select a subset of arcs which upon interdiction minimizes the spread of infection or minimizes the loss from multi-stage attacks. In particular, we define four novel network connectivity-based metrics and develop interdiction models to optimize the metrics. Direct network representation of
the physical cyber network is used as the underlying network in this case. Two of the interdiction models prove to be very effective arc removal methods for minimizing the spread of infection. We also develop multi-level network interdiction models that remove a subset of arcs to minimize the loss from multi-stage attacks. Our models capture the defender-attacker interaction in terms of stackelberg zero-sum games considering the attacker both as a complete rational and bounded rational agents. Our novel solution algorithms based on constraint and column generation and enhanced by heuristic methods efficiently solve the difficult multi-level mixed-integer programs with integer variables in all levels in reasonable times.

Key words: Network interdiction, cyber security, attack graph, constraint and column generation, bi-level program, mixed integer linear programming
DEDICATION

To my mother and late father.
ACKNOWLEDGEMENTS

I am grateful to my advisor for all that he has done for me. I am also grateful to all the other members of my dissertation committee. I am grateful to my wife for tolerating the indisciplined and sometimes stressful life during the course of my PhD. I cannot express how much I am indebted to my parents. They did not have anything, but they have done everything for me.
# TABLE OF CONTENTS

DEDICATION ........................................................................................................... ii

ACKNOWLEDGEMENTS ......................................................................................... iii

LIST OF TABLES ......................................................................................................... vii

LIST OF FIGURES ....................................................................................................... viii

CHAPTER

1. INTRODUCTION ................................................................................................. 1

2. METHODS FOR REMOVING LINKS IN A NETWORK TO MINIMIZE THE SPREAD OF INFECTIONS ................................................................. 16

   2.1 Introduction .................................................................................................... 16
   2.2 Problem Description ....................................................................................... 25
   2.3 Model Formulations ....................................................................................... 27
      2.3.1 Definition 1 (Transmission path) ............................................................. 27
      2.3.2 Definition 2 (Connection) ....................................................................... 28
      2.3.3 Definition 3 (Susceptible node at risk of infection) ............................... 28
      2.3.4 MINCONNECT Model ........................................................................... 28
         2.3.4.1 Lemma 1 (Implied lower bound) ....................................................... 30
         2.3.4.2 Case 1 .............................................................................................. 30
         2.3.4.3 Case 2 .............................................................................................. 30
         2.3.4.4 Corollary 1 ...................................................................................... 30
      2.3.5 MINATRISK Model .............................................................................. 31
         2.3.5.1 Lemma 1 .......................................................................................... 33
         2.3.5.2 Case 1 .............................................................................................. 33
         2.3.5.3 Case 2 .............................................................................................. 34
      2.3.6 MINPATHS Model .................................................................................. 36
      2.3.7 MINWPATHS Model .............................................................................. 37
   2.4 Solution Algorithms ....................................................................................... 38
      2.4.1 Heuristic algorithm for the MINCONNECT Model ............................... 39
      2.4.2 Heuristic algorithm for the MINATRISK Model ................................. 40
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.2</td>
<td>Lower Bound</td>
<td>120</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Algorithm MINBMAX</td>
<td>123</td>
</tr>
<tr>
<td>4.4</td>
<td>Results and Discussion</td>
<td>124</td>
</tr>
<tr>
<td>4.5</td>
<td>Conclusion</td>
<td>131</td>
</tr>
<tr>
<td>5.</td>
<td>CONCLUSION AND FUTURE RESEARCH</td>
<td>133</td>
</tr>
<tr>
<td>5.1</td>
<td>Conclusion</td>
<td>133</td>
</tr>
<tr>
<td>5.2</td>
<td>Publications</td>
<td>137</td>
</tr>
<tr>
<td>5.3</td>
<td>Future Research</td>
<td>137</td>
</tr>
</tbody>
</table>

REFERENCES                                                                 139
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Parameters and their values in the experiments</td>
<td>52</td>
</tr>
<tr>
<td>3.1</td>
<td>Parameters</td>
<td>77</td>
</tr>
<tr>
<td>3.2</td>
<td>Variables</td>
<td>78</td>
</tr>
<tr>
<td>3.3</td>
<td>Parameters and their values in the experiments.</td>
<td>98</td>
</tr>
<tr>
<td>3.4</td>
<td>Growth of computation time with graph size.</td>
<td>99</td>
</tr>
<tr>
<td>3.5</td>
<td>Performance of the heuristic method.</td>
<td>100</td>
</tr>
<tr>
<td>3.6</td>
<td>Performance of the MAXBREACHBM and MAXBREACHD formulations.</td>
<td>101</td>
</tr>
<tr>
<td>3.7</td>
<td>Computation times (clock seconds) of the algorithm.</td>
<td>103</td>
</tr>
<tr>
<td>3.8</td>
<td>Sensitivity of total breach loss to parameter uncertainty.</td>
<td>109</td>
</tr>
<tr>
<td>4.1</td>
<td>Parameters</td>
<td>117</td>
</tr>
<tr>
<td>4.2</td>
<td>Variables</td>
<td>127</td>
</tr>
<tr>
<td>4.3</td>
<td>Computation times with and without bounded rationality.</td>
<td>125</td>
</tr>
<tr>
<td>4.4</td>
<td>Examples with and without the impact of bounded rationality.</td>
<td>130</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

1.1 Local Area Network (LAN) ................................................. 1
1.2 Wide Area Network (WAN) ................................................. 2
1.3 Undirected graph ............................................................. 5
1.4 Directed graph ............................................................... 6
1.5 Spread of infection on a graph .............................................. 7
1.6 Directed attack graph ......................................................... 7
2.1 Solution to MINATRISK-Z without the binary constraint \( z \in \{0, 1\} \) \((b = 1)\) ............................................................ 35
2.2 Solution to MINATRISK-Z \((b = 1)\) ............................................. 35
2.3 Solution to MINATRISK-Z \((b = 4)\) ............................................. 36
2.4 Computational performance of the MINCONNECT model .................. 42
2.5 Performance of the MINATRISK and MINATRISK-Z formulations ............ 44
2.6 Computational performance of the MINATRISK-Z model .................... 46
2.7 Fraction of solutions that are optimal ....................................... 47
2.8 Average optimality gap of the algorithms ................................... 48
2.9 Computational performance of the heuristic algorithms ...................... 48
2.10 Effectiveness of the methods in minimizing spread 1 ......................... 55
2.11 Effectiveness of the methods in minimizing spread 2 ......................... 56
2.12 Effectiveness of the methods in minimizing spread 3 ......................... 56
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.13</td>
<td>Effectiveness of the methods in minimizing spread 4</td>
<td>57</td>
</tr>
<tr>
<td>2.14</td>
<td>Effectiveness of the methods in slowing down spread 1</td>
<td>58</td>
</tr>
<tr>
<td>2.15</td>
<td>Effectiveness of the methods in slowing down spread 2</td>
<td>58</td>
</tr>
<tr>
<td>2.16</td>
<td>Effectiveness of the methods in slowing down spread 3</td>
<td>59</td>
</tr>
<tr>
<td>2.17</td>
<td>Effectiveness of the methods in slowing down spread 4</td>
<td>59</td>
</tr>
<tr>
<td>2.18</td>
<td>Effectiveness of the methods in minimizing spread 5</td>
<td>61</td>
</tr>
<tr>
<td>2.19</td>
<td>Effectiveness of the methods in minimizing spread 6</td>
<td>61</td>
</tr>
<tr>
<td>2.20</td>
<td>Effectiveness of the methods in slowing down spread 5</td>
<td>62</td>
</tr>
<tr>
<td>2.21</td>
<td>Effectiveness of the methods in slowing down spread 6</td>
<td>62</td>
</tr>
<tr>
<td>3.1</td>
<td>Example attack graph</td>
<td>77</td>
</tr>
<tr>
<td>3.2</td>
<td>Bipartite attack graph</td>
<td>83</td>
</tr>
<tr>
<td>3.3</td>
<td>Iteration 1: Attacker’s solution, Upper bound = 35</td>
<td>90</td>
</tr>
<tr>
<td>3.4</td>
<td>Iteration 1: Defender’s solution, Lower bound = 0</td>
<td>90</td>
</tr>
<tr>
<td>3.5</td>
<td>Iteration 2: Attacker’s solution, Upper bound = 35</td>
<td>91</td>
</tr>
<tr>
<td>3.6</td>
<td>Iteration 2: Defender’s solution, Lower bound = 15</td>
<td>91</td>
</tr>
<tr>
<td>3.7</td>
<td>Iteration 3: Attacker’s solution, Upper bound = 25</td>
<td>92</td>
</tr>
<tr>
<td>3.8</td>
<td>Defender’s solution, Lower bound = 25</td>
<td>109</td>
</tr>
<tr>
<td>3.9</td>
<td>Attack graph generated using our approach</td>
<td>110</td>
</tr>
<tr>
<td>3.10</td>
<td>Average computation times using different techniques</td>
<td>110</td>
</tr>
<tr>
<td>3.11</td>
<td>Variation of breach loss with defender’s budget</td>
<td>111</td>
</tr>
<tr>
<td>4.1</td>
<td>Example 1</td>
<td>126</td>
</tr>
<tr>
<td>4.2</td>
<td>Example 2</td>
<td>127</td>
</tr>
<tr>
<td>4.3</td>
<td>Example 3</td>
<td>128</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

Organizations nowadays rely heavily on a network of computers, smart devices, and smart equipments to streamline flow of information for proper functioning and increased productivity. This network of computers are sometimes organized as a local area network (LAN) as shown in Figure 1.1, if the computers are located in close proximity.

![Local Area Network (LAN)](image1)

Figure 1.1
Local Area Network (LAN)

Sometimes, especially for large organizations, when the computers are not in close proximity, the network is organized as a wide area network (WAN) as shown in Figure 1.2.
through different means such as the internet. The network of an organization is also connected to the internet to exchange information with customers and other stakeholders. Unfortunately, this connected nature of organizations are also making them prone to frequent and major cyber attacks to steal information, disrupt business operations, disclose confidential information, and cause harms in other ways [77].

![Wide Area Network (WAN)](image)

Figure 1.2

Wide Area Network (WAN)

FBI received a total of 269,422 complaints in 2014 with an adjusted dollar loss of $800,492,073 [50]. The adjusted dollar loss is in reality billions of dollars considering that only a small proportion of all victims file directly to IC3. The sheer number of complaints and the dollar losses give some idea about the magnitude of the problem. However, the IC3 report is mainly based on individual victims, and this dissertation is concerned with cyber attacks and losses to organizations. According to the State of Cybersecurity report by [65],
the average cost of cybercrime for US retail stores was $8.6 million per company in 2014 which was more than double of the average cost in 2013. The PricewaterhouseCoopers reported in its “Global State of Information security report 2015” that the total number of security incidents detected by respondents grew to 42.8 million around the world, up 48% from 2013.

[52] reported that, among the the high profile cyber attacks against big businesses in 2014, hackers got access to confidential information of up to 11 million customers of Primera Blue Cross - a Washington-based health insurer. Hackers breached a database of 80 million former and current customers of Anthem. In an attack against Home Depot, hackers compromised 56 million payment cards. The estimated cost to Home Depot from the breach was $62 million. In an attack against JPMorgan Chase, account information about 83 million households and small businesses were compromised. Refer to [52, 62] for several other high profile cyber attacks against big businesses in 2014. Therefore, it is quite expected to see large increase in cyber security budgets of corporations and governments [65]. In a survey by [65], 56.09% of cybersecurity professionals reported an expected increase in cybersecurity budget in 2015 compared to the budget in 2014. President Barack Obama in his 2015 budget proposed a sharp increase in spending on cybersecurity, to $14 billion.

Organizations suffer different types of cyber attacks such as hacking attempts, Malware, social engineering, phishing, watering hole, and so on. In the survey by [65], 50.14% and 66.48% of the respondents, respectively reported that their organizations suffered hacking attempts and malware attacks. In fact, hacking and malware attacks are the
second and third most frequent attack types after phishing attacks. Although attacks are divided into the above mentioned categories for simplicity of analysis and discussion, more often than not, one attack type is used as part of another attack type. For example, phishing attack is frequently used as part of a malware attack; malware attack is used as one of the attack vectors in a hacking attack. While the attacks reported in [52, 62] are mainly hacking attacks, there is no lack of malware attacks in recent times. In one of the high profile malware attacks, 1000 to 1500 centrifuges were destroyed in the Natanz nuclear facility of Iran by spreading a computer worm called Stuxnet [134]. [146] provided a brief discussion on 8 deadly computer viruses that appeared at different times over the years.

Malware in malware attacks almost always come with a spreading mechanism, i.e., after one or more computers in a network is infected with a malware, the malware can replicate itself and infect many other computers through activities among infected and uninfected (susceptible) computers. This phenomenon of infection spreading is very similar to infectious disease epidemiology [32, 102]. [72] provided the first serious attempt to adapt mathematical epidemiology to model the spread of computer viruses. [73] in a subsequent article discussed the analogies between the spread of computer viruses (malware) and biological diseases. [171] provided a mathematical model for the propagation of the internet worm called Code Red. [112] and [172] investigated the spread of computer viruses through the email network resulted from the email address books of the victims. Using a combination of analytic modeling and simulation, [97] described the design space of a worm containment system in terms of parameters such as reaction time, containment strategy, and deployment scenario. [75] studied the propagation of computer worms through
the internet using the SIS and modified SIR epidemiological models. [168] proposed a proactive worm propagation model in unstructured P2P networks.

In this dissertation, we study both the problem of spread of malware infection and multi-stage attacks (hacking) against the cyber network of an organization. In particular, we explore methods to minimize the spread of infection and develop models and algorithms to minimize the potential damage from multi-stage attacks. The cyber network of an organization can be represented as a simple graph with the computers as the nodes and the connections among the computers as the arcs (Figure 1.3).

![Undirected graph](image)

Figure 1.3

Undirected graph

The resulting graph can be either undirected (Figure 1.3) or directed (Figure 1.4). In our study the spread of infections, we represent the underlying cyber network as an undirected graph with a set of initially infected and susceptible nodes. Red and white nodes in Figure 1.5 represent the initially infected and susceptible nodes, respectively. In contrast,
in our study of the multi-stage attacks, we use a special type of directed graph called the attack graph which is generated from the underlying cyber network (Figure 1.6).

![Directed graph]

[119] first proposed the attack graph as a tool to analyze multi-stage attacks. A node in an attack graph represents an attack state or a security condition, and an arc represents an atomic attack or exploit/vulnerability. An attack graph represents all the paths from the initial security conditions (initially vulnerable nodes) to some desired states (goal nodes). Thus, an attack graph represents all the paths that can be potentially used by an attacker to compromise a goal node. In Figure 1.6, the salmon nodes with dashed borders represent the initially vulnerable nodes, the blue nodes represent the transition nodes, and the green nodes represent the goal nodes.

The following inputs are used to generate an attack graph: a database of common attacks broken down into atomic steps, information on network topology and configuration,
Figure 1.5

Spread of infection on a graph

Figure 1.6

Directed attack graph
and an attacker profile. [144] proposed an automated attack graph generation tool using these inputs. Since then, researchers have proposed and studied a multitude of attack graph variations such as attack trees [155] and defense trees [22]. Refer to [86] and [79] for a comprehensive review of different types of attack graphs. In this dissertation, we used randomly generated synthetic attack graphs as the underlying graph in our models and algorithms for multi-stage attacks. However, we envisage using the attack graph generated by a tool similar to the one developed by [144] in future.

Interdiction in general means thwarting something from happening. Network interdiction is the act of reducing the capacities of some of the components (nodes and arcs) of a network or removing the components altogether from the network to minimize the activities of some agents. Thus, network interdiction is the most appropriate in modeling a problem when the activities in question and the system in which the agents are operating can be represented using a network. The spread of infection of a malware can be represented as the stochastic propagation of infection from the nodes infected by the malware to the susceptible nodes through the arcs of the network. A multi-stage attack can be represented as the step by step movement of an attacker in an attack graph toward the goal nodes in an effort to reach them starting from the initially vulnerable nodes. Network interdiction in the context of this dissertation means the complete removal of a subset of arcs from the network.

The contributions of this dissertation are as follows. In regard to minimizing spread of infections, we define effective connectivity related metrics, develop novel mixed-integer programming models to optimize the metrics, and propose efficient heuristic methods to
solve them quickly (Chapter 2). In regard to minimizing the damage from multi-stage cyber attacks, first, we develop a bi-level network interdiction model to represent the zero-sum stackelberg game between an attacker and a defender and propose efficient exact and heuristic solution algorithms (Chapter 3). Second, we develop tri-level network interdiction model to study the game between an attacker and a defender when the attacker is boundedly rational (chapter 4). In all of our studies, the objective is to select an optimum subset of arcs to remove them from the graph in an effort to optimize different goals.

In chapter 2, we propose effective methods to remove a subset of arcs from a network to minimize the spread of infection. We define four metrics on network connectivity, develop network interdiction models to minimize the metrics through arc removal, and then use the models as the arc removal methods for minimizing spread of infection. We compare the effectiveness of these four models with the effectiveness of two existing arc removal methods [47, 76] and random arc removal in terms of both the amount of new infections and the speed of the spread. Comparisons show that two of our methods are very effective in minimizing the amount of spread and minimizing the speed of spread, respectively. We provide valuable insights that can be used to formulate effective isolation and quarantine policies.

As it takes relatively long time to solve large instances of the interdiction models developed in chapter 2, we developed heuristic algorithms based on Monte Carlo simulation. Monte carlo method entails the estimation of the qualities of all the solutions (population) via evaluation of a sample from the population using random numbers. [55] presented one of the first applications of the Monte Carlo method. [37] presented a heuristic algo-
gorithm based on the Monte Carlo method. [124] developed a Monte Carlo based algorithm for iteratively computing projective clusters. [93] proposed a Monte Carlo based global optimization method named simulated tempering. Refer to [54] for a comprehensive discussion on the current state of Monte Carlo based methods.

In chapter 3, we present a bi-level attacker-defender network interdiction model capturing the stackelberg zero sum game between an attacker and a defender played on an attack graph. In particular, the bi-level model selects a subset of arcs which upon removal minimizes the maximum damage that can be caused by the attacker. We proposed a novel exact algorithm, enhancements, and heuristic methods to speed up the solution. The exact algorithm is a novel application of the constraint and column generation method.

Application of game theory for cyber security is not new. Game theoretic approaches are appropriate because, most of the times, the behavior of the attackers of a cyber network are driven by the defensive measures undertaken by the defender suggesting an interaction between these two parties. [8] investigated the possible application of game theoretic concepts to develop a decision and control framework. In particular, they model and analyze a game between an attacker and a network of intrusion detection system sensors within a two-person, non-zero sum game. [2] defined a cooperative game among the sensor nodes in a mobile wireless sensor networks. They cluster the nodes based on some similarity of a payoff function and defines one strategy set for each node that guarantees an equilibrium point. [29] investigated the value of game theoretic modeling in information security. They found that a firm enjoys a higher payoff if the security decisions are derived from game theoretic models versus decision theoretic models. [28] developed a game theoretic model
to find the optimal patch updating frequency to balance the operational cost associated with patch updating and the damage associated with unpatched security vulnerabilities. [87] presented the interaction between an attacker and the administrator of a network as a two-player stochastic game and formulate the game as a non-linear programming model to find the Nash equilibrium. [67] provided a good discussion on the problem of applying game theory in security and current successes in solving real-world security problems. [83] examined a security game in which the defender is monitoring the vertices of a graph. They found that the applicability of the approach directly is highly dependent on factors such as type of graph, type of schedules, and the type of defender resources. Calculating the marginal probability of individual resource deployment, [131] proposed a stochastic game theoretic modeling approach that can be used to predict the security and dependability behavior of a system. Refer to [130, 89, 84] for a comprehensive review of game theory as applied to security of computer and communication networks.

Our studies in chapter 3 and 4 complements and extends the existing literature on attack graph cut-set generation [3, 7, 40, 68, 114]. However, cut-set generation literature does not apply game-theoretic approaches in most of the cases. Among the game theoretic approaches on attack graphs, [40] proposed a multi-objective optimization model posing the attacker-defender interaction as an arms race. They use genetic algorithm and competitive co-evolution to solve the model. [169] proposed an automated response approach called the response and recovery engine posing the attacker-defender interaction as a two-player stackelberg stochastic game. [45] presented a game-theoretic model capturing the game between a network administrator and an attacker played on an attack graph.
They use Markov Decision Process and associated policy search technique to solve the model. [138] proposed a game-theoretic model based on Vulnerability Dependency Graph - a variation of attack graph. In this game, the attacker tries to maximize his impact, and the defender tries to both minimize the worst-case impact by patching vulnerabilities or removing some software and maximize the productivity within the enterprise by retaining important software.

A special type of game called the stackelberg game follows a sequential game structure in which the defender is the leader making a defensive decision first, and the attacker is the follower making her decision based on the defender’s decision. The sequential nature of the stackelberg game makes it very effective in modeling network security games. [80] presented a game-theoretic modeling approach for security in a stackelberg game setting. [80] accounted for the uncertainty in the attacker’s surveillance capability. They provide valuable insights from different game settings such as multiple versus single target for the attacker and interchangeability of the Nash equilibria. [120] presented a Bayesian Stackelberg game based approach for randomized defensive actions. Their implemented software agent is able to appropriately weight different actions and account for uncertainty in adversary type. [122] discussed the design choices and challenges in the implementation of a software system called the GAURDS. Compared to the previously studied approaches, [122] included the following features: reasoning about many heterogeneous security activities, reasoning about diverse potential threats, a system designed for hundreds of end users. [117] proposed an efficient exact algorithm for Bayesian Stackelberg games. The algorithm is based on a novel mixed-integer linear programming formulation. One important
assumption in Stackelberg games is that the followers are completely rational. Realizing that leader sometimes faces human followers, [121] proposed an approach to find robust solutions to stackelberg games in which the followers are boundedly rational. [74] presented new scalable models and algorithms for Stackelberg security games.

Because of the multi-stage or decentralized nature of decision making in Stackelberg games and also in network interdiction problems, these classes of problems can be naturally formulated as multi-level mixed-integer linear programs. A problem should be formulated as a bi-level or a tri-level programming model depending on whether the decisions in the problem are made sequentially in two stages or three stages, respectively. [98] showed one of the earlier applications of bi-level programming. Among some other existing studies, [143] presented a bi-level programming model with the upper level determining the optimal location of logistics centers and the lower level determining the equilibrium customer demand distribution. [126] proposed a bi-level quadratic programming model to derive robust knock strategies in the optimization procedure identifying gene knockouts for targeted biochemical overproduction. Refer to [15] for a tutorial on different properties of bi-level programming. [51] extended bi-level linear programming showing its capacity to reformulate mixed-integer linear programs. The author defined a natural generalization of the bi-level linear programming problem and show that most of the existing algorithms originally developed for bi-level linear programs can be adapted to this new class of bi-level programs.

