A three-layered robustness analysis of cybersecurity:

Attacks and insights

By

David Schweitzer

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

Mississippi State, Mississippi

December 2019
A three-layered robustness analysis of cybersecurity:

Attacks and insights

By

David Schweitzer

Approved:

Stanley F. Bullington
(Major Professor)

Qian Zhou
(Minor Professor)

Hugh R. Medal
(Committee Member)

Mohammad Marufuzzaman
(Committee Member)

Maxwell Young
(Committee Member)

Linkan Bian
(Graduate Coordinator)

Jason M. Keith
Dean
Bagley College of Engineering
Cybersecurity has become an increasingly important concern for both military and civilian infrastructure globally. Because of the complexity that comes with wireless networks, adversaries have many means of infiltration and disruption of wireless networks. While there is much research done in defending these networks, understanding the robustness of these networks is tantamount for both designing new networks and examining possible security deficiencies in preexisting networks. This dissertation proposes to examine the robustness of wireless networks on three major fronts: the physical layer, the data-link layer, and the network layer. At the physical layer, denial-of-service jamming attacks are considered, and both additive interference and no interference are modeled in an optimal configuration and five common network topologies. At the data-link layer, data transmission efficacy and denial-of-sleep attacks are considered with the goal of maximizing throughput under a constrained lifetime. At the network layer, valid and anomalous communications are considered with the goal of classifying those anomalous communications apart from
valid ones. This dissertation proposes that a thorough analysis of the aforementioned three layers provides valuable insights to robustness on general wireless networks.

Key words: Network flow, cybersecurity, mathematical programming, deep learning, mixed-integer programming
DEDICATION

This work is dedicated to Brooke Scharff and Skylar Nallen. Without either of you, I would never have completed this. The friendships you two give me will always be cherished, and I hope we will always be there for one another.
ACKNOWLEDGEMENTS

First, I would like to thank my mother, Sylvie Jones, for her unwavering support. It is through her parenting throughout my entire life that this feat was made possible. Thank you for the sacrifices you made to bring me to this Earth and raise me to be the best person I could be. Next, I would like to thank my friends, Brooke Scharff and Skylar Nallen, for their support in helping me to keep my head focused in the most trying of times. I am who I am now because of the friendships they have provided me. To Tanveer and Sarah, for all the struggles we overcame together, and all the good times we had. As we three went on this journey together, we were able to stay headstrong through continuous encouragement. And I would like to thank God, from whom all things are possible, for blessing me with such support.

I want to thank Dr. Stanley Bullington for his time and patience in assisting me with the requirements for this work, for whom I have significant respect. The courses I took from him, though not directly related to my dissertation, were enjoyable and insightful because of the knowledge you brought to them.

I would also like to thank Dr. Maxwell Young, Dr. Qian “Michelle” Zhou, and Dr. Mohammad Marufuzzaman. Although I never took any courses with any of you, your willingness to serve on my committee and participate in the forward progression of my work is appreciated.
CONTENTS

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

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

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

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

CHAPTER

I. INTRODUCTION ....................................................... 1

1.1 Background and Motivation ........................................ 1
1.2 Related Literature .................................................. 4
1.3 Contributions ....................................................... 6

II. PHYSICAL LAYER: WIRELESS LAN TRANSMITTER LOCATION UNDER
THE THREAT OF JAMMING ATTACKS ................................. 8

2.1 Introduction ......................................................... 8
2.2 Related Literature .................................................. 9
2.3 Problem Description ............................................... 11
  2.3.1 Illustrative Example ........................................... 12
2.4 Mathematical model ............................................... 14
  2.4.1 Tri-level mixed-integer programming model ................. 14
  2.4.2 Additive Interference ......................................... 22
2.5 Solution Methodologies ........................................... 27
  2.5.1 Branch-and-bound ............................................. 27
  2.5.2 Implicit enumeration ......................................... 29
  2.5.3 Dynamic constraint generation for additive model ........ 30
2.6 Computational analysis ........................................... 32
  2.6.1 Run-time of solution methodologies and model comparison 35
  2.6.2 Solution Quality for Different Topologies ................. 38
  2.6.3 Problem sensitivity analysis .................................. 43
  2.6.4 Utility function changes ..................................... 47
| 2.6.5 | Trend experimental results | 50 |
| 2.7  | Conclusions               | 53 |
| 2.7.1 | Discussion               | 53 |
| 2.7.2 | Future Work              | 55 |

### III. DATA-LINK LAYER: MAXIMUM NETWORK LIFETIME UNDER THE THREAT OF DENIAL-OF-SLEEP ATTACKS

| 3.1  | Introduction            | 56 |
| 3.2  | Related Literature      | 58 |
| 3.3  | Problem Description     | 60 |
| 3.4  | Mathematical model       | 63 |
| 3.5  | Solution Methodologies   | 70 |
| 3.5.1 | Branch-and-bound         | 70 |
| 3.5.2 | Implicit enumeration     | 70 |
| 3.6  | Illustrative example    | 72 |
| 3.7  | Computational analysis   | 76 |
| 3.7.1 | Experimental parameters | 76 |
| 3.7.2 | Run-time of solution methodologies | 78 |
| 3.7.3 | Problem sensitivity analysis | 79 |
| 3.7.3.1 | Effect of the number of DoSL devices | 79 |
| 3.7.3.2 | Effect of the battery lifetime | 81 |
| 3.7.3.3 | Effect of the number of DoSL devices and DoSL energy | 83 |
| 3.8  | Conclusions             | 85 |
| 3.8.1 | Discussion              | 85 |
| 3.8.2 | Future Work             | 86 |

### IV. NETWORK LAYER: DETECTING VARIETIES OF BOTNETS THROUGH GENERALIZED COMMUNICATION BEHAVIORS

<p>| 4.1  | Introduction            | 88 |
| 4.2  | Related Literature      | 89 |
| 4.3  | Problem Description     | 92 |
| 4.4  | Data                    | 93 |
| 4.5  | Methodology             | 96 |
| 4.5.1 | Features               | 96 |
| 4.5.2 | Loss Function           | 98 |
| 4.5.3 | Neural Network Architecture | 99 |
| 4.6  | Experiments and Results | 100 |
| 4.6.1 | Experimental Setup      | 100 |
| 4.6.2 | Results and Analyses    | 102 |
| 4.6.3 | Comparison to Other Work | 107 |
| 4.7  | Conclusions             | 109 |</p>
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>List of Parameters for $[TL - MIP]$</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>Run-time of non-additive model</td>
<td>35</td>
</tr>
<tr>
<td>2.4</td>
<td>Run-time of additive model</td>
<td>37</td>
</tr>
<tr>
<td>2.5</td>
<td>$n$ Access Point Additions</td>
<td>45</td>
</tr>
<tr>
<td>2.6</td>
<td>Jamming Impact</td>
<td>48</td>
</tr>
<tr>
<td>2.7</td>
<td>Jammer Movement Trend Results</td>
<td>51</td>
</tr>
<tr>
<td>3.1</td>
<td>List of Parameters for $[BL - MIP]$</td>
<td>65</td>
</tr>
<tr>
<td>3.2</td>
<td>Network parameters</td>
<td>78</td>
</tr>
<tr>
<td>3.3</td>
<td>Run-time of algorithms</td>
<td>78</td>
</tr>
<tr>
<td>4.1</td>
<td>Data and Label Summary of CTU-13</td>
<td>95</td>
</tr>
<tr>
<td>4.2</td>
<td>Onyx HPC Environment</td>
<td>100</td>
</tr>
<tr>
<td>4.3</td>
<td>Results</td>
<td>102</td>
</tr>
<tr>
<td>4.4</td>
<td>Confusion Matrices for CTU-13</td>
<td>104</td>
</tr>
<tr>
<td>4.5</td>
<td>Comparison of Performance Metrics</td>
<td>108</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>1.1</td>
<td>Physical Layer</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Example Network</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>Additive Interference from Jammers</td>
<td>23</td>
</tr>
<tr>
<td>2.3</td>
<td>Wireless Topologies</td>
<td>33</td>
</tr>
<tr>
<td>2.4</td>
<td>Percentage Below Optimal</td>
<td>39</td>
</tr>
<tr>
<td>2.5</td>
<td>Additive/Non-Additive Relative Error</td>
<td>40</td>
</tr>
<tr>
<td>2.6</td>
<td>Visual Comparison of Optimal and Median Topologies</td>
<td>43</td>
</tr>
<tr>
<td>3.1</td>
<td>DoSL Attack</td>
<td>74</td>
</tr>
<tr>
<td>3.2</td>
<td>CMU and MIT networks</td>
<td>77</td>
</tr>
<tr>
<td>3.3</td>
<td>Number of DoSL devices versus lifetime throughput</td>
<td>80</td>
</tr>
<tr>
<td>3.4</td>
<td>Battery lifetime versus lifetime throughput</td>
<td>82</td>
</tr>
<tr>
<td>3.5</td>
<td>Problem 1: 10 DoSL devices at 30 units/sec vs. 20 DoSL devices at 15 units/sec</td>
<td>83</td>
</tr>
<tr>
<td>3.6</td>
<td>Problem 2: 20 DoSL devices at 30 units/sec vs. 40 DoSL devices at 15 units/sec</td>
<td>84</td>
</tr>
<tr>
<td>4.1</td>
<td>Trends of Accuracy Metrics Across all of CTU-13</td>
<td>106</td>
</tr>
<tr>
<td>4.2</td>
<td>Loss Trend Across All CTU-13</td>
<td>107</td>
</tr>
</tbody>
</table>
1.1 Background and Motivation

Communication networks that enable the aerial transferal of data without the use of cables or wires are referred to as wireless networks. There are a variety of types of these networks, such as the Wireless Sensor Network (WSN), Ad Hoc Network (AHN), and the Wireless Local Area Network (WLAN). WSNs, typically found in locations that do not share much human traffic, such as polar tundras or deserts, consist of a large number of sensor nodes typically designed to run at low power and continuous gather data, such as weather data; this data is then fed to some operator. AHNs are self-contained, limited-run networks designed to be built and allow wireless communication where there is not much infrastructure, and these are typically not meant to be permanent installations; examples of AHNs include military outposts or in areas hit by severe natural disasters such as hurricanes. The WLAN is perhaps the most abundant and well-known as it is used by businesses and homes all over the world, relying on towers and satellites to act as transceivers and keep phones and computers in near-constant communication. Unfortunately, though these networks are easily maintained and moved, sending information over the air allows for adversaries to more easily disrupt the networks.

There are a variety of types of attacks of wireless networks that each target different attributes, or layers, of the networks. Wireless networks are made of seven abstract layers [105]: the physical
layer, the data-link layer, the network layer, the transport layer, the session layer, the presentation layer, and the application layer. The physical layer is the transceiver that drives the signals on the network. Attacks on this layer are well understood and referred to as denial-of-service (DoS), which include jamming attacks, attacks that send enough signal that communication signals are overwhelmed and nodes are completely disrupted. The data-link layer is responsible for creating the frames that move across the network, ensuring that communication is maintained and consistent, and while attacks on this layer are varied, one of the more known types of attacks are the denial-of-sleep (DoSL) attacks, such as vampire attacks, that force transceivers to continually transmit information (legitimate or otherwise) at the expensive of the limited battery. The network layer is responsible for creating the packets that move across the network and generates packets of communication information, such as IP addresses at both the source and destination, and botnets and malware typically attack this layer. The transport layer establishes the connection between applications on different hosts, while the session layer provides the means for opening, closing, and managing sessions between application processes. The presentation layer translates the data for the network, and the application layer is a group of applications requiring network communications. Some viruses may attack each of these layers. This dissertation focuses on the physical, data-link, and network layers. Figure 1 [35] illustrates these layers.
The development of more powerful computers saw an increase in modeling optimization problems for examining wireless networks’ robustness to attacks and attack efficacy on networks as well as the advent of machine and deep learning for identifying malicious communications such as malware. In this dissertation, we examine these networks that are under attack at the three bottom most layers: the physical layer, the data-link layer, and the network layer. We look how these networks withstand DoS and DoSL attacks, including how to create a network that can best mitigate a jamming attack and how the throughput of a network under a lifetime constraint is diminished by a DoSL attack, and we also examine the efficacy of a network to identify malicious communication. In any scenario, civilian or military, it is important not only to diminish an adversary’s attack but also examine the robustness of one’s own network under the threat of such attacks. Thus this dissertation outlines and summarizes three new works studying these attacks. The rest of this dissertation is outlined as follows: Section 2 specifies the actual problems being solved and outlines
the current works in progress, Section 3 summarizes the importance of this work and makes clear the contributions, and Section 4 provides a timeline of the remainder of this dissertation

1.2 Related Literature

This literature review provides a summary of the literature related to general wireless network robustness, including attacks and defenses, which is the core of the areas discussed in this dissertation. More detailed reviews of the literature pertaining to each individual area are contained within their respective chapters of the dissertation that follow.

Vadlamani, et al. detail a variety of attacks on wireless networks that primarily affect the physical layer [119]. The majority of these techniques constitute general denial-of-service (DoS) attacks, of which jamming is a widely researched topic within. General DoS attacks on any layer in the network have also been detailed along with means of detecting such attacks [16]. These types of attacks have been localized to specific types of networks, such as wireless sensor networks [63, 126] and other energy-constrained networks [129] as well as ad hoc networks [43, 131]. From an operations point-of-view, the impacts of these types of attacks have been studied [1] with the goal of making them more secure, as well as the feasibility of such attacks and defenses occurring [127], including the use of specific functionalities like cognitive radios and directed antennae [108]. In general, however, much of this work assumes the network has been pre-established and does not consider building an optimal network from the ground-up. Those that do [27] do not consider the actual impacts of jammers nor how topologies may be exploited to mitigate such attacks. In this dissertation, we consider how topology plays a special role in the efficacy of jammers by considering several pre-specified topologies and examining jammer behaviors across each of them.
In the case that a network is built from the ground-up, we demonstrate that significant connectivity can be established, thus demonstrating the importance of initial planning and development.

Energy efficiency in lifetime-constrained networks, such as wireless sensor networks, is also of interest, where authors have studied numerous attempts to maximizing network lifetime under varying conditions [96, 130, 42]. As well, work has been done in maximizing not lifetime but throughput [47, 48, 79], with some works seeking to maximize both as a multi-objective criterion [2]. Such maximum flow-type problems, however, do not consider attacks by denial-of-sleep (DoSL) devices that seek to drain the batteries of nodes in these networks. While work has been done in trying to prevent such attacks [13, 12], the actual impacts of such attacks are largely unknown. Here, we provide a first look at how much impact DoSL devices have on throughput by not only considering their optimal placement but also show how not only are DoSL devices likely to bottleneck a network, but trying to give nodes more powerful batteries does not significantly reduce the efficacy of the DoSL attack.

Of course, all the aforementioned works have one major goal in mind: allow information to transmit across the network, with the goal of having communication at all or sending specific communication from one point to another efficiently. Whether this information is normal or anomalous is a widely researched topic [55, 133]. As anomalous communications, such as botnets, become more stealthy, there is ever a need to consider not specific protocols within the botnet but the underlying structure. As such, graph analytics have been employed with mixed results, often relying on a priori knowledge of botnet architecture [124, 87, 32]. What has helped the detection of botnets has been the advent and rapidly growing interest in machine learning and deep learning. Many suitable techniques have been examined and benchmarked thoroughly [10, 67], though these
techniques typically fail on larger or real datasets. To overcome the issues of traditional machine learning, deep learning and artificial neural networks have been deployed [118, 90]. However, these, too, fall in the trap of considering only specific botnets. While some efforts have been made at detecting general botnets on large-scale networks [41], accuracy has generally been mixed. This is likely a result of too few botnets existing in the data and models generally being unable to detect smaller, more hidden botnets. Our work moves away from this to examine the communication behaviors directly rather than inferencing upon a given node, an important consideration given the stealthy improvements botnets have made [29].