Although there is an abundance of studies on bi-level programming, only a few of those studies include integer variable in the lower level [136, 159, 132, 164, 39, 158]. Solution
of this class of bi-level problem poses additional challenges due to the integer variable in the lower level. [136] presented a bi-level mixed-integer programming formulation of the r-interdiction median problem with fortification. They solve the problem through an implicit enumeration. [159] proposed a new exact algorithm based on Branch and Bound. Compared to the existing algorithms, their algorithm relies on a fewer and weaker assumptions, explicitly considers finite optimal, infeasible, and unbounded cases, and terminates correctly in finite number of iterations. [38] presented a bi-level mixed-integer programming model for vulnerability analysis of electric grid and developed an algorithm based on Multi-start Benders decomposition. [164] provided a generic algorithmic framework based on column-and-constraint generation for bi-level programming problems including those with integer lower levels. Refer to the dissertations by [39] and [158] for a comprehensive discussion on different properties and solution challenges associated with bi-level programming problems.

In chapter 4, we relax the assumption in typical game theoretic models that the attacker is completely rational. Specifically, we consider that the rationality of an attacker is bounded, and it causes the attacker to be able to inflict only a fraction of the maximum damage that can be inflicted with complete rationality. We formulate the problem as a tri-level mixed-integer linear programming network interdiction model. We also develop customized column and constraint generation algorithm to solve the model. Our results demonstrate that incorporating bounded rationality of the attacker could be important depending on the characteristic of the underlying problem.
[11] provided one of the earlier applications of the tri-level linear program. [6] presented a defender-attacker-defender problem to address the problem of minimizing vulnerability of electric power grid against multiple contingencies. Their solution algorithm first reformulates the tri-level program into a bi-level program, and then applies implicit enumeration to solve the resulting bi-level program. [160] proposed a tri-level programming model based on conditional value at risk for a three-stage supply chain management problem. Their solution method also involved reformulation of the tri-level program into an equivalent bi-level program. [64] proposed a tri-level programming model for disaster preparedness planning and solved the model using an iterative dual-ascent solution approach. [167] presented an algorithm called the kth-best algorithm for the general tri-level linear programming model.

We envision the work of this dissertation as a big step toward developing a comprehensive modeling and algorithmic framework that will be used inside a cyber security decision support software system. In addition to the security models and algorithms, the proposed software system will also include other supporting modules such as the module for automated attack graph generation. Guided by organizational policies and protocols and personal judgment, cyber security personnel would use the proposed software to decide about which components of the cyber network to harden under different scenarios including defending against state sponsored hackers, hackers seeking financial gains, etc. or a combination thereof.
CHAPTER 2

METHODS FOR REMOVING LINKS IN A NETWORK TO MINIMIZE THE SPREAD OF INFECTIONS

2.1 Introduction

The spread of harmful infections are common in real life networks: infectious diseases spread through social and transportation networks, computer viruses and malware spread in computer networks, and propaganda and rumors spread in online social networks. Minimizing the spread of these infections is very important because they can cause significant economic and social damage. Worldwide, infectious diseases cause over 10 million deaths each year, accounting for 23% of the total disease related deaths [156]. In the well-known influenza pandemic of 1918, 30 to 50 million people are estimated to have died [60]. According to [134], 1000 to 1500 centrifuges of an Iranian nuclear power plant were destroyed by spreading a computer worm called Stuxnet. Lethal worms and viruses such as Stuxnet can easily fall in the hands of terrorist organizations and rogue nations.

The simplest and traditional way of modeling spread of infections assumes the population as a homogeneous mix of individuals and then compartmentalizes them based on their infection status. Although this simple compartmental framework has been extended to include some upper level host heterogeneities such as contact patterns among age groups, differing spatial structure, inclusion of individual contact structure is a fairly recent phe-
nomenon [13]. The maturation of network science is enabling researchers to find limitations in the homogeneous mixing assumption and discover the value of network modeling [96, 111, 118, 153]. It is indeed important to capture the underlying network because the ability of networks to maintain connectivity when subjected to selective or random removal of nodes or links depends on the particular network topology [4, 61]. Connectivity is a popular measure for networks, and it represents the ability of a network’s nodes to communicate with one another, thereby, facilitating the spread of infections. Connectivity is applicable to infection control because removing the nodes and links in reducing the connectivity of a network is analogous to immunization of individuals and preventing contacts in reducing the spread of infections. In one of the earlier studies that applied network modeling, [135] showed that the tolerance of scale-free networks [14] against random node or link removal does not allow a homogeneous approximation of connectivity in infectious disease spread modeling and results in overestimation of the epidemic threshold. [118] even found the absence of an epidemic threshold in scale-free networks.

Epidemic threshold as defined by [118] is the minimum value of the effective reproduction rate \( \frac{I}{R} \) for which the infection spreads and turns into an epidemic; otherwise, infection dies out. The implication of the work by [135] is that even in networks with very small average connectivity, epidemics can occur, and to find the most influential set of individuals to immunize or quarantine, one must consider the underlying network topology, irrespective of whether the topology resembles a scale-free, random [48], small-world [153], or some other type of network. Immunizing or quarantining randomly based on
homogeneous mixing assumption does not ensure effective reduction of the connectivity of the network, thereby, not preventing the spread of infections effectively.

It is possible to reduce the connectivity of a network and consequently minimize or slow down the spread of infections in the network by interdicting the network in two ways: removing links and removing nodes. [147] and [76] proposed heuristic algorithms to minimize spread in a network by removing a subset of links. [47] proposed a network interdiction model with a non-linear programming formulation that minimizes the number of nodes at risk of infection. [58] proposed a model that removes a set of both links and nodes to minimize the total cost composed of the cost of infection and and the cost of preventing infection. Their model can also be used as a node and link removal method to minimize spread by accounting for the cost of prevention in a budget constraint. [78] analyzed the behavior of basic reproduction number as defined originally by [111] with respect to link removal and proposed a new definition for basic reproduction number. [161] proposed a method to control a special type of spread known as the traffic-driven outbreak by removing links using different link ranking metrics. [31] studied the efficacy of several centrality based link removal strategies on the spread of infectious diseases through the global airline network. [81] proposed approximate algorithms for link removal to minimize complex (threshold-based) contagion.

In a field related to controlling the spread of infections, other authors have studied the problem of removing nodes in a network in order to maximize the fragmentation of the network, minimizing connectivity [1, 12, 33, 42, 139, 148, 150, 149]. This problem is known as the critical node detection problem (CNDP). These studies on the CNDP optimize one
or more of the following network fragmentation metrics: 1) the number of connected node pairs (minimize), 2) the largest connected component size (minimize), and 3) the number of connected components (maximize). Although the interdiction models related to CNDP can be used to identify a set of nodes to remove to reduce the spread of infections, they are not expected to reduce spread effectively as they assume all the nodes to be of the same type rather than dividing the nodes into infectious and susceptible categories. The aforementioned fragmentation metrics also need to be significantly modified before they can be used for interdicting a network to minimize spread; For example, to minimize spread, it is enough to have just two connected components, one with all the infectious nodes and the other with all the susceptible nodes, instead of maximizing the number of connected components.

The field of network vulnerability and robustness analysis [20, 27, 44, 95, 103, 137, 141, 59, 43] is a closely related field of the critical node detection problem. The problem analyzed in this field is functionally opposite of the critical node detection problem because in this case, critical components (nodes and links) are those, whose hardening results in maximizing the connectivity of the network. The fragmentation metrics that are minimized in the CNDP are maximized in the robustness problem and vice versa. Moreover, although some of the studies in this field consider cascading failure of components, most of the studies do not consider spreading agents such as infections.

In this paper, we study the problem of detecting a subset of critical links in a network, whose removal minimizes the spread of infections. Therefore, the problem studied in this paper is similar to the critical node detection problem, but it is customized for minimizing
spread by removing links instead of nodes. One of the reasons the link removal problem is very important is that link removal allows finer control than node removal. If a node is removed, all of the links connected to the node are automatically removed. In contrast, if a link is removed, only that link is removed. A node can still be removed in this paradigm by removing all of the links connected to it. Therefore, the link removal problem has the potential to provide additional insights into problems that have only been studied under node removal [24, 153, 23, 145, 82, 127]. [90, 91] compared several generic edge and node ranking metrics in reducing the global spread of influenza through an airline network. According to their results, link ranking metrics are usually more effective than node ranking metrics. Moreover, in many situations the node removal option is unattractive or not available at all, whereas there is still a way to remove links. For example, it might be very difficult to find and eliminate terrorists in a terrorist network, but there might be a way to block their communication channels, in effect removing links among them. The loss associated with completely shutting down an entire airport might be enormous, but it might be possible to temporarily suspend the flights between two specific airports during a global disease pandemic. However, we should also note that there are cases where node removal is feasible but link removal is not. For example, if the nodes in the network represent individuals and the links represent the interactions between individuals, it may be possible to target specific individuals for vaccination, but preventing specific individual-individual interactions is likely not a feasible strategy.

Although many previous research studied the evolution of an spread of infection with respect to link removal, only a small portion of them studied link removal methods that
minimize the spread of infections. The algorithm proposed by [147] is based on the finding that the leading eigenvalue of the network adjacency matrix determines whether a spread will turn into an epidemic [151, 123]. They report the effectiveness of their algorithm by showing that it reduces the leading eigenvalue more than other eigenvalues. They also report the effectiveness based on the comparison of the fraction of infected nodes produced by a simulation. However, they do not use information about which nodes are infected and which are not in the link selection mechanism. Also, they do not compare their effectiveness with any existing well-known link removal methods. The algorithm proposed by [76] is based on bond percolation. The main virtue of the algorithm proposed by [147] and the algorithm along with the speeding mechanism proposed by [76] is that they are fast. [76] evaluates the effectiveness of their algorithm in terms of an indirect measure called contamination degree. The model proposed by [47] minimizes connectivity between infectious and susceptible nodes, and later in a subsequent work, [46] compare the effectiveness of the model along with some other link removal approaches. Although [47] present a heuristic procedure, it is unclear if their non-linear programming formulation can be solved to optimality for problem instances much larger than the instances with 15 nodes that they solve using complete enumeration. They do not propose any exact procedure other than complete enumeration. One common aspect of all of the studies including the studies that control the spread of infections by removing links is that they use one or more indirect metrics such as the leading eigenvalue [147], contamination degree [76], susceptible nodes at risk of infection [47], etc. These metrics are assumed to be representing infection spread, and in most of the cases, they are not evaluated against a direct
metric such as the average new infections. Although the interdiction models developed in this work do not optimize any direct infection spread metric, they are evaluated in terms of two direct infection spread metrics.

In this paper, we propose four network interdiction models and formulate them as mixed-integer linear programs. We call the models as link removal methods, in that solution of each model provides a set of links whose removal optimize an interdiction metric. Then, we evaluate the link removal methods by estimating two direct metrics of infection spread through two different types of simulations. The first model (\text{MinCONNECT}, sub-section 2.3.4) minimizes the number of connections between infected and susceptible nodes. The second model (\text{MinATRISK}, sub-section 2.3.5) minimizes the number of susceptible nodes having one or more connections with infected nodes. The third model (\text{MinPATHS}, sub-section 2.3.6) minimizes the number of paths between infected and susceptible nodes. The fourth model (\text{MinWPATHS}, sub-section 2.3.7) minimizes the total weight of the paths between infected and susceptible nodes. Weight of a path is calculated as the product of the transmission probabilities of the links on that path. Since, there are only a few existing link removal methods that minimize spread, we compare our link removal methods with random link removal, a link removal method proposed by [76], and a method based on adapted betweenness centrality defined by [47]. After applying a link removal method to select a set of links to remove, we remove the links from the network and test the residual network using two types of simulations: susceptible-infectious (SI) and susceptible-infectious-recovered (SIR). We estimate the average number of new infections (measures the occurrence of new infections) from the SIR simulation and the time to
infect half of the susceptible nodes (measures the slowing down of the spread) from the SI simulation.

Our methods have several advantages over the existing methods. First, unlike the methods in [76, 147], that directly propose algorithms, we propose both mathematical programming models and heuristic algorithms as faster solution alternatives. One advantage of a mathematical programming model is that it allows one to take advantage of the rich algorithms and solvers developed for mathematical programs, in our case, integer programs. Having a mathematical model in our view also enables future researchers to develop efficient exact algorithm based on the model itself such as cutting plane algorithms along with novel valid inequalities or to use the model as the benchmark to develop more efficient and effective approximate algorithms to solve the underlying problem.

Second, as we argue in section 2.3.4 that the MINCONNECT model is a general case of the critical link detection problem, the MINCONNECT model and the MINATRISK model which is an extension of the MINCONNECT model can have additional applications such as finding important links in a protein-protein network, anti-terrorism network, and brain functionality [23, 82, 71, 145]. To the best of our knowledge, the MINCONNECT model is one of the few interdiction models that minimize the number of pairwise connections by removing a set of links. Thus, the MINCONNECT model has potential applications in problems such as synthesizing distributed firewall configurations in a computer network [165]. Recall that the other two models considering pairwise connectivity [150, 43] assume all the nodes to be of the same type.
Third, the M\textsc{InATRisk} model with the same interdiction objective as that of [47] can be solved to optimality for networks with 200 nodes within reasonable time for several parameter combinations. These 200 node networks are much larger than the 15 nodes networks that [47] solved to optimality. In addition, our heuristic algorithms solve problem instances with 200 nodes in less than 11 minutes compared to 2 hours taken by [47] using their approximate algorithm. The M\textsc{InATRisk} model can also be applied in a similar real-world setting such as the network of residential hotels in which some of the hotels have number of injection drug users less than a threshold (susceptible), and the other hotels have number of injection drug users greater than a threshold (infectious) [47]. The problem in this case is to maximize the number of susceptible hotels completely isolated from the infectious hotels. [46] found that reactive approaches including the model proposed by [47] outperform preventive approaches when the surveillance information is not highly erroneous. We think our paper will complement both of these studies [47, 46] very well in the sense that we propose four optimization based reactive approaches and compare six reactive approaches along with a random approach.

Fourth, to the best of our knowledge, no existing link removal method attempts to control the speed of spread. The speed of spread measures how fast susceptible nodes become infected, and the less time it takes to infect the same number of susceptible nodes, the greater the speed of spread. Although the M\textsc{InWPaths} model does not use a speed of spread metric in the objective function, this model along with the algorithm to solve it performs very well when applied to slow down the spread. Our models can also be used as part of an algorithm to solve other network interdiction problems [107]. In summary,
although our models might be limited in direct applicability, they are useful in developing insight in minimizing spread of infections under different scenarios. They can also be applied in interdiction problems other than infection control.

Specifically, the contributions of this paper are as follows: 1) four novel network interdiction models formulated as mixed-integer linear programs (MILP), 2) two new heuristic algorithms for the interdiction models, 3) a comparison of our link removal methods with several existing methods in minimizing spread, and 4) recommendations, based on our results about which link removal method is the most appropriate in different infection control scenarios (e.g., reduction of the occurrence of new infections versus slowing down of the speed of spread).

The rest of the paper is organized as follows. In Section 2.2, we describe the problem of removing a set of links to minimize the spread of infections in a network. In Section 2.3, we propose mixed-integer programming formulations of our network interdiction models and prove several structural properties. In Section 2.4, we provide heuristic algorithms for solving our models. In Section 2.5, we discuss the computational tractability of our models and algorithms by reporting the results of a set of experiments. In Section 2.6, we compare all of the link removal methods via simulation. Finally, in Section 2.7, we conclude our paper.

2.2 Problem Description

Our mathematical models consist of an undirected graph $G = (N,A)$, where $N$ is a set of nodes and $A = \{(i,j) : i \in N, j \in N, i < j\}$ is a set of links. We assume that $G$ is arbi-
tary, and our knowledge about its topology and other attributes is complete. Depending on the type of infection, a node might represent a computer, a person, an account on a social media site, and so on. Similarly, a link might represent a communication channel between two computers, social contact between two persons, friendship status between two user accounts, and so on. The models input the state of a system prior to an outbreak, in which some of the nodes in the network are infected and the rest are susceptible to infection. Let $I \subseteq N$ be the set of infected nodes, and $S = N \setminus I$ be the set of susceptible nodes. The problem is to remove a set of links $L \subseteq A$ to minimize the spread of infection in the resulting network, such that the cardinality of $L$ is no greater than some integer parameter $b$. Parameter $b$ represents the maximum number of links that can removed using an available budget.

The spread of infection is represented in this paper by the following metrics: 1) average number of new infections and 2) average time to infect half of the susceptible nodes to capture the number and speed of occurrence of new infections, respectively. The spread of infections through a network is inherently stochastic, with an infectious node infecting its neighbors with some probability [76]. However, existing stochastic optimization approaches such as stochastic programming and simulation-optimization are often computationally intensive. Thus, this paper proposes four deterministic network interdiction models minimizing four interdiction metrics, and each act as a link removal method to minimize the spread of infection measured in terms of the two aforementioned spread of infections metrics. In turn, this paper evaluates and compares the average performances
of these four methods along with some existing methods in an stochastic environment by estimating the spread of infections metrics using a stochastic simulation.

2.3 Model Formulations

In terms of the objectives and the formulations, MINCONNECT and MINATRISK models are similar, and MINPATHS and MINWPATHS models are similar. MINCONNECT and MINATRISK models optimize the number of connections. In contrast, MINPATHS and MINWPATHS models optimize the number of transmission paths. We used the same notations for all the parameters and variables that are common in all the formulations to avoid repetition. We should mention here that any link between any pair of infected nodes is removed before building the corresponding formulations because these links are unable to transmit any infection.

The following terms are used in describing our four models:

2.3.1 Definition 1 (Transmission path)

If a path between two nodes (at least one of them is susceptible) contains no other infected nodes, it is a transmission path. If both of the nodes are susceptible, infection can transmit through the path when one of them becomes infected, and no other nodes on the path become infected at the same time. Thus, a transmission path in the initial state of the network might not remain a transmission path after infections start spreading because both of the end nodes, or one or more of the intermediate nodes might become infected. However, our models take only the initial state of the network into account.
2.3.2 Definition 2 (Connection)

Two nodes, at least one of them susceptible, have one pairwise connection if there is at least one transmission path between them.

2.3.3 Definition 3 (Susceptible node at risk of infection)

A susceptible node is at risk of infection if it has at least one pairwise connection with the infected nodes, and the susceptible node is saved from infection if it is no longer at risk of infection in the interdicted network.

2.3.4 MinConnect Model

The first model, MinConnect, minimizes the number of pairwise connections between infected and susceptible nodes.

Let, $N_i$ be the set of neighbors of node $i$, and $\Omega = \{(i, j) : (i, j) \in N \times N, j > i, (i, j) \notin I \times I\}$. $\Omega$ is the set of distinct pairs of nodes, and at least one of the nodes in each pair is susceptible. The decision variables are as follows.

The MinConnect model is formulated as follows.
\begin{align*}
\text{(MinConnect)} \quad & \min \sum_{i \in S} \sum_{j \in I} x_{ij} \quad (2.1) \\
\text{s.t.} \quad & x_{ij} + y_{ij} \geq 1 \quad \forall (i, j) \in A \\
& x_{k} - x_{j} + y_{ji} \geq 0 \quad \forall (k, i) \in \Omega, \quad (2.3) \\
& \forall j \in N, k \neq j, j \notin I \\
& \sum_{(i, j) \in A} y_{ij} \leq b \quad (2.4) \\
& x_{ij} \geq 0 \quad \forall (i, j) \in \Omega \quad (2.5) \\
& y_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \quad (2.6)
\end{align*}

The objective function (2.1) counts the total number of pairwise connections between the infected and susceptible nodes. Constraint (2.2) makes sure that two neighboring nodes $i$ and $j$ are connected ($x_{ij} = 1$) if the link $(i, j)$ between them is not removed. Constraint (2.3) makes sure that a node $k$ is connected to node $i$ if node $k$ is connected to node $j$ and the link $(i, j)$ is not removed. Constraints (2.2) and (2.3) together with the objective function (2.1) ensure that $x_{ij} = 1$ if and only if there exists at least one path of unremoved links between $i$ and $j$. There is no constraint (2.3) for nodes $i$ and $k$ if both of them are infected. There is no constraint (2.3) for nodes $i$ and $k$ through node $j$ if node $j$ is infected. Constraint (2.4) is the budget constraint. Corollary 2.3.4.4 shows that $x_{ij}$ variables will be binary in an optimal solution.
2.3.4.1 Lemma 1 (Implied lower bound)

An implied lower bound on $x_{ij}$ is either 0 or 1 for all $(i, j) \in \Omega$ in any feasible solution of the MinCONNECT model.

Proof: The proof is divided into two cases. Cases 1 and 2 are mutually exclusive for a pair of nodes $(i, j)$.

2.3.4.2 Case 1

Nodes $i$ and $j$ have one or more transmission paths. If $y_{pq} = 0$ where $(p, q) \in A$ for all of the links on a transmission path, then $x_{pq} \geq 1$ for all the links on that transmission path. From constraint (2.3), $x_{ij} \geq 1$. If $y_{pq} = 1$ for at least one link on the transmission path, from constraint (2.3), $x_{ij} \geq 0$. In this way, either $x_{ij} \geq 0$ or $x_{ij} \geq 1$ resulting from that transmission path. The same is true for all the transmission paths between nodes $i$ and $j$.

2.3.4.3 Case 2

If there is no transmission path between nodes $i$ and $j$, constraint (2.5) ensures that $x_{ij} \geq 0$ for all $(i, j) \in \Omega$.

It is clear from the above cases that the implied lower bound on $x_{ij}$ is either 0 or 1 for all $(i, j) \in \Omega$ in any feasible solution of the MinCONNECT model.

2.3.4.4 Corollary 1

An optimal solution to MinCONNECT exists such that $x_{ij} \in \{0, 1\}$ for all $(i, j) \in \Omega$. 

\[30\]
From Lemma 2.3.4.1, it is obvious that \( x_{ij} \in \{0, 1\} \) for all \( (i, j) \in \Omega \) in any optimal solution of the MINCONNECT model because it is a minimization problem with an objective function of the sum of \( x_{ij} \) variables.

We now show that the MINCONNECT model is a general case of the critical link detection problem in [150] when the metric, number of pairwise connections is minimized in this problem. Let both of the sets \( S \) and \( I \) be substituted by \( N \) in the MINCONNECT formulation. In addition, redefine the set \( \Omega \) as \( \Omega = \{(i, j) : (i, j) \in N \times N, j > i\} \), and redefine the constraint (2.3) as \( x_{ki} - x_{kj} + y_{ji} \geq 0 \quad \forall (k, i) \in \Omega, \forall j \in N, k \neq j \). With these modifications, the MINCONNECT formulation now minimizes the number of pairwise connections between any pair of nodes. Hence, the MINCONNECT Model is a general case of the critical link detection problem.

2.3.5 MINATRISK Model

The second model, MINATRISK, minimizes the number of susceptible nodes at risk of infection. This metric is the same as the metric in [47]. However, the formulation in [47] is a non-linear program, while we present a mixed-integer linear programming formulation below.

Compared to the MINCONNECT formulation, this formulation uses an additional decision variable \( z_i \) as defined below.

\[
z_i = \begin{cases} 
1 & \text{if susceptible node } i \text{ is at risk of infection} \\
0 & \text{otherwise} 
\end{cases}
\]

The MINATRISK model is formulated as follows.
\[ \text{(MINATRISK)} \min \sum_{i \in S} z_i \] 

\[ \text{s.t.} \quad (2.2) \quad - \quad (2.6) \] 

\[ z_i - x_{ij} \geq 0 \quad \forall i \in S, j \in I \] 

The objective function (2.7) counts the number of susceptible nodes at risk of infection. The additional constraint (2.9) makes sure that a susceptible node is at risk of infection if it is connected to one or more of the infected nodes. Lemma 2.3.5 proves that the \( z_i \) variables will always be binary in an optimal solution.

An optimal solution to \text{MINATRISK} exists such that \( z_i \in \{0, 1\} \) for all \( i \in S \).

In the \text{MINATRISK} formulation, \( x_{ij} \geq 0 \) or \( x_{ij} \geq 1 \) for all \( (i, j) \in \Omega \) from Constraint (2.8) based on Lemma 2.3.4.1. Now, from constraint (2.9), \( z_i \geq 0 \) or \( z_i \geq 1 \) for all \( i \in S \).

Hence, \( z_i \in \{0, 1\} \) for all \( i \in S \) in the optimal solution since \text{MINATRISK} is a minimization problem with an objective function of the sum of \( z_i \) variables.

Let, \text{MINATRISK-Z} be a variation of the \text{MINATRISK} formulation such that the binary constraint \( y_{ij} \in \{0, 1\} \) for all \( (i, j) \in A \) in (2.8) is replaced with \( 0 \leq y_{ij} \leq 1 \) for all \( (i, j) \) in \( A \), and the constraint \( z_i \in \{0, 1\} \) for all \( i \in S \) is added. Lemma 2.3.5.1 shows that any optimal fractional \( y_{ij} \) solution of the \text{MINATRISK-Z} formulation can be converted to an optimal binary solution by simply setting the fractional \( y_{ij} \) values to 0. This finding is particularly important because \text{MINATRISK-Z} is much more computationally efficient than the \text{MINATRISK} formulation (see Section 2.5).
2.3.5.1 Lemma 1

Any optimal solution of the MINATRISK-Z formulation which has some fractional $y_{ij}$ values can be converted to $y_{ij} \in \{0, 1\}$ for all $(i, j) \in A$ by setting the fractional $y_{ij}$ values to 0, and the resulting objective function value is equal to the optimal objective function value of the MINATRISK formulation.

Proof: The proof is divided into two cases as follows.

2.3.5.2 Case 1

Let us first examine the solution of the MINATRISK formulation if the binary constraint $y_{ij} \in \{0, 1\}$ for all $(i, j) \in A$ is removed. Suppose, the set of all the susceptible nodes is divided into two sets $S_1$ and $S_2$. Nodes in $S_1$ are connected to nodes in $I$ through a set of links $L_1$, where $|L_1| > b$, and nodes in $S_2$ are connected to nodes in $I$ through a set of links $L_2$, where $|L_2| = b$. From constraints (2.8) and (2.9), it is possible to have a solution such that $0 \leq y_{ij} = \frac{b}{|L_1|} < 1$ for all $(i, j) \in L_1$ and $y_{ij} = 0$ for all $(i, j) \in L_2$ where $\sum_{(i,j) \in L_1} y_{ij} = b$ if $|S_1| \gg |S_2|$. This is due to the reduction in the objective function $|S_1| \times \frac{b}{|L_1|} > |S_2|$. The quantity in the right side of this inequality is the reduction in the objective function if the solution is $y_{ij} = 1$ for all $(i, j) \in L_2$ and $y_{ij} = 0$ for all $(i, j) \in L_1$. So, the fractional solution is superior to the binary solution (Figure 2.1). However, for the MINATRISK-Z formulation and for the same fractional solution, the reduction in the objective function is $0 < |S_2|$ because $z_i = 1$ in any solution in which $z_i > 0$ from constraint $z_i \in \{0, 1\}$ for all $i \in S$. So, the binary solution is superior to the fractional solution (Figure 2.2). For the MINATRISK formulation, the binary solution is automatically selected. Figures 2.1
and 2.2 are for the same problem, and figure 2.3 is for a different problem. Red and blue nodes are infected and susceptible, respectively. Links with fractional values on them are partially removed; links without any value are not removed; and links with value 1 are completely removed.

2.3.5.3 Case 2

If $|L_2| < b < |L_1|$, and more than $b - |L_2|$ links need to be removed to save one more node (Figure 2.3), the extra budget $b - |L_2|$ cannot be used to save any more nodes after saving the nodes in $S_2$. So, in the optimal solution if $0 < y_{ij} < 1$ for any $(i, j) \in L_1$, they can be set to 0 without altering the objective function. Also, if $b > |L|$ in the trivial case, where $L$ is the set of all the links connected to the infected nodes, the extra budget $b - |L|$ cannot be used because there is no remaining susceptible nodes to be saved (Figure 2.3). In this figure, after removing the three links connected to node 7, the additional link allowed to be removed is not able to save another node. This results in arbitrary partial removal of some of the links. So, in the optimal solution if $0 < y_{ij} < 1$ for any $(i, j) \in A \setminus L$, these $y_{ij}$s can be set to 0 without altering the objective function. For the MINATRISK formulation, in both $|L_2| < b < |L_1|$ and $b > |L|$ scenarios, links corresponding to the extra budget are automatically set to either 0 or 1.

In both cases, after setting the relevant $y_{ij}$s to 0, the optimal solution will be $y_{ij} \in \{0, 1\}$ for all $(i, j) \in A$, and the optimal objective function value of the MINATRISK formulation is equal to the optimal objective function value of the MINATRISK-Z formulation.
Solution to MINATRISK-Z without the binary constraint $z \in \{0,1\} \ (b = 1)$

Solution to MINATRISK-Z ($b = 1$)
Figure 2.3

Solution to MINATRISK-Z (b = 4)

2.3.6 MinPaths Model

The MinPaths model maximizes the number of transmission paths removed from the network.

Let, $P_{uv}$ be the set of transmission paths between infected node $u$ and susceptible node $v$, and $L_{uvw}$ be the set of links belonging to the $w^{th}$ transmission path in $P_{uv}$. We use a modified depth-first search algorithm to find all the $L_{uvw}$ sets. The additional decision variable used in this model is as follows.

$$t_{uvw} = \begin{cases} 
1 & \text{if the transmission path } \text{in } P_{uv} \text{is removed} \\
0 & \text{otherwise} 
\end{cases}$$

The MinPaths model is formulated as follows.
\[
\text{(MinPaths)} \quad \text{max} \quad \sum_{u \in I} \sum_{v \in S} \sum_{w \in P} t_{uvw} \\
\text{s.t.} \quad t_{uvw} - \sum_{(i,j) \in L_{uvw}} y_{ij} \leq 0 \quad \forall u \in I, v \in S, w \in P_{uv} \\
\sum_{(i,j) \in A} y_{ij} \leq b \\
y_{ij} \in \{0,1\} \quad \forall (i,j) \in A
\]

Objective function (2.10) counts the number of transmission paths removed from the network. Constraints (2.13) and (2.14) are the same as the constraints (2.4) and (2.6), respectively, in the MinConnect model. Constraint (2.12) makes sure that if none of the links on the \(w^{th}\) transmission path between infected node \(u\) and susceptible node \(v\) is removed, the \(w^{th}\) transmission path is not removed. Constraint (2.14) guarantees that the values of the path removal variables do not exceed 1. Note that although we do not restrict \(t_{uvw}\)s to be binary, the formulation guarantees binary values of this variable in an optimal solution of the model, satisfying the binary requirement in our definition of transmission paths.

### 2.3.7 MinWPaths Model

The MinWPaths model minimizes the total weight of the transmission paths between all the infected nodes and all the susceptible nodes in the induced network. The weight of a transmission path is the product of the transmission probabilities on each of the links on that path. We assume that the probability of transmission on a link is readily available. Although use of transmission probability is a common feature in epidemiological
studies [94], estimation of this probability might be very difficult in reality. Note that the 
\textsc{MinWPaths} and the \textsc{MinPaths} models are the same except for their objective functions.

Let, \( p \) be the probability of an infected node infecting a neighboring susceptible node, 
and let \( p_{uvw} \) be the probability of transmission from infected node \( u \) to susceptible node \( v \) 
on the \( w^{th} \) transmission path between them. Then, \( p_{uvw} \) is the weight of this path, and it is 
calculated as \( p_{uvw} = p^{|I_{uvw}|} \). The \textsc{MinWPaths} model is formulated as follows.

\[
\text{(MinWPaths)} \quad \max \sum_{u \in I} \sum_{v \in S} \sum_{w \in P_{uv}} p_{uvw} t_{uvw} 
\quad \text{(2.11)--(2.14)} 
\quad \text{(2.16)}
\]

Objective function (2.15) ascertains the total number of weighted paths between all the 
infected and susceptible nodes.

2.4 Solution Algorithms

At first, we solve the interdiction models for some of the problem instances using the 
commercial solver CPLEX [36] to understand the need for developing any algorithm. It 
is clear that the \textsc{MinConnect} and \textsc{MinRisk} models cannot be solved for large prob-
lems, e.g., problems with 300 nodes and average node degree 4 within reasonable time 
(two hours) for most of the parameter combinations. Motivated by the previous studies 
that successfully develop computationally efficient algorithms using Benders decompo-
sition [35], we also develop and test algorithm based on these decomposition technique. 
However, Benders decomposition of the \textsc{MinRisk} formulation is slower than its direct 
solution using CPLEX. Refer to the supplemental material for a complete description of
the Benders formulation, algorithm, and computational results. On the other hand, there are potentially exponential number of paths between all the infected nodes and all the susceptible nodes in a network. It means that the MILP formulations of the \textsc{MinPaths} and the \textsc{MinWPaths} models can only be built and solved for small networks (less than 20 nodes). Thus, we develop Monte Carlo-based heuristic algorithms for the four models in the following sub-sections and demonstrate the performance of the algorithms in the computational experiments in section 2.5.

\subsection*{2.4.1 Heuristic algorithm for the \textsc{MinConnect} Model}

\begin{algorithm}
\caption{Heuristic algorithm for the \textsc{MinConnect} Model}
\begin{algorithmic}
\State \textbf{Input}: Budget = $b$.
\State \textbf{Output}: Set of arcs selected = $L$.
\State 1. Evaluate each arc, $a$ in $A$.
\State 2. Randomly select $b - |L| - 1$ other arcs.
\State 3. Select the arc removing most number of connections.
\State 4. $A = A \setminus a$, $L = L + a$.
\State 5. If $|L| = b$, return. Else, go to step 1.
\end{algorithmic}
\end{algorithm}

Algorithm 2.4.1 is the heuristic algorithm for the \textsc{MinConnect} model. The potential of a link to minimize the number of connections is estimated by temporarily removing this link along with some other links to fill out the budgeted quota of links that can be removed and then, counting the number of connections between the infected and susceptible nodes in the resulting network.

The parameter $M$ controls the number of times a set of links is randomly removed in the \textsc{MinConnect} and the \textsc{MinAtRisk} models and the number of trees generated randomly
from each of the infected nodes at the beginning in the algorithms for the MINPATHS and the MINWPATHS models. In other words, $M$ is the number of replications in the random selection processes of the heuristic algorithms.

2.4.2 Heuristic algorithm for the MINATRISK Model

We do not present the heuristic algorithm for the MINATRISK model separately because this algorithm and the heuristic algorithm for the MINCONNECT model are very similar. In this algorithm, instead of $C_{A_i}$, $R_{A_i}$ is calculated in each replication of a link, where $R_{A_i}$ is the number of susceptible nodes at risk of infection in that replication. Then, $i_{best}$ at any iteration of the while loop is selected based on $TR_{A_i}$ rather than $TC_{A_i}$, where $TR_{A_i}$ is the total number of susceptible nodes remaining at risk of infection when link $A_i$ is evaluated.

2.4.3 Heuristic algorithm for the MINWPATHS Model

Input: Budget = $b$.

Output: Set of arcs selected = $L$.

1. Randomly generate $M$ trees from the graph.
2. Evaluate each arc, $a$ in $A$.
3. Select the arc removing most number of paths in the $M$ trees.
4. Select the arc $a$ removing most number of connections.
5. $A = A \setminus a$, $L = L + a$.
6. If $|L| = b$, return. Else, go to step 2.

Algorithm 2.4.3 is the algorithm for the MINPATHS model. In this algorithm, $M$ random trees, each consisting of paths from the infected nodes to the susceptible nodes are
generated initially. Then, at each iteration of the main loop, the link that removes the maximum number of paths between the infected and susceptible nodes in the trees is removed from the network. At an iteration, the number of paths removed by a link is calculated by temporarily removing the link from the network and counting the number of paths removed from the remaining paths as a result of removing the link under consideration.

2.5 Computational Experiments

In this section, we investigate the computation times of the models for several different network sizes and problem configurations. Network size here means the number of nodes, and problem configuration means a specific combination of the fraction of nodes infected and fraction of links removed. Unless mentioned otherwise, Random network [48] is the underlying network type used in the experiments of this paper. The other network type used is the Scale-free network [14]. Note that Random networks are different than randomly-generated networks. All the networks (both Random and Scale-free) used in this paper are randomly generated and have an average node degree of 4. Thus, the number of links in any of the networks is approximately double of the number of nodes. However, two networks with the same network size might have a different number of links between infected nodes, and these links are removed before building the models. Therefore, the number of links might have minor variation even for the same network size when they are input into the optimization models. All the experiments were carried out on a computer with an Intel core i7 2.90GHz processor and 8GB RAM.
At first, we report the computation times of direct solutions of the MinCONNECT and the MinAtRisk models by CPLEX. We do not report the computation times of the Min.Paths and the Min.WPaths models by CPLEX as they can be solved for only small networks. Then, we evaluate the quality of the solutions by the heuristic algorithms on both Random and Scale-free networks in terms of the proportion of their solutions that are optimal and the average optimality gap. We also report the computation times required by the heuristic algorithms to solve problems with up to 200 nodes.

![Figure 2.4](image)

**Figure 2.4**

Computational performance of the MinCONNECT model

### 2.5.1 MinCONNECT Model

We found that the computation time for solving this model using CPLEX varies a great deal even for problem instances with similar numbers of nodes, links, and budget ($b$). This
is in fact true for both the MINCONNECT and the MINATRISK models. This indicates that problem instances become easy or difficult to solve depending on the positions of the infected and susceptible nodes in the network. Figure 2.4 shows the variation of the average computation time with respect to the network size for different problem configurations. The average computation time corresponding to a specific network size and problem configuration is taken over 10 problem instances.

### 2.5.2 MINATRISK Model

Recall that the MINATRISK model and the model proposed by [47] have the same network interdiction objective. However, unlike the non-linear programming formulation in [47], we formulate the problem as a mixed-integer linear program which can be solved by a commercial solver such as CPLEX for relatively large problems. [47] present an approximate algorithm that can solve problem instances with 200 nodes in about 2 hours. The equivalent MILP formulation proposed in this paper, especially the MINATRISK-Z formulation, can be solved to optimality by CPLEX for problems with 150 nodes in less than 2 hours for almost all the parameter combinations, and for problems with 200 nodes for several parameter combinations (figure 2.6).

Figure 2.5 juxtaposes the average computation times of the MINATRISK-Z and the MINATRISK formulations. The average computation times are taken over 10 problem instances for each network size. Figure 2.5 shows that the average computation time of the MINATRISK-Z formulation is less than that of the MINATRISK formulations, and also the former increases slower than the latter as the number of nodes increases. In fact, the com-
putation times for the MinAtRisk-Z formulation are always less than the corresponding computation times for the MinAtRisk formulation irrespective of the problem configuration and network size. The main reason for the faster computation of MinAtRisk-Z is that MinAtRisk-Z has approximately half as many binary variables as in the MinAtRisk formulations for the networks with average node degree of 4. Therefore, MinAtRisk-Z is expected to perform even better compared to the MinAtRisk formulation for denser networks.

![Figure 2.5](image.png)

**Figure 2.5**

Performance of the MinAtRisk and MinAtRisk-Z formulations

Figure 2.6 shows the variation of the average computation time with respect to the network size for different problem configurations. Most of the 100 node problem instances with 10% nodes infected and 10% links removed cannot be solved. Thus, the average computation time is 7200 seconds because the experiments are terminated after that time. The
same applies for this problem configuration for the 150 and 200 node problem instances. Most of the 200 node problem instances with 20% infected and 20% links removed also cannot be solved. Of the 200 node problem instances with 20% infected and 10% links removed, 29% can be solved to optimality in 2 hours, and the instances that cannot be solved to optimality for this configuration have an average optimality gap of 5.6% at termination. All the 200 node problem instances for the other 2 configurations are solved to optimality.

Figure 2.6 demonstrates an interesting behavior of the computation times of the MI-NATRISK-Z formulation for different problem configurations. The average computation time corresponding to a specific network size and problem configuration is taken over 10 problem instances. Apparently, the computation time varies significantly with the ratio between the fraction of nodes infected and the fraction of links removed. When the fraction of nodes infected and the fraction of links removed are equal, the number of links that can be removed is half of the number of links connected to the infected nodes. This likely increases the number of feasible solutions, making the problem combinatorially more difficult. Thus, the average computation time increases as the ratio becomes closer to one.

2.5.3 Heuristic Algorithms

To evaluate the performance of the heuristic algorithms, we randomly generated 100 Random networks, each with 12 nodes, for several problem configurations. Recall that the MINPATHS and MINWPATHS models can only be solved by CPLEX for small networks. Therefore, to keep the same basis of comparison for all the algorithms, we used networks
with 12 nodes for the performance evaluation. The values of $M$ (number of replications) are set to 100 in all of the heuristic algorithms.

According to Figure 2.7, ALGORITHM-MINCONNECT and ALGORITHM-MINATRISK found optimal solutions more than 60% and 70% of the times, respectively, on Random networks. The solid and dash bordered columns show the average performance on Random and Scale-free network, respectively. On the other hand, ALGORITHM-MINPATHS and ALGORITHM-MINWPATHS found optimal solutions only about 15% and 23% of the times, respectively, on random networks. However, according to Figure 2.8, their average optimality gaps were both less than 5%. All of the algorithms perform slightly better on Scale-free networks. Optimality and gap of a solution is determined by comparing with the optimal solution found by CPLEX. The solid and dash bordered columns show the average performance on Random and Scale-free network, respectively.
Figure 2.9 presents average run times of the heuristic algorithms for different network sizes and problem configuration with 20% initially infected, 10% fraction of arcs removed, and 0.15 as the probability of transmission. The average computation times are taken over 10 problem instances for each network size. Figure 2.9 shows that on average, the heuristic algorithm for the MINATRISK model takes about 600 seconds to solve the 200 nodes problems, and this is the longest time taken by any algorithm. The approximate algorithm proposed by [47] takes 2 hours to solve a problem of similar size. However, unlike our heuristic algorithms, their approximate algorithm provides performance guarantee. The heuristic algorithms for the MINPATHS and the MINWPATHS models solve the 200 nodes problems within a few seconds.
Figure 2.8

Average optimality gap of the algorithms

Figure 2.9

Computational performance of the heuristic algorithms
2.6 Comparison of Link Removal Methods in Minimizing Spread

Recall that our network interdiction models have connectivity-related interdiction objectives. These objectives are different than spread related objectives such as minimizing or slowing down the spread, measured respectively, by the metrics, average new infections and time to infect half of the susceptible nodes. Therefore, to evaluate the effectiveness of these models and their associated heuristic algorithms as link removal methods in minimizing or slowing down the spread, we estimate the average new infections and time to infect half of the susceptible nodes using simulation after removing the links from the network prescribed by these methods. To compare our methods with three existing methods, we estimate the above two metrics for the existing methods also using simulation. Thus, the methods evaluated in this section are:

1. Optimal solution of MINCONNECT / Heuristic algorithm solution of MINCONNECT.
2. Optimal solution of MINATRISK / Heuristic algorithm solution of MINATRISK.
3. Optimal solution of MINPATHS / Heuristic algorithm solution of MINPATHS.
4. Optimal solution of MINWPATHS / Heuristic algorithm solution of MINWPATHS.
5. RANDDEL. In this link removal method, a set of $b$ links are randomly removed from the network. Then, a simulation is run on the residual network to estimate the spread related metrics. The performance of this method is evaluated as the average simulated performance over $M$ replications.
6. GREEDYDEL. In this method, proposed by [76], links are iteratively removed from the network using a metric called the minimum average contamination degree. A link to be evaluated in the remaining network is temporarily removed from the network. Next, a random number is generated for each of the links in the network, and the links having corresponding random numbers greater than the probability of transmission are temporarily removed from the network. Then, the contamination of the link under evaluation is calculated as the total number of susceptible nodes at risk of infection in the remaining network. The above two steps are carried out $M$ times, and a total of $M$ contamination degrees are estimated. The average contamination degree is then calculated as the average of the $M$ contamination degrees. Average
contamination degree is calculated for all the other links in the same way, and the link with the minimum average contamination degree is removed from the network. The algorithm proceeds to the next iteration and recalculates the average contamination degrees of the remaining links. The algorithm terminates after removing a total of \( b \) links.

7. **BetweenDel.** The algorithm removes links in order of an adapted version of betweenness centrality proposed by [47]. The betweenness centrality score \( c(a) \) of a link is a standard metric for evaluating the importance of a link in maintaining the connectivity of the network. This metric is determined by calculating the proportion of shortest paths between two nodes that pass over the link and then summing the proportions over all pairs of nodes in the network. The adapted betweenness measure proposed by [47] is calculated in the same way, but it only includes paths between pairs of one infected and one susceptible node (Equation (2.17)).

\[
c(a) = \sum_{(i,j) \in I \times S} \frac{\phi(i,j|a)}{\phi(i,j)}
\] (2.17)

Here, \( \phi(i,j|a) \) is the number of shortest paths between infected node \( i \) and susceptible node \( j \) on which link \( a \) is one of the links. \( \phi(i,j) \) is the total number of shortest paths between node \( i \) and node \( j \). Then, the heuristic algorithm works as follows. At each iteration, the link among the remaining links having the maximum centrality score is removed from the network. Then, similar to the GreedyDel algorithm, this algorithm also proceeds to the next iteration and recalculates the centrality score of all the remaining links. The algorithm terminates after removing a total of \( b \) links.