1.3 Contributions

There are several contributions within this dissertation. For each of the three layers, a model is provided to answer the broad question of how much legitimate throughput is successfully transmitted from a source to a destination. At the physical layer, a tri-level mixed-integer programming model seeks to maximize total connectivity in a network is utilized. At the data-link layer, a bi-level mixed-integer programming model seeks to maximize network throughput. At the network layer, an artificial neural network model seeks to classify botnet and nonbot (or legitimate) communication a network while minimizing false positives (bots detected as nonbots). Within the first two layers, we provide several insights into topological considerations by exploring random topologies as well as two established topologies, the Carnegie Melon University topology and the Massachusetts Institute of Technology topology, as well as five common topologies, all of which will be illustrated later. Three major topological considerations are considered. The first involves how (optimally) adding access points affects the connectivity. The second involves jam-
mer movement behaviors across several pre-established topologies. The final demonstrates that simply adding one or two access points is not enough–there is definitely a need to add a certain number before significant results are attained. Resulting from our topological considerations, we point out jammer movement behavior trends and that topology plays a significant role in how jammers behave; we demonstrate that the utility of connectivity matters, and whether signal strength or number of connections is more important plays a vital role in determining overall connectivity; we show that fewer, powerful denial-of-sleep devices are less impacting than more, weaker devices; we show that improving battery lifetime has a largely linear relationship to improving the performance of a network; and we show that topology is also important for mitigating the effects of multiple denial-of-sleep devices. For both the physical and data-link layers and their mixed-integer programming models, we apply the Implicit Enumeration algorithm to show that they are solvable in very good time even compared to commercial solvers like Gurobi or traditional algorithms like Branch-and-Bound. We also explore the effects of additive interference jammers may make use of in attacking a network, something that has never been considered in other jamming papers. Finally, we make use of more generalized features to detect malicious communications and lay a framework for how general botnets behave with the intention that our model can be any dataset, not just the one we use, and attain good accuracy.
CHAPTER II

PHYSICAL LAYER: WIRELESS LAN TRANSMITTER LOCATION UNDER THE THREAT OF JAMMING ATTACKS

2.1 Introduction

Wireless networks are widely used in a variety of environments ranging from businesses to college campuses; a key feature of such networks is that they require little to no infrastructure. Consisting of a set of transmitters and receivers, maintaining connection to them does not necessarily require fixed positions, and thus mobility is nearly unrestricted. Both the mobility and the limited infrastructure provide great benefit to a variety of recipients, from those who need response in environmental disasters (where actual infrastructure may be deployed; such networks are called ad hoc networks) to those who need to perform business transactions while in flight.

Most wireless networks are contextualized to some spatial distribution of access points, such as transmitters, routers, or signal towers, and demand points, the users or infrastructure that need to connect to the network. The loss of an access point can remove connections for several demand points, while a loss of critical demand points can result in tasks being unable to be completed. Unfortunately, wireless networks are vulnerable to attacks, the most prominent one being the jammer attack that seeks to remove connections by overwhelming all other signals.

But network signals are not typically “on/off.” In fact, attackers can find themselves in a position where their jammers are powerful enough that they can degrade signals beyond any jammer’s
radius, and by combining this effect with other jammers, more connections are destroyed; thus attackers have an advantage in disrupting the network. As such, defenders not only need to safeguard networks against direct jamming but also overcome aggregate interference a well-planned attack may cause. The best way to ensure robustness against jamming is to optimally place access points such that direct jamming and interference are mitigated, establishing a crucial role for the defender in planning and developing a wireless network design.

The goal of this chapter is to examine the topological considerations of designing an array of wireless access points that is resilient to jamming attacks. We consider a wireless local area network (WLAN) that is subject to jamming attacks. To mitigate the impact of jamming, we seek to optimally locate a set of wireless access points (devices that send signals, such as towers or routers) over a set of potential sites and model the impact of a jamming attack as the loss in user connectivity due to the unavailability of access points. Toward this end, we develop a tri-level multi-period mixed-integer programming model (TL-MIP) that maximizes total network performance under the threat of jamming.

2.2 Related Literature

The physical layer of any network is the most easily understood layer because it consists of physical devices. As such, attacking these devices directly is simpler, in theory, than attacking other layers that require direct network access. The wireless network jamming problem involves placing a set of jammers in a network so as to sever connections. This problem is similar to general network interdiction in that a set of nodes are introduced to a network such that some edges are removed. Network interdiction is an area of literature which has been extensively studied.
Network interdiction studies have examined a variety of optimization objectives such as maximizing a shortest path [102, 50, 15], minimizing the maximum flow [3, 123], and minimizing network connectivity [89, 5]. Researchers have also expanded this literature to include problems such as multiple commodities [64] and multiple time periods [72], stochastic interdiction [81], and dynamic interactions between attackers and defenders [68]. There are also studies that examine randomness, such as random topologies [45, 46] or random behaviors from the attackers [82, 95]. Many specific applications have also resulted, some of which involve waterway commodity flows [8], power grid vulnerability assessment [107], and interdicting nuclear smuggling [94]. Another significant application of network interdiction is the wireless communication network interdiction, and the forefront of this area is the jammer placement problem.

Wireless network jamming attacks are a type of denial-of-service attack on the physical layer of a wireless network [98], and there have been several studies on specifically the placement of jammers [91, 27, 26]. There are also a variety of studies that provide details of jamming attacks [92, 61] and how they affect network performance [91, 9]. Expectantly, there are some studies that focus on how to find jamming devices [99] or how to respond to an attack [69, 52], and these studies offer some strategies for safeguarding networks. Such strategies include channel-hopping [86], key management techniques [34], and spread spectrum techniques [66]. However, these papers all focus solely on the placement of jammers or the improvement of an already existing wireless infrastructure in order to mitigate attacks, such as enhancing signal antennae or changing antennae direction, switching channels, etc. None of these papers specifically focus on the initial construction of the wireless infrastructure with the focus of mitigating an optimal jamming attack, nor do they consider any topological establishments.
The key to maximizing a wireless network’s robustness amounts to placing access points so as to increase the network’s performance (e.g., connectivity, throughput), but doing so in such a way as to increase the number of connections or signal strength has not been considered. Further, most jamming attacks are treated as distance-only protocols, where connections are jammed only if access or demand points fall within a jammer’s radius. More realistically, signals are likely to create an additive interference effect [51], and the placement of jammers that takes advantage of this effect has never been studied. Thus, there are currently gaps in the literature involving the optimal placement of access points designed to mitigate optimal jamming attacks, the consideration of how resilient common and pre-established topologies are, and how any wireless network fares against additive interference from jammers, thus preventing a realistic study of the vulnerability of large-scale wireless networks to jamming attacks.

2.3 Problem Description

Our problem consists of a set of access points (e.g., signal towers), each with some capacity, a set of possible access point locations, a set of jammers, and a set of possible jammer locations. Demand points (e.g. users, laptops, cell phones) attempt to connect to some access points, but an established connection will only occur between a demand point and a single access point if two conditions are met: 1) the access point must not be at capacity, and 2) the new connection must be establish-able. This means that a new connection should not be established within the model if it falls within a jammer’s radius or could be automatically jammed under the additive interference, which will be discussed later.
We seek to solve the access-jammer placement problem under such a condition, that is to optimally place the set of jammers such that the network connectivity is minimized while access points are optimally placed to mitigate this attack. However, the attacker and defender do not know a priori the locations of the demand points. Demand points may relocate between time; thus, demand points strive to maximize their connectivity, and will place themselves accordingly, particularly if they are located near a capacitated access point. Should a connection be jammed, the demand point may be reconnected to another nearby access point if the next access point also satisfies the previous two conditions. Thus, it is to the attacker’s advantage if the attacker not only disrupts the total number of actual connections, but forces the jammers to be placed in such a way that other access points reach capacity and prevent demand points from remaining connected to the network, effectively bottle-necking the network.

2.3.1 Illustrative Example

Figure 2.1 shows a network that is being jammed. Circles represent demand points, triangles represent jammers, the circular fields around the triangles represent their jamming radius, and squares represent access points. The subscripts under each access point index represents the capacity of that access point. The solid lines represent a connection, while dashed lines represent possible secondary connections that could exist if two conditions are met: 1) the demand point is within range of the access point, and 2) the access point is not already at capacity. For example, consider access point 2 in Figure 2.1a. Demand points 4, 5, 6, and 7 are within its range and thus could connect to it. Although its capacity allows for only 2 demand points to connect to it, only demand point 5 connects to it. Demand point 4 is closer to access point 1 and establishes
this connection without violating 1’s capacity, and demand points 6 and 7 connect to access points 3 and 4, respectively, under the same reasoning. Thus only access points 1 and 3 are at capacity. Now consider Figure 2.1b, an example where jammers have been (not necessarily optimally) placed. Jammers 1 and 2 completely contain access points 1 and 4 and demand point 8, thus any connections these three might have with any other demand points or access points are rendered impossible. Demand point 4 is now assigned to access point 2 because access point 2’s capacity was not met and demand point 4 is the first of the other possible demand points that could connect to it. Also, as demand points 4 and 5 bring access point 2 to capacity, demand point 7 can no longer connect to the network. Thus, even though no jammer is directly removing demand point 7 from the network, the indirect effect of forcing access point 2 to reach capacity has forced the network to not attain the maximum possible number of connections, thus improving the impact of jamming.

However, the illustration in Figure 2.1b may not be the optimal placement of the access points. By keeping the jammers and demand points fixed, a slight shift in the access points’ location as in Figure 2.1c produces 3 more established connections, suggesting the location of the access points can have a significant influence on the impact of jamming. Thus, there is a need to study the optimal placement of access points given a threat of an attack by optimal jammer placement, which is precisely the focus on this chapter.
2.4 Mathematical model
2.4.1 Tri-level mixed-integer programming model

To model the problem described in Section 2.3, we present the following Tri-Level Multi-Period Mixed-Integer Programming $[TL-MIP]$ model. This model has been adopted from the $r$-interdiction median model, which involves locating facilities to maximize the total distance to a set of demand points. Said problem is an inverse of the $p$-median problem, which strives to choose a set
of facilities to remove with the goal of minimizing the distance. For our problem, the access points correspond to the facilities to be removed; specifically, our model strives to remove (by jamming) a set of access points in order to minimize the number of demand point-access point connections. The objective function is thus similar to those used in the facility location problems [24], where an attacker seeks to minimize the amount of resource allocation (here, network connectivity) through interdiction, and the defender seeks to maximize the resource allocation with the knowledge of an attack. Our model allows for the demand points and jammers to relocate between time periods, while jammers have a movement restriction $R$ and a limited jamming radius. Access points also have finite capacity, and demand points maybe reassigned to another access point if the preferred access point is at capacity or is jammed. Our model is unique in that we also consider the additive effect of the jammers, which we discuss further in Section 2.4.2.
Table 2.1: List of Parameters for $[TL - MIP]$  

<table>
<thead>
<tr>
<th><strong>Indices</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>index for demand points</td>
</tr>
<tr>
<td>$j$</td>
<td>index for access points</td>
</tr>
<tr>
<td>$k$</td>
<td>index for locations of jammers</td>
</tr>
<tr>
<td>$l$</td>
<td>index for jammers</td>
</tr>
<tr>
<td>$t$</td>
<td>index for time periods</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Parameters</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{ijkl}$</td>
<td>1 if a jammer $l$ located in position $k$ disturb the connectivity between access point $j$ and demand point $i$ at period $t$; 0 otherwise</td>
</tr>
<tr>
<td>$d_{ab}$</td>
<td>the Euclidean distance between $a$ and $b$</td>
</tr>
<tr>
<td>$R$</td>
<td>the greatest distance that a jammer may relocate to between time</td>
</tr>
<tr>
<td>$S_{ijt}$</td>
<td>the signal strength from access point $j$ to demand point $i$ during time period $t$</td>
</tr>
<tr>
<td>$n$</td>
<td>the number of jammers available</td>
</tr>
<tr>
<td>$C_k$</td>
<td>the jamming radius of jammer $k$</td>
</tr>
<tr>
<td>$m$</td>
<td>the number of access points available</td>
</tr>
<tr>
<td>$c$</td>
<td>the capacity of an access point</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Sets</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$</td>
<td>the set of demand points</td>
</tr>
<tr>
<td>$J$</td>
<td>the set of locations for access points</td>
</tr>
<tr>
<td>$K$</td>
<td>the set of existing jammers</td>
</tr>
<tr>
<td>$L$</td>
<td>the set of potential locations for jammers</td>
</tr>
<tr>
<td>$T$</td>
<td>the set of time periods</td>
</tr>
<tr>
<td>$A_{it} = { j \in J \mid 0 &lt; S_{ijt} }$</td>
<td>the set of access points with capacity such that demand point $i$ at period $t$ has some positive signal strength</td>
</tr>
<tr>
<td>$T_{ijt} = { q \in A_{it} \mid q \neq j, S_{ijt} \leq S_{ijt} }$</td>
<td>the set of existing access points (not including at location $j$) that are at most the same signal from access point $j$ for demand point $i$ at time period $t$</td>
</tr>
<tr>
<td>$P_k = { m \in K \mid m \neq k, d_{km} &gt; R }$</td>
<td>the set of locations of jammers (excluding $k$) whose distance from $k$ is greater than $R$</td>
</tr>
</tbody>
</table>
Using the above notation, our MIP is split into the following three stages.

\[ Z_1 = \text{Max } W(y) \]  \hspace{1cm} (2.1a)

subject to:

\[ \sum_{j \in J} w_j = m \]  \hspace{1cm} (2.1b)

\[ w_j \in \{0, 1\} \quad \forall j \in J \]  \hspace{1cm} (2.1c)

where

\[ W(y) = \text{Min } Z_2 \]  \hspace{1cm} (2.2a)

subject to:

\[ \sum_{k \in K} y_{klt} \leq 1 \quad \forall l \in L, t \in T \]  \hspace{1cm} (2.2b)

\[ \sum_{l \in L} y_{klt} \leq 1 \quad \forall k \in K, t \in T \]  \hspace{1cm} (2.2c)

\[ \sum_{n \in P_k} y_{n,l,t+1} \leq 1 - y_{klt} \quad \forall k \in K, l \in L, t \in T \]  \hspace{1cm} (2.2d)
\( y_{klt} \in \{0, 1\} \quad \forall k \in K, l \in L, t \in T \) \hspace{1cm} (2.2e)

where

\[
Z_2 = \text{Max} \sum_{i \in I} \sum_{j \in J} \sum_{t \in T} S_{ijt} x_{ijt}
\] \hspace{1cm} (2.3a)

subject to:

\[
\sum_{j \in A_{it}} x_{ijt} = 1 \quad \forall i \in I, t \in T \quad [\alpha_{it}]
\] \hspace{1cm} (2.3b)

\[
\sum_{i \in I} x_{ijt} \leq cw_j \quad \forall j \in A_{it}, t \in T \quad [\beta_{jt}]
\] \hspace{1cm} (2.3c)

\[
x_{ijt} \leq 1 - \sum_{k \in K} \sum_{l \in L} D_{ijklt} y_{klt} \quad \forall i \in I, j \in J, t \in T \quad [\gamma_{ijt}]
\] \hspace{1cm} (2.3d)

Here, \( x_{ijt} \in \{0, 1\} \) refers to if a demand point \( i \) can establish a connection with access point in location \( j \) during time period \( t \); \( y_{klt} \in \{0, 1\} \) refers to if a jammer \( k \) is in location \( l \) during time period \( t \); and \( w_j \in \{0, 1\} \) refers to if an access point is in location \( j \). In the first stage (2.1), the objective function (2.1a) strives to maximize the total signal strength through the deployment of access points, where constraint (2.1b) requires that all available access points are placed.

In the second stage (2.2), the objective function (2.2a) strives to minimize the total signal strength through the deployment of jammers. The sets of constraints (2.2b) and (2.2c) force at most only one jammer to exist in any given location and that all available jammers be deployed.
The set of constraints (2.2d) restrict where the jammer may relocate based on \( R \) between time periods.

In the third stage (2.3), the objective function (2.3a) strives to maximize the signal strength based on the demand distribution, where the demand should seek access points with available capacity. The set of constraints (2.3b) maintains that each demand point must be connected to exactly one access point. In general, this may be impossible as some demand points may be too far away from an unjammed access point with available capacity. In this case, the demand point is assigned to a dummy access point with zero signal strength. The set of constraints (2.3c) ensures that for each access point, only so many demand points may be connected to it as determined by the capacity. Finally, the set of constraints (2.3d) forces all interdicted connections to have zero signal strength. For the sets of constraints (2.3b), (2.3c), and (2.3d), we have labeled the dual variables \( \alpha_{it}, \beta_{jt}, \) and \( \gamma_{ijt} \), respectively, that we will use later.

For the demand point placement distribution, we use a spatial Poisson process, which is a multi-dimensional generalization of the Poisson point process. Precisely, a spatial Poisson process with uniform intensity \( \beta > 0 \) is a point process in \( \mathbb{R}^2 \) such that for each bounded, closed, disjoint set \( B_i, 1 \leq i \leq M \) for some integer \( M > 0 \), the count \( N(B_i) \) has a Poisson distribution with mean \( \beta \lambda(B_i) \) where \( \lambda(B_i) \) is the area of \( B_i \) and \( N(B_1), \ldots, N(B_m) \) are independent. We divide our region \( R \) into \( a \times b \) cells, and each cell \( B_i \) satisfies the inter-arrival property \( \mathbb{P}(N(B_i) = k \mid N(R) = n) = \binom{n}{k} p^k (1 - p)^{n-k} \), where \( n \) is the maximum number of demand points within our region \( R \) and \( k \) is the number of demand points arriving into cell \( B_i \), where \( p = \lambda(B_i) / \lambda(R) \). Realistically, demand points can only move so far between time periods. However, for the sake
of our model’s tractability, we reassign the demand distribution between time periods according to the above process.

Though we use a specific utility function given by equation (2.3a), there are a variety of interpretations on maximizing network connectivity, and we discuss this further in Section 2.6.4, where we consider three variants.