To evaluate any of these seven methods on a particular problem instance, we do the following: 1) generate the network, 2) obtain a solution from the method, 3) remove the links from the network that are prescribed by the solution, and then 4) run a simulation of infection spread on the residual network.

We use discrete time stochastic susceptible-infectious (SI) and susceptible-infectious-recovered (SIR) simulations to compare the performances of all the link removal methods.
In both the SI and SIR simulations, at each iteration (tick), a random number is generated for each of the links. Then, all the infected nodes infect their susceptible neighbors if the corresponding random numbers are less than the probability of transmission. In the SI simulation, infected nodes do not recover from infection. Whereas, in the SIR simulation, there is a fixed probability of recovery for the infected nodes. In this simulation, if the random number generated corresponding to an infected node is less than the probability of recovery, that infected node recovers from infection and becomes immune from further infection.

After running a simulation, two performance metrics are estimated: 1) Expected number of new infections ($E(I_{\text{new}})$) and 2) Expected time to infect half of the susceptible nodes ($E(T)$). $E(I_{\text{new}})$ is estimated from the SIR simulation. $E(T)$ is estimated from the SI simulation. $E(I_{\text{new}})$ and $E(T)$ represents the occurrence and speed of the spread of infections, respectively. $E(I_{\text{new}})$ is estimated from the SIR simulation because it is realistic enough to be applicable in most of the spreading scenarios. However, the spread might die out before infecting half of the susceptible nodes if the nodes recover from infection making it difficult to estimate $E(T)$. Hence, $E(T)$ is estimated from the SI simulation.

### 2.6.1 Experimental Setup

A total of 1000 simulation replications were carried out to estimate the performance metrics for each of the different combinations of the following parameters: probability of transmission, initial fraction of nodes infected, fraction of links that can be removed, and
network size (number of nodes). Table 2.1 shows the different values of parameters for which simulations were run.

Table 2.1

Parameters and their values in the experiments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmission probability</td>
<td>0.05, 0.15, 0.25, 0.9</td>
</tr>
<tr>
<td>Initial fraction infected</td>
<td>0.1, 0.2, 0.3</td>
</tr>
<tr>
<td>Fraction of links removed</td>
<td>0.1, 0.2</td>
</tr>
<tr>
<td>Network size</td>
<td>12, 50, 150</td>
</tr>
</tbody>
</table>

We compare the link removal methods both based on the optimal solutions of our models and also based on the solutions using the heuristic algorithms. Recall that the MinPaths and the MinWPaths models can only be solved for small networks. Thus, we use networks of 12 nodes for the comparisons involving optimal solutions and networks of 150 nodes for the comparisons involving heuristic algorithm solutions. We performed the former set of comparisons (Figures 2.10, 2.11, 2.12, 2.13, 2.14, 2.15, 2.16, and 2.17) only on Random networks, but the latter set of comparisons (Figures 2.18, 2.19, 2.20, 2.21) on both Random and Scale-free networks. (See the plots from experiments involving heuristic algorithm solutions on random networks in section 2 of the supplemental material. To be concise, we provide plots for only two parameter combinations in figures 2.18, 2.19, 2.20, 2.21 as opposed to plots for four combinations in figures 2.10, 2.11, 2.12, 2.13, 2.14, 2.15, 2.16, and 2.17. However, the omission does not cause any loss of generality of the associated discussion and conclusion because the omitted plots exhibit similar patterns as the
plots included in the paper.) Networks of 150 nodes are used in the experiments involving Scale-free networks. Note that the legend above figure 2.10 is applicable for all the plots used in the comparisons.

2.6.2 Comparison of the methods

Figures 2.10 (Infected = 10%, Links removed = 10%), 2.11 (Infected = 20%, Links removed = 20%), 2.12 (Infected = 30%, Links removed = 10%), 2.13 (Infected = 30%, Links removed = 20%), 2.14, 2.15 (Infected = 20%, Links removed = 20%), 2.16 (Infected = 30%, Links removed = 10%), 2.17 (Infected = 30%, Links removed = 20%), 2.18 (Infected = 20%, Links removed = 20%), 2.19 (Infected = 30%, Links removed = 20%), 2.20 (Infected = 30%, Links removed = 20%), 2.21 (Infected = 30%, Links removed = 20%) show how the relative effectiveness of the link removal methods vary with respect to transmission probability. This plots are created using normalized values of $E(I_{NEW})$ and $E(T)$. The value in the denominator is the maximum taken across the seven methods for each combination of parameter settings. Models are solved to optimality using CPLEX. Random network, n=12.

It is clear from figures 2.10, 2.11, 2.12, 2.13, 2.18, 2.19 that the MINATRISK model is much more effective than all the other methods in minimizing the spread of infections when both the transmission probability and the fraction of initially infected nodes are not very low, and the relative effectiveness of the MINATRISK model increases monotonically with transmission probability irrespective of the problem configuration and topological characteristics. It is also clear from these figures that as the fraction of infected nodes in-
creases relative to the fraction of links that can be removed, the transmission probability beyond which the MINATRISK model is more effective than other methods decreases. For example, in figure 2.18, 0.1 is the transmission probability beyond which the MINATRISK model is the most effective, and in figure 2.19 that transmission probability is 0.05. The ratio \( \frac{\text{Infected Links removed}}{\text{Links removed}} \) is equal to 1 (20\% \div 20\%) in figure 2.18 and 1.5 (30\% \div 20\%) in figure 2.19. Both of these findings suggest that when the infection is highly virulent or too many nodes are infected, probability is very high that infection will reach a susceptible node if the susceptible node is not completely isolated from the infected nodes because there are too many high probability transmission paths through which infection can transmit to the susceptible node.

Figures 2.10, 2.11, 2.12, 2.12, 2.18, 2.19 also show that when the transmission probability is very low, MINWPATHS is the most effective in minimizing the spread of infections. It suggests that the infected nodes recover before spreading infections through paths of low transmission probability. Thus, only the paths of highest transmission probability need to be removed. GREEDYBET and GREEDYDEL methods are comparable with the MINWPATHS model in most of the scenarios with low transmission probability. One possible explanation for the similarity of the performances of the GREEDYBET model and the MINWPATHS model is that both of them are calculating two different types of centrality of a link. The GREEDYBET model calculates centrality considering the paths of highest weight between all pairs of infected and susceptible nodes, whereas the MINWPATHS model calculates centrality considering the paths of highest weight regardless of whether the paths represent all infected-susceptible node pairs or not. One possible reason for the
similarity between the GREEDYDEL model and the MINWPATHS model is that both of these models consider transmission probability as a criterion for evaluating a link.

![Effectiveness of the methods in minimizing spread](image)

**Figure 2.10**

Effectiveness of the methods in minimizing spread

Figures 2.14, 2.15, 2.16, 2.17, 2.20, 2.21 demonstrate that the MINWPATHS model is the most effective in slowing down the spread when the probability of transmission is not very high. Effectiveness of the GREEDYDEL and the GREEDYBET methods are the closest to the MINWPATHS model for low transmission probabilities. However, as the probability of transmission increases, effectiveness of the MINWPATHS model drops and becomes worse than the GREEDYBET and GREEDYDEL methods in most scenarios. The effectiveness of the GREEDYBET method is not particularly consistent. In some scenarios, it is the second or the third most effective method at low transmission probabilities and the most effective method at high transmission probabilities (Figures 2.14, 2.15, 2.16, 2.17,
Figure 2.11
Effectiveness of the methods in minimizing spread 2

Figure 2.12
Effectiveness of the methods in minimizing spread 3
Effectiveness of the methods in minimizing spread

and 2.20). But, in some other scenarios, the effectiveness of this method is worse than most of the other methods (Figure 2.21). The MINATRISK model, which is very effective in minimizing the spread of infections, is inferior in slowing down the speed of spread of infections in most scenarios. Inferior performance of the MINATRISK model is due to the fact that it spends all the resources needed to isolate a susceptible node without taking into account the probability of infection transmitting to that node. However, the MINATRISK model performs quite well in figure 2.21. This is an indication that when there are many infected nodes and not enough links can be removed, complete isolation of the susceptible nodes is a good method also to slow down the spread.

Figures 2.18 (Infected = 20%, Links removed = 20%), 2.19 (Infected = 30%, Links removed = 20%), 2.20 (Infected = 20%, Links removed = 20%), and 2.21 (Infected = 30%, Links removed = 20%) show the effectiveness of the link removal methods to minimize
Figure 2.14
Effectiveness of the methods in slowing down spread 1

Figure 2.15
Effectiveness of the methods in slowing down spread 2
Figure 2.16
Effectiveness of the methods in slowing down spread 3

Figure 2.17
Effectiveness of the methods in slowing down spread 4
the average number of new infections (models are solved by heuristic algorithms) and the effectiveness in slowing down the spread. Scale-free network with n = 150 are used.

The patterns of the plots in figures 2.10, 2.11, 2.12, 2.12, 2.14, 2.15, 2.16, 2.17 are similar to their counterparts in figures 2.18, 2.19, 2.20, and 2.21. Recall that the figures 2.10, 2.11, 2.12, 2.12, 2.14, 2.15, 2.16, 2.17 are generated using the optimal solutions (by CPLEX) of our models, whereas, the figures 2.18, 2.19, 2.20, and 2.21 are generated using the solutions by the heuristic algorithms. The fact that the patterns are similar is another validation of the effectiveness of the heuristic algorithms. Moreover, the similarity of the patterns in figures 2.10, 2.11, 2.12, 2.12, 2.14, 2.15, 2.16, 2.17 with the patterns in figures 2.18, 2.19, 2.20, and 2.21 suggests that the relative effectiveness of the link removal methods are not highly sensitive to the topological characteristics of the networks. See the random network counterparts of figures 2.18, 2.19, 2.20, and 2.21 in the supplemental material. However, the superiority of the MINATRISK model with respect to the other models in minimizing the number of new infections is even clearer in the Scale-free network. This is an indication of the fact that Scale-free networks have a few highly connected nodes and links, thus making it relatively easy for the MINATRISK model to isolate many susceptible nodes.

2.7 Conclusion

This paper investigates the problem of removing a set of links from a network to minimize the spread of infections. For that purpose, we developed four network interdiction models and formulated them as mixed-integer programs. The interdiction models optimize
Figure 2.18
Effectiveness of the methods in minimizing spread 5

Figure 2.19
Effectiveness of the methods in minimizing spread 6
Figure 2.20

Effectiveness of the methods in slowing down spread 5

Figure 2.21

Effectiveness of the methods in slowing down spread 6
four different connectivity-based interdiction metrics. We also proposed heuristic algorithms for the models. Then, we compared our methods along with random link removal, a link removal method proposed by [76], and a method based on modified betweenness centrality proposed by [47] in minimizing the spread of infections. The spread of infection metrics used in the comparison are average number of new infections and average time to infect half of the susceptible nodes.

We found that when the probability of transmission is moderate to high and infected nodes can recover from infection (SIR simulation), the most effective method in minimizing the number of new infections is to remove links in order to minimize the number of susceptible nodes at risk of infection (MINATRISK model). This method is also the most effective when a large fraction of the nodes are infected, and only a small number of links can be removed. Thus, when the infection is highly virulent or prevalent, as many susceptible nodes as possible should be completely isolated from the rest of the nodes. The effectiveness of this method relative to other methods increases with the probability of transmission and the ratio between the fraction of nodes initially infected and the fraction of arcs removed. Possible reason for the inferior performance of the MINATRISK model at low transmission probability is that it spends all the resources needed to completely isolate a node even if it is far from the infected nodes. In contrast, when infected nodes do not recover from infection (SI simulation) and the transmission probability is low to moderate, the most effective method in slowing down the spread of infections is to remove links to reduce the total weight of the transmission paths between infected and susceptible nodes (MINWPATHS model). The MINWPATHS, GREEDYBet, and the GREEDYDel models
show similar performance under many scenarios because all of them are based on different types of link centrality. Therefore, intervention policies should be based on removing the paths of highest transmission probabilities to slow down the spread for relatively less virulent infections. This intervention of removing paths of highest transmission probability can be quite useful at the beginning of an outbreak to allow some time before other interventions become available. The effectiveness of this link removal method relative to other methods increases as the probability of transmission decreases. Probability of transmission is clearly an important parameter influencing the effectiveness of the link removal methods.

Results also reveal the computation tractability of the models and algorithms. For one, the MINCONNECT model for 125 node problems can be solved to optimality within a reasonable time (2 hours). In addition, the MINATRISK-Z formulation can be solved to optimality for 150 node problems for most parameter combinations and for 200 node problems for several parameter combinations within 2 hours. The equivalent mixed-integer non-linear programming formulation proposed by [47] can only be solved to optimality for much smaller networks.

Our heuristic algorithms for the MINCONNECT and MINATRISK models can solve problem instances with a network size of 200 nodes in 11 minutes which is less than 2 hours taken by the approximate algorithm proposed by [47]. The heuristic algorithms for the MINPATHS and MINWPATHS models are even faster taking less than a minute to solve a problem with 200 nodes. On average, more than 60% of the heuristic algorithm solutions for the MINCONNECT model are optimal, and more than 70% of the heuristic
algorithm solutions for the MINATRISK model are optimal. Average optimality gaps of the heuristic solutions for the MINPATHS and MINWPATHS models are both less than 5%.

In developing our methods, we assume that our knowledge about the topology and other attributes such as the probability of transmission on the links is complete. These assumptions might limit the direct applicability of our methods in real life settings. However, our methods can still be useful in conceptual analysis of the characteristics of spread of infections under different scenarios. Various questions related to how infection will spread with respect to different network topologies, probabilities of transmission, budget, prevalence of infection, etc., can be answered using the methods of this paper. We recommend our methods to be used more as tools for exploration than for decision making.

For future work, more sophisticated algorithms should be developed to solve larger problems with performance guarantees. More efficient cutting plane algorithms with novel valid inequalities should be developed to solve the mixed-integer programming models. In order to make our methods more realistic, one or more of the assumptions perfect knowledge about the probability of transmission should be relaxed in future works. In a real life setting, usually both the node-based and link-based interventions are available at the same time. Thus, this paper can be extended to combined node and link-based infection control. It will be interesting to study the spread of infection on a transportation network using our methods. For example, it is not always possible to prevent individual-individual interactions, but interactions of metapopulations such as the connection between two airports can be prevented. It will be interesting to study the link removal problem considering the network of metapopulations along with the assumption of homogeneous mixing inside each
metapopulation. Also, determining the topology of a transportation network is expected
to be much easier than determining a human contact network. A study such as this may
produce better results compared to the existing studies such as the study by [90] because
it will capture the initial infection status of the metapopulation, demographic properties
of the metapopulation, and also due to the fact that our interdiction models perform bet-
ter than the betweenness centrality based models as demonstrated in this paper. Finally,
more experiments should be performed on real life data sets including data sets from other
applications such as the interdiction of a terrorist network.
CHAPTER 3
INTERDICTING ATTACK GRAPHS TO PROTECT ORGANIZATIONS FROM CYBER ATTACKS: A BI-LEVEL DEFENDER-ATTACKER MODEL

3.1 Introduction

In order to increase operational efficiency and functionality, organizations, individuals, and devices are becoming more and more connected, spawning new phenomena such as the “Internet of Things.” The result is the increasing vulnerability of information to theft and even disruption of services provided by critical infrastructures. According to the FBI Internet crime report for 2013 [49], more than 260,000 individuals reported complaints about their accounts being compromised with a total adjusted loss of more than $781 million. This loss is an increase of 48.8% from $581 million in 2012. In addition to the plethora of low-profile cyber attacks, there have been several recent high-profile attacks on corporations such as JPMorgan Chase & Co. and Target. The cyber attack on Target in 2013 resulted in the theft of 40 million credit card numbers and 70 million different pieces of personal information [128], and the attack on JPMorgan in 2014 compromised information from about 76 million households and 7 million small businesses [154]. Therefore, maximizing the security of cyber systems is becoming one of the most important tasks of IT teams in many organizations.
The security of a network can be enhanced by hardening selected components of the network in various ways, such as deploying firewalls and intrusion prevention and/or detection systems, finding and patching vulnerabilities, and making configuration changes. One of the tools for analyzing the security of a network and finding ways to deploy countermeasures is the attack graph, which contains all the paths that can be used to penetrate a network to breach critical nodes (goal nodes). In this paper, we develop models and algorithms based on the attack graph of an organization to optimally deploy security countermeasures to protect its informational assets.

[119] first proposed attack graphs as an analysis tool. Attack graphs can be generated manually or automatically using the following inputs: a database of common attacks broken into atomic steps, information on network topology and configuration, and an attacker profile [119]. [144] later proposed an automatic attack graph generation tool. Since then, many researchers have studied attack graphs and proposed a multitude of attack graph variations including attack trees [155], attack countermeasure trees [129], defense trees [22], and exploit dependency graphs [115]. For a detailed review of many other attack graph variations, refer to the studies by [86] and [79]. Regardless of the type, an attack graph can be aggregated to different levels such as a graph of hosts, or a graph of subnets using different underlying network regularities [114]. As a result, the models and algorithms developed in this paper are also applicable to a physical computer network in which some of the computers are designated as critical assets, and the attacker tries to breach the critical assets starting from some noncritical assets and using some other critical or noncritical as-
sets along the way. In fact, only a few studies so far analyze security based on the physical computer network rather than attack graphs [9, 166].

Our work complements the existing information security research on attack graph cut set generation. A cut set can be either complete or partial. Based on the specific study, a complete cut set, which is also a minimum cut set, is usually defined as the set of initial security conditions, a set of exploits, or a set of arcs in general; the removal or hardening of such a cut set also removes all the paths to the goal nodes. In contrast, a partial cut set is usually generated considering a limited budget for the defender, and the cut set removes only the most important subset of paths to the goal nodes. [3] propose a formal cost model that estimates the cost if a critical node is breached. Their cost model can be used to select the minimum set of initial security conditions, which upon removal, also removes all the paths from initial conditions to the goal conditions. They also propose an approximation algorithm to find the minimum set. [7] develop an approach based on genetic algorithm to find the minimum cut set in dependency attack graphs. This is one of the few studies that considers arc removal as opposed to node removal. In this paper, we consider that it might not be optimal to deploy countermeasures on the minimum cut set because both the defender and the attacker have limited budgets. The study by [68] is another that uses arc removal for network hardening. [40] propose a multi-objective optimization model to select a subset of security-hardening measures that minimizes the total damage and the total cost of security hardening. Their work is one of the few studies that constrain the total amount spent on security measures. [114] develop a mechanism to cover (i.e., completely
protect) an attack graph by placing the fewest number of Intrusion Detection Systems (IDS) sensors.

This work fits into the field of network interdiction, a subset of network optimization field. The complete or partial removal of an arc is in general referred to as the interdiction of that arc in the network interdiction literature. The problem in this work is a network interdiction problem because the goal is to generate a partial cut set of arcs to be interdicted (completely removed). In fact, our work is able to utilize some of the existing network interdiction ideas and methodologies. At the same time, our work also extends the network interdiction literature by providing an effective new model and algorithm and applying them in a new manner. Our model and algorithm can also be adapted for variations of existing network interdiction applications, such as interdiction of a nuclear weapons project to maximize the minimum completion time of the project, [25, 125], interdicton of an electric power grid to assess the vulnerability [132, 133], monitoring drinking water supply for quick detection of contamination [17, 18, 19, 101, 100, 152], interdiction in hazardous materials transportation to minimize the discharge damage and transportation cost [163], and interdiction of a nuclear smuggling network [99, 109, 116, 142]. From the modeling perspective, our work is most closely related to the work by [116]; in both their work and our work, the network is interdicted to prevent some flow from reaching a set of destinations. However, [116] interdict arcs in a nuclear material smuggling network to minimize a stochastic maximum reliability path, and our work interdicts arcs in an attack graph to minimize the maximum damage that can be caused through breach of critical assets (goal nodes) by an attacker. Although the selection of an origin destination pair by
an attacker in their work is stochastic, they assume that the attacker selects only a single origin-destination pair in a specific realization. In our work, an attacker can use any one or more origins (initially vulnerable nodes) to initiate attacks and breach any one or more destinations (goal nodes).

In this paper, we study the strategic interaction between the defender of an organization and an attacker, modeling a two-player resource allocation game over an attack graph. Related to our work, [41] propose a multi-objective optimization model that poses the interaction between the attacker and the defender as an arms race. They use a genetic algorithm to solve the defender’s problem and the attacker’s problem, and they use competitive co-evolution to solve the combined problem to find the Nash equilibrium. [170] propose a new automated response approach called the Response and Recovery Engine. In this engine, adversaries are modeled as opponents in a two-player Stackelberg stochastic game.

To the best of our knowledge, most of the existing literature related to cut set generation analyzes attack graphs composed of only one goal node. None of these studies consider multiple goal nodes that each have different costs of breach. Goal nodes are indeed different; for example, the loss due to the breach of a database server of an e-commerce portal that contains credit card information is probably not the same as the loss due to the breach of an internal mail server. None of these studies consider intermediate nodes as goal nodes. In reality, merging all the attack graphs of an organization produces intermediate goal nodes. Thus, inclusion of intermediate nodes as goal nodes can increase flexibility both in generation and analysis of attack graphs because the merged attack graphs can be
analyzed without removing the intermediate goal nodes. There is a scarcity of rigorous mathematical models that analyzes the attack graphs, especially the mathematical programming models, as most of the models are logic-based ad hoc models [30, 40]. There is also scarcity of literature that incorporates both the interest of the attacker and the interest of the defender in an attack graph setting. The existing studies are applicable only to the specific graph types used as the analysis platforms. Also, to the best of our knowledge, all of the proposed algorithms solving the attacker-defender games on attack graphs are either heuristic or simulation based. Since most of the studies develop methods to ensure complete security, only a few of the previous studies consider a limited budget for the defender, and none of the previous studies consider limited budgets for both the defender and the attacker.

In this paper, we develop an attacker-defender bi-level network interdiction model and formulate it using mixed-integer linear programming. We provide two alternative formulations for the inner problem. Our model has binary variables in both of the levels. As a result, there is no easy way to merge the two levels into a single-level formulation; thus, the bi-level formulation cannot be solved directly using a commonly used mixed-integer programming solver. Moreover, although bi-level programming models are aplenty in the network interdiction literature [10, 19], there are not many bi-level programming models with binary variables in both levels [25]. Therefore, we develop a customized exact algorithm to solve the model. [5] provide a comprehensive discussion on the modeling and algorithmic strategies for quantifying the resilience of infrastructure systems to disruptive events. Our work is different from theirs in two ways: 1) their operational models are
mainly based on different types of network flow, whereas ours is based on reachability from a set of source nodes to a set of destination nodes, 2) their problems are functionally opposite from ours in the sense that they try to maximize the performance whereas we try to minimize the reachability. Our algorithm is based on the algorithmic framework proposed by [25] and subsequently generalized by [110]. We also propose several enhancements to the base algorithm. We show through experimentation that our algorithm is capable of solving relatively large problems for different parameter combinations. We argue that with further enhancements to our algorithm, along with implementation of the existing graph simplification mechanisms [10, 63, 69, 85, 114], which reduces graph size by up to 99% in some cases, our algorithm has potential to solve even larger problems. We also show that both of the inner problem formulations perform well under specific parameter settings.

The rest of the paper is organized as follows. Section 3.2 describes the optimization problem studied in this paper: select a subset of arcs to minimize the maximum breach loss. Section 3.3 provides the model formulations. Section 3.4 discusses the base algorithm, the formulations of the master problem, and several enhancements to the base algorithm. Section 3.5 describes the plan of experiments, explains the procedure for generating the data sets, and provides the outputs from the experiments. Section 3.6 discusses the experimental outputs and the resulting insights. Finally, section 3.7 concludes the paper.

3.2 Problem Description

A node in an attack graph might represent several different objects including an attack state, a security condition, a vulnerability, or an exploit. On the other hand, an arc might
represent a change of state caused by an atomic action or attack by an attacker, exploit, etc. If the nodes in an attack graph represent security conditions, the tail node of an arc is a pre-condition, and the head node of that arc is a post-condition of that pre-condition. In the attack graphs used in this paper, a node represents a security condition, and an arc represents an attacker action or an exploit. An attacker action on any one of the pre-condition nodes of a post-condition node is enough to activate the post-condition node. This is in contrast with some variations of attack graphs in which some of the post-conditions are connected to pre-conditions via an AND logical gate; that is, the attacker must take action on all or a specific subset of the pre-conditions to activate a post-condition.

Given an attack graph, an attacker tries to maximize the total reward by compromising as many high-value goal nodes as possible starting from the initially vulnerable (initial security condition) nodes. A single path from an initially vulnerable node to a goal node is an attack path, and one or more attack paths constitute an attack plan. The defender or owner of the network selectively thwarts a subset of attacker actions or disables a subset of exploits. On the other hand, if the graph is a computer network, the defender selectively blocks a subset of the communication links by deploying countermeasures such as firewalls. These subsets of arcs constitute an interdiction plan. The defender observes the results of different interdiction plans and attempts to minimize the maximum total reward possible for the attacker by interdicting or protecting the best subset of the arcs. We refer to this as the optimal interdiction plan of the defender. The defender’s and the attacker’s problems are represented in the outer and inner levels, respectively, in the bi-level formulation of the problem, which we call the MINMAXBREACH problem. Both the attacker and
the defender have limited budgets. Loss associated with different nodes, cost of attacks through different arcs, and cost of security countermeasures to interdict different arcs are all different. We use protection and interdiction of an arc interchangeably in this paper. Interdicting an arc is the same as protecting it from being attacked. Moreover, when a goal node is breached, the attacker gains a reward, and the defender suffers a loss equal in magnitude to the reward. Thus, the terms “reward” and “loss” used for a goal node also refer to the same quantity. We also refer to an initially vulnerable node as a vulnerability node to be concise.

Let the attack graph be denoted as $G = (N, A)$, where $N$ is the set of nodes, and $A$ is the set of arcs. $N$ is partitioned into three subsets: $N_I$ is the set of vulnerability nodes, $N_R$ is the set of transition nodes, and $N_T$ is the set of goal nodes. Nodes in $N_T$ can act also as transition nodes. An attacker initiates his attacks using one or more of the nodes in $N_I$, continues his attacks by visiting one or more transition nodes in $N_R$, and then culminates his attacks by reaching one or more goal nodes in $N_T$, upon which he receives some reward. We assume that after a goal node is breached once, complete damage is done, and no further reward can be gained by attacking it again. Thus, we do not allow attacking a goal node more than once in the model. To move from one node to another, the attacker must attack using the arc connecting the two nodes, incurring a cost to attack. However, he incurs this arc attack cost only once even if the arc is used on more than one attack path. The defender may also incur a cost to employ a countermeasure on an arc to protect it from being attacked.
Figure 3.1 shows an example attack graph. Information inside the nodes includes the labels of the nodes and the losses due to breach separated by a hyphen (e.g., the label “6-5” denotes node 6, which has a loss of 5 if breached). Salmon nodes with dashed outlines are the vulnerability nodes, light blue nodes are the transition nodes, and green nodes are the goal nodes. To make the example simple, assume that the cost of countermeasures and attacks is 1 for all of the arcs in this graph. Also, assume that the budgets of the attacker and the defender are 3 and 0, respectively. So, the attacker attacks without any interdiction by the defender. In this case, an optimal attack plan for the attacker is to attack goal nodes 5 and 7 using the attack paths \{(0,3),(3,5)\} and \{(0,3),(3,7)\}, resulting in a total reward of 35. Now, assume that the defender’s budget is 1, and the defender interdicts arc (3,5). In this case, an optimal attack plan is to attack goal nodes 4 and 5 using the attack paths \{(0,2),(2,4)\} and \{(0,2),(2,5)\}, resulting in a total reward of 25. By interdicting arc (3,5), the defender minimizes the maximum total reward achievable by the attacker. Note that in the first attack plan, the attacker uses arc (0,3) to breach two goal nodes. However, the attacker incurs the arc attack cost only once to attack the goal nodes using this arc. The same applies for arc (0,2) for the second attack plan.

3.3 Mathematical Formulations

In this section, we formulate the MINMAXBREACH problem as a bi-level mathematical programming model. The outer level represents the defender, and the inner level represents the attacker. Tables 3.1 and 3.2 list the notation used in the rest of this paper.
Figure 3.1

Example attack graph

Table 3.1

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L(i)$</td>
<td>Set of arcs leaving node $i$</td>
</tr>
<tr>
<td>$E(i)$</td>
<td>Set of arcs entering node $i$</td>
</tr>
<tr>
<td>$c^b_t$</td>
<td>Loss due to breach of a goal node $t \in N_T$</td>
</tr>
<tr>
<td>$c^d_{ij}$</td>
<td>Cost of deployment of countermeasures on arc $(i, j) \in A$</td>
</tr>
<tr>
<td>$c^d_{ij}$</td>
<td>Cost of using arc $(i, j) \in A$ in one or more attack paths</td>
</tr>
<tr>
<td>$B_d$</td>
<td>Defender’s budget</td>
</tr>
<tr>
<td>$B_a$</td>
<td>Attacker’s budget</td>
</tr>
<tr>
<td>$M$</td>
<td>Large number enforcing the upper bound on variables</td>
</tr>
<tr>
<td>$c^b_p$</td>
<td>Loss due to breach of a goal node through path $p$</td>
</tr>
<tr>
<td>$A_p$</td>
<td>Set of arcs in path $p$</td>
</tr>
<tr>
<td>$P_k$</td>
<td>Set of paths in attack (iteration) $k$</td>
</tr>
</tbody>
</table>
### Table 3.2

#### Variables

<table>
<thead>
<tr>
<th>Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z_t$</td>
<td>[ \begin{cases} 1 &amp; \text{if goal node } t \text{ is breached} \ 0 &amp; \text{otherwise} \end{cases} ]</td>
</tr>
<tr>
<td>$x_{ij}$</td>
<td>[ \begin{cases} 1 &amp; \text{if countermeasures are deployed on arc } (i, j) \ 0 &amp; \text{otherwise} \end{cases} ]</td>
</tr>
<tr>
<td>$y_{ij}$</td>
<td>[ \text{Number of attacks through arc } (i, j) ]</td>
</tr>
<tr>
<td>$y_{ij}^f$</td>
<td>[ \begin{cases} 1 &amp; \text{if goal node } t \text{ is attacked using arc } (i, j) \ 0 &amp; \text{otherwise} \end{cases} ]</td>
</tr>
<tr>
<td>$w_{ij}$</td>
<td>[ \begin{cases} 1 &amp; \text{if arc } (i, j) \text{ is used for one or more attacks} \ 0 &amp; \text{otherwise} \end{cases} ]</td>
</tr>
<tr>
<td>$u_p$</td>
<td>[ \begin{cases} 1 &amp; \text{if path } p \text{ is removed} \ 0 &amp; \text{otherwise} \end{cases} ]</td>
</tr>
</tbody>
</table>

The objective of the outer problem is to minimize the maximum reward achievable by the attacker. The outer problem MINMAXBREACH is formulated as follows.

\[
\min \quad f(X) \tag{3.1}
\]
\[
\sum_{(i,j) \in \mathcal{D}} c_{ij}^d x_{ij} \leq B_d \tag{3.2}
\]
\[
x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in \mathcal{D} \tag{3.3}
\]

The function $f(X)$ in (3.1) computes the maximum total reward achievable by the attacker for a given countermeasure vector $X$. Constraint (3.2) ensures that the total countermeasure cost does not exceed the budget.

The objective of the inner problem is to maximize the total reward possible by breaching the goal nodes given the values in the countermeasure vector $X$ from the outer problem.
We refer to the following formulation of the inner problem as MAXBREACHBM. The BM at the end of MaxBreachBM indicates the presence of big $M$ in this formulation.

\[ f(X) = \max \sum_{t \in N_T} c_t^b z_t \quad (3.4) \]

\[ z_t - \sum_{(i,t) \in E(i)} y_{it} = 0 \quad \forall t \in N_T \quad (3.5) \]

\[ \sum_{(i,j) \in L(i)} y_{ij} - \sum_{(j,i) \in E(i)} y_{ji} = 0 \quad \forall i \in N_R \quad (3.6) \]

\[ z_t \leq 1 \quad \forall t \in N_T \quad (3.7) \]

\[ y_{ij} \leq M w_{ij} \quad \forall (i, j) \in A \quad (3.8) \]

\[ w_{ij} \leq 1 - x_{ij} \quad \forall (i, j) \in A \quad (3.9) \]

\[ \sum_{(i,j) \in A} c_{ij}^a w_{ij} \leq B_a \quad (3.10) \]

\[ y_{ij} \geq 0 \quad \forall (i, j) \in A \quad (3.11) \]

\[ w_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \quad (3.12) \]

The objective function (3.4) calculates the total reward acquired by the attacker. Constraint (3.5) enforces that if at least one of the arcs going into a goal node is attacked, then the goal node is attacked. Constraint (3.6) ensures that an attack does not stop after visiting a transition node. Constraint (3.7) ensures that a goal node is not attacked more than once. Constraint (3.8) makes sure that if an arc is not used, no attack can be carried out through that arc. Intuitively, the attacker does not need to carry out more attacks through an arc than there are goal nodes; therefore, we set $M = |N_T|$. Constraint (3.9) ensures that an arc is not used by the attacker if the arc is interdicted by the defender. Constraint (3.10) ensures that the attacker does not spend more than the budget. Constraints (3.8) and (3.12) together en-
sure that even if more than one attack is carried out through arc \((i, j)\), the attacker incurs the cost of attacking through \((i, j)\) only once. Notice that the \textsc{MaxBreachBM} formulation resembles a network design problem, especially because of constraint (3.8) accompanied by constraints (3.5) and (3.6). Constraints (3.5) and (3.6) are equivalent to the flow balance constraints of a network design problem.

Notice that constraint (3.5) in its current form does not allow intermediate goal nodes. Constraint (3.5) is reformulated as follows to allow intermediate goal nodes.

\[
    z_t = \sum_{(i,t) \in E(t)} y_{it} - \sum_{(t,i) \in L(t)} y_{ti} \quad \forall t \in N_T \tag{3.13}
\]

Initial computational experiments using our algorithm (see section 3.4) on small problems showed that the majority of computation time spent solving the problem is spent solving the inner problem. Initial investigation into the inner problem solutions also shows that its LP relaxations are not tight because of the big \(M\) in constraint (3.8) despite using \(|N_T|\) as the value of \(M\). Therefore, in an effort to remove the big \(M\), we reformulate the inner problem, which we refer to as \textsc{MaxBreachD} (the D at the end of \textsc{MaxBreachD} indicates that this is a disaggregated version of the \textsc{MaxBreachBM} formulation). In fact, \textsc{MaxBreachD} resembles a multi-commodity network design problem where goal nodes are analogous to the commodities. To be concise, we commonly refer to the \textsc{MaxBreachBM} and the \textsc{MaxBreachD} formulations as the \textsc{MaxBreach} formulation whenever that is appropriate. The solution of \textsc{MaxBreach} produces an attack plan, \(\bar{A}\), for the attacker, that is, \(w_{ij} = 1 \forall (i, j) \in \bar{A}\). \textsc{MaxBreachD} is formulated as follows.
The objective function (3.14) of MAXBreachD is the same as the objective function (3.4) of MAXBreachBM. Also, constraints (3.18), (3.20), (3.21), and (3.23) in MAXBreachD are directly equivalent to constraints (3.7), (3.9), (3.10), and (3.12), respectively, in MAXBreachBM. Constraint (3.15) enforces that if attack is sent for goal node $t$ using at least one of the arcs going into the goal node, then the goal node is attacked. Constraint (3.16) ensures that any attack sent for a goal node $t$ does not stop at any other goal node. Constraint (3.17) requires that an attack does not terminate at a transition node. Constraint (3.19) has the same meaning as constraint (3.8) except that now it is
enforced for attacks sent for each of the goal nodes. Because one attack, at most, can be sent for a goal node, constraint (3.19) does not require a big $M$, unlike constraint (3.8) in \textsc{MaxBreachBM}.

However, \textsc{MaxBreachD} eliminates the big $M$ at the expense of adding a large number of variables and constraints because the number of $y_{ij}'$ variables is much higher than the number of $y_{ij}$ variables in \textsc{MaxBreachBM}. The index $t$ in $y_{ij}'$ means that attacks through an arc are now disaggregated into attacks aimed at each of the goal nodes. Also, \textsc{MaxBreachD} has constraints for all $t$ in constraints (3.16) and (3.17). Refer to sections 3.5 and 3.6 for a comparative analysis of the performances of \textsc{MaxBreachBM} and \textsc{MaxBreachD} formulations.

\section{3.4 Solution Approach}

Bi-level mixed-integer programs with binary variables only in the outer level can be converted to a single level by taking the dual of the inner level and merging the inner level with the outer level [157, 66]. Because our bi-level formulation has binary variables in both levels, the inner problem has a nonzero duality gap, requiring it to take a different approach. Thus, we develop a customized algorithm to solve our model. The framework of our algorithm is motivated by the algorithm in [25]. The algorithm requires two models to generate the upper and lower bounds. We refer to the model generating the upper bound as the sub-problem and the model generating the lower bound as the master-problem. The algorithm alternates between the master-problem and the sub-problem in an effort to reduce the optimality gap (i.e., the gap between the two bounds) at every iteration. The
master-problem finds an optimal interdiction plan for a given set of alternative attack plans generated by the sub-problem so far, whereas the sub-problem finds an optimal attack plan for a given interdiction plan generated by the master-problem.

The solution of the sub-problem (MAXBREACH) for a feasible interdiction plan produces an upper bound. However, there is no easy way to generate a good lower bound. Thus, to generate the lower bound, we adapt a model (MINATRISK) from a study by [104] on interdicting networks to prevent the spread of infections.

3.4.1 Theorem 1 (NP-hard)

The attacker problem (MAXBREACH) is NP-hard.

Proof: If there are only two levels in an attack graph, the initially vulnerable nodes in the upper level and the goal nodes in the lower level, the attack graph simplifies into a directed bipartite graph as shown in Figure 3.2.

![Figure 3.2](image-url)

**Figure 3.2**

Bipartite attack graph.

The attacker problem on a bipartite graph is a binary knapsack problem. Let, $s_t$ be a binary variable representing the breach of a goal node $t$ in an attack plan. If $s_t = 1$, goal
node \( t \) is breached, and it costs the attacker an amount, \( c_t^m = \min(c_t^a) \). Then, the binary knapsack attacker problem can be formulated as follows: 
\[
\max \sum_{t \in N_T} c_t^b s_t : \sum_{t \in N_T} c_t^m s_t \leq B_a, \quad s_t \in \{0, 1\} \ \forall t \in N_T.
\]
We know that the binary knapsack problem is NP-hard. Therefore, the attacker problem on a general attack graph is NP-hard.

### 3.4.2 Lemma 1

An attack is a tree in the attack graph.

**Proof:** In a residual graph after interdiciton, every node can be breached through zero or more minimum cost paths. The attacker incurs additional cost by breaching any node through more than one path because of using additional arcs. However, the attacker does not gain any additional reward. Thus, in an attacker solution, any node will always be breached through at most one path which will result in a tree. The set of constraints (3.6) and (3.7) in the MAXBREACHBM formulation, and the set of constraints (3.16), (3.17), and (3.18) in the MAXBREACHD formulation ensure that any node is breached at most once in an attacker solution.

### 3.4.3 Upper Bound

We refer to the MAXBREACH model for a specific interdiction plan \( \hat{X} \) as MAXBREACH\( \hat{X} \). The optimal objective value of MAXBREACH\( \hat{X}^k \) is an upper bound for our algorithm at iteration \( k \). Then, the optimal objective value of MAXBREACH\( \hat{X}^k \) is

\[
U(\hat{X}^k) = \max_{w,y} f(\hat{X}^k).
\]  

(3.24)
3.4.4 Lower Bound

The MINATRISK model in [105] minimizes the number of susceptible nodes at risk of infection from infected nodes. The idea in the MINATRISK model is that for a specific interdiction plan of the defender, if the attacker is able to build a path from any of the vulnerability nodes to a goal node, that goal node is at risk of breach. We reformulate the MINATRISK model as follows in an effort to improve it before using the model for our purpose. In our reformulation, we also incorporate the fact that the attacker has a limited budget as opposed to no attacker budget restriction in the MINATRISK model. We refer to the new formulation as MINBREACHNODE\(A^k\) generated at iteration \(k\) of the algorithm (presented below). Here, \(A^k\) is the set of arcs used by the attacker to carry out attack at iteration \(k\) of the algorithm. Suppose \(L(A^k)\) is the optimal objective value, and \(\hat{X}^k\) is the optimal solution of MINBREACH\(A^k\). Then, \(\hat{X}^k\) is the new interdiction plan generated at iteration \(k\) of the algorithm. \(L(A^k)\) is the lower bound at the \(k^{th}\) iteration because \(\hat{X}^k\) is generated considering only a subset of the alternative attack plans.
L(A^k) = \min \eta \quad (3.25)

\eta \geq \sum_{i \in N^k_i} c^b_i \cdot z^k_i \quad \forall k \in K \quad (3.26)

z^k_j \geq z^k_i - x_{ij} \quad \forall (i, j) \in A^k, \forall k \in K \quad (3.27)

z^k_i = 1 \quad \forall i \in N^k_i, \forall k \in K \quad (3.28)

\sum_{(i, j) \in A} c^d_{ij} x_{ij} \leq B_d \quad (3.29)

z^k_i \geq 0 \quad \forall i \in N^k, \forall k \in K \quad (3.30)

x_{ij} \in \{0, 1\} \quad \forall (i, j) \in \bigcup_{k \in K} A^k \quad (3.31)

The objective function (3.25) and the constraint (3.26) ensure that the objective of this model is the maximum total reward acquired by the attacker from all the attacks generated through iteration \(k\). Constraint (3.27) ensures that if a node \(i\) is at risk of breach in attack (iteration) \(k\), and there exists an arc \((i, j)\), this arc must be interdicted for node \(j\) to not be at risk of breach through node \(i\) in this attack. All the arcs entering into node \(j\) which have tail nodes that are at risk of breach must be interdicted for node \(j\) to be saved from breach. According to constraint (3.28), vulnerability nodes used in attack \(k\) are already at risk of breach in this attack. Constraint (3.2) is the defender’s budget constraint. According to Lemma 3.4.2, an attack is a tree in the attack graph. A distinct set of nodes and arcs represent the associated attack tree, and to differentiate an attack from any other in the model, a new set of variables are generated for the nodes used in this attack, and a new set of constraints (3.27) and (3.28) represent the connectivity among these nodes. Note that a new variable \(z^k_j\) is generated for a node \(j\) if it is used in the attack \(k\).
Constraint (3.27) performs a similar function as the optimality cuts in Benders decomposition [16] in that it forces the master-problem to move closer to the optimal solution at each iteration. However, whereas each optimality cut in Benders decomposition cuts away a fraction of the feasible region of the master-problem, each constraint in (3.27) actually creates a new feasible region in a higher dimension by including more and more new variables (arcs). When a sufficient number of variables and constraints has been included, the solution of the master-problem is the optimal solution.

3.4.4.1 Theorem 1

The master problem (MINBREACHNODE) provides a valid lower bound.

Proof: According to Lemma 3.4.2, an attack is a tree in the attack graph. Constraints (3.26), (3.27), and (3.28) from a specific attack in the MINBREACHNODE formulation above adds the attack tree in the master problem. All the distinct attack trees are represented separately in this formulation, thereby implicitly taking the attacker budget into consideration. The MINBREACHNODE formulation adds \( k \) trees through iteration \( k \). Because of adding only a subset of all the possible alternative attack trees through \( k \), the objective value of the MINBREACHNODE model provides a lower bound to the defender problem.

3.4.5 Algorithm MINMAX

Input: Parameter values for MINMAXBREACH and tolerance \( \varepsilon \geq 0 \).

Output: Subset of arcs \( X^* \) on which to deploy countermeasures with a maximum optimality gap of \( \varepsilon \).

1. Upper bound, \( UB := \infty \), lower bound \( LB := 0 \), current interdiction plan, \( X^* := \hat{X}^1 := 0 \), iteration counter \( k := 1 \).
2. Given \(\hat{\mathbf{X}}^k\), solve the sub-problem \(\text{MAXBREACH}(\hat{\mathbf{X}}^k)\) to determine the attacker’s optimal attack plan, \(\hat{\mathbf{W}}^k\) and the associated \(\mathcal{M}^k\).

3. If \(U(\hat{\mathbf{X}}^k) < UB\), \(UB := U(\hat{\mathbf{X}}^k)\). Set \(\hat{\mathbf{X}}^K\) as the new best interdiction plan. \(X^* := \hat{\mathbf{X}}^k\).

4. If \(UB - LB \leq \epsilon\), go to END.

5. Generate new variables \(z^k_i\) for the nodes used in the current attack plan. Add constraints (3.27) corresponding to the set of nodes and arcs used in this attack.

6. Solve \(\text{MINBREACH}(A^k + 1)\). If \(L(A^k + 1) \geq LB\), \(LB := L(A^k + 1)\) and \(X^* := \hat{\mathbf{X}}^k + 1\). If \(UB - LB \leq \epsilon\), go to END.

7. \(k = k + 1\), go to 2.

8. END: return \(X^*\) as the \(\epsilon\)-optimal solution.

### 3.4.5.1 Lemma 1

The master problem produces a new solution at each iteration until convergence.

Proof: Let us assume that the master problem solution from iteration \(k\) is the same as the master problem solution from a prior iteration \(q\). Then, the master problem objective value from iteration \(k\) will be at least as large as the sub-problem objective value from iteration \(q\). This can only be true if the sub-problem objective value from iteration \(q\) equals the current upper bound. The upper bound and the lower bound becomes equal at this point, and the algorithm terminates. Hence, the master problem must produce a new solution at each iteration partially or completely interdicting all the attack plans generated so far until convergence.