The \([TL - MIP]\) program can be simplified into a bi-level program by taking the dual of the third stage and combining this dual and the second stage into a single stage. This invokes strong duality which is normally a problem for our third stage, where \(x_{ijt}\) is a binary variable. However, by fixing the decision variables and relaxing the binary constraint, the third stage becomes an assignment problem in which the constraint matrix is known to be totally uni-modular. Thus we can guarantee the solutions of \(x_{ijt}\) will be 0 or 1 under relaxation, and this relaxation allows us to proceed with the dual reformulation. The dual of the third stage is presented below.

\[
Z_D = \text{Min} \sum_{i \in I} \sum_{j \in J} \sum_{t \in T} (\alpha_{it} + c \beta_{jt} + (1 - \sum_{k \in K} \sum_{l \in L} D_{ijkl} y_{kl}) \gamma_{ijt}) \tag{2.4a}
\]

subject to:

\[
\sum_{t \in T} (\sum_{i \in I} \alpha_{it} + \sum_{j \in A_{it}} \beta_{jt} + \sum_{i \in I} \sum_{j \in J} \gamma_{ijt}) \geq 1 \quad \forall i \in I \tag{2.4b}
\]

\[
\alpha_{it}, \beta_{jt}, \gamma_{ijt} \geq 0 \quad \forall i \in I, j \in A_{it}, t \in T \tag{2.4c}
\]
Substituting this dual into the second stage, we can now formulate the modified Bi-Level Mixed-Integer program \([BL - MIP]\) as follows.

\[
Z_1 = \max W(y) \tag{2.5a}
\]

subject to:

\[
\sum_{j \in J} w_j \leq m \tag{2.5b}
\]

\[
w_j \in \{0, 1\} \quad \forall j \in J \tag{2.5c}
\]

where

\[
W(y) = \min \left( \sum_{i \in I} \sum_{j \in J} \sum_{t \in T} (\alpha_{it} + c\beta_{ijt} + (1 - \sum_{k \in K} \sum_{l \in L} D_{ijkl} y_{kl}) \gamma_{ijt}) \right) \tag{2.6a}
\]

subject to:

\[
\sum_{k \in K} y_{klt} \leq 1 \quad \forall l \in L, t \in T \tag{2.6b}
\]

\[
\sum_{l \in L} y_{klt} \leq 1 \quad \forall k \in K, t \in T \tag{2.6c}
\]

\[
\sum_{n \in P_k} y_{n,l,t+1} \leq 1 - y_{klt} \quad \forall k \in k, l \in L, t \in T \tag{2.6d}
\]
\[
\sum_{t \in T} \left( \sum_{i \in I} \alpha_{it} + \sum_{j \in J} \beta_{jt} + \sum_{i \in I} \sum_{j \in J} \gamma_{ijt} \right) \geq 1 \quad \forall i \in I, j \in A_{it}, t \in T \tag{2.6e}
\]

\[
\alpha_{it}, \beta_{ijt}, \gamma_{ijt} \geq 0 \quad \forall i \in I, j \in J, t \in T \tag{2.6f}
\]

\[
y_{klt} \in \{0, 1\} \quad \forall k \in K, l \in L, t \in T \tag{2.6g}
\]

In the objective function, the product of \( y_{klt} \) and \( \gamma_{ijt} \) can be linearized in the usual way by defining a new non-negative variable \( z_{ijklt} \) and imposing the following constraints:

\[
z_{ijklt} \leq y_{klt} \gamma_{ijt},
\]

\[
z_{ijklt} \leq \gamma_{ijt}, \text{ and } z_{ijklt} \geq \gamma_{ijt} - (1 - y_{klt})M,
\]

where these constraints are given for all \( i \in I, j \in J, k \in K, l \in L, \) and \( t \in T \) where appropriate, and \( M \) is a large number. The mixed-integer nature of this problem makes it difficult to solve. To overcome this difficulty, our solution methodologies, discussed in Section 2.5, will incorporate relaxation on these integer constraints. We conclude this section with the remark that the above model assumes that jammers work independently from one another; access or demand points within the radius of any jammer are completely jammed, and adding more jammers to an area does not increase jamming impact where jammers’ effects overlap. The above model is referred to as the non-additive model.

### 2.4.2 Additive Interference

As the previous section assumes no additive effects from multiple jammers, we now consider the presence of additive interference [51]. Here, the jammers may work together to interfere with connections even when access or demands are not within their radii. Figure 2.2 illustrates this. Un-
der a non-additive effect, connection is allowed. However, under an additive effect, the connection is interfered with, even though no single jammer captures the access or demand point within their radii. This is because in the additive model the jamming interference present at the connection is the sum of the interference caused by each jammer. If the interference is large enough, the connection is jammed in the sense that no connection can be established between the access point and demand point.

![Figure 2.2: Additive Interference from Jammers](image)

This effect is known as the additive (or physical) interference model. In this model, the signal strength for access point $j$ trying to connect to a given demand point $i$ during time period $t$ is given by the formula $S_{ijt} = P_j G_{ijt}$, where

$$G_{ijt} = \left( \frac{\omega}{d_{ijt}} \right)^\eta$$

(2.7)
is the gain between access point \( j \) and demand point \( i \) during time period \( t \), \( P_j \) is the transmission power of access point \( j \), \( \omega \) is a positive parameter and \( \eta \) is the path loss exponent. The signal to interference plus noise ratio (SINR) at demand point \( i \) from access point \( j \) at period \( t \) is

\[
SINR_{ijt} = \frac{S_{ijt}}{N + \sum_{q \in K \setminus \{k\}} S_{iqt}}
\]  (2.8)

where \( N \) is the ambient noise. A connection is considered successful if \( SINR_{ijt} \geq \sigma \), where \( \sigma \) is some SINR threshold. These equations have been well-demonstrated [51]. The original model (equations (2.1a)-(2.3d)) considers a connection to be nonexistent under the set of constraints (2.3d), where access or demand points and jammers only need to be within some distance from each other. Under this additive interference model, however, we may replace the aforementioned constraint with the following new one, which, after removing the fractions, is:

\[
P_j G_{ijt} x_{ijt} + M_{ijt} (1 - x_{ijt}) \geq \sigma \left( \sum_{q \in K \setminus \{k\}} P_q G_{iqt} y_{qlt} + N \right) \quad \forall i \in I, j \in J, k \in K, l \in L, t \in T
\]  (2.9)

where \( M_{ijt} \) is a large number. This end result comes from Capone, et al. [14]; as they explain, the \( M_{ijt} \) parameter in this inequality makes the problem substantially more difficult to solve. However, they derived an exact reformulation that simplifies the above equation into a covering constraint problem instead of a SINR constraint problem, transforming the equation into (with corresponding dual variable \( \iota_{ijlt} \))
\[
\sum_{q \in C} y_{qlt} \leq |C| \cdot x_{ijt} \quad \forall i \in I, j \in J, l \in L, t \in T, C \subset C : \sum_{q \in C} G_{qit} > r_{ijt} \tag{2.10}
\]

where the set \( C \subset K \backslash \{k\} \) is a cover if \( \sum_{q \in C} G_{qjt} > r_{ijt} \) and \( C \) is the set of all covers. The set of constraints (2.10) demands that if jammers are placed at all of the locations in a cover, there can be no connection for the access-demand point \((i, j)\) pair. The construction of the cover demonstrates the additive effect of the jammers, where the jammers work together to create more interference than the signal strength of an access-demand point \((i, j)\) pair. Here, \( r_{ijt} = \frac{S_{ijt}}{\sigma} - N \) is effectively the normalization of an access-demand point connection as the ratio of the signal strength \( S_{ij} \) to some lower limit threshold \( \sigma \) and free of noise \( N \); thus, \( r_{ijt} \) is a noise threshold such that if the cumulative noise caused by a set of jammers is more than \( r_{ijt} \), then that set forms a cover. From here, the third stage in the new additive tri-level mixed-integer program \([ATLMIP]\) is

\[
Z_2 = \text{Max} \sum_{i \in I} \sum_{j \in J} \sum_{t \in T} S_{ijt} x_{ijt} \tag{2.11}
\]

subject to:

\[
\sum_{j \in A_{it}} x_{ijt} = 1 \quad \forall i \in I, t \in T \quad [\alpha_{it}] \tag{2.12}
\]

\[
\sum_{i \in I} x_{ijt} \leq cw_j \quad \forall j \in J, t \in T \quad [\beta_{jt}] \tag{2.13}
\]
\[
\sum_{l \in L} \sum_{q \in C} y_{qlt} \leq |C| - x_{ijt} \quad \forall i \in I, j \in J, t \in T, C \subset \mathcal{C} : \sum_{q \in C} G_{qit} > r_{ijt} \quad [t_{ijlC}] \quad (2.14)
\]

Fixing \( y_{qlt} \), we can take the dual of this and combine it with the second stage under the same idea as in the \([TL - MIP]\) which allows us to relax a binary constraint on \( x_{ijt} \), giving us the new second stage for the additive bi-level program \([ABLMIP]\)

\[
W(y) = \text{Min} \quad \sum_{i \in I} \sum_{j \in J} \sum_{t \in T} [\alpha_{it} + c \beta_{ijt} + \sum_{C \in \mathcal{C}} t_{ijlC}(|C| - \sum_{l \in L} \sum_{q \in C} y_{qlt})] \quad (2.15a)
\]

subject to:

\[
\sum_{k \in K} y_{klt} \leq 1 \quad \forall l \in L, t \in T \quad (2.15b)
\]

\[
\sum_{l \in L} y_{klt} \leq 1 \quad \forall k \in K, t \in T \quad (2.15c)
\]

\[
\sum_{n \in P_k} y_{n,l,t+1} \leq 1 - y_{klt} \quad \forall k \in L, l \in L, t \in T \quad (2.15d)
\]

\[
\sum_{t \in T} (\sum_{i \in I} \alpha_{it} + \sum_{j \in J} \beta_{ijt} + \sum_{i \in I} \sum_{j \in J} \sum_{l \in L} t_{ijl}) \geq 1 \quad \forall i \in I, j \in J, t \in T \quad (2.15e)
\]

\[
\alpha_{it}, \beta_{ijt}, t_{ijlC} \geq 0 \quad \forall i \in I, j \in J, t \in T, C \in \mathcal{C} \quad (2.15f)
\]
\[ y_{klt} \in \{0, 1\} \quad \forall k \in K, l \in L, t \in T \] (2.15g)

Thus, we propose two models: the non-additive model given by (2.1a)-(2.3d), and the additive model which replaces the set of constraints (2.3d) with the set of constraints (2.10). Again, we note that the product of \( y_{qlt} \) and \( \iota_{ijtC} \) can be linearized in a similar manner as the previous model.

2.5 Solution Methodologies

2.5.1 Branch-and-bound

Because of the mixed-integer nature of the problem, the classic branch-and-bound algorithm as developed by Bard and Moore [7] can be employed on \([BL - MIP]\) directly without modification. We use this basic algorithm as a benchmark for computational tractability since commercial solvers cannot easily solve a multi-level mixed-integer program. We define for each sub-problem \( s \) the leader problem \( L \) as the top level, the follower problem \( F \) as the bottom level, and the high-point problem \( HPP \) as both levels with the objective value of \( F \) removed. We define \( \alpha^{Ls} \) and \( \beta^{Ls} \) as the vector of lower and upper bounds, respectively, of the integer variables controlled by \( L \), and we define \( \alpha^{Fs} \) and \( \beta^{Fs} \) as the vector of lower and upper bounds, respectively, of the integer variables controlled by \( F \). The sets of these bounds, respectively, are \( H^L_s = \{(\alpha^{Ls}, \beta^{Ls})\} \) and \( H^F_s = \{(\alpha^{Fs}, \beta^{Fs})\} \). The sets of indices of the integer variables restricted in sub-problem \( s \) are, respectively, \( S^L_s = \{ q : \alpha^{Ls}_j > 0 \text{ or } \beta^{Ls}_j < UB^L_j \} \) for \( L \), with \( F \) done similarly. Algorithm 2.1 details the algorithm.
Algorithm 2.1 Branch and Bound

1: **Initialization**: Set the iteration number $s = 0$, the current optimal solution $z = -\infty$, the parameters of the sets of bounds, and put the sets of indices for the bounds as empty.

2: **Upper bounds and fathoming**: Attempt to find the solution of $HPP$ and determine the optimal objective value, $z_{HPP}$. If $z_{HPP} \leq z$ or $HPP$ is infeasible, go to Step 7.

3: **Continuous solution**: Attempt to solve the relaxed bi-level program. If infeasible, go to Step 7. Otherwise, store and fix the current solution.

4: **Branching**: If integrality requirements are satisfied by the current solution, go to Step 5. Otherwise, select a fractional-valued variable and place a bound on it. Iterate $s = s + 1$, and update the sets of bounds and indices for the bounds. Return to Step 2.

5: **Bi-level feasible solution**: Fix the current solution $w^{(m)} = (w^{(m)}_j)_{j \in J}$ and solve $F$ to obtain $y^{(m)} = (y^{(m)}_{klt})_{k \in K, l \in L, t \in T}$. Compute $Z$, and if $Z > z$, set $z = Z$, storing the new optimal solution.

6: **Integer branching**: If each upper and lower bound is equal to each other for both of $L$ and $F$, go to Step 7. Otherwise, select one inequality from either of the sets of bounds and update the bound by decreasing the upper bound or increasing the lower bound, depending on the inequality chosen and which set it comes from. Iterate $s = s + 1$ and update the sets of bounds and the sets of indices for the bounds. Return to Step 2.

7: **Backtracking**: If not undetermined node exists, go to Step 8. Otherwise, branch to the newest undetermined node, iterate $s = s + 1$, and update the sets of bounds and sets of indices for the bounds.

8: **Stopping criterion**: If $z = -\infty$, no feasible solution exists. Otherwise, terminate algorithm with the optimal solution.
2.5.2 Implicit enumeration

The tri-level model with binary variables makes the above models (both the additive and non-additive) difficult to solve. In $[BL - MIP]$, the binary bottom-level decision variables make this problem computationally complex, and we cannot reduce the problem to a single level mixed-integer program. An implicit enumeration methodology was derived by Scaparra and Church [111] for planning the defense of a logistic network, and we can employ the methodology directly here. This methodology seeks to find an optimal solution through a search tree, wherein the optimal set of access-demand point connections must identify at least one “critical” location, namely the location that can establish some maximal utility for some set of demand points that could be interdicted. Failure to place an access point at this location would force the utility to become minimized, thus creating a “worst-case” scenario of access-demand point connections that can never be established.

Borrowing terminology from the branch-and-bound algorithm, the algorithm starts at the root of the search tree by solving the lower level of the $[BL - MIP]$ such that no access points are placed, thus seeking to jam as many demand points as possible. The optimal interdiction solution provides branches of the root with location placements for the access points, and at least one of these solutions must be realized. Branching continues until there are no more access points (transforming the branch into a leaf) or there are no more feasible locations to place the access points. The optimal solution is the feasible placement with the largest value of the upper-level objective function. Call the optimal solution value from either the additive or non-additive model $Z$. Algorithm 2.2 details the algorithm.
Algorithm 2.2 Implicit Enumeration

1: **Initialization**: Initialize the node set $M$ with the root node associated with no placement of access points, i.e., $w_j = 0, \forall j \in J$, and set the optimal solution to $z = -\infty$.

2: **Node processing**: Select and remove a node $m$ from $M$. If $n$ is the root node or was created from setting any $w_j = 1$, then solve the corresponding lower-level problem for the corresponding vector $w^{(m)} = (w_j^{(m)})_{j \in J}$, providing the vector $y^{(m)} = (y_{klt}^{(m)})_{k \in K, l \in L, t \in T}$. Define $O_J^{(m)}$ as the set of suitable locations for all the access points, and determine then a set of suitable locations for each $w_j^{(m)}$ in $O_J^{(m)}$. If $Z > z$, set $z = Z$, storing the new optimal solution.

3: **Pruning**: If there is no better placement for the access points, or if $O_J^{(m)}$ is empty, the branch is a leaf, and go to Step 5.

4: **Branching**: Choose a location $j$ from the set of possible locations $O_J^{(m)}$ and create two new nodes. One node forces $w_j = 1$, the other forces $w_j = 0$. Add the newly created nodes to the set $M$.

5: **Stopping criterion**: If the node set $M$ is empty, terminate the algorithm with the optimal solution; otherwise, return to Step 2.

---

2.5.3 Dynamic constraint generation for additive model

Under the additive model, the set of constraints (2.14) poses a computational challenge as the number of covers grows exponentially. To address this issue we consider a restricted problem in which our set of covers $C$ is not complete. Let $\bar{C} \subset C$, and so we desire to add covers to $\bar{C}$ as needed.

We propose a method of dynamically generating such sets using a cutting plane approach. Cutting plane approaches have been used frequently within the literature. Medal [75] uses a similar method to solve a jamming problem subject to a different kind of interference. Israeli and Wood [50] developed the method to maximize the shortest path an interdicted network. Zeng and Zhao [128] developed a more general algorithm for solving general two-stage robust optimization problems. And Dallaire, et al. [30] also developed such an algorithm to solve a transit crew scheduling problem. Let $\bar{C}$ be the set of covers found so far, then the restricted form of this set of constraints is:
\[
\sum_{l \in L} \sum_{q \in C} y_{qlt} \leq |C| - x_{ijt} \quad \forall i \in I, j \in J, t \in T, C \in \bar{C}
\] 

(2.16)

In order to generate new constraints (2.14) that are maximally violated for the incumbent solution, we introduce new, binary variables \( u_l \) that decide whether a given location for a jammer (and so whether a given jammer) is placed within the set of covers. We then solve the following separation problem for each \((i, j, t)\) triplet corresponding to a given access and demand point at some time period:

\[
z^*_{ijt}(\hat{x}, \hat{y}) = \text{Max} \sum_{l \in L} \hat{y}_{klt} u_l - \sum_{l \in L} u_l + \hat{x}_{ijt} \quad (2.17a)
\]

subject to:

\[
\sum_{l \in L} \sum_{k \in K} G_{kjt} u_l > r_{ijt} \quad \forall j \in J, t \in T \quad (2.17b)
\]

\[
u_l \in \{0, 1\} \quad \forall l \in L \quad (2.17c)
\]

Here, \( \hat{x} \), \( \hat{x}_{ijt} \), \( \hat{y} \), and \( \hat{y}_{klt} \) are the incumbent solutions found in the algorithm. In the branch-and-bound algorithm, whenever a new incumbent solution is found, the separation problem is solved for each \((i, j, t)\) triplet, and whenever \(\sum_{l \in L} \sum_{q \in C} \hat{y}_{qlt} > |C| - \hat{x}_{ijt}\), we append the new cover \(\{l \in L : u_l^* = 1\}\) to \(\bar{C}\), where \(u_l^*\) is the optimal solution to the separation problem. In the implicit enumeration algorithm, the same procedure is done during the node processing step when an incumbent \(y^{(m)}\) vector is found. Specifically, the dynamic constraint generation process
is implemented as such: first, a feasible solution in \((x, y)\) is found; second, the dynamic constraint generation methodology is employed; third, the dual parameter(s) for the new constraint(s) are added into the bi-level program; finally, the algorithm continues, and this process repeats with new incumbent solutions.

2.6 Computational analysis

In this section we examined how different topologies of the wireless access points affected the placement and movement trends of the jammers. In many cases, access points either have already been placed and, if relocatable, are limited in movement, or the planner of the network must locate access points according to the layout of structures across the region. In these cases, we were interested in analyzing how common topologies compared against the optimal placement of access points. For these prespecified access point topologies we did not consider the first stage in the original model, instead using the simplified bi-level mixed-integer program \([BLMIP]\) that considered only equations (2.6a)-(2.6g). We considered five topologies, named Partite, Perimeter, Dense, Spacious, and Median, illustrated in Figure 2.3, which also includes one instance of an optimal topology, named Optimal, generated from solving one problem instance. All experiments were run using a Dell OptiPlex 7050 running Windows 10 on 8 dedicated Intel Core i7-7700 processors running at 3.6 GHz with 16 GB RAM. The algorithms were encoded using Python 2.7 with sub-problems solved using Gurobi 7.5.1 and Pyomo 5.2.
The Partite topology had three distinct clusters of access points such that each cluster was far enough from each other cluster such that a demand point too centered between two clusters would possibly be unable to connect to any access point in any cluster. Thus demand points should
have needed to cluster around and within each group. This is akin to several, central facilities, each requiring its own dedicated network, with each partition having its own, often high, demand needs. The Perimeter topology had all access points uniformly distributed across the perimeter of the region with some distance between each access point and the edge of the region to allow demand points to move freely around any access point. Here, demand points would typically be far away from other demand points. The Dense topology had all access points clustered around a central hub, which we took as a ball centered in the region. This is akin to a critical location that may operate on its own network and requires constant connectivity with large demand, regardless of the rest of the region’s demand for connectivity. The Spacious topology had all access points distributed randomly across the entire region such that no two access points were too close to one another. Here, demand points were almost always in the range of at most one or two access points. This was the most random topology, where overall demand for connectivity was uniform across the entire region. The Median topology had all access points distributed uniformly across the diagonals and central vertical and horizontal lines within the region. We note that there is some clustering in the center of the region under this topology. This topology is akin to a campus for which there is some network connectivity over the entire campus, but there is a main “center” (not necessarily the center of the region) with high demand for connectivity. We note that the images are examples only for illustration purposes and do not necessarily contain the coordinates used in our experiments.

For all five topologies, we considered three experiments of 10, 25, and 50 access points, each with a capacity of 15, 5 jammers with jamming radius of 150 feet, and 5 time periods, giving 15 experiments. Each region was 1 square mile. The demand was realized ten times, with 100 demand
points, and the results were averaged. Each of the ten realizations was repeated across all fifteen experiments such that realization \( i \) for experiment \( m \) is the same as realization \( i \) for experiment \( n \).

In the first half of Section 2.6.2 with regards to Figure 2.4, we compared both the additive and non-additive models. However, because the additive is more interesting and more realistic, beginning with the second half of Section 2.6.2 with regards to Table 2.5 and onward, we only showed the results of the additive model ([\textit{ATLMIP}] from Section 2.4.2).

### 2.6.1 Run-time of solution methodologies and model comparison

First, we examined the run-time solutions of the non-additive model for two methodologies: branch-and-bound and implicit enumeration. Table 2.3 shows the results.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Branch-and-Bound (secs)</th>
<th>Implicit Enumeration (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal-10</td>
<td>356</td>
<td>86</td>
</tr>
<tr>
<td>Optimal-25</td>
<td>588</td>
<td>123</td>
</tr>
<tr>
<td>Optimal-50</td>
<td>1253</td>
<td>165</td>
</tr>
<tr>
<td>Partite-10</td>
<td>93</td>
<td>27</td>
</tr>
<tr>
<td>Partite-25</td>
<td>183</td>
<td>41</td>
</tr>
<tr>
<td>Partite-50</td>
<td>307</td>
<td>59</td>
</tr>
<tr>
<td>Perimeter-10</td>
<td>67</td>
<td>18</td>
</tr>
<tr>
<td>Perimeter-25</td>
<td>126</td>
<td>29</td>
</tr>
<tr>
<td>Perimeter-50</td>
<td>237</td>
<td>41</td>
</tr>
<tr>
<td>Dense-10</td>
<td>92</td>
<td>30</td>
</tr>
<tr>
<td>Dense-25</td>
<td>192</td>
<td>55</td>
</tr>
<tr>
<td>Dense-50</td>
<td>401</td>
<td>74</td>
</tr>
<tr>
<td>Spacious-10</td>
<td>127</td>
<td>47</td>
</tr>
<tr>
<td>Spacious-25</td>
<td>336</td>
<td>63</td>
</tr>
<tr>
<td>Spacious-50</td>
<td>771</td>
<td>84</td>
</tr>
<tr>
<td>Median-10</td>
<td>105</td>
<td>28</td>
</tr>
<tr>
<td>Median-25</td>
<td>221</td>
<td>43</td>
</tr>
<tr>
<td>Median-50</td>
<td>518</td>
<td>60</td>
</tr>
</tbody>
</table>
Each experiment is listed as the name of the topology and the number of access points for that experiment. For example, Partite-25 was the experiment with the partite topology and 25 access points. The Optimal experiment was the original optimal placement of the access points \([TL - MIP]\). The implicit enumeration algorithm significantly outperformed the branch-and-bound algorithm. Furthermore, the scalability of the implicit enumeration scheme was relatively stable, where even as the access points doubled, the solution run-time was nearly linear, unlike the branch-and-bound which seemed to grow as rapidly as the as the access points did. As expected, the Optimal experiments took longer than the prespecified topologies because the Optimal required the top level of the tri-level model, where the predetermined locations precluded the need. Excluding the Optimal experiments, the Spacious topology took the longest to solve while the Perimeter took the shortest, with an average of 30 seconds between them. This detail and the disparities across all the topologies showed that the topology played a significant role in computational time.

We then examined the run-time solutions of the additive model for three methodologies: branch-and-bound with dynamically generated covers, implicit enumeration, and implicit enumeration with dynamically generated covers. We omitted the standard branch-and-bound because the smallest problems with only 10 access points each took over thirty minutes to solve, and so we highlighted the significance of the other techniques. Table 2.4 shows the results.
Table 2.4: Run-time of additive model

<table>
<thead>
<tr>
<th>Experiment</th>
<th>BBDGC (secs)</th>
<th>IE (secs)</th>
<th>IEDGC (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal-10</td>
<td>102</td>
<td>103</td>
<td>14</td>
</tr>
<tr>
<td>Optimal-25</td>
<td>158</td>
<td>161</td>
<td>18</td>
</tr>
<tr>
<td>Optimal-50</td>
<td>222</td>
<td>231</td>
<td>21</td>
</tr>
<tr>
<td>Partite-10</td>
<td>38</td>
<td>43</td>
<td>6</td>
</tr>
<tr>
<td>Partite-25</td>
<td>61</td>
<td>71</td>
<td>8</td>
</tr>
<tr>
<td>Partite-50</td>
<td>97</td>
<td>108</td>
<td>9</td>
</tr>
<tr>
<td>Perimeter-10</td>
<td>20</td>
<td>29</td>
<td>3</td>
</tr>
<tr>
<td>Perimeter-25</td>
<td>46</td>
<td>58</td>
<td>6</td>
</tr>
<tr>
<td>Perimeter-50</td>
<td>81</td>
<td>90</td>
<td>8</td>
</tr>
<tr>
<td>Dense-10</td>
<td>69</td>
<td>76</td>
<td>6</td>
</tr>
<tr>
<td>Dense-25</td>
<td>120</td>
<td>127</td>
<td>10</td>
</tr>
<tr>
<td>Dense-50</td>
<td>173</td>
<td>182</td>
<td>14</td>
</tr>
<tr>
<td>Spacious-10</td>
<td>77</td>
<td>83</td>
<td>7</td>
</tr>
<tr>
<td>Spacious-25</td>
<td>140</td>
<td>148</td>
<td>12</td>
</tr>
<tr>
<td>Spacious-50</td>
<td>191</td>
<td>207</td>
<td>15</td>
</tr>
<tr>
<td>Median-10</td>
<td>40</td>
<td>52</td>
<td>5</td>
</tr>
<tr>
<td>Median-25</td>
<td>80</td>
<td>89</td>
<td>9</td>
</tr>
<tr>
<td>Median-50</td>
<td>124</td>
<td>132</td>
<td>12</td>
</tr>
</tbody>
</table>

The BBDGC column shows the run-time of the branch-and-bound algorithm with dynamically generated covers. The IE column shows the run-time of the implicit enumeration algorithm. The IEDGC shows the run-time of the implicit enumeration algorithm with dynamically generated covers. Although the implicit enumeration fared better than branch-and-bound, without dynamically generated covers, it was only feasible on very small problems because all the covers had to be determined at the start. Using dynamically generated covers significantly improved overall computational speed. By applying it to only the branch-and-bound algorithm, it solved the model only slightly faster than the basic implicit enumeration algorithm. However, combining it with the implicit enumeration algorithm gave the most significant speed increase, where the Optimal
experiment only took longer because it had the additional top level to compute. Across all pre-specified topologies, the Perimeter topology was solved in the fastest time while the Dense and Spacious topologies were solved the slowest. Finally, the additive model took longer to solve than the non-additive model across all experiments when comparing the branch-and-bound and implicit enumeration (both without dynamically generated covers) algorithms, again because of the generation of the covers.

These conclusions demonstrate that more work than a traditional branch-and-bound method is necessary to solve even smaller problem sets. For more realistic problems in which access points serve hundreds or thousands of demand points individually, the more stable growth in complexity from the implicit enumeration plus the speed that dynamic constraint generation provides is clearly preferred. Table 2.3 shows a 67%-91% drop in speed, with comparable results between the standard branch-and-bound with dynamic constraint generation and the implicit enumeration with dynamic constraint generation.

### 2.6.2 Solution Quality for Different Topologies

We also compared each of the five topologies from the bi-level program with the optimal placement of access points from the tri-level program, taking $S_{ij} = 1$ and thus we considered only maximizing the total number of connections. Figure 2.4 shows the results.
The experiment names are as before. The Percentage Below Optimal value gives the percentage difference from the optimal solution of total signal strengths jammed, calculated as \( \frac{O - P}{O} \times 100\% \), where \( O \) is the optimal objective value from the original tri-level program, and \( P \) is the objective value from the given experiment. It is clear from Figure 2.4 that placement of the access points near expected concentrations of demand points is essential for any network given some topology since following certain geometries can lead to significantly diminished connectivity should an attack occur. Access points spread too far will not allow demand points to connect to the network (as can be seen from the Perimeter topology). Access points too clustered together give jammers an easier time of severing connections. The Spacious topology proved to be the closest to the
optimal placement of the access points, indicating the Spacious topology naturally gave a strong likelihood of connecting and maintaining connection. The non-additive model consistently gave weaker results except in the Partite topology, likely because of this clustering. In the additive model, connections only needed to be close to jammers, and clusters of access points implied clusters of access-demand point connections were close enough to jammers for them to be more effective.

We then compared the additive and non-additive models directly. Because the additive model is a more accurate representation of how the interference of jammers sums together to mitigate network connectivity, we took its objective value as a base and measured the relative error of the non-additive model to it. This relative error was calculated as $\frac{A - N}{A} \times 100\%$, where $A$ is the objective value of the additive model and $N$ is the objective value of the non-additive model. Figure 2.5 shows the results.

![Figure 2.5: Additive/Non-Additive Relative Error](image-url)
There was a significant amount of relative error across all topologies and the optimal configuration, demonstrating the large differences between the additive and non-additive models. The Dense and Median topologies had the largest relative errors as well as the most significant changes as more access points were added because of the clustering of demand points around clusters of access points. A similar reasoning can be applied to the Partite topology; however, because the Partite topology confined where access points may be placed, demand points’ ability to relocate to maintain connectivity was limited. Limiting the locations for connections to be made allowed jammers to not have to work as hard to jam more connections under the non-additive model. This confinement explained why the Partite topology not only did not give as much relative error as the Dense and Median topologies in spite of similar clustering problems but also why there was not much of a change in relative error despite more access points placed. The Perimeter provided the least relative error for the exact opposite reason: demand points and access points were so spread out that jammers working together under the additive model or requiring direct interference under the non-additive model were not able to produce as much of an effect.

One final point concerns our choice of $S_{ij} = 1$. To summarize, the Spacious topology had the smallest Percentage Below Optimal value while the Perimeter had the largest, and the Perimeter topology had the smallest Additive/Non-additive Relative Error value while the Median had the largest and the other three topologies were all higher than the Optimal. Under the consideration of $S_{ij}$ taking any value between 0 and 33, each of these conclusions was the same (with different numerical values) except two: both the Spacious and the Partite topologies had smaller Additive/Non-additive Relative Error values than the Optimal topology, and the Perimeter Additive/Non-additive
Relative Error values were worse than the Optimal topology. Thus, depending on how a defender defines the utility function, certain topologies have a stronger or weaker disparity between additive interference and no interference. Because [51] cautions strongly that the additive interference models are more realistic, then the Perimeter topology (which had larger relative errors than the Optimal) can behave very differently in reality (using the additive interference model) than it does in expectation (without considering interference).

Finally, as a visual representation, we include Figure 2.6 which overlays the Optimal Topology and Median Topology together, showing only the access point distribution and omitting numerical details. Intuitively, robustness against jamming attacks might involve a non-clustering of access points, of which the Median topology does except near the center. However, depending on the demand point distribution, it may be the case that jammers are not located at specific cluster of demand points, yet a single access point or even two may not have enough capacity to satisfy all the demand points. As can be seen in the figure, there are several clusters of access points, and so a key takeaway from the figure is that preconceived topological notions of access point distribution may not yield an optimal planning strategy.
2.6.3 Problem sensitivity analysis

We were also interested in seeing the effects of optimally adding $n$ access points to a given topology and improving the overall network connectivity in terms of additional connections. Here, we ran the original model, only all initial access points’ locations had been predefined according to the given topologies, and the first stage was considered with these locations and the optimal placement of 1, 3, and 5 access points to each of the experiments. The optimal placement of additional access points did not require that the points fit within any given topology. That is, theoretically, they could have been placed anywhere, regardless of the given topology. We kept the same utility function as before, so this experiment measured how additional access points allowed for additional connections in the network. We then repeated the experiments with a total of 10
jammers to see the impacts of additional access points as more jammers attacked the network.