### 3.4.5.2 Theorem 1

The algorithm \(\text{MINMAX}\) converges within a finite number of iterations.

Proof: According to Lemma 3.4.5.1, the master problem will keep producing a new solution at each iteration until the last iteration. If there are \(K\) possible alternative interdiction
plans, in the worst case, the algorithm will run through \( K \) iterations adding all the possible attack plans associated with the \( K \) interdiction plans. At this point, the algorithm must converge because of exhausting all the possibilities. Therefore, the MinMax algorithm will terminate within a finite number \((\mathbb{K})\) of iterations. In reality, the algorithm usually terminates within a small fraction of the \( \mathbb{K} \) maximum iterations.

\[\blacksquare\]

### 3.4.6 Algorithm Example

Figures 3.3, 3.4, 3.5, 3.6, 3.7, and 3.8 demonstrate how the algorithm works using the example attack graph in Figure 3.1. The defender’s budget is 1, and the attacker’s budget is 3 in this demonstration. See Figure 3.1 for the original colors of the nodes and their meanings. The dark blue solid arcs in figure 3.3, 3.4, 3.5, 3.6, 3.7, and 3.8 are either used to attack (in the attacker’s solutions) or not interdicted by the defender (in the defender’s solution). The dashed arcs in the defender’s solutions are interdicted by the defender. The gray arcs in the attacker’s solutions are not used by the attacker. The input graph in each of the sub-figures related to the attacker’s solutions is the whole graph in that sub-figure with all the arcs in dark blue solid except the arcs interdicted by the defender in the previous iteration. The input graphs in each of the sub-figures related to the defender’s solutions are the whole graphs in that sub-figure with all the arcs in dark blue solid.

At iteration 3, the algorithm terminates as the lower bound (25) and the upper bound (25) become equal. An optimal interdiction plan is to interdict arc (3,7), and a corresponding optimal attack plan is to attack using the set of arcs \{(0,2),(2,5),(2,4)\} comprised of the attack paths \{(0,2),(2,5)\} and \{(0,2),(2,4)\}.
Figure 3.3

Iteration 1: Attacker’s solution, Upper bound = 35

Figure 3.4

Iteration 1: Defender’s solution, Lower bound = 0
Figure 3.5

Iteration 2: Attacker’s solution, Upper bound = 35

Figure 3.6

Iteration 2: Defender’s solution, Lower bound = 15
Notice from figures 3.4, 3.6, and 3.8 that the number of sub-graphs input into the MinBreach model is growing at each iteration. Eventually, the MinBreach model finds an optimal interdiction plan by solving for a sufficiently large number of sub-graphs of the original graph.

### 3.4.7 Enhancements to the MinMax Algorithm

Computational experiments show that the basic MinMax algorithm takes too long to terminate for graphs larger than 100 nodes. The reason is that the computation time of the master problem increases exponentially because of adding a large number of variables and constraints at each iteration.

#### 3.4.7.1 Path Based Formulation of the Master Problem (S)

We exploit the fact that an attacker solution can be represented by a distinct set of paths instead of a distinct set of nodes and arcs as in the MinBreathNode formulation. At each
iteration of the algorithm, we run a search algorithm on the attacker solution to find the set of paths used in this attack. If a path found is new, a binary variable and associated constraints are generated and added to the master problem. If a path was used in a previous attack, we just attach the path to the current attack. Following is the path based formulation of the master problem, \( \text{MINBREACHPATH}(A^k) \).

\[
L(A^k) = \min \eta \\
\eta \geq \sum_{p \in P_k} c_p^b (1 - u_p) \quad \forall k \in K \tag{3.32}
\]

\[
u_p \leq \sum_{(i, j) \in A_p} x_{ij} \quad p \in \bigcup_{k \in K} P_k \tag{3.33}
\]

\[
u_p \leq 1 \quad p \in \bigcup_{k \in K} P_k \tag{3.34}
\]

\[
\sum_{(i, j) \in A_c} c_{ij}^d x_{ij} \leq B_d
\]

\[
u_p \geq 0 \quad p \in \bigcup_{k \in K} P_k \tag{3.36}
\]

\[
x_{ij} \in \{0, 1\} \quad \forall (i, j) \in \bigcup_{k \in K} A^k \tag{3.37}
\]

The right hand side of constraint (3.33) calculates the total attacker reward from all the attacks through iteration \( k \). Objective function (3.32) and constraint (3.33) together ensure that the maximum attacker reward is minimized. Constraints (3.34) and (3.35) together ensure that a path is not removed if none of the arcs on the path is not removed. Constraint (3.36) is the defender budget constraint. Constraints (3.37) and (3.38) are the sign restriction and binary constraints, respectively. The \text{MINBREACHPATH} formulation usually has a significantly less number of variables and constraints than the \text{MINBREACHNODE} for-
mulation. Hence, the computational efficiency of the MinBreachPath formulation is much superior to that of the MinBreachNode formulation.

### 3.4.7.2 Add Multiple Sub-Problem Solutions to the Master Problem (Ms)

If only one sub-problem solution is added to the master problem solution, only a slightly different master problem is solved at each iteration requiring the algorithm to run through many iterations to provide the master problem enough sub-problem information for convergence. To overcome this issue along with the fact that Gurobi (commercial solver used in this paper) is able to return multiple optimal and sub-optimal solutions from a solution, we add multiple sub-problem solutions to the master problem at each iteration. Adding multiple sub-problem solutions indeed reduces the number of iterations and the average computation time. Experiments show that adding 33% of the available solutions produces good results.

### 3.4.7.3 Stabilize Master Problem Solutions (TR)

One problem with the MinBreach formulation is that it produces very divergent solutions at initial iterations of the algorithm slowing the convergence of the algorithm. In an effort to stabilize the master problem solution, we added trust region cut at the initial iterations (20) of the algorithm. Suppose \( \hat{x}_{ij} \) is the master problem solution from iteration \( k \) and \( \hat{X}^k = \{(i, j) : \hat{x}_{ij} = 1\} \). Then, we add the following trust region cut to the master problem at the next iteration.
\[
\sum_{(i,j) \not\in \hat{X}^k} x_{ij} + \sum_{(i,j) \in \hat{X}^k} (1 - x_{ij}) \leq 0.33 \times 2 \times |\hat{X}^k| \tag{3.39}
\]

The left hand side of constraint (3.39) calculates the Hamming distance [56] between the interdiction plan from iteration \( k \) and the interdiction plan from iteration \( k + 1 \). The right hand side of constraint (3.39) ensure that a maximum of one-third of all the arcs interdicted at the current iteration is replaced at the next iteration. Master problem with the trust region cut does not provide a valid lower bound. Thus, we update the lower bound only after we stop adding the trust region cut. Experiments show that although the impact of adding trust region cut is not significant, the average computation time is slightly lower with the trust region cut in the master problem.

### 3.4.7.4 Apply Heuristic to Solve the Master Problem (Hf)

We greedily select a set of arcs to be removed using the following steps at each iteration.

1. Initialize, \( X_h = \emptyset, t\text{Budget} = 0 \).

2. Evaluate a metric, \( \text{Score}_{ij} = \frac{\text{Saved}_{ij}}{\text{SecurityCost}_{ij}} \) for each of the arcs not selected yet and the removal of the arc does not exceed the budget. Here, \( \text{Saved} \) is the difference between current maximum total reward gained by the attacker and the maximum reward gained if the arc is removed. And, \( \text{SecurityCost} \) is the arc security cost.

3. Select the arc \((i, j)\) with the maximum \( \text{Score} \) calculated in step 2 and add to \( X_h \). \( t\text{Budget} = t\text{Budget} + \text{SecurityCost}_{ij} \).

4. If \( t\text{Budget} < B_d \), go to step 2, else return \( X_h \).

The solution \( X_h \) generated by the above heuristic can be used on its own as the master problem solution. \( X_h \) can also be used as a warm start to the master problem at each iteration. We apply the later approach along with limiting the number of nodes explored
in the master problem solution to 1 as our heuristic method. We refer to this heuristic as
the heuristic method with master heuristic. Exploration of the master problem solution
can be limited to 1 node without applying the master problem heuristic. We refer to the
latter method as the heuristic method without master heuristic. Neither of the heuristic
methods is expected to produce a valid lower bound. Nevertheless, we use the master
problem solution from the heuristic methods as the lower bound for the convergence of the
algorithm keeping in mind that the final solution might not be optimal.

3.5 Computational Experiments

In this section, we perform experiments to ascertain the effects of the model parameters
and different topological attributes of attack graphs on the computation times and the loss
due to breach. All the experiments are performed on synthetic attack graphs generated
using the approach described in the following paragraph. All the experiments are carried
out on a laptop with an Intel core i7 2.70 GHz processor and 16 GB RAM. Gurobi [53] is
used to solve both the MAXBREACH and the MINBREACH problems.

In most of the attack graphs in the literature, nodes are organized in a hierarchical
structure with nodes in one level having directed arcs incident to nodes in the subsequent
level [85]. A level usually means the level of access acquired into the system. Thus, if
an attack path has five levels, an attacker will have to acquire five successive levels of
access, possibly by using five different exploits, in order to reach a goal node. This is
very similar to attack trees except that any node might have more than one incoming arc,
making it a graph. Keeping that in mind, we generate the graphs for this paper randomly
using the following approach. The whole set of nodes are arbitrarily divided into \( r \) levels. All the nodes in the first level are designated as vulnerability nodes, and all the nodes in the last level are designated as the goal nodes. Thus, all the nodes in the intermediate levels are transition nodes. In this way, the distance between the vulnerability nodes and the goal nodes equals \( r - 1 \) levels. First, directed arcs are generated randomly from the first level to the second level. Arcs are generated in a similar manner from all the levels to their subsequent levels, and no arcs are generated in the opposite direction. Second, arcs are also generated randomly between any pair of nodes within a level. Finally, incoming arcs are generated randomly to each of the transition and goal nodes without any incoming arc after the first two steps. Tail nodes of the arcs generated in the final step are selected randomly from the corresponding prior levels. The final step makes sure that all the transition and goal nodes have at least one incoming arc. Any of the vulnerable nodes will not have any incoming arcs. Some of the goal nodes will have outgoing arcs because of the second step.

We used values of 2, 5, 7, and 10 for the parameter \( r \). Figure 3.9 shows a five-level graph generated using our approach.

Table 3.3 shows the different parameters and their values used in the computational experiments. We perform experiments on 4 different network sizes (defined by the number of nodes) to show how the computation time is impacted by the size of the graph. The number of arcs in each of the graphs is approximately 2.15 times the number of nodes. Three and two different level values are used for graphs with 50 nodes and 100 nodes, respectively to examine the variation of computation time with respect to the number of levels. Loss due to the breach of goal nodes, the costs of attack on arcs, and the costs of
countermeasures on arcs are generated using uniform distribution with different parameters values.

Table 3.3
Parameters and their values in the experiments.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network size (nodes)</td>
<td>50, 100, 150, 200</td>
</tr>
<tr>
<td>Network size and number of levels combinations (size, levels)</td>
<td>(50,5), (50,7), (50,10), (100,2), (100,5), (150,5), (200,5)</td>
</tr>
<tr>
<td>Arcs</td>
<td>≈2.15×Nodes</td>
</tr>
<tr>
<td>Network sizes and defender budget combinations (size, low budget, intermediate budget, high budget)</td>
<td>(50, 75), (100, 100, 150, 250), (150, 275), (200, 375)</td>
</tr>
<tr>
<td>Network sizes and attacker budget combinations (size, low budget, high budget)</td>
<td>(50, 125), (100, 150, 300), (150, 325), (200, 425)</td>
</tr>
<tr>
<td>Loss due to breach of the goal nodes</td>
<td>˜Uniform(500, 1500), ˜Uniform(1000, 2000)</td>
</tr>
<tr>
<td>Cost of attacks on arcs</td>
<td>˜Uniform(10, 30), ˜Uniform(30, 50)</td>
</tr>
<tr>
<td>Cost of countermeasures on arcs</td>
<td>˜Uniform(10, 30), ˜Uniform(30, 50)</td>
</tr>
</tbody>
</table>

In table 3.4, number of levels = 5 and other parameters are at their low levels. Values in italic mean that those graphs cannot be solved in 2 hours using exact method.

Table 3.4 reports the computation times for 4 random instances of each of the network sizes. Experiments in this table used a single level (low) of budgets. The purpose of this table is to show how the computation time increases with the graph size.
Table 3.4

Growth of computation time with graph size.

<table>
<thead>
<tr>
<th>Graph Params</th>
<th>Heuristic Method</th>
<th>Exact Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Heuristic Method</td>
<td>Exact Method</td>
</tr>
<tr>
<td></td>
<td>Iters</td>
<td>Time</td>
</tr>
<tr>
<td>Nodes</td>
<td>Arcs</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>118</td>
<td>22</td>
</tr>
<tr>
<td>50</td>
<td>126</td>
<td>21</td>
</tr>
<tr>
<td>50</td>
<td>119</td>
<td>21</td>
</tr>
<tr>
<td>50</td>
<td>116</td>
<td>21</td>
</tr>
<tr>
<td>100</td>
<td>237</td>
<td>22</td>
</tr>
<tr>
<td>100</td>
<td>235</td>
<td>20</td>
</tr>
<tr>
<td>100</td>
<td>230</td>
<td>20</td>
</tr>
<tr>
<td>100</td>
<td>248</td>
<td>25</td>
</tr>
<tr>
<td>150</td>
<td>374</td>
<td>167</td>
</tr>
<tr>
<td>150</td>
<td>374</td>
<td>123</td>
</tr>
<tr>
<td>150</td>
<td>345</td>
<td>97</td>
</tr>
<tr>
<td>150</td>
<td>352</td>
<td>109</td>
</tr>
<tr>
<td>200</td>
<td>431</td>
<td>230</td>
</tr>
<tr>
<td>200</td>
<td>437</td>
<td>445</td>
</tr>
<tr>
<td>200</td>
<td>425</td>
<td>208</td>
</tr>
<tr>
<td>200</td>
<td>424</td>
<td>131</td>
</tr>
</tbody>
</table>

In table 3.5, master model NodeLimit = 1, Gurobi initial heuristic NodeLimit (sub-MIPNodes) = 2000. In the columns under the heuristic method without master heuristic, master solution was limited to single node exploration, and master problem heuristic was not applied.

Table 3.5 reports the breach losses from the solutions of the exact method, from the solutions of the heuristic method with master problem heuristic, and from the heuristic method without the master problem heuristic. The main purpose of this table is to demonstrate the high quality of the solutions from the heuristic methods.
Table 3.5
Performance of the heuristic method.

<table>
<thead>
<tr>
<th>Graph Params</th>
<th>Exact method</th>
<th>Heuristic method with master heuristic</th>
<th>Heuristic method without master heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>Levels</td>
<td>BreachLoss</td>
<td>masTime</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
<td>3990</td>
<td>0.6</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
<td>4893</td>
<td>1.1</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
<td>2999</td>
<td>0.3</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
<td>3564</td>
<td>0.5</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>2273</td>
<td>0.4</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>1995</td>
<td>0.3</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>1226</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>2343</td>
<td>0.5</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>5122</td>
<td>2.8</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>5097</td>
<td>1.8</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>4749</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>5839</td>
<td>27.3</td>
</tr>
</tbody>
</table>

Table 3.6 reports the average computation times of the MaxBreachBM and the MaxBreachD formulations for the sub-problem. The purpose of this table is to compare the computational performance of the two formulations under different scenarios. Columns 6 and 7 contain average computation times (clock seconds) of the sub-problem at each iteration averaged over a maximum of 10 iterations.
Table 3.6

Performance of the MaxBreachBM and MaxBreachD formulations.

<table>
<thead>
<tr>
<th>Graph Params</th>
<th>Defender Budget</th>
<th>Attacker Budget</th>
<th>Average sub-problem runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>Arcs</td>
<td>Levels</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>118</td>
<td>5</td>
<td>75</td>
</tr>
<tr>
<td>50</td>
<td>126</td>
<td>5</td>
<td>75</td>
</tr>
<tr>
<td>50</td>
<td>119</td>
<td>5</td>
<td>75</td>
</tr>
<tr>
<td>50</td>
<td>116</td>
<td>5</td>
<td>75</td>
</tr>
<tr>
<td>50</td>
<td>117</td>
<td>7</td>
<td>75</td>
</tr>
<tr>
<td>50</td>
<td>116</td>
<td>7</td>
<td>75</td>
</tr>
<tr>
<td>50</td>
<td>112</td>
<td>7</td>
<td>75</td>
</tr>
<tr>
<td>50</td>
<td>109</td>
<td>7</td>
<td>75</td>
</tr>
<tr>
<td>50</td>
<td>117</td>
<td>10</td>
<td>75</td>
</tr>
<tr>
<td>50</td>
<td>124</td>
<td>10</td>
<td>75</td>
</tr>
<tr>
<td>50</td>
<td>126</td>
<td>10</td>
<td>75</td>
</tr>
<tr>
<td>50</td>
<td>123</td>
<td>10</td>
<td>75</td>
</tr>
<tr>
<td>100</td>
<td>237</td>
<td>5</td>
<td>250</td>
</tr>
<tr>
<td>100</td>
<td>235</td>
<td>5</td>
<td>250</td>
</tr>
<tr>
<td>100</td>
<td>230</td>
<td>5</td>
<td>250</td>
</tr>
<tr>
<td>100</td>
<td>248</td>
<td>5</td>
<td>250</td>
</tr>
<tr>
<td>100</td>
<td>220</td>
<td>2</td>
<td>250</td>
</tr>
<tr>
<td>100</td>
<td>225</td>
<td>2</td>
<td>250</td>
</tr>
<tr>
<td>100</td>
<td>236</td>
<td>2</td>
<td>250</td>
</tr>
<tr>
<td>100</td>
<td>224</td>
<td>2</td>
<td>250</td>
</tr>
<tr>
<td>150</td>
<td>374</td>
<td>5</td>
<td>275</td>
</tr>
<tr>
<td>150</td>
<td>374</td>
<td>5</td>
<td>275</td>
</tr>
<tr>
<td>150</td>
<td>345</td>
<td>5</td>
<td>275</td>
</tr>
<tr>
<td>150</td>
<td>352</td>
<td>5</td>
<td>275</td>
</tr>
</tbody>
</table>

Table 3.7 reports the computation times and the objective function values for different combinations of the following parameters: attacker budget, defender budget, loss due to breach of goal nodes, cost of attack on arcs, and cost of countermeasures on arcs. This table is set up as a factorial experiment with two levels of each of the aforementioned parameters.
parameters. The same graph with 100 nodes is used in all the resulting 32 experiments in this table. Nodes=100, arcs=238, Binaries=476. Solved using the heuristic method.
Table 3.7

Computation times (clock seconds) of the algorithm.

<table>
<thead>
<tr>
<th>No.</th>
<th>Defender parameters</th>
<th>Attacker parameters</th>
<th>WithMaxBreachBM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Budget</td>
<td>ArcCost</td>
<td>GoalNodeLoss</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>30-50</td>
<td>1000-2000</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>30-50</td>
<td>1000-2000</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>30-50</td>
<td>1000-2000</td>
</tr>
<tr>
<td>4</td>
<td>250</td>
<td>30-50</td>
<td>1000-2000</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>30-50</td>
<td>1000-2000</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>30-50</td>
<td>1000-2000</td>
</tr>
<tr>
<td>7</td>
<td>250</td>
<td>30-50</td>
<td>1000-2000</td>
</tr>
<tr>
<td>8</td>
<td>250</td>
<td>30-50</td>
<td>1000-2000</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>10-30</td>
<td>1000-2000</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>10-30</td>
<td>1000-2000</td>
</tr>
<tr>
<td>11</td>
<td>250</td>
<td>10-30</td>
<td>1000-2000</td>
</tr>
<tr>
<td>12</td>
<td>250</td>
<td>10-30</td>
<td>1000-2000</td>
</tr>
<tr>
<td>13</td>
<td>100</td>
<td>10-30</td>
<td>1000-2000</td>
</tr>
<tr>
<td>14</td>
<td>100</td>
<td>10-30</td>
<td>1000-2000</td>
</tr>
<tr>
<td>15</td>
<td>250</td>
<td>10-30</td>
<td>1000-2000</td>
</tr>
<tr>
<td>16</td>
<td>250</td>
<td>10-30</td>
<td>1000-2000</td>
</tr>
<tr>
<td>17</td>
<td>100</td>
<td>30-50</td>
<td>500-1500</td>
</tr>
<tr>
<td>18</td>
<td>100</td>
<td>30-50</td>
<td>500-1500</td>
</tr>
<tr>
<td>19</td>
<td>250</td>
<td>30-50</td>
<td>500-1500</td>
</tr>
<tr>
<td>20</td>
<td>250</td>
<td>30-50</td>
<td>500-1500</td>
</tr>
<tr>
<td>21</td>
<td>100</td>
<td>30-50</td>
<td>500-1500</td>
</tr>
<tr>
<td>22</td>
<td>100</td>
<td>30-50</td>
<td>500-1500</td>
</tr>
<tr>
<td>23</td>
<td>250</td>
<td>30-50</td>
<td>500-1500</td>
</tr>
<tr>
<td>24</td>
<td>250</td>
<td>30-50</td>
<td>500-1500</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>10-30</td>
<td>500-1500</td>
</tr>
<tr>
<td>26</td>
<td>100</td>
<td>10-30</td>
<td>500-1500</td>
</tr>
<tr>
<td>27</td>
<td>250</td>
<td>10-30</td>
<td>500-1500</td>
</tr>
<tr>
<td>28</td>
<td>250</td>
<td>10-30</td>
<td>500-1500</td>
</tr>
<tr>
<td>29</td>
<td>100</td>
<td>10-30</td>
<td>500-1500</td>
</tr>
<tr>
<td>30</td>
<td>100</td>
<td>10-30</td>
<td>500-1500</td>
</tr>
<tr>
<td>31</td>
<td>250</td>
<td>10-30</td>
<td>500-1500</td>
</tr>
<tr>
<td>32</td>
<td>250</td>
<td>10-30</td>
<td>500-1500</td>
</tr>
</tbody>
</table>
3.6 Discussion

Figure 3.10 compares the average computation times from the different computational methods based on 4 attack graphs for each of the base or accelerated computation techniques. Nodes = 100, defender budget = 150, and attacker budget = 150. In this figure, All, N, S, SHf, SMs, and STR have the following meanings: All - path based (MINBREACHPATH) method with all the enhancements, N - node based (MINBREACHNODE) method, S - path based method without any enhancements, SHf - path based method with master problem heuristic, SMs - path-based method with the addition of multiple attacker solutions at an iteration, and STR - path based method with the addition of trust region cut to the master problem. This figure clearly demonstrates that the path based method is much superior to the node based method. Average computation time seems to increase slightly if the master problem heuristic is applied with the path based method.