Table 2.5 shows the results, where each result indicates the average number of connections the additional access points allow.
### Table 2.5: \( n \) Access Point Additions

<table>
<thead>
<tr>
<th>Experiment</th>
<th>( n = 1 )</th>
<th>( n = 3 )</th>
<th>( n = 5 )</th>
<th>( n = 1 )</th>
<th>( n = 3 )</th>
<th>( n = 5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partite-10</td>
<td>5.87</td>
<td>12.32</td>
<td>25.41</td>
<td>15.00</td>
<td>40.25</td>
<td>56.05</td>
</tr>
<tr>
<td>Partite-25</td>
<td>5.24</td>
<td>10.33</td>
<td>21.49</td>
<td>14.88</td>
<td>38.08</td>
<td>55.15</td>
</tr>
<tr>
<td>Partite-50</td>
<td>4.98</td>
<td>10.10</td>
<td>21.56</td>
<td>14.80</td>
<td>36.55</td>
<td>54.92</td>
</tr>
<tr>
<td>Perimeter-10</td>
<td>2.30</td>
<td>4.70</td>
<td>8.35</td>
<td>15.00</td>
<td>18.10</td>
<td>20.00</td>
</tr>
<tr>
<td>Perimeter-25</td>
<td>1.93</td>
<td>4.56</td>
<td>6.21</td>
<td>15.00</td>
<td>18.08</td>
<td>19.95</td>
</tr>
<tr>
<td>Perimeter-50</td>
<td>1.72</td>
<td>3.05</td>
<td>5.83</td>
<td>15.00</td>
<td>18.00</td>
<td>19.95</td>
</tr>
<tr>
<td>Dense-10</td>
<td>9.87</td>
<td>15.48</td>
<td>21.03</td>
<td>14.90</td>
<td>36.48</td>
<td>50.07</td>
</tr>
<tr>
<td>Dense-25</td>
<td>4.22</td>
<td>10.36</td>
<td>15.87</td>
<td>14.90</td>
<td>32.50</td>
<td>47.70</td>
</tr>
<tr>
<td>Dense-50</td>
<td>2.15</td>
<td>8.32</td>
<td>14.27</td>
<td>14.50</td>
<td>32.09</td>
<td>45.55</td>
</tr>
<tr>
<td>Spacious-10</td>
<td>9.32</td>
<td>18.74</td>
<td>28.83</td>
<td>15.00</td>
<td>45.00</td>
<td>59.92</td>
</tr>
<tr>
<td>Spacious-25</td>
<td>5.73</td>
<td>11.24</td>
<td>15.83</td>
<td>15.00</td>
<td>38.67</td>
<td>52.44</td>
</tr>
<tr>
<td>Spacious-50</td>
<td>2.11</td>
<td>4.83</td>
<td>7.07</td>
<td>15.00</td>
<td>36.60</td>
<td>51.86</td>
</tr>
<tr>
<td>Median-10</td>
<td>14.57</td>
<td>20.32</td>
<td>38.55</td>
<td>15.00</td>
<td>45.00</td>
<td>63.30</td>
</tr>
<tr>
<td>Median-25</td>
<td>9.31</td>
<td>11.12</td>
<td>16.37</td>
<td>15.00</td>
<td>40.16</td>
<td>62.12</td>
</tr>
<tr>
<td>Median-50</td>
<td>8.37</td>
<td>10.44</td>
<td>16.00</td>
<td>15.00</td>
<td>40.04</td>
<td>60.75</td>
</tr>
</tbody>
</table>
When considering the original experiment of only 5 jammers, the addition of a single access point did not really improve overall connectivity, except in the Median topology with the fewest access points. This was clearest when there were already a larger number of access points available. However, in the presence of more jammers, the addition of more access points became significantly more important. For all experiments, the optimal placement of a single new access point allowed its capacity to be fully realized or almost so. Three new access points greatly increased the number of additional connections across all experiments, as did five new access points. Regardless of clustering that, as stated previously, aided jammers in severing connections, the optimal placement of a single access point was not enough to allow for more connectivity. Thus the capacity of a network was critical here, as adding more access points began to greatly increase total connectivity when few access points were in place. In fact, in the presence of clustering (such as the Partite, Dense, and the center of the Median topologies), adding more capacity significantly increased overall connectivity with even a few access points. Finally, there was a mitigating factor present in all columns where, as the number of access points increased for any topology, the amount of improved connectivity was lessened. This observation was quickly seen in the presence of more jammers, where increasing from 1 additional access point to 3 additional access points provided a significant gain in total connections, but 5 additional access points did not provide as much of an increase in number of connections. The placement of the access points was significant, but if the topology already had high connectivity, then there was less of a need for demand points to connect to new access points.
2.6.4 Utility function changes

There are a variety of ways attackers seek to disrupt networks. Some may be more interested in only attacking the most critical connections, while others may be more interested in severing as many connections as possible, regardless of the strength of each connection. Equation (2.3a), \( \text{Max} \sum_{i \in I} \sum_{j \in J} \sum_{t \in T} S_{ij} x_{ijt} \), is a generalized utility function that may be interpreted in different ways, and we examine three different possibilities, which we describe below.

1. **Range**: each access-demand point connection has an associated strength \( S_{ij} \) which takes a value in the range \([0, 33]\), as was used by Schweitzer, et al. [112], and any nonzero signal strength implies an established connection.

2. **Unity**: each connection has a unit signal strength, i.e., \( S_{ij} = 1 \), and links between demand points and access points are either connected or not. This is equivalent to having a signal quality greater than a threshold of 0.

3. **Tolerance**: each connection has some associated signal strength \( S_{ij} \), but if \( S_{ij} < \delta \) where \( \delta \) is some nonzero threshold, the connection is considered too weak to be useful and so the link is considered unconnected.

In the first case, it was more critical for jammers to be located where maximum signal strengths occur, regardless of the number of signals actually being jammed, which is plausible whenever certain devices are significantly more critical to being connected to the network than others. The second case considered our model minimizing the sum total of \( x_{ijt} \) variables, or the total number of connected demand points. In this case, it was more critical that jammers sought the densest concentrations of demand points. In the final case, which is a combination of the previous two, jammers needed to work harder to seek larger concentrations of stronger signals, ignoring those signals that are considered too weak. In general, this effort can be more difficult because there may be instances where there is a large cluster of connections whose combined signal strengths...
and number of connections are larger than other clusters; however, they must be ignored because individually the connections are too weak.

To quantify the results, we ran the original tri-level program to obtain the optimal value $J$, which was the total utility function value after jamming has taken place. Next, we fixed the access and demand points and remove the jammers, and we recalculated the total network connectivity based on the utility function, calling this $U$, the total utility function value if no jamming occurred. Finally, we measured a relative error, $\frac{U - J}{U} \times 100\%$, and we called this our jamming impact. We note that in the case of $S_{ij} = 1$, $U$ will always attain total connectivity (100% of demand points are connected), and so the results are a measure of the percentage of jammed connections. For our experiments, we took $\delta = 10$ for the Tolerance utility. Table 2.6 shows the results.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Range (%)</th>
<th>Unity (%)</th>
<th>Tolerance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal-10</td>
<td>5.35</td>
<td>12.17</td>
<td>17.55</td>
</tr>
<tr>
<td>Optimal-25</td>
<td>4.83</td>
<td>7.32</td>
<td>6.73</td>
</tr>
<tr>
<td>Optimal-50</td>
<td>4.12</td>
<td>6.98</td>
<td>6.02</td>
</tr>
<tr>
<td>Partite-10</td>
<td>13.11</td>
<td>23.48</td>
<td>19.98</td>
</tr>
<tr>
<td>Partite-25</td>
<td>9.76</td>
<td>12.99</td>
<td>11.67</td>
</tr>
<tr>
<td>Partite-50</td>
<td>8.00</td>
<td>12.24</td>
<td>11.43</td>
</tr>
<tr>
<td>Perimeter-10</td>
<td>9.55</td>
<td>16.73</td>
<td>17.31</td>
</tr>
<tr>
<td>Perimeter-25</td>
<td>5.26</td>
<td>10.05</td>
<td>9.38</td>
</tr>
<tr>
<td>Perimeter-50</td>
<td>4.97</td>
<td>9.32</td>
<td>8.67</td>
</tr>
<tr>
<td>Dense-10</td>
<td>20.41</td>
<td>27.34</td>
<td>24.04</td>
</tr>
<tr>
<td>Dense-25</td>
<td>9.02</td>
<td>14.55</td>
<td>13.49</td>
</tr>
<tr>
<td>Dense-50</td>
<td>8.87</td>
<td>14.09</td>
<td>12.52</td>
</tr>
<tr>
<td>Spacious-10</td>
<td>17.34</td>
<td>29.60</td>
<td>22.16</td>
</tr>
<tr>
<td>Spacious-25</td>
<td>10.42</td>
<td>16.02</td>
<td>13.68</td>
</tr>
<tr>
<td>Spacious-50</td>
<td>9.87</td>
<td>15.67</td>
<td>12.51</td>
</tr>
<tr>
<td>Median-10</td>
<td>16.78</td>
<td>26.06</td>
<td>19.45</td>
</tr>
<tr>
<td>Median-25</td>
<td>9.14</td>
<td>15.38</td>
<td>11.56</td>
</tr>
<tr>
<td>Median-50</td>
<td>8.65</td>
<td>14.43</td>
<td>10.03</td>
</tr>
</tbody>
</table>
Here, the Range column gives a percentage of how much total signal strength was jammed, the Unity column gives a percentage of how many signals were jammed, and the Tolerance column gives a percentage of how many signals fell below the tolerance after jammers were optimally placed to reduce their signal quality. There was a very clear trend in the data showing that the Unity column was larger than Tolerance column which was larger than the Range column. Also, there were large disparities among the three columns. Notable exceptions included the Partite topology, the differences between the Unity and Tolerance columns in the Perimeter and Dense topologies, and the differences between the Tolerance and Range columns in the Median topology. The low disparities between the Unity and Tolerance columns in the Perimeter and Dense topologies were likely because any random placement of demand was likely to place the demand farther than desired from any given access point. Thus, if an access-demand point connection was susceptible to jamming at all, it was immediately jammed. Contrast the Range column where jammers were likely to avoid clusters of demand points that might be farther away from access points, simply seeking out the strongest signal strengths. This analysis is further backed by Figure 2.4 for the Perimeter topology, which had the worst objective value compared to the other topologies. A similar analysis can be applied to the Median topology and the low disparity between the Range and Tolerance columns, where the demand points near access points, especially those near the center of the region, were likely to have a strong signal strength. This last point is also extendable to the Partite topology and the low disparities among all three columns because the access points themselves were clustered together.
2.6.5 Trend experimental results

Between each time period, jammers were allowed to move no more than $R$ feet away from their current positions. Ideally, understanding how the jammers behaved as they moved between time periods allows further insights into the overall network robustness against an attack. Continuously and optimally relocating jammers between time periods under a movement restriction should force the jammers to behave in such a way that they seek out dense gatherings of demand points or seek those access points that have most of their capacity filled. Under this knowledge, the defender may be able to identify certain locations or access points that are highly critical to the overall connectivity of the network. To facilitate network robustness insights, we examined four features: the average percentage of connections jammed, the average distance jammers moved between time periods, the average percentage of connections jammers shared, and the average distance any jammer had with the closest jammer to it. With regards to the average percentage of connections jammers shared, we illustrate this idea with an example. Suppose there are 100 demand points and 2 jammers, and both jammers are placed such that 5 demand points are within the radii of both jammers. Then the percentage of connections shared across the jammers is 5%. We note that access and demand points need not fall directly within a jammer’s radius for the connection to be jammed as in the additive model, as illustrated in Figure 2.2. However, this metric is not necessarily concerned with the effects of jamming itself. Examining only the shared demand points allows a defender to see which areas in the network are under threat of an attack; measures could then be taken to mitigate an attack as clusters of jammers could identify critical infrastructures. For all values, we calculated them for each jammer and then took the average of all five jammers. Each jammer had a movement restriction of 100 feet. Table 2.7 shows the results.
<table>
<thead>
<tr>
<th>Experiment</th>
<th>Interfered (%)</th>
<th>Distance</th>
<th>Shared (%)</th>
<th>Neighbor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal-10</td>
<td>14.33</td>
<td>39.38</td>
<td>0.00</td>
<td>406.73</td>
</tr>
<tr>
<td>Optimal-25</td>
<td>9.67</td>
<td>45.22</td>
<td>0.00</td>
<td>336.21</td>
</tr>
<tr>
<td>Optimal-50</td>
<td>8.33</td>
<td>63.21</td>
<td>1.33</td>
<td>237.33</td>
</tr>
<tr>
<td>Partite-10</td>
<td>23.33</td>
<td>4.30</td>
<td>18.00</td>
<td>125.31</td>
</tr>
<tr>
<td>Partite-25</td>
<td>26.67</td>
<td>18.31</td>
<td>20.33</td>
<td>78.52</td>
</tr>
<tr>
<td>Partite-50</td>
<td>29.33</td>
<td>32.19</td>
<td>18.67</td>
<td>65.57</td>
</tr>
<tr>
<td>Perimeter-10</td>
<td>61.33</td>
<td>48.3</td>
<td>0.00</td>
<td>2358.38</td>
</tr>
<tr>
<td>Perimeter-25</td>
<td>46.00</td>
<td>60.07</td>
<td>1.00</td>
<td>3203.90</td>
</tr>
<tr>
<td>Perimeter-50</td>
<td>10.67</td>
<td>90.33</td>
<td>5.25</td>
<td>3374.44</td>
</tr>
<tr>
<td>Dense-10</td>
<td>38.67</td>
<td>10.01</td>
<td>14.67</td>
<td>207.49</td>
</tr>
<tr>
<td>Dense-25</td>
<td>34.67</td>
<td>14.48</td>
<td>18.33</td>
<td>180.96</td>
</tr>
<tr>
<td>Dense-50</td>
<td>32.00</td>
<td>17.62</td>
<td>20.00</td>
<td>157.73</td>
</tr>
<tr>
<td>Spacious-10</td>
<td>18.00</td>
<td>15.23</td>
<td>3.50</td>
<td>660.86</td>
</tr>
<tr>
<td>Spacious-25</td>
<td>17.33</td>
<td>20.07</td>
<td>5.00</td>
<td>501.28</td>
</tr>
<tr>
<td>Spacious-50</td>
<td>17.33</td>
<td>18.90</td>
<td>10.33</td>
<td>473.33</td>
</tr>
<tr>
<td>Median-10</td>
<td>16.00</td>
<td>40.27</td>
<td>0.00</td>
<td>1280.39</td>
</tr>
<tr>
<td>Median-25</td>
<td>25.33</td>
<td>28.93</td>
<td>8.00</td>
<td>1147.62</td>
</tr>
<tr>
<td>Median-50</td>
<td>38.67</td>
<td>15.56</td>
<td>16.67</td>
<td>1002.27</td>
</tr>
</tbody>
</table>

The Interfered column shows the average percentage of total of connections that all jammers were able to interfere with across all 5 time periods. The Distance column shows the average number of feet the jammers moved between each time period for all time periods. The Shared column shows the average percentage of jammed connections jammers share. The Neighbor shows the average distance any jammer has with the nearest jammer to it.

The table showed robustness in the optimal placement of access points under the original model with the smallest percentage of connections interfered as expected. Based only on the percentage of connections interfered, the Spacious topology was closest to the original model. Interestingly, it was also robust from the attacker’s perspective, as there was not much change in the percentage
of connections interfered. Schweitzer, et al. [112] found that increasing the number of access points (not necessarily optimally) produced a mitigating effect on the overall connections added (and thus the amount of connections that can be jammed), which was backed here. The Partite topology also showed robustness against adding access points, an intuitive conclusion since all the access points were clustered together. Because of the larger spacing between access points in the Perimeter topology, particularly with fewer access points, demand points tended to cluster closer to one another in order to establish connections. As more access points were added, these clusters lost density, and so this topology had the most significant changes to the interference percentage. According to its topology, this meant there were fewer other access points to possibly be reconnected to in the event of jamming, thus providing the largest percentage of interfered connections. Adding more access points, however, showed a significant improvement in network robustness against jamming.

In all cases, although the jammers could have moved up to 100 feet between time periods, none of them ever reached this limit. With the exception of a few problem instances, many jammers shared connections, which was necessary in maximizing the impact under the additive model, especially if the goal was to maximize jamming impact under a signal quality utility function, either through a signal strength or a quality tolerance as explained in Section 2.6.4. Finally, with the exception of the Partite and Dense topologies, jammers also tended to stay relatively far away from one another.
2.7 Conclusions
2.7.1 Discussion

Ensuring the robustness of wireless networks against attackers, as well as the development of networks that can mitigate the effects of an attack, is an important area of research. This chapter examines the optimal placement of access points as a means of maximizing connectivity under the optimal placement and movement of jammers. A tri-level mixed-integer programming model is developed, and both additive and non-additive models are considered. To solve the model, a branch-and-bound algorithm is implemented, and an implicit enumeration scheme is developed to produce optimal results faster. In both algorithms for the additive model, the inclusion of covers by dynamic constraint/cover generation also significantly improved computational speed, and the results in Section 2.6.1 show that this problem is solvable within a reasonable amount of time for both models.