However, table 3.5 shows that the heuristic method with the master problem heuristic usually finds more optimal solutions and higher quality solutions (if not optimal) compared to the heuristic method without the master problem heuristic. Thus, application of master problem heuristic can be very useful when the heuristic method is used. Addition of multiple sub-problem solutions seems to slightly decrease the average computation time. Addition of trust region cut also seems to slightly decrease the average computation time. Application of all the enhancements on the path based method decreases the average computation time compared to the application of only one of the enhancements on the path based method. Note however that the computational methods and the enhancements only impact the computation time of the master problem.
From the *Time* columns in table 3.4, we see that total computation time increases sharply with respect to graph size, especially when the exact method is used. Graphs with 150 and 200 nodes cannot be solved in two hours using the exact method. However, the heuristic method solves most of the 150-node and 200-node graphs in less than 15 minutes.

One important observation is that the time taken by the algorithm to solve the sub-problem is much greater than the time taken to solve the master-problem during the initial iterations of the algorithm. However, as the number of iterations increases, the master problem becomes bigger sharply increasing its computation time. We see from table 3.4 that when the exact method is used, the majority of the total time is spent in solving the master problem. In contrast, when the heuristic method is used, the majority of the total time is spent in solving the sub-problem. Comparision of the masTime columns under the exact and the heuristic methods makes it clear that the heuristic method significantly decreases the computation time of the master problem.

From table 3.5, we see that 9 and 8 out of the 12 solutions using the heuristic method with the master heuristic and the heuristic method without the master heuristic, respectively are optimal. So, a large fraction of the solutions from both of the heuristic methods are optimal. However, the heuristic method with the master heuristic seems to find optimal solutions more frequently and solutions with higher quality if it does not find the optimal solution.

Comparing the computation times of the graphs with the same number of nodes and levels, it is clear that significant variation remains. Although the number of nodes and the levels are the same, their topologies are different because the graphs are generated
randomly. In fact, topology plays an important role not just in computation times but also in the total breach loss.

We see from table 3.6 that the average computation time of the MaxBreachBM formulation increases with the number of levels. Upon examination of the computation times of the 50-node graphs, it is apparent that the computation times of the 50-node graphs with 10 levels are much higher on average than the computation times of the 50-node graphs with 5 levels. Because of the higher number of levels, the average distance between the vulnerability nodes and the goal nodes are much longer in the 10-level graphs than in the 5-level graphs. Thus, the 10-level graphs have more paths than the 5-level graphs, and the paths are longer, making it relatively more difficult for the attacker to decide whether to attack a specific node. When the graph size is significantly large relative to the number of levels, computation time of the MaxBreachBM formulation decreases significantly with graph size. Computation times of the 100-node graphs with 2 levels are significantly smaller than the computation times of the 50-node graphs with 10 levels. Because of the small number of levels, there are relatively more short paths and possibly fewer total paths from vulnerability nodes to the goal nodes in the 100-node graphs with 2 levels, making it easier for the attacker to decide whether to attack some goal nodes. However, increasing the number of levels while keeping the number of nodes unchanged does not monotonically increase the computation time because that makes the graph easier for the defender to defend given the same defense budget.

Average computation time of the \texttt{MAXBREACHD} formulation is smaller than the average computation time of the \texttt{MAXBREACHBM} formulation under many network size
and number of level combinations, especially when the number of levels is large relative to the graph size. However, variation of computation times is also much more for the MaxBreachD formulation. This is probably an indication that the MaxBreachD model is more affected by the topology of the attack graph than the MaxBreachBM model. Computation times of the MaxBreachD formulation are smaller than the computation times of the MaxBreachBM formulation for the 100-node graphs with 2 levels in table 3.6. Recall that for the MaxBreachD formulation, the number of constraints and variables depends on the number of goal nodes, and as the number of goal nodes increases, the number of constraints and variables increases. Because of the smaller number of levels, the 100-node graph with 2 levels have 50 goal nodes instead of 20 goal nodes as in the 100-node graphs with 5 levels. Therefore, we can conclude that the MaxBreachD formulation usually performs worse than the MaxBreachBM formulation in cases when the number of goal nodes is relatively high, or, in other words, the number levels is smaller for the same graph size.

Table 3.7 shows that the heuristic method is able to solve the problem very quickly for most of the parameter combinations. In fact, the median computation time of all the parameter combinations is less than 2 minutes. Therefore, the heuristic method has the potential to solve relatively large problems within a reasonable amount of time, especially for suitable combinations of parameters and topology of the graph.

From table 3.7, by comparing the Breachloss values of the pairs of problem instances in which the defender’s budget is low in one and high in the other (e.g., instances 1 and 17), we see that the total breach loss for the instances with a high defender’s budget is always
lower than the total breach loss for the ones with a low defender’s budget. Similar compar-
isons reveal that the total breach loss increases if the attacker’s budget increases. If the average individual breach loss increases, total breach loss increases. If the average individual countermeasure deployment cost increases, total breach loss also increases. Finally, if the average individual attack cost increases, total breach loss decreases.

Figure 3.11 shows the variation of breach loss with defender’s budget. This chart is generated using five randomly generated graphs with 100 nodes. Graph0, Graph1, Graph2, Graph3, and Graph4 have 228, 232, 228, 221, and 238 arcs, respectively. All the parameters except the defender’s budget are at their low levels. Average breach loss for a specific budget is determined by averaging the breach losses over all the graphs.

Although there are some random variation in the plots of figure 3.11, the common pattern of relationships between the total breach loss and the amount of the defender’s budget is that the breach loss drops sharply for small increases in budget at the beginning, and it then forms a concave shape until the breach loss becomes zero. This relationship becomes even clearer from the plot of the average breach losses. This relationship between the breach loss and the defender’s budget implies that the defender will need to spend much more to reduce per unit of breach loss after the initial set of relatively easy security challenges are successfully confronted. Decision makers should investigate breach losses for a wide range of defense budgets before allocating a specific budget to ensure high return on investment.

Finally, it might not be easy to know or estimate the budget of the attacker correctly. Suppose the attacker’s actual budget is an unknown amount between 100 and 200, but the
Defender’s solution, Lower bound = 25

Table 3.8

Sensitivity of total breach loss to parameter uncertainty.

<table>
<thead>
<tr>
<th>Graphs</th>
<th>Extra Loss (%) for 50% Overestimate</th>
<th>Extra Loss (%) for 25% Underestimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph0</td>
<td>11.34</td>
<td>10.14</td>
</tr>
<tr>
<td>Graph1</td>
<td>27.62</td>
<td>4.93</td>
</tr>
<tr>
<td>Graph2</td>
<td>33.54</td>
<td>3.82</td>
</tr>
<tr>
<td>Graph3</td>
<td>13.1</td>
<td>1.37</td>
</tr>
<tr>
<td>Graph4</td>
<td>21.03</td>
<td>14.5</td>
</tr>
</tbody>
</table>
Figure 3.9

Attack graph generated using our approach

Figure 3.10

Average computation times using different techniques
Figure 3.11

Variation of breach loss with defender’s budget

defender’s estimate of the attacker’s budget is 150. Assume also that the defender’s budget is 250, the other parameters are at their low levels, and the five graphs are the same as those used in figure 3.11. Values in table 3.8 are the extra breach losses due to error in the estimates of the attacker’s budget. The second column corresponds to the extra breach losses due to a 50% overestimate of the attacker’s budget (the actual attacker budget is 100, and the defender’s perception of the attacker’s budget is 150), and the third column corresponds to the extra breach losses due to a 25% underestimate of the attacker’s budget (the actual attacker budget is 200, and the defender’s perception of the attacker’s budget is 150). For example, with respect to Graph0, the defender incurs an extra breach loss of 11.34% due to a 50% overestimate, and an extra breach loss of 10.14% due to a 25% underestimate of the attacker’s budget. The average extra loss is 14.17% corresponding
to an average error of 37.5%. Therefore, the quality of an interdiction plan is relatively insensitive on average with respect to the error in the defender’s knowledge about the attacker’s budget. Moreover, the extra losses due to the underestimate are significantly smaller than the extra losses due to the overestimate of the attacker’s budget. Hence, it is preferrable to have an underestimate rather than an overestimate of the attacker’s budget to be conservative.

3.7 Conclusion

This paper presents the breach of information resources of an organization by an attacker and the defensive measures of the organization as an attacker-defender bi-level network interdiction model. The inner level represents the attacker trying to maximize the total reward, and the outer level represents the defender trying to minimize the maximum total reward achievable by the attacker. Both the inner- and the outer levels are formulated as mixed-integer linear programs. We provide two alternate formulations for the inner problem. As both of the levels have binary variables, we also develop a customized algorithm to solve the model. The path based method with the enhancements is much faster than the node based method. Heuristic method with or without the application of master problem heuristic is capable of solving relatively large problems for various parameter settings. The following are some of the most important findings. 1) Topology is an influential factor in computation times. 2) The computation time of the two sub-problem formulations does not monotonically increase with random graph size. In fact, average computation time of the MaxBreachBM formulation decreases with graph size when the
graph size is significantly large relative to the number of levels. 3) The majority of com-
putation time is spent solving the sub-problem when the heuristic method is used. 4) The 
\text{MAXBREACHD} formulation without big M is a computationally better formulation than
the \text{MAXBREACHBM} formulation with the big M when the number of levels is relatively
high for a specific graph size. 5) Breach loss drops sharply for small increases in the de-
fense budget at the beginning, then levels off before finally dropping sharply again with
increasing defense budget. and 6) Quality of an interdiction plan is relatively insensitive
with respect to the correctness of the attacker’s budget.

In this work, we focused on accelerating the master problem solution. Further research
should emphasize on speeding up the sub-problem solution. Decomposition techniques
such as Benders decomposition and Lagrangian relaxation aided by additional inequalities
should be investigated. This work can be extended to have actual security countermeasures
with multiple levels of defense of arcs (interdiction) rather than binary defense. It will be
valuable to relax the implicit assumptions that the attacker has complete information about
the topology of the attack graph, and that the countermeasures are capable of completely
protecting the arcs. It will also be more realistic to consider the interdiction of arcs as
stochastic; that is, the interdiction effect is probabilistic, and the probability is uncertain
within a range. Finally, it will be worthwhile to perform experiments on real attack graphs
rather than the synthetic ones used in this paper.
4.1 Introduction

In multi-agent problems, including those in defender-attacker settings, it is usually assumed that given the actions chosen by the defender, the attacker is able to completely rationalize her actions and select the best action [5, 26, 57, 116]. In reality, the rationality of the attacker might be bounded, making it impossible for the attacker to accomplish the maximum reward attainable within a set of resource and other constraints. [140] first presented the idea of bounded rationality. [140] argued that rationality can be bounded in different ways such as the uncertainty in parameters, incomplete information about the alternatives, and extreme complexity in evaluating the decisions. [92] explained that the bounded rationality assumption results from the limitations in incorporating uncertainty about future consequences in decision making. [70] argued that decision makers are intendedly rational but fail to materialize complete rationality occasionally due to human cognitive and emotional limitations. [34] discussed four reasons for incorporating bounded rationality in economic models: 1) Empirical evidence is abundant that bounded rationality is important, 2) There are several impressive works showing the importance of bounded rationality, 3) The logic behind the assumption of unbounded rationality is unconvincing,
and 4) Reasoning to find good decisions is a costly activity. [88] is one of the first to include bounded rationality in operations research, incorporating it in the analysis of user behavior in transportation systems.

The impact of incorporating bounded rationality in security games can be significant, because in security games, adversaries are almost always humans. [121] presented a game-theoretic model for stackelberg security game incorporating bounded rationality. The model developed by [121] is able to find a robust solution under many different rationality levels of the attackers. [162] developed new algorithms for finding optimal strategic solutions using prospect theory and quantal response equilibrium. [113] showed that explicitly incorporating human behavior models representing bounded rationality can be more effective than finding robust solutions considering only the least favorable rationality levels.

Our work in this chapter is closely related to the work of [121] in that we also pose the problem as an optimization model that minimizes the maximum potential loss. However, we extend the literature on bounded rationality and the literature on network interdiction in the following ways. To the best of our knowledge, we are the first to incorporate bounded rationality in a stackelberg game in which the game is played on a graph. Our formulation is a tri-level mixed integer program with the outer level representing the defender and the inner levels representing the attacker. All the three levels are formulated as mixed integer programs making this work also the first attempt to incorporate bounded rationality when the attacker problems are integer programs. We propose a novel exact algorithm based on constraint and column generation to solve the model. Therefore, our exact algo-
rithm is also the first to solve a tri-level mixed integer program with integer variables in all levels. We demonstrate through experiments that the algorithm has the potential to solve large problems in a reasonable timeframe. Finally, we provide valuable insights about the characteristics of the problems in which modeling bounded rationality of the attacker is especially important.

4.2 Mathematical Formulations

Let the attack graph be denoted as $G = (N,A)$, where $N$ is the set of nodes, and $A$ is the set of arcs. $N$ is partitioned into three subsets: $N_V$ - the set of vulnerability nodes (initial security conditions), $N_R$ - the set of transition nodes, and $N_T$ - the set of goal nodes. Nodes in $N_T$ can act also as transition nodes. The attacker has a limited budget and is boundedly rational, i.e., given a defense plan, the best solution the attacker is able to determine is limited by a rationality factor which has a value between 0 and 1. If the value of the rationality factor is 0, the attacker is not able to find any solution. In contrast, if the value of the rationality factor is 1, the attacker is completely rational and able to find the optimal solution.

This problem is an example of a Stackelberg game with one leader and one follower, and the game is played as follows. At first, the defender selects a subset of arcs to be interdicted. The attacker initiates his attack using one or more of the nodes in $N_V$, continues his attacks by visiting one or more transition nodes in $N_R$, and then culminates his attacks by reaching one or more goal nodes in $N_T$, upon which he receives some reward. The attacker determines the optimal attack given the interdiction plan, attacker budget, and the
rationality factor. The problem of the defender is to find the optimal subset of arcs which upon interdiction minimizes the maximum reward of the bounded rational attacker. Tables 4.1 and 4.2 define all the parameters and variables used in the formulation.

Table 4.1

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L(i)$</td>
<td>Set of arcs leaving node $i$</td>
</tr>
<tr>
<td>$E(i)$</td>
<td>Set of arcs entering node $i$</td>
</tr>
<tr>
<td>$\Omega$</td>
<td>Set of bounded rational attacks</td>
</tr>
<tr>
<td>$\Delta_{p'}$</td>
<td>Set of all attacks with respect to which $p'$ is bounded rational, $c_{p'}^b \leq \varepsilon c_p^b$</td>
</tr>
<tr>
<td>$\Pi$</td>
<td>Set of all attacks</td>
</tr>
<tr>
<td>$l_t$</td>
<td>Loss due to breach of a goal node $t \in N_T$</td>
</tr>
<tr>
<td>$c_p^b$</td>
<td>Loss associated with an attack $p$</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>Parameter bounding the rationality of an attacker ($1 \geq \varepsilon \geq 0$)</td>
</tr>
<tr>
<td>$c_{ij}^d$</td>
<td>Cost of interdiction of arc $(i, j) \in A$</td>
</tr>
<tr>
<td>$c_{ij}^a$</td>
<td>Cost of attacking through arc $(i, j)$</td>
</tr>
<tr>
<td>$B_d$</td>
<td>Defender’s budget</td>
</tr>
<tr>
<td>$B_a$</td>
<td>Attacker’s budget</td>
</tr>
</tbody>
</table>

The defender’s problem $\text{MNMXLOSS}$ is formulated as follows.

$$\min_x f(x, \varepsilon) \quad (4.1)$$

$$\sum_{(i,j) \in A} c_{ij}^d x_{ij} \leq B_d \quad (4.2)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i,j) \in A \quad (4.3)$$
Objective function (4.1) calculates the maximum loss caused by the bounded rational attacker. Constraint (4.2) enforces the limited budget of the defender. Constraint (4.3) is the binary constraint.

Given an attacker budget and an interdiction plan from, the completely rational attacker’s problem $\text{MaxLossC}(\hat{x}^K)$ (i.e., rationality factor equals 1) is formulated as follows.

\[
\begin{align*}
    f(x, 1) &= \max_{x, w, z} \sum_{t \in N_T} l_t z_t \quad (4.4) \\
    z_j &\leq \sum_{(i,j) \in E(j)} w_{ij} \quad \forall j \in N \setminus N_V \quad (4.5) \\
    z_j &\leq 1 \quad \forall j \in N_T \quad (4.6) \\
    w_{ij} &\leq z_i \quad \forall (i,j) \in A \\
    w_{ij} &\leq 1 - x_{ij} \quad \forall (i,j) \in A \\
    \sum_{(i,j) \in A} c_{ij} w_{ij} &\leq B_a \quad (4.9) \\
    w_{ij} &\in \{0, 1\} \quad \forall (i,j) \in A \quad (4.10)
\end{align*}
\]

Objective function (4.4) calculates the maximum loss that can be caused using a specific attacker budget. Each constraint (4.5) ensures that node $j$ is not attacked if none of its incoming arcs is used to attack $j$. Each constraint (4.6) ensures that a node is attacked at most once. Each constraint (4.7) prevents the attacker from using arc $(i, j)$ if node $i$ is not attacked. Each constraint (4.8) prevents the attackers from using arc $(i, j)$ if arc $(i, j)$ is interdicted by the defender. Constraint (4.9) enforces the limited budget of the attacker. Finally, constraint (4.10) ensures binary usage of arcs.
Then, given a rationality level of the attacker and the optimal objective value of the complete rational attacker from MXLOSSC, the attacker problem with bounded rationality MXLOSSB(\(\hat{x}^K\)) is formulated as follows.

\[
f(x, \varepsilon) = \max_{x,w,z} \sum_{t \in N_T} l_t z_t \tag{4.11}
\]

\[
\sum_{t \in N_T} l_t z_t \leq \varepsilon f(x, 1) \tag{4.12}
\]

Objective function (4.11) is the same as (4.4). Constraint (4.12) is the bounded rationality constraint ensuring that the attacker is not able to find an objective value which is greater than \(f(x, B_a, 1)\) multiplied by the rationality factor \(\varepsilon\).

4.3 Solution Approach

The framework of the algorithm used to solve the problem in this work is the same as the framework of the MINMAX algorithm in [108]. In this framework, a sub-problem which is the the attacker problem provides the upper bound, and a customized master problem provides the lower bound. Solution from the sub-problem is added to the master problem at each iteration using newly created variables and constraints. Addition of the sub-problem solution at each iteration pushes the master problem to implicitly enumerate all the alternative attacks and find the overall optimal solution.
4.3.1 Upper Bound

The optimal objective value of MAXLOSSR(\(\hat{x}^k\)) at iteration \(k\), \(f(\hat{x}^k, \varepsilon)\) is an upper bound for our algorithm at this iteration. MAXLOSSR(\(\hat{x}^k, \varepsilon\)) is the sub-problem for our algorithm. If \(UB\) is the minimum of \(f(\hat{x}^k, \varepsilon)\) found through iteration \(k\),

\[
UB \leq f(\hat{x}^k, \varepsilon)
\]  
(4.14)

4.3.2 Lower Bound

Our model for generating the lower bound which we call the master problem is similar to the models MINBREACHNODE and MINBREACHPATH in [108]. Notice that despite solving for the same objective functions, i.e., minimizing the maximum loss, the MINBREACHNODE and MINBREACHPATH models are developed at different granularity levels. Specifically, in the MINBREACHNODE model, a node is the smallest unit with an associated loss. Whereas, a path is the smallest unit with an associated loss in the MINBREACHPATH model. The master problem of this work is developed at even a higher granularity level. The smallest unit with an associated loss in the new master problem is an attack which is larger than a path. In the MINBREACHNODE model, the goal nodes associated with an attack are combined to represent the loss associated with the attack. In the MINBREACHPATH model, the paths associated with an attack are combined to represent the loss associated with the attack. In the master model of this work, the loss associated with an attack is directly represented by treating the attacks as modeling units. We refer to the new master model as MINMAXLOSSB(\(\Omega^k, \Pi^k\)) generated at iteration \(k\) of the algorithm. Here, \(\Omega^k\) and \(\Pi^k\) are the sets of bounded rational attacks and all attacks, resep-
tively, generated by solving the MXLOSSC and the MXLOSSB problems through iteration $k$ of the algorithm. $\text{MINMAXLOSSB}(\Omega^k, \Pi^k)$ is the master-problem for our algorithm. Suppose, $L(\Omega^k, \Pi^k)$ is the optimal objective value, and $\hat{x}^k$ is the optimal solution of $\text{MINMAXLOSSB}(\Omega^k, \Pi^k)$. Then, $\hat{x}^k$ is the new interdiction plan generated at iteration $k$ of the algorithm. $L(\Omega^k, \Pi^k)$ is a lower bound for our algorithm at iteration $k$ because $\hat{x}^k$ is generated considering only a subset of the alternative attack plans. If $LB$ is the maximum of $L(\Omega^k, \Pi^k)$ found through iteration $k$,

$$LB = L(\Omega^k, \Pi^k)$$ (4.15)

$\text{MINMAXLOSSB}(\Omega^k, \Pi^k)$, the master problem model, is formulated as follows.

$$L(\Omega^k, \Pi^k) = \min_{\eta, y, u, v} \eta$$ (4.16)

s.t.

$$\eta \geq c_{p'^{b}}^b(1 - v_{p'^{b}}) \quad \forall p'^{b} \in \Omega^k$$ (4.17)

$$y_{p'^{b}} \leq u_{p} \quad \forall p'^{b} \in \Omega^k, \ p \in \Delta_{p'^{b}}^k$$ (4.18)

$$v_{p'^{b}} \leq y_{p'^{b}} + u_{p} \quad \forall p'^{b} \in \Omega^k$$ (4.19)

$$u_{p} \leq \sum_{(i,j) \in A_{p}} x_{ij} \quad \forall p \in \Pi^k$$ (4.20)

$$\sum_{\forall (i,j) \in A} c_{ij}^d x_{ij} \leq B_d$$ (4.21)

$$v_{p'^{b}} \leq 1 \quad \forall p'^{b} \in \Omega^k$$ (4.22)

$$u_{p} \leq 1 \quad \forall p \in \Pi^k$$ (4.23)

$$u_{p} \geq 0 \quad \forall p \in \Pi^k$$ (4.24)

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A$$ (4.25)
Each constraint (4.17) calculates the loss from bounded rational attack $p'$, and all constraints (4.17) together ensure that the minimized loss objective is at least as large as the maximum of the losses through iteration $k$. The set of constraints (4.18) for each bounded rational attack $p'$ ensures that the complete rational attacks $p \in \Delta_{p'}^k$ can make $p'$ unattainable by the attacker if all $p \in \Delta_{p'}^k$ are interdicted. Each pair of constraints (4.19) and (4.22) ensures that a bounded rational attack $p'$ can be made unattainable by the attacker if either all $p \in \Delta_{p'}^k$ are interdicted, or $p'$ itself is interdicted. Each pair of the constraints (4.20) and (4.23) for an attack $p$ together ensure that at least one of the arcs on the attack must be interdicted to interdict the attack. Constraint (4.21) is the defender budget constraint. Constraints (4.24) and (4.25) are the sign restriction and binary constraints, respectively. Any solution of the MinMaxLossB formulation produces binary values for the $u_p, v_{p'}, y_{p'}$ variables without the binary domain restriction on these variables.