The results in the remainder of Section 2.6 consider a topological analysis of networks in designing a network that is more robust against jamming attacks. We demonstrate that for a specified topology, optimally placing $n$ additional access points provides the largest overall benefit to the Partite topology regardless of the number of access points already in place, provided enough access points are added; meanwhile, the Median topology benefits the most when a considerable number of jammers are deployed. In general, the Partite and Median topologies are the most robust against jamming attacks when considering utility as total signal strength, number of connections, or as a tolerance allowance. Although our results actually show the Perimeter topology as the most robust, there is a caveat: in general, the Perimeter topology has low utility when the demand is not distributed along the perimeter, and so while the topology may be robust against jamming, it is not
necessarily an optimal topology for network connectivity. In a direct comparison to the optimal placement of access points and jammers, however, the Spacious topology is actually the closest in terms of relative error of the objective function. Furthermore, jammers are least successful in jamming overall connections in the Spacious topology. Jammer movement is also relatively stable for the Spacious topology, allowing a defender to assume independence between jammer movement and number of access points and thus strategize with less variability. Across all topologies, higher movement among jammers between time periods keeps them relatively farther away from each other, which is an important consideration for the additive model since jammers must work together to effectively interfere with the network. As expected, the Dense and Partite topologies have significant sharing of connections among jammers, and with their low jammer movement and distance, finding ways of mitigating jammers’ efficacy becomes more difficult (but more necessary).

We finally note that while the problems we have solved in this chapter are small in comparison to real-world networks, where there may be hundreds of access points and thousands or millions of demand points, our results show general trends of the common topologies. Individual demand points here may be similar to clusters of demand points in the real world, where the signal strength would increase accordingly and the threshold utility (now interpreted as the proportion of demand points able to still access the network) may become more appropriate. For a decision maker or defender, our conclusions provide insights into certain behaviors of jammers and robustness against attacks, such as the Partite topology being fairly robust against an attack but perhaps not the most desirable topology given the amount of empty space inside the region demand points would inevitably find themselves in.
2.7.2 Future Work

One possible future research regards the placement of demand. As demand points roam between time periods, they should have a higher probability of moving towards those unjammed access points with available capacity under the presence of jamming. This would likely lead to a stochastic programming interpretation of the problem.

Another extension relates to the access points. Our model assumed fixed access points, but mobile access points could be considered. This idea could lead to a game-theoretic approach, where the defender might have a constraint that considers the network “active” if the utility can be maintained above a certain threshold for all time periods, while the attacker needs to only interfere with the network enough to drop the utility below this threshold. This attack could be modeled with the jammers degrading the signal strength rather than assuming jammers can always completely jam a signal.
CHAPTER III

DATA-LINK LAYER: MAXIMUM NETWORK LIFETIME UNDER THE THREAT OF
DENIAL-OF-SLEEP ATTACKS

3.1 Introduction

One goal of a wireless communication network is to transmit as much information as possible. This point is even more significant when the network itself may have a limited lifespan, such as when all the access and demand points have limited battery capacities. Such networks include ad hoc mesh networks (AHMNs) and wireless sensor networks (WSNs). In both cases, nodes have zero or very limited mobility. AHMNs, like other ad hoc networks, do not rely on preestablished infrastructure and may be put together quickly but require extensive connectivity over the entire region through a variety of devices that all communicate with one another or otherwise transmit information (thus the “mesh” component). When a given node is somehow removed from the network, the remaining nodes can all still communicate with one another. Many U.S. military forces rely on AHMNs using ruggedized laptops, but a more civilian example includes modernized electric smart readers, where electric meters deployed on residences communicate to one another until they eventually reach the central office for billing and thus do not need a human reader. WSNs, on the other hand, generally acquire a specific kind of data, usually environmental conditions for research laboratories, and all the equipment is standardized to use the same transmission types. They are generally larger and more spread out than AHMNs, as well. Preventing several transmitters
from transmitting information is a priority for an attacker. There are a variety of methods of how attackers may deny service to these networks, collectively referred to as denial-of-service (DoS) attacks. Jamming, where malicious devices broadcast signals that overwhelm legitimate signals and disrupt network traffic, is one such DoS attack. When networks have limited lifespans, the effective transmission of information becomes the highest priority, and rather than denying service explicitly, an attacker may instead choose to reduce the lifespan of the network, an attack known as a denial-of-sleep (DoSL) attack. For nodes in a network with limited lifetime, the device that forces the nodes to be continually active and thus drain their energy more rapidly are dangerous. These kinds of attacks can be done more stealthily than direct jamming attacks because the amount of information transmitted is too small be detected by a human operator, and the networks vulnerable to them, such as WSNs and AHMNs, typically have large numbers of transceiver nodes that the addition of a new, malicious transmitter node is not readily noticed. While there has been significant research both in maximizing throughput in a network that has a limited lifetime as well as research in what denial-of-sleep devices are, how they behave, and how to detect and respond to them, nothing has been done to merge the two ideas together. That is, we still do not know the actual impacts DoSL devices have on a network when they are deployed by intelligent attackers. The goal of this chapter is to study the placement of denial-of-sleep devices for disrupting a network’s ability to transmit information given battery lifetimes for each node. To this end, we develop a bi-level mixed-integer programming (BL-MIP) model that minimizes the maximum lifetime of the network. We also deploy an implicit enumeration scheme and analyze the impacts DoSL devices have on networks to gain insight to this particular attack.
3.2 Related Literature

Some wireless networks have such a unique structure that attacks on them should not typically come from jammers but instead should come from attacking the battery source. As such, we move from the physical layer and into the data-link layer. Padmavathi and Shan [93] provided a survey of a variety of attacks specific to WSNs, while Satish, et al. [119] provided a survey of jamming attacks to wireless networks in general. Deng, et al. [31] discussed path-based attacks and defenses for general denial-of-service (DoS) attacks. Physical layer security has been paramount to defending wireless networks, pioneered primarily by Wyner [125] with a memory-less wiretap channel that was discrete and examined for secure communications in case of an eavesdropper. Foschini and Gans [36] considered multiple antennas while Zou, et al. [134] considered cooperative relays, thus mitigating placements of malicious nodes. These methods are improved by Wang, et al. [121] with hybrid cooperative beam-forming and jamming. A joint physical-application layer framework for security was given by Zhou, et al. [132] by examining and exploiting both physical signal processing and authentication of data transmissions.

With regards to energy draining attacks on WSNs, a focus of our research, Manju, et al. [73] proposed a method to defend against DoSL devices by considering the network organization and selective level authentication. Choi, et al. [23] adds to a security model by providing an advanced encryption standard algorithm. Peres, et al. [100] considered generating secret keys by deriving these keys from a single shared key all nodes in the network understood. Rafik, et al. [101] further improves the previous method by considering user authentication and symmetric cryptography to automate the process. Some work has been done by resource exhaustion where networks have finite amounts of data they can transmit rather than attacking the batteries themselves. Malicious cycles
as by Chan and Perrig [17] are also discussed, where information is continuously transmitted in loops over the network rather than seeking out a network’s sink, thus forcing nodes to continuously communicate with one another. Flood attacks, where multiple connection requests are sent to the server, are another kind of energy draining attack, and such attacks have been defeated with cryptography as Aura, et al. [6] had done. Other defensive attempts have been made in minimal-energy routing by minimizing transmission energy or distance as by Chang and Tassiulas [19]. While the research is clear on how to defend against DoSL devices, none of these papers have considered the impacts DoSL devices have on networks.

The maximum flow with limited network lifetime, another topic relevant to our research, has been studied in a variety of contexts. One context is the maximum lifetime of the entire network, where once some prespecified node (usually a critical node) has been depleted, the network is no longer active [53, 54]. This has been somewhat generalized to include an arbitrary pair of nodes with some battery capacity at each node [70], and efforts to conserve energy as much as possible has also been studied [18]. Further, the network lifetime problem has been studied in the context of constrained link usage as a multi-objective optimization problem [21]. For security purposes, the deployment of redundant nodes to enhance lifetime has also been studied [59]. An application of the original problem with the lifetime of one battery source for disaster recovery has also been studied [135]. The general problem has also been redefined in the context of a shortest path aggregation tree [113]. As before, none of these papers have considered an adversary whose goal is to directly attack the lifetime of a network.

The biggest threat to any node with a limited lifespan is a device that forces it to run continuously to drain the life, known as a denial-of-sleep (DoSL) device. There are several papers that
discuss what DoSL devices are, how they work, and how to defend against them \[104, 103, 20, 57\], as well as examining how to conserve energy by DoSL prevention \[84\]. However, to the best of our knowledge, no author has examined how these DoSL devices impact the lifetime-constrained maximum flow problem.

All of the aforementioned topics consider the network as a physical entity and attack physical parts of the network, either the devices themselves or the batteries supplying the devices’ power. In the network layer, direct access to the network must be achieved, and this first layer is particularly vulnerable to malicious and otherwise anomalous communication attempts. Botnets are an ever-increasing problem in wireless communication activity, and while many methodologies exist, most of the literature assumes the botnet to be identified is of a certain type or behaves a certain way, and many methods are created using small and/or synthetic datasets that do not scale well to large datasets. The purpose of this of examining this layer is to detect arbitrary botnets by training on a very large, real dataset. Defining the correct framework and selecting appropriate features are the most important aspects of this research topic.

### 3.3 Problem Description

Our problem consists of a network with a set of nodes which act as transceivers: they are able to both receive data and transmit data to other nodes. Without loss of generality, we isolate two nodes as a source and sink where the source needs to communicate to the sink specifically, and so all other nodes act as intermediate nodes to help facilitate the transmission of data. Each intermediate node is powered by a battery that has a limited lifespan. Once the battery has been depleted, that node is considered removed from the network, and so the source may need to seek
a less-than-optimal route through the network to communicate to the sink. Each node has some required transmission power to facilitate communication along a link and a rate at which information is generated; both of these determine the rate at which the battery drains, or, more specifically, determine the lifetime of a given node. Each link has a maximum load or capacity it can maintain to facilitate the communication. The goal of the source is to maximize the throughput, or flow, across the network.

Seeking to cripple the network, an attacker seeks to optimally deploy a set of DoSL devices so as to drain the most critical nodes and minimize the throughput of the network. These DoSL devices send out illegitimate information at a high rate so as to communicate with intermediate nodes at a high intensity, thus draining the nodes’ batteries rapidly; the rapid draining is a result of intermediate nodes passing more information than originally intended. Intermediate nodes cannot discern legitimate information from illegitimate information and simply see a DoSL device as some other node desiring to pass information along to the sink node via path-based injection; this information, illegitimate in the sense that it should not be there (and thus we will refer to it as illegitimate information throughout the chapter), is generally acceptable information as far as the network is concerned. This information is typically eavesdropped and given minor modifications by an adversary before an attack begins, and then this information is transmitted across the network during an attack [31]. This extra load of information requires more energy to transmit. We seek to solve the problem of optimally placing DoSL devices to create such an attack on a lifetime-constrained maximum flow problem. Naïvely, the attacker should attempt to deploy the DoSL devices near the nodes that contain the links with the largest capacities, thus reducing the potential maximum flow. However, it is well known from the literature on maximum flow network
interdiction that this is not an optimal strategy [4]. Another naïve approach is for the attacker to deploy DoSL devices near nodes that have the largest battery lifespan. However, links from these nodes do not necessarily have high throughput rates, and so attacking these nodes may not damage the network much at all.

Instead, the attacker should attempt to merge the previous two strategies, deploying the DoSL devices near those nodes that are being used the most, have large battery lifespans, and can contribute greatly to the communication rate. These DoSL devices also create an additive effect; dummy information is generated by these devices that requires the nodes in the network to receive the information which further drains the battery. More devices generate more dummy information, draining the battery at a faster rate. An analogy of two people conversing while filtering out extraneous conversations going on around them (noise) is appropriate; more noise requires more effort for the people conversing to not become distracted and pay more attention to their immediate conversation. That said, there is no apparent limit on how many DoSL devices can attack a node; theoretically, enough devices could rapidly drain a battery. Realistically, a node is more likely to become congested with information based on its upload/download rate before it is drained, and so a cluster of DoSL devices would become more similar to a DoS attack by overwhelming attempts at service through the node than a DoSL attack. Also, unlike with DoS attacks where a defender simply needs to keep an eye on which nodes are active, a DoSL attack may be more difficult to spot as there could be a number of reasons a battery is draining more rapidly than expected, one of which being that the defender may simply mistake dummy information for legitimate information. Unless information is being examined directly at a given node, there is not necessarily any
strong reason to not consider the possibility that users are communicating more frequently over a network, such as in an emergency.

3.4 Mathematical model

To model the program described in Section 3.3, we present the following Bi-Level Mixed-Integer Programming \([BL - MIP]\) model, where the top level solves the DoSL device placement problem and the bottom level solves the energy-constrained maximum flow problem. We start with a set of nodes \(N\) indexed by \(i\). As node \(i\) attempts to communicate to all other nodes \(j \in V_i\), where \(V_i\) is the set of nodes connected to node \(i\), there is a transmission power \(P_{ij}\) required for node \(i\) to communicate with node \(j\) that drains the transmitter node \(i\)'s battery which has energy \(B_i\) and a transmission rate \(T_{ij}\) that allows communication. Here, the total energy used at node \(i\) over the lifetime of the network is \(\sum_{j \in V_i} E_{ij} f_{ij} = \sum_{j \in V_i} (\frac{P_{ij}}{T_{ij}}) f_{ij}\), where \(E_{ij}\) is the energy spent by node \(i\) to transmit one unit of information to node \(j\), \(f_{ij}\) is the total lifetime flow from node \(i\) to node \(j\) and thus is the amount of information actually transmitted. We assume the links are symmetric so that \(E_{ij} = E_{ji} \forall i \in N, j \in V_i\). Further, the rate at which information is generated at node \(i\) is \(S_i\), which acts as a supply for each node, and this information must be communicated to the sink. Since each node has a predefined battery power \(B_i\), there exists a corresponding lifetime for each node \(L_i\) under a given flow \(f = (f_{ij})_{i \in N, j \in V_i}\), which is defined as

\[
L_i(f) = \frac{B_i}{\sum_{j \in V_i} E_{ij} T_{ij} f_{ij}} \tag{3.1}
\]

For the purposes of this model, however, we classify \(L_i\) as a variable and enforce relationship (3.1) implicitly in our mathematical programming model.
We also have a list of possible locations $K$ for a set of DoSL devices, indexed by $k$. These DoSL devices operate by sending packets to nearby nodes, either as dummy information or legitimate information that is simply copied, with the goal of forcing nodes to continue to pass this information along to other nodes and thus deplete the battery faster. As these attacks occur frequently, there is an associated rate of energy loss, $D_{ik}$, that a device at location $k$ has on node $i$. These attacks must be small, however, or these devices may be detected early, particularly in WSNs where operators are monitoring information and may notice a sudden spike. In general, these devices are most effective when used to cause information to become rerouted and drain more of the network to focus exclusively on draining individual batteries completely.
Table 3.1: List of Parameters for $[BL − MIP]$

<table>
<thead>
<tr>
<th>Indices</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i, j$</td>
<td>index for nodes</td>
</tr>
<tr>
<td>$k$</td>
<td>index for locations of DoSL devices</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{ij}$</td>
<td>the maximum flow a link from node $i$ to node $j$ can support</td>
</tr>
<tr>
<td>$B_i$</td>
<td>the initial battery capacity at node $i$</td>
</tr>
<tr>
<td>$Y$</td>
<td>the maximum number of DoSL devices available</td>
</tr>
<tr>
<td>$P_{ij}$</td>
<td>the required transmission power to transmit from node $i$ to node $j$</td>
</tr>
<tr>
<td>$T_{ij}$</td>
<td>the transmission rate from node $i$ to node $j$</td>
</tr>
<tr>
<td>$E_{ij} = \frac{P_{ij}}{T_{ij}}$</td>
<td>the required energy to transmit one unit of information from node $i$ to node $j$</td>
</tr>
<tr>
<td>$S_i$</td>
<td>the rate amount of information generated at node $i$</td>
</tr>
<tr>
<td>$D_{ik}$</td>
<td>the rate of energy a DoSL device at location $k$ can draw from node $i$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sets</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>the set of nodes</td>
</tr>
<tr>
<td>$V_i$</td>
<td>the set of nodes node $i$ connects to (forward star)</td>
</tr>
<tr>
<td>$K$</td>
<td>the set of potential locations for DoSL devices</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{ij}$</td>
<td>the total legitimate flow from node $i$ to node $j$ during the lifetime of the network</td>
</tr>
<tr>
<td>$g_{ij}$</td>
<td>the total illegitimate flow from node $i$ to node $j$ during the lifetime of the network</td>
</tr>
<tr>
<td>$y_k$</td>
<td>1 if a DoSL device is placed at location $k$; 0 otherwise</td>
</tr>
<tr>
<td>$L_i$</td>
<td>the lifetime of node $i$</td>
</tr>
</tbody>
</table>

Using the above notation, we define the bi-level mixed integer program $[BL − MIP]$ below.

$$Z = Min \ W(f)$$ \hspace{1cm} (3.2a)$$

subject to:
\[ \sum_{k \in K} y_k = Y \] (3.2b)

\[ y_k \in \{0, 1\} \quad \forall k \in K \] (3.2c)

where

\[ W(f) = \text{Max} \quad f_{t,s} \] (3.3a)

subject to:

\[ \sum_{j \in V_i} (f_{ij} + g_{ij} - f_{ji} - g_{ji}) = L_i \left( S_i + \sum_{k \in K} \frac{D_{ik}}{E_{ki}} y_k \right) \quad \forall i \in N \setminus \{t, s\} \quad [\alpha_i] \] (3.3b)

\[ \sum_{j \in V_s} (f_{sj} - f_{js}) = f_{ts} \quad [\alpha_s] \] (3.3c)

\[ \sum_{j \in V_t} (f_{tj} - f_{jt}) = -f_{ts} \quad [\alpha_t] \] (3.3d)

\[ \sum_{j \in V_i} E_{ij} (f_{ij} + g_{ij}) + L_i \sum_{k \in K} D_{ik} y_k \leq B_i \quad \forall i \in N \quad [\beta_i] \] (3.3e)
\[ 0 \leq f_{ij} + g_{ij} \leq L_i R_{ij} \quad \forall i \in N, j \in V_i \quad [\gamma_{ij}] \]  

(3.3f)

The top level objective function (3.2a) strives to minimize the throughput of the network by optimally deploying the DoSL devices. The set of constraints (3.2b) ensures that all available devices are deployed, and the set of constraints (3.2c) is the binary constraint. In the bottom level, our objective function (3.3a) strives to maximize the lifetime throughput by solving a variant of the maximum flow problem, where \( f_{ts} \) is the total flow from the source \( t \) to the sink \( s \). The set of constraints (3.3b) is the flow balance constraints between each pair of nodes to include, on the left-hand side, both legitimate and illegitimate flow, and, on the right-hand side, legitimate information and illegitimate information generated by nodes. Here, the DoSL device acts as a dummy intermediate node, and its illegitimate information generation is equivalent to the ratio of the rate of energy it can draw from a node to the amount of energy required to transmit the illegitimate information. The sets of constraints (3.3c) and (3.3d) are, respectively, the flow balance constraints for the source and sink nodes. The set of constraints (3.3e) is the energy conservation constraint for each node. This set of constraints also ensures that the battery of node \( i \) cannot be drained by a DoSL device unless the device is near the node and also incorporates the flow of illegitimate information. Finally, the set of constraints (3.3f) ensures a maximum flow rate between each pair of nodes across both legitimate and illegitimate information. The first term is the total energy expenditure as a product of the energy transmission and the amount of information being sent. The second term is the total amount of energy the DoSL device is pulling from the battery by feeding the battery illegitimate information. Also, for the sets of constraints (3.3b)–(3.3f), we
provide the dual variables at the end of each constraint to facilitate taking the dual formulation. To linearize the product of $L_i$ and $y_k$ in the set of constraints (3.3b) and (3.3e) we use McCormick constraints [74], introducing the variable $z_{ik}$ and defining the following additional constraints:

$$z_{ik} \geq L_i - (1 - y_k)M_i \quad \forall i \in N, k \in K \quad [\nu_{ik}] \quad (3.4)$$

$$z_{ik} \geq 0 \quad \forall i \in N, k \in K \quad [\mu_{ik}] \quad (3.5)$$

By adding the above constraints to the model and replacing the problem variables in the set of constraints (3.3e) with $\sum_{k \in K} D_{ik}z_{ik}$, we are able to properly solve the problem because our continuous variable $L_i$ is bounded above some $M_i$ since each node has a finite battery. Ordinarily, the variable $z_{ik}$ would need upper bound constraints for each $i \in N, k \in K$. However, because of the direction of inequality of the set of constraints (3.3e), they are unnecessary. Finally, we can reformulate the above as a single level program by taking the dual of the lower level as such:

$$Z = \text{Min} \sum_{i \in N} (B_i \beta_i + M_i \sum_{k \in K} \nu_{ik}) \quad (3.6a)$$

subject to:

$$\alpha_t - \alpha_s \geq 1 \quad (3.6b)$$

$$\sum_{i \in N} (\alpha_i + E_{ij} \beta_i + \sum_{j \in V_i} \gamma_{ij}) \geq 0 \quad \forall j \in V_i \quad (3.6c)$$
\[
\sum_{i \in \mathbb{N} - s, t} S_i \alpha_i - \sum_{i \in \mathbb{N}} \left( \sum_{j \in V_i} R_{ij} \gamma_{ij} - \sum_{k \in K} (\epsilon_{ik} + \nu_{ik}) \right) \geq 0 \quad (3.6d)
\]

\[
\sum_{i \in \mathbb{N}} M_i \sum_{k \in K} (\nu_{ik} + \delta_{ik}) \leq 0 \quad (3.6e)
\]

\[
\sum_{i \in \mathbb{N}} \left( D_{ik} \beta_i + \frac{D_{ik}}{E_{ij}} \alpha_i + \sum_{k \in K} (\delta_{ik} + \epsilon_{ik} + \mu_{ik} + \nu_{ik}) \right) \geq 0 \quad \forall j \in V_i, k \in K \quad (3.6f)
\]

\[
\sum_{i \in \mathbb{N}} (\alpha_i + E_{ij} \beta_i) \geq 0 \quad \forall j \in V_i \quad (3.6g)
\]

\[
\alpha_i \quad \text{urs} \quad \forall i \in \mathbb{N} \quad (3.6h)
\]

\[
\beta_i, \gamma_{ij}, \delta_{ik}, \epsilon_{ik} \geq 0 \quad \forall i \in \mathbb{N}, j \in V_i, k \in K \quad (3.6i)
\]

\[
\nu_{ik}, \mu_{ik} \leq 0 \quad \forall i \in \mathbb{N}, k \in K \quad (3.6j)
\]

Thus, the reformulated model, a single-level integer program, has objective function (3.6a) subject to the constraints (3.2b)-(3.2c) and (3.6b)-(3.6j).
3.5 Solution Methodologies
3.5.1 Branch-and-bound

A bi-level mixed-integer program can be difficult for commercial solvers to handle. In the upper level, we have \(|K|\) total locations for \(Y\) DoSL devices. In the bottom level, we have a max flow problem. In a worst case scenario, we are solving \(|K|\) choose \(Y\) max flow problems, which is computationally complex. However, the branch-and-bound algorithm developed by [80] can be employed without much difficulty, but as we will see in Section 3.7.2, the main purpose of this methodology is for tractability. This algorithm is presented as follows. Given a sub-problem \(s\), set the top level leader problem \(L\) and the bottom level follower problem \(F\). Consider both levels with \(F\)'s objective value removed to be the high-point problem (HPP). Define the vectors of the lower and upper bounds, respectively, to be \(\alpha^{Ls}\) and \(\beta^{Ls}\) and \(\alpha^{Fs}\) and \(\beta^{Fs}\). These are controlled by \(L\) and \(F\) where indicated. Define \(S^{F}_s = \{ q : \alpha^{F_s}_j > 0 \text{ or } \beta^{F_s}_j < UB_j^F \}\) for the sets of indices of restricted (integer) variables in \(s\) for \(F\) and similarly for \(L\) via \(S^{L}_s\). Finally, set \(H^F_S = \{ (\alpha^{F_s}, \beta^{F_s}) \}\) and \(H^L_S = \{ (\alpha^{Ls}, \beta^{Ls}) \}\) as the sets of bounds. Algorithm 3.1 details the algorithm.

3.5.2 Implicit enumeration

While branch-and-bound is a classic algorithm, it does not solve this problem efficiently (as will be seen in Section 3.7.2). To overcome this issue, we employ the implicit enumeration algorithm developed by Scaparra and Church [111]. Here, a search tree is developed and there must be some critical location in an optimal throughput \(f = (f_{ij})_{i \in N, j \in V_i}\) that can be most impacted by the place of a DoSL device. If there is an \(f_{ij}\) in the set of elements comprising the vector \(f\) that cannot be realized, the maximum throughput becomes worsened. Continuing down other \(f_{ij}\) elements would thus cause \(f_{ts}\) to be minimized. Implicit enumeration starts at the root of the search tree
Algorithm 3.1 Branch and Bound

1: **Initialization**: Set $s = 0$, the current optimal solution $z = -\infty$, the parameters of the sets of bounds, and the sets of indices for bounds empty.

2: **Upper bounds and fathoming**: Determine the solution of $HPP$ and the optimal objective value, $z_{HPP}$. If $z_{HPP} \leq z$ or $HPP$ is infeasible, go to Step 7.

3: **Continuous solution**: Solve the relaxed single level program. If infeasible, go to Step 7. Otherwise, fix the current solution then store it.

4: **Branching**: If integrality requirements are satisfied by the current solution, go to Step 5. Otherwise, select any fractional-valued variable and bound it. Update $s = s + 1$ and update the sets of bounds and indices for the bounds. Return to Step 2.

5: **Bi-level feasible solution**: Fix the current solution $y^{(m)} = (y_k^{(m)})_{k \in K}$ and solve $F$ to obtain $f^{(m)} = (f_{ij}^{(m)})_{i \in N, j \in V_i}$ and $f_i^{(m)}$. Compute the optimal $Z$. If $Z > z$, set $z = Z$, updating the new optimal solution.

6: **Integer branching**: If each $UB$ and $LB$ are equal for both $L$ and $F$, go to the next step. Otherwise, select one inequality from either of the sets of bounds and update the bound by decreasing the upper bound or increasing the lower bound (choice dependent on which inequality is chosen and which set it comes from). Update $s = s + 1$ and update the sets of bounds and sets of indices for the bounds. Return to Step 2.

7: **Backtracking**: If no undetermined node exists, go to Step 8. Otherwise, branch to newest undetermined node, update $s = s + 1$, and update the sets of bounds and sets of indices for the bounds.

8: **Termination**: If $z = -\infty$, no feasible solution exists. Otherwise, terminate algorithm with the optimal solution.
and solves the lower level of the bi-level program such that no DoSL devices are placed. An optimal throughput is thus established, providing suitable location for DoSL devices, or branches in the search tree, and at least one of these solutions that places DoSL devices must be realized. Branching continues until all DoSL devices are placed, at which point the branch becomes a leaf, or until feasibility no longer exists (i.e., no more locations to place DoSL devices). The feasible placement of DoSL devices with the largest value of the top-level object function is optimal, which we call $Z$. Algorithm 3.2 details the algorithm.

Algorithm 3.2 Implicit Enumeration

1: **Initialization**: Initialize the node set with the root node associated with no placement of DoSL devices, i.e., $y_k = 0 \forall k \in K$, and set the optimal solution to $z = -\infty$.

2: **Processing**: Select and remove a node $m$ from $M$. If $n$ is the root node or was created from setting any $y_k = 1$, then solve the corresponding lower-level problem for the corresponding vector $y^{(m)} = (y_k^{(m)})_{k \in K}$, providing the vector $f^{(m)} = (f_{ij}^{(m)})_{i \in \mathcal{N}, j \in \mathcal{V}_i}$. Define $O_K^{(m)}$ as the set of suitable locations for all the DoSL devices, and determine then a set of suitable locations for each $y_k^{(m)}$ in $O_K^{(m)}$. If $Z > z$, set $z = Z$, storing the new optimal solution.

3: **Pruning**: If there is no better placement for the DoSL devices, or if $O_K^{(m)}$ is empty, the branch is a leaf, and go to Step 5.

4: **Branching**: Choose a location from the set of possible locations $O_K^{(m)}$ and create two new nodes. One node forces $y_k = 1$, the other forces $y_k = 0$. Add the newly created nodes to the set $M$.

5: **Termination**: If the node set $M$ is empty, terminate the algorithm with the optimal solution; otherwise, return to Step 2.

### 3.6 Illustrative example

We now illustrate how the DoSL attack is performed and how it affects the network using Figure 3.1. Here, circles represent nodes in the network, where $S$ is the source node, $T$ is the
terminal node, and the other numbered nodes are intermediate. The numbers following the hyphens within nodes are the current battery levels. The arrows represent the directions that nodes may communicate along, where the numbers along the edges represent how much information is being transmitted along an arc compared to how much (maximal) information can be transmitted along an arc. The triangle represents a DoSL device, and the larger circle surrounding it represents its radius. Within the image names, the $t = X$ refers to the current time step (taken as arbitrary units), so $t = 10$ in Figure (3.1e) refers to the tenth time step of the network. For illustrative purposes, node $S$ only collects data a rate of 15 units/time and the DoSL device to only send 5 units per arc it establishes with any node.
Figure 3.1: DoSL Attack

In Figure 3.1a, we have a normal network, where information has begun being transferred from the source node to the terminal node for a total of 15 units of information per unit time; we...
emphasize that the flows on the arcs is a per-unit flow and not a total lifetime flow. Each battery currently has 300 units of capacity, and they will drain at a rate of 1 unit of battery per 1 unit of information sent. As an example, after 3 units of time have passed with all nodes sending 5 units of information per unit of time, nodes will have drained $5 \times 3 = 15$ total units from their battery. At $t = 0$ in Figure 3.1b, a single DoSL device has been placed such that it can affect nodes 1 and 2, though the dashed line means it is not yet active.

At $t = 1$ in Figure 3.1c the DoSL device has begun transmitting illegitimate information at the same rate as the legitimate information is sent (5 units per unit time). The intermediate nodes cannot discriminate between legitimate and illegitimate information, and so the intermediate nodes proceed to transmit this “extra” information along with the legitimate information, increasing the amount of information being transferred along each arc. Transmitting more information requires more energy, and so nodes 1 and 2 will have their batteries drained more quickly, which is illustrated in Figure 3.1c as evidenced by the significant drop in battery capacity compared to the others (down to 90 units compared to the others’ 95 units) and the increase in information along the arcs (increased from 5 to 10). We do not consider the power loss of receiving energy, only transmitting energy as in Section 3.3. Flow balance is also not violated here because the DoSL device is seen by the network as another intermediate node that has acquired information from the source node and desires to pass this (illegitimate) information to the sink node. Thus arc $(1, 2)$ is increased because node 1 is transmitting extra information, and arc $(2, T)$ is also increased for the same reason. Node 2 is also receiving even more (illegitimate) information from the DoSL device, but it is transmitting at capacity, so it cannot transmit more than 10 units of information along arc $(2, T)$.
After 5 time units, we see a more significant effect the DoSL device has had on the network in terms of draining the batteries (see Figure 3.1d). Finally, after ten time units have passed, in Figure 3.1e, nodes 1 and 2 are effectively removed from the network as their batteries have drained completely and they can no longer relay information. The network continues passing information from $S$ to $T$ in Figure 3.1f and onward until finally, at $t = 17$, although $S$ still has 50 units of battery and can thus continue to send information, there are no nodes that can transmit information as their batteries have drained fully. Ordinarily, the total amount of information $S$ should have been able to send to $T$ was 300 units. However, with the DoSL device placed and affecting an entire path, it could only send a total of 250 units over the lifetime of the network.

3.7 Computational analysis
3.7.1 Experimental parameters

We consider three network topologies: the Carnegie Melon University (CMU) [25], Massachusetts Institute of Technology (MIT) [78], and a random network (Random) topology. Figure 3.2 shows the MIT and CMU topologies. The random network topology was created by uniformly assigning 100 nodes to random locations over a region of 1 square mile with random edges dispersed such that it was possible to create a path from any node to any other node, and each edge carries a maximum of 50 bits. The set of flows was generated by randomly selecting origin-destination node pairs in the network, and the origin nodes were selected at random without replacement such that each trial had a unique origin node. 10 trials, or 10 random networks, were created this way, and all intermediate nodes for each trial were selected by determining the shortest path for each flow in the set with a predefined total number of flows for the network. The same total number of flows was constant across all 10 trials. Table 3.2 gives the parameter values, which were taken from
Chen, et al. [20], and we note that we set the rate the DoSL devices are able to drain information from the nodes’ batteries equal to the nodes’ transmission powers. This setting is both for tractability purposes as well as allowing our DoSL devices to realistically blend in with other nodes as all nodes, malicious or honest, transmit at the same threshold. All experiments were run on a 64-bit Dell Alienware laptop running Windows 10 with 8 Intel(R) Core(TM) i7-6700HQ CPU processors at 2.60 GHz with 16 GB of RAM. The model and algorithms were encoded using Python 2.7, and the modified single-level problem, each of the sub-problems in the branch-and-bound algorithm, and each branch in the implicit enumeration’s search tree were all solved by Gurobi 7.5.1.