MinMaxLossB($\Omega^k, \Pi^k$) is the master problem formulation used to find the bounded rational solution. The MinMaxLossC formulation below is the equivalent master problem formulation used to find the complete rational solution. We use this formulation instead of the MinBreachPath formulation in the experiments to compare the computational times for the bounded rational problem and the complete rational problem. MinMaxLossC formulation is used instead of MinBreachPath formulation for the comparison because MinMaxLossC formulation is a direct special case of MinMaxLossB formulation, thus provides a fairer comparison.
\[ L(\Omega^k, \Pi^k) = \min_{\eta, u, x} \eta \]
\[ \text{s.t.} \quad \eta \geq c_p^b(1 - u_p) \quad \forall p \in \Pi^k \]
\[ u_p \leq \sum_{(i,j) \in A_p} x_{ij} \quad \forall p \in \Pi^k \]
\[ \sum_{(i,j) \in A} c_{ij}^d x_{ij} \leq B_d \]
\[ u_p \leq 1 \quad \forall p \in \Pi^k \]
\[ u_p \geq 0 \quad \forall p \in \Pi^k \]
\[ x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \]

4.3.3 Algorithm MinBMMax

Column and constraint generation algorithm.

Input: Values of all the parameters and tolerance \( \delta \geq 0 \).

Output: Subset of arcs \( x^* \) to interdict with a maximum optimality gap of \( \delta \).

1. Upper bound, \( UB := \infty \), lower bound \( LB := 0 \), current interdiction plan, \( x^* := \hat{x}^k := 0 \), iteration counter \( k := 1 \).

2. Given defender solution \( \hat{x}^k \), solve the sub-problem \( \text{MXLOSSC}(\hat{x}^k) \) to determine the complete rational attack plan, \( p^k \) and the associated objective function value \( f(\hat{x}^k, 1) \). Compare \( p^k \) with each \( p \in \Pi^{k-1} \). If \( p^k \) is bounded rational with respect to \( p \), add \( p \) into \( \Delta_{p^k} \), and add \( p^k \) into \( \Omega^k \). If \( p \) is bounded rational with respect to \( p^k \), add \( p^k \) into \( \Delta_{p} \), and if \( p \notin \Omega^{k-1} \), add \( p \) into \( \Omega^k \). Add \( p^k \) into \( \Pi^k \).

3. Given defender solution \( \hat{x}^k \) and \( f(\hat{x}^k, 1) \), solve the sub-problem \( \text{MXLOSSB}(\hat{x}^k) \) to determine the bounded rational attack plan, \( p'^k \) and the associated objective function value \( f(\hat{x}^k, \epsilon) \). Compare \( p'^k \) with each \( p \in \Pi^{k-1} \). If \( p'^k \) is bounded rational with respect to \( p \), add \( p \) into \( \Delta_{p'^k} \), and add \( p'^k \) into \( \Omega^k \). If \( p \) is bounded rational with respect to \( p'^k \), add \( p'^k \) into \( \Delta_{p} \), and if \( p \notin \Omega^{k-1} \), add \( p \) into \( \Omega^k \). Add \( p'^k \) into both \( \Omega^k \) and \( \Pi^k \).

4. Calculate \( U(\hat{x}^k) = f(\hat{x}^k, \epsilon) \). If \( U(\hat{x}^k) < UB \), \( UB := U(\hat{x}^k) \). Set \( \hat{x}^k \) as the new best interdiction plan. \( x^* := \hat{x}^k \).
5. If $UB - LB \leq \delta$, go to END.

6. Create the new variables $u_p \in \Pi^k \setminus \Pi^{k-1}$, $y_{p'} \in \Omega^k \setminus \Omega^{k-1}$, $v_{p'} \in \Omega^k \setminus \Omega^{k-1}$ and constraints corresponding to the new variables.

7. Solve $\text{MINMAXLOSSB}(\Omega^k, \Pi^k)$. $\hat{x}^{k+1}$ is the new interdiction plan. If $L(\Omega^k, \Pi^k) \geq LB$, $LB := L(\Omega^k, \Pi^k)$ and $x^* := \hat{x}^{k+1}$. If $UB - LB \leq \delta$, go to END.

8. $k = k + 1$, go to 2.

9. END: return $x^*$ as the $\delta$—optimal solution.

### 4.4 Results and Discussion

In this section, we run experiments to analyze the computational efficiency of the algorithm $\text{MINBMAX}$. We also develop insights about both the impact of bounded rationality on the potential losses and the characteristics of problems on which bounded rationality has an impact. We run experiments on both the bounded rational algorithm and the complete rational algorithm. To find the bounded rational objective value from the complete rational solution, we simply apply the optimal interdiction plan generated by the complete rational algorithm on $\text{MXLOSSB}$ and find $f(\hat{x}, \epsilon)$. The experiments are performed on the same set of 50 and 100 nodes synthetic attack graphs as those used in [108]. The experiments are carried out on a laptop with an Intel core i7 2.70 GHz processor and 16 GB RAM. The algorithm is implemented in Python 3.4 with Gurobi [53] as the solver for all the embedded mixed integer programs (i.e., master and subproblems).
Table 4.3

Computation times with and without bounded rationality.

<table>
<thead>
<tr>
<th>Graph size</th>
<th>Budgets</th>
<th>Bounded rational</th>
<th>Complete rational</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BObj</td>
<td>Iterations</td>
</tr>
<tr>
<td>Node</td>
<td>Arc</td>
<td>Bd</td>
<td>Ba</td>
</tr>
<tr>
<td>50</td>
<td>118</td>
<td>75</td>
<td>125</td>
</tr>
<tr>
<td>50</td>
<td>126</td>
<td>75</td>
<td>125</td>
</tr>
<tr>
<td>50</td>
<td>119</td>
<td>75</td>
<td>125</td>
</tr>
<tr>
<td>50</td>
<td>116</td>
<td>75</td>
<td>125</td>
</tr>
<tr>
<td>100</td>
<td>237</td>
<td>100</td>
<td>150</td>
</tr>
<tr>
<td>100</td>
<td>235</td>
<td>100</td>
<td>150</td>
</tr>
<tr>
<td>100</td>
<td>230</td>
<td>100</td>
<td>150</td>
</tr>
<tr>
<td>100</td>
<td>248</td>
<td>100</td>
<td>150</td>
</tr>
</tbody>
</table>

Table 4.3 shows both the computation times and bounded rational objective function values of both the bounded rational solution (uses the bounded rational algorithm) and complete rational solution (uses the complete rational algorithm). Rationality factor = 0.7. According to table 4.3, the maximum time taken by the bounded rational solution is 73.3 seconds for a problem with 100 nodes. This table also shows that the bounded rational solution takes significantly longer total time than the complete rational solution. The values in the SubCTime columns under the bounded rational solution and complete rational solutions are comparable, meaning that the time to solve the complete rational sub problems using the two algorithms are not very different. In contrast, the master problem computation time (Mtime) for the bounded rational solution is much greater than the master problem computation time for the complete rational solution. Moreover, the bounded rational solution has one additional time component which is the time to solve the bounded rational sub problem (SubBTime). Table 4.3 shows that the time to solve the bounded ra-
tional sub problem is significantly longer than the time to solve the complete rational sub problem

Although the 50 and 100 nodes problems are solved reasonably quickly, there is an order of magnitude growth in the computation time from the 50 node problems to 100 node problems, meaning that the algorithm, without any speedup mechanism, might not be able to solve significantly larger problems in a reasonable timeframe. One aspect of the bounded rational algorithm which might be inflating the computation time is the significantly larger number of iterations compared to the MINMAX algorithm in [108]. Experiments show that the lower bound remains 0 for most of the iterations of the algorithm except only in the last few iterations. One idea to improve the lower bound will be to constrain the sub problems to not generate solutions that are highly overlapping in the set of arcs used. Improving the lower bound will most probably decrease the number of iterations needed for the convergence of the algorithm which in turn will reduce the total computation time.

Figure 4.1
Example 1
Table 4.2

Variables

<table>
<thead>
<tr>
<th>Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z_i$</td>
<td>(\begin{cases} 1 &amp; \text{Node } i \text{ is breached} \ 0 &amp; \text{otherwise} \end{cases})</td>
</tr>
<tr>
<td>$x_{ij}$</td>
<td>(\begin{cases} 1 &amp; \text{if arc } (i, j) \text{ is interdicted} \ 0 &amp; \text{otherwise} \end{cases})</td>
</tr>
<tr>
<td>$w_{ij}$</td>
<td>(\begin{cases} 1 &amp; \text{if arc } (i, j) \text{ is used for one or more attacks} \ 0 &amp; \text{otherwise} \end{cases})</td>
</tr>
<tr>
<td>$u_p$</td>
<td>(\begin{cases} 1 &amp; \text{if attack } p \text{ is interdicted} \ 0 &amp; \text{otherwise} \end{cases})</td>
</tr>
<tr>
<td>$v_{p'}$</td>
<td>(\begin{cases} 1 &amp; \text{if bounded rational attack } p' \text{ is interdicted} \ 0 &amp; \text{otherwise} \end{cases})</td>
</tr>
<tr>
<td>$y_{p'}$</td>
<td>(\begin{cases} 1 &amp; \text{if all attacks } p \in \Delta_p' \text{ is interdicted} \ 0 &amp; \text{otherwise} \end{cases})</td>
</tr>
<tr>
<td>$\eta$</td>
<td>Maximum potential loss in the residual graph</td>
</tr>
<tr>
<td>$x, w, y, u, v, z$</td>
<td>Vector representations of the variables $x_{ij}, w_{ij}, y_{p'}, u_p, v_{p'}, z_i$, respectively</td>
</tr>
</tbody>
</table>

Figure 4.2

Example 2
One important observation from table 4.3 is that, in seven out of eight experiments, the complete rational solution and bounded rational solution produced the same bounded rational objective function value (BObj). In this work, we are investigating the change in defender’s interdiction plan and the consequent change in the loss suffered by the defender due to bounded rationality, and the prior observation suggests that bounded rationality rarely has an impact. However, a deeper look into the reasons for the equal bounded rational objective values from the complete rational and bounded rational solutions reveals that it is indeed quite easy to come up with many examples with a high impact of bounded rationality. We created the six example graphs in figures (4.1), (4.2), (4.3), (4.4), (4.5), and (4.6) to demonstrate the characteristics of problems in which bounded rationality has an impact. To simplify the discussion, we assume that both the budgets of the defender and the attacker are equal to 1, which makes each of the problem having 7 non-overlapping attacks. The bounded rationality factor value is 0.7.

The examples in figures 4.1, 4.2, 4.3, 4.4, 4.5, and 4.6 are again used to characterize the problems having impact of bounded rationality. Csol and Bsol in the SolType column
Table 4.4

Examples with and without the impact of bounded rationality.

<table>
<thead>
<tr>
<th>Examples</th>
<th>SoType</th>
<th>Attack 1</th>
<th>Attack 2</th>
<th>Attack 3</th>
<th>Attack 4</th>
<th>Attack 5</th>
<th>Attack 6</th>
<th>Attack 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Csol</td>
<td>30</td>
<td>20</td>
<td>13</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Bsol</td>
<td>30</td>
<td>20</td>
<td>13</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Csol</td>
<td>30</td>
<td>24</td>
<td>19</td>
<td>15</td>
<td>12</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Bsol</td>
<td>30</td>
<td>24</td>
<td>19</td>
<td>15</td>
<td>12</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Csol</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>17</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Bsol</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>17</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Csol</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>17</td>
<td>10</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Bsol</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>17</td>
<td>10</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>Csol</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>17</td>
<td>15</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Bsol</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>17</td>
<td>15</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>Csol</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Bsol</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>17</td>
</tr>
</tbody>
</table>

Mean complete rational solution and bounded rational solution, respectively. In a specific solution (row), the cell highlighted in blue indicates the interdicted attack, and the cell in yellow indicates the optimal bounded rational attack. The number in a cell is the loss from that attack. For clarity, attacks 1–7 are sorted in descending order from left to right.

According to table 4.4, for examples 1 and 2, bounded rational objective values from the bounded rational solution and the complete rational solution are equal. On the other hand, for the remaining four examples, bounded rational objective values are different from those two solutions. Notice that the complete rational solution always interdicts the attack with the largest objective function value because, if not interdicted, the complete rational attacker will always choose this attack. Notice also that in example 1, the maximum bounded rational attacks without and with interdiction are different, and those are attack 2 without interdiction and attack 3 after interdiction. The same applies for example 2, i.e., attack 3 and attack 4 are the maximum bounded rational solutions, respectively without
and with interdiction. The pattern is that both the complete rational solution and bounded rational solution forces the maximum bounded rational attack to be the next largest attack. However, this pattern does not hold for example 3. In this example, the interdiction of attack 1 by the complete rational solution does not move the maximum bounded rational attack from the original maximum bounded rational attack, which is attack 4. Attack 4 is still maximum bounded rational in this case because it is bounded rational with respect to the uninterdicted attacks 2 and 3. So, the bounded rational objective value from the complete rational solution is 17. In contrast, the bounded rational solution interdicts attack 4, making attack 5 the new maximum bounded rational attack with an objective function value of 5. Thus, the bounded rational solutions results in a 71% decrease in the bounded rational objective function value. The pattern in example 3 does apply to examples 4, 5, and 6. In summary, regardless of the number of attacks that are larger than the maximum bounded rational attack in the uninterdicted graph, as long as the interdiction from the complete rational solution does not change the maximum bounded rational attack, the bounded rational solution produces smaller losses than the complete rational solution.

### 4.5 Conclusion

In this work, we relax the assumption of complete rationality made in typical defender-attacker game theoretic models. Specifically, we incorporate the idea of bounded rationality of the attacker in a Stackelberg defender-attacker zero-sum game and formulate the problem as a tri-level mixed-integer program with integer variables in all levels. The first level represents the defender problem which minimizes the maximum loss caused by the
bounded rational attacker. The second level represents the complete rational attacker. The third level represents the bounded rational attacker which maximizes the loss satisfying the constraint that the bounded rational attacker is able to achieve only a fraction of the objective value achievable by the complete rational attacker.

To solve the problem we develop a customized column and constraint generation algorithm. Although the algorithm is able to solve 50 and 100 nodes problems reasonably quickly, the order of magnitude growth of computation time from 50 nodes to 100 nodes problems suggests the need for considerable speed up the algorithm. Computation times suggest that bounded rational problem is much harder to solve than the complete rational problem. Experiments also show that, in many cases, incorporating bounded rationality in the model does not make any difference in the loss caused by the bounded rational attacker. However, problems with specific characteristics can be impacted significantly by the incorporation of bounded rationality in the model.

As a future work, we will conduct more experiments to thoroughly analyze the characteristics of the problems in which inclusion of bounded rationality makes a significant difference in the solution found by our model. In addition, in this work we worked with a single attacker with a known rationality level. We will extend this work by including multiple attackers with varying levels of rationality and budgets. We also plan to accelerate the algorithm in different ways which include the generation and addition of non-overlapping attacker solutions into the master problem during the initial iterations. We also plan to parallelize the accelerated algorithm and implement it on high performance computing platform to solve problems with tens of thousands of nodes and arcs.
CHAPTER 5

CONCLUSION AND FUTURE RESEARCH

5.1 Conclusion

Cyber security is a major concern nowadays for organizations of all types and sizes. Several high profile attacks in recent past on large corporations resulted in losses amounting to tens of millions of dollars. The cyber network of an organization can be represented both directly as the physical network of computers and other devices and also as as a logical or virtual graph of vulnerabilities and security conditions resulting from the specific network topology, attacker profile, and set of vulnerabilities existing in the network. In this dissertation, we use a special type of logical graph called the attack graph which represents all the attack paths from the initially vulnerable nodes to the goal nodes. Interdiction of arcs on the physical network or the attack graph naturally maps into application of security countermeasures on components of the network.

This dissertation studies several security-related problems. First, we develop network interdiction models and algorithms to select an optimal subset of arcs to minimize spread of infections. Our results show removing a subset of arcs from the physical network based on two of our proposed connectivity-based metrics reduces the speed and amount of spread of infections more than existing arc removal methods. Second, we develop robust optimization models to select subset of arcs to minimize the maximum loss from multi-stage cyber
attacks. The optimization models capture the interaction between the defender and the attacker as stackelberg zero-sum game. Third, we developed models to consider the attacker both as a complete rational agent and a bounded rational agent. In addition to minimizing loss, our results provide several important insights. This dissertation extends the network interdiction literature both by formulating the problems as novel multi-level mixed-integer linear programs with integer variables in all levels and by developing efficient exact algorithms based on constraint and column generation and heuristics algorithms based on monte-carlo simulation.

In chapter 2, we tackle the problem of minimizing the spread of infections in a network. One of the ways of removing the spread of infections in a network is to remove a subset of the arcs from the network. In this study, we study the effectiveness of different link removal methods in minimizing the spread of infections. Specifically, we propose four novel connectivity metrics and propose arc removal methods based on the optimization of the four connectivity metrics. We also develop heuristic algorithms to solve our network interdiction models. We compare the effectiveness of arc removal recommended by our methods with the effectiveness of several other methods in minimizing spread. The other methods are the method developed by [76], the method developed by [47], and random arc removal. We found through experimentation that one of our methods which maximizes the number of susceptible nodes completely isolated from infectious nodes is the most effective in minimizing the amount of new infections. We also found that another of our methods which maximizes the total transmission probability of the paths removed from the netowrk is the most effective in slowing down the spread of infections. Probability
of transmission plays an important in determining the relative effectiveness of the different methods in minimizing the amount and speed of spread.

In chapter 3, we study the problem of minimizing the losses from multi-stage cyber attacks. In that effort, attack graphs are used to represent all the possible attack paths from the initially vulnerable nodes to the valuable goal nodes in a network. An attack graph of an organization is generated using the attacker profile, topology of the network, and the vulnerabilities existing in the network. An attacker with a specific budget can chose from a set of alternative attack plans composed of different combinations of attack paths. In this work, we develop a network interdiction model to find the optimal subset of arcs that when removed minimizes the maximum loss from the alternative attacks that can be chosen by an attacker. We formulate the model as a bi-level mixed-integer linear program with integer variables in all levels, and the model captures the defender-attacker interaction as a stackelberg zero-sum game. We also develop efficient exact and heuristic algorithms to solve the model. Experiments show that the quality of an interdiction plan is relatively insensitive with respect to the error in the defender’s knowledge of the attacker’s budget. Experiments also show that with increase in the security budget, the breach loss drops sharply at the beginning, then levels off before dropping sharply again.

In the defender-attacker game theoretic model developed in chapter 3, one underlying assumption is that the attacker is completely rational meaning that she is able to find the optimal solution. This is a typical assumption in game-theoretic models. In reality, cyber attackers are usually human beings, and human beings are usually not completely rational. Thus, in chapter 4, we depart from the assumption that the attacker is completely rational.
We consider that the rationality of the attacker is bounded meaning that the attacker is able to inflict only a fraction of the maximum damage that can be inflicted with a given budget. We capture the bounded rationality of the attacker using a rationality factor less than 1. We capture the resulting defender-attacker interaction as a network interdiction model and formulate the model as a tri-level mixed-integer linear program with integer variables in all levels. Similar to chapter 3, our model in this work also captures the defender-attacker interaction as a stackelberg zero-sum game with limited budgets for both the defender and the attacker. We also develop customized algorithm based on constraint and column generation to solve the model. Experiments show the importance of considering bounded rationality and the characteristics of the problems with a significant impact of bounded rationality.

We imagine a decision support software for cyber security as the ultimate outcome of the work of this dissertation. We believe that the models and algorithms developed in this dissertation, or some extended versions of them, can work as the vehicle for representing the real life attack and defense scenarios, helping the software application to find the most effective decisions for the cyber security personnel in an organization. Given the topology of a cyber network, the profiles of the potential attackers, and the set of existing vulnerabilities, the proposed security software can work as follows. 1) An attack graph generation module generates the attack graph. 2) Using the attack graphs and other parameters as inputs, an optimization module finds the best subset of arcs in the graph. Our models and algorithms work as the backbone of this module. And, finally 3) A mapping module maps the arcs selected on the attack graph to the actual vulnerabilities, devices, or connections
that need to be secured. A system can either be setup to automatically implement the recommenda-
tions of the software with the oversight of security personnel, or it can work as a what if analysis tool in the manual decision making process of the security personnel.

5.2 Publications

Both of our works in chapter 2 and chapter 3 have been published as separate journal papers in Computers and Operations Research in 2016 [106, 108]. We are currently extending the work in chapter 4 in terms of the number of attackers and the uncertain knowledge of the defender on the rationality and budget of the attackers. We plan to submit the extended work as a journal paper to IIE Transactions by the end of 2016. In addition to the above published or planned journal papers that are direct outcomes of this dissertation, we also published a conference proceedings article related to this dissertation [21].

5.3 Future Research

We are currently extending the model and algorithm in chapter 4 to enable them to represent the reality more closely. Specifically, the extended model and algorithm will incorporate multiple attackers as well as the uncertain knowledge of the defender about the budget and rationality levels of the attackers. We will formulate the extended model as a tri-level stochastic mixed-integer program and adapt the algorithm developed in chapter 4 to solve the new formulation.

Parallel to the extension of chapter 4, we are also extending the work in chapter 3 with respect to the following two aspects. First, we are incorporating multiple attackers with different probability of realization and budget for each attacker. Second, in the resulting
defender-attacker stochastic problem, we minimize both the overall expected value of the potential losses and the expected value of the largest losses. The second component of the objective function makes the model protect against attacks with very large losses. We formulate the network interdiction model as a bi-level mixed-integer stochastic program with binary variables in both levels. We extend the algorithms in chapter 2 to solve the model developed in this work. We are currently working on parallelizing the algorithm and implement it on high performance computing to solve problems with tens of thousands of nodes and arcs.

Further, the underlying defender-attacker games in all the models of this dissertation are zero-sum meaning, that the gain of the attacker is equal to the loss of the defender. However, in reality the gain of the attacker might be very different than the loss of the defender. Thus, we plan to extend the work of this dissertation by considering the underlying game as stackelberg non-zero sum game. In addition, some variations of attack graphs have both AND and OR nodes. Thus, in this extended work, we also plan to incorporate both AND and OR nodes instead of only OR nodes, as considered in this dissertation.
REFERENCES