![CMU](image1.png)  ![MIT](image2.png)

Figure 3.2: CMU and MIT networks
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel bandwidth</td>
<td>20 kbps</td>
</tr>
<tr>
<td>DoSL devices</td>
<td>5</td>
</tr>
<tr>
<td>DoSL locations</td>
<td>30</td>
</tr>
<tr>
<td>Battery power</td>
<td>50 units</td>
</tr>
<tr>
<td>Data packet length</td>
<td>10 kb</td>
</tr>
<tr>
<td>Data generation rate</td>
<td>10 kbps</td>
</tr>
<tr>
<td>Data transmission power</td>
<td>1 unit/sec</td>
</tr>
</tbody>
</table>

### 3.7.2 Run-time of solution methodologies

First, we examined the run-time solutions of the two algorithms plus Gurobi on each of the three topologies. To better illustrate the power or lack of power in a particular algorithm, three experiments were made for each topology. The first experiment had 10 nodes and 1 DoSL device. The second experiment had 50 nodes with 5 DoSL devices. The final experiment had 100 nodes and 10 DoSL devices. Table 3.3 shows the results.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>BB (secs)</th>
<th>Gurobi (secs)</th>
<th>IE (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMU-10-1</td>
<td>1711</td>
<td>131</td>
<td>25</td>
</tr>
<tr>
<td>CMU-50-5</td>
<td>11853</td>
<td>373</td>
<td>50</td>
</tr>
<tr>
<td>CMU-100-10</td>
<td>72242</td>
<td>792</td>
<td>182</td>
</tr>
<tr>
<td>MIT-10-1</td>
<td>1970</td>
<td>118</td>
<td>19</td>
</tr>
<tr>
<td>MIT-50-5</td>
<td>13114</td>
<td>380</td>
<td>38</td>
</tr>
<tr>
<td>MIT-100-10</td>
<td>59776</td>
<td>712</td>
<td>191</td>
</tr>
<tr>
<td>Random-10-1</td>
<td>1801</td>
<td>129</td>
<td>33</td>
</tr>
<tr>
<td>Random-50-5</td>
<td>15663</td>
<td>450</td>
<td>64</td>
</tr>
<tr>
<td>Random-100-10</td>
<td>71848</td>
<td>993</td>
<td>200</td>
</tr>
</tbody>
</table>
Each row is named according to the topology used (CMU, MIT, or Random), followed by the number of nodes in the network, followed by the number of DoSL devices placed, all separated by hyphens. So, CMU-50-5 means the CMU topology is used with 50 nodes and 5 DoSL devices. The BB column shows the run-time of the branch-and-bound algorithm, the IE column shows the run-time of the implicit enumeration algorithm, and the Gurobi column shows the run-time of running the modified single-level problem directly on Gurobi. Even on the smallest problems, a traditional branch-and-bound algorithm was near 30 minutes and is not feasible on larger problems. Gurobi performed well on all experiments, showing some stability in the computational time versus problem size growth. The implicit enumeration, however, solved the problem regardless of its size significantly faster than the other two algorithms, never taking longer than 3.5 minutes. In both the branch-and-bound and implicit enumeration schemes, as the problem size grew, there was a significant increase in the computational time, suggesting a possible exponential increase. Gurobi, however, remained mostly linear.

### 3.7.3 Problem sensitivity analysis

#### 3.7.3.1 Effect of the number of DoSL devices

In our first experiment, we examined how increasing the number of DoSL devices affects the lifetime throughput. We start with 1 DoSL device and increase this number by 2 for each experiment until we have examined 15 total DoSL devices. Figure 3.3 shows the results.
The significant differences in the curves between the MIT and CMU topologies, regardless of actual values, gives an idea that topology plays an important role in the amount of throughput received by a sink node. Under the CMU topology, which contains two bipartite regions surrounded by the nodes, many devices must be placed in order to overcome attempts at generating throughput. On other hand, the MIT topology, which is essentially a large cluster (and would be more indicative of sensor networks), there is a largely linear relationship with a quick drop. Under the Random topology, because of the randomness there is no immediate understanding of how topology plays a role. However, what can be said is that except in perhaps more optimal topologies (like the MIT topology), there comes a point where once enough DoSL devices have been placed the throughput drop becomes very significant, as the CMU and Random topologies coincide. Military
ad hoc mesh networks that have overall fewer nodes (because they are smaller installations) may benefit from these non-clustered topologies since there must be a significant number of DoSL devices present to get any meaningful drop in the throughput. Sensor networks, however, are exactly the opposite in size and have hundreds of nodes already in place, and so they may benefit from totally clustered topologies.

3.7.3.2 Effect of the battery lifetime

In our second experiment, we examined how much increasing the battery lifetime mitigates the damage done by DoSL devices. We started with all batteries at normal capacity and solved the problem where no DoSL devices were placed, so the batteries drained normally; this was our base result. We then placed 10 DoSL devices and resolved the problem which gave us our normal result. From here, we decreased each of the batteries’ lifespans by 5% for each run until they were operating at a minimum of 85% capacity, and then we increased the lifespans by 5% until they had a maximum of 150% capacity. Figure 3.4 shows the results.
Across all three topologies, on average, there is no real significance to the battery power itself in maximizing throughput given the mostly linear relationships. We caution and remind that the focus of this chapter is on throughput maximization; other researchers have shown the significance of battery power in maximizing the lifetime of a network [53, 54], and so these results are inherently different from our own. Of course, more throughput is able to be generated if batteries can last longer, but the takeaway here is that, under the idea that a network may be attacked, rallying to improve the battery powers of each node may not be optimal. Instead, as in Section 3.7.3.1, the role of the topology of the nodes may be far more critical.
3.7.3.3 Effect of the number of DoSL devices and DoSL energy

For our final experiment, we considered the trade-off of having more, weaker DoSL devices versus fewer, stronger DoSL devices. We considered two problems: one problem with 10 DoSL devices with the ability to generate 30 units/second of illegitimate information to drain nodes compared against 20 DoSL devices with the ability to generate only 15 units/second and another problem where the numbers of DoSL devices for each case are doubled. Figures 3.5 and 3.6 show the results.

Figure 3.5: Problem 1: 10 DoSL devices at 30 units/sec vs. 20 DoSL devices at 15 units/sec
Having more devices, though weaker, generates a significantly larger impact on the network as a result of more arcs being removed from the network with more nodes being removed; this result is present even though nodes are taking longer to be removed comparatively. Simply doubling the number of devices at half the draining efficacy yields between 25-50% more throughput being unable to be sent. Combining these results with those in Section 3.7.3.1 shows that placing a significant number of weak (and and more importantly, harder to detect) devices is crucial to removing as much potential throughput from the network as possible. In Figure 3.5 with fewer devices present overall, the MIT topology suffered roughly 50% more dropped throughput while the Random and CMU topologies suffered roughly 25%. In Figure 3.6 with more devices overall, the CMU and MIT topologies saw nearly 50% dropped throughput while the Random topology...
saw roughly 25%. In this figure with more devices, only so much more throughput can be dropped since so many devices have already been placed, again following the results in Section 3.7.3.1. As both the CMU and MIT topologies are considerably different, there is some evidence to suggest that, at some point, the topological considerations (resulting in the previous sections) actually stop having as much of an impact in mitigating a DoSL attack once the network is overwhelmed with DoSL devices.

3.8 Conclusions
3.8.1 Discussion

For wireless sensor networks (WSN) and ad hoc mesh networks (AHMN) where battery lifespan is critical to the transmission and reception of information, the impact of denial-of-sleep (DoSL) devices on these networks was previously not understood. While efforts have been made to understand DoSL devices and how to detect them, their impacts on networks has not been explored. This chapter examines the optimal placement of DoSL devices as a means of minimizing network throughput under a lifetime constrained network. A bi-level mixed-integer programming model is developed. To solve the model, an implicit enumeration scheme is implemented and outperforms both Gurobi and a traditional branch-and-bound algorithm significantly.

Several experiments are also considered in the number of DoSL devices to better impact a network, the effects of battery power in an attempt to mitigate such an attack, and whether using fewer stronger or more weaker devices is preferred. We show that topological considerations are very important in mitigating overall attack efficacy by increased numbers of DoSL devices, but there is some point where topological considerations do not have much effect anymore. Overall, however, a few DoSL devices do not significantly damage a network, and there is a point to which
the number of devices begins to significantly affect the network’s performance. On the other hand, initial battery levels do not readily mitigate DoSL attacks regardless of topological considerations as we show a mostly linear relationship in the amount of throughput sent over a network to how much battery power exists for each node. Finally, we see that quantity is in fact better than quality, as an abundance of weaker devices will have a stronger effect than a handful of stronger devices, and this overwhelming nature is what will mitigate topological defensive considerations. This result comes from that fact that more nodes are being overwhelmed, eliminating more arcs more quickly, and thus the network becomes significantly more bottle-necked along remaining arcs in an attempt to transmit as much throughput as possible. A decision maker would need to greatly consider both the placement of their nodes relative to the scope of the network: a WSN that needs to stay for a long time should be fairly clustered with possible communications from node to node as large as possible, while AHMNs that will likely not be set up for too long should employ a more sparse topology.

### 3.8.2 Future Work

One immediate extension of this work is with regards to the usage of mobile DoSL devices. As nodes’ batteries become weakened, it may not be prudent to allow this node to continue to be drained knowing it will be depleted soon, and so a DoSL device could be relocated to a stronger battery source(s). Furthermore, investigation into the preference of this tactic versus waiting for a node to completely drain before relocating the DoSL device would be needed. Further incorporating mobility, inductive charging allows for wireless charging using electromagnetic induction between two objects; a defender could carry a charging device on (for example) a drone vehicle
to patrol the network and continuously recharge batteries, giving both the attacker and defender a
need to identify “critical paths” that are most important to the relaying of information across nodes
from source(s) to sink(s).
CHAPTER IV

NETWORK LAYER: DETECTING VARIETIES OF BOTNETS THROUGH GENERALIZED COMMUNICATION BEHAVIORS

4.1 Introduction

Botnets—programs designed to infect devices and receive commands from a master—are perhaps the most debilitating problems civilian networks face and are a threat to national security in military networks. Dubbed the biggest malware attack in history, WannaCry, a malicious botnet that infected 230,000 computers over 150 countries, occurred as recently in May 2017 [88]. Not only were computers affected, but also medical equipment. CryptoLocker, an attack in 2013, afflicted 500,000 computers over a smaller area, demanding users pay $400 or have users’ files, which became encrypted with the malware, destroyed [58].

Anti-malicious software are, unfortunately, largely reactive solutions. They need to first see an attack before a proper response can be encoded. In order to see attacks, malware and botnets must first be detected. Knowing what kind of malware or botnet one wants to find can help tremendously in detecting them, be they peer-to-peer botnets or e-mail phishing attempts. And generally, these malware have similar behaviors to one another in the same category. Unfortunately, many efforts are made to make malware more stealthy, and thus identifying them becomes more difficult as they are detected as legitimate communications or users. One way around this obstacle is to examine very large amounts of data. Unfortunately, traditional machine learning attempts are easily made
over-fit by very large data and may be incapable of detecting stealthy malware. Further, significant effort must be put into feature engineering for machine learning to work properly, where feature engineering constitutes deriving the correct features, such as an IP address or a communication protocol.

This chapter will examine the detection of botnets without any prior assumptions about the type of botnet being detected in a very large, very real dataset. This chapter will seek to identify peculiar communication patterns by using a deep learning framework, or an artificial neural network (ANN) directly on changes in communication where botnet traffic consists of a very small percentage of the total amount of traffic. We deploy a artificial neural network (ANN), but we pay attention to the data, or feature vectors, fed into the ANN. Because botnets have adapted in how they communicate, we focus more on the actual communications than the means of communication. We then examine a variety of accuracy metrics to see how the detection scheme performs.

4.2 Related Literature

However, cybersecurity, specifically in malware detection, is a boon of research topics that implement machine and deep learning. Wang, et al. [122] used the CNN idea from above by taking pictures of the botnet traffic, an unusual approach that shows both the power of deep learning and the novelty of varying techniques. An original work in using deep learning specifically on malware intrusions [39], which was only in 2010, relied on very basic deep learning frameworks and revealed how this field has evolved since then. Meidan, et al. [76] actually used real botnets on their own equipment with great results, but they only considered two botnets on nine machines. Meanwhile, Chen, et al. [22] demonstrated the weaknesses of ANNs that focus on specific botnets
by intentionally attacking the ANN itself with few anomalies and managed to get more than 90% success rate, demanding a need for a robust ANN framework. Grosse, et al. [40] also pointed out the flaws of malware detectors that rely on focused features by simply having their malware switch continuous variables to binary variables. Strayer, et al. [117] relied on a more generalized approach to feature selection by using flow-based communication features but limited themselves to a small and synthetic dataset. Narang, et al. [85] converted time-domain communication parameters to frequency-domain communication patterns, allowing for more ambiguous behavior but focused only on peer-to-peer bots. Karim, et al. [56] used a simple logistic regression to detect botnet applications but designed their code specific for mobile phones. Dah, et al. [28] overcame the computational complexity of running an ANN on a large dataset with dimension reduction, but they also relied on simpler algorithms like logistic regression that are not necessarily robust against stealthy anomalies as their targets were clearly identifiable. A non-parametric method was used by Saxe and Berlin [110], but could not be realistically used in large datasets of billions of pings of data where they only tested in on a small twenty-thousand. Huang and Stokes [49] developed a robust multi-tiered ANN to detect malware, but they focused on only two types of malware. Linda, et al. [65] perhaps are the closest in developing a truly robust intrusion detection framework, but they focused it entirely on critical infrastructure; noncritical infrastructure can be used to attack critical infrastructure, which they point out as one of their own shortcomings. Pascanu, et al. [97] attempt to get at the most abstract methodologies malware anomalies, and their results prove promising but rely on synthetic malware training.

Aside from deep learning, traditional machine learning techniques have also been used. A support vector machine [77] was applied to flow features, though this dataset was very small.
Without relying on inherent communication based features, purely mathematical ones, similar in idea to our own, were used with mixed success as the authors could only get a true positive rate of 65% overall [37]. Further, a system was proposed [115] using similar features on a very large, real dataset (CTU-13), but results have not been shown. However, it is well known that traditional machine learning does not scale well to larger datasets, and there are problems with over-fitting, whereas deep neural networks perform significantly better [62]. Nevertheless, ensemble methods [60, 11] may help overcome these limitations to some degree, with the authors here reaching an accuracy of 98.2% for mobile threats and better than 90% on general peer-to-peer botnets.

While there is an abundance of research, a significant amount of these traditional efforts rely on smaller or synthetic botnet datasets. Few traditional machine learning algorithms have established themselves on larger, real networks; when attempts have been made to scale, accuracy metrics have led to mixed results [38]. Nevertheless, many have tried with mixed results. When limited to specific botnets, higher accuracy can be achieved [114], but these results were still on synthetic data. Further improvements to traditional machine learning attempts require a significant amount of time on feature selection, and while fine-tuning models is a firm practice, this kind of work can not only increase model complexity but also yield negligible improvements [109]. Shishir, et al. [83] was one of major pioneers in noticing that botnets tend to be precisely structured, giving way to the use of generalized graph-based analytics for detecting them. They were able to detect even stealthy botnets, but they, too, relied on synthetic datasets, and they mention one possible improvement is the use of change in communication behaviors over time, which this chapter focuses on. More recently, and similar to our own methods, Harun, et al. [44] attempted to examine critical communication features in looking at communication frequency, and they used
several traditional machine learning techniques; however, their attempts to isolate limited botnet nodes from normal nodes resulted in being unable to detect some bots.

Stevanovic and Pederson [116] provide a more comprehensive overview of machine learning tactics for botnet and malware detection, while Saad, et al. [106] examine several botnet detection methods and make clear that there are shortcomings that researchers need to address. These shortcomings with the literature are reliance on training and detecting anomalies on synthetic networks, only detecting specific malware detections such as DDoS attacks or phishing attempts or peer-to-peer botnets, or modeling entirely on small networks with very limited data types. Most authors have only considered fine-tuning their architecture to do better than what was previously done, which, as stated, yields negligible improvements, and so there is a reliance on filtering the data and applying what remains directly into the ANN architecture. No author has considered a mathematical treatment of the data through rates of change to see how botnets behave in their communication attempts, which we will accomplish.

4.3 Problem Description

A botnet is a set of nodes where at least one comprises the bot-master and the rest comprise of bots, where the bot-master provides instructions to each of the bots that they must then carry out. Botnets are typically relatively large where their goals may involve distributed denial-of-service attacks or spam. Regardless of the purpose of the botnet, all bots receive commands directly from the bot-master, and this bot-master seeks to infect as many nodes as possible and to have the subordinate bots do the same, maximizing the botnet size (this size is not necessarily limitless). As botnets are increasingly stealthier, detecting them outright is becoming more difficult. Instead,
the network must be examined as a whole and anomalous communication behaviors should be
detected, though this is nearly impossible for a human operator to do as even day-to-day data may
be quite large. For our purposes, we consider just few hours of a large network, of which only a
small portion is botnet traffic, and these botnets come in a variety of types.

The goal is as thus: Detect anomalous communications (botnet flows) with high accuracy
while minimizing Type II errors. Type II errors imply that a botnet communication was flagged
as legitimate communication, and thus botnet is allowed into the network. No a priori assumption
about what kind of botnet or how it behaves in the network should be assumed, as botnets can be
drastically different depending on their nature.

4.4 Data

The dataset used is a real, large dataset known as CTU-13 [38]. This is a botnet traffic dataset
that was captured in the CTU University in the Czech Republic in 2011. In it, three types of traffic
are labeled: botnet, normal, and background traffic. There are 13 scenarios that comprise different
botnet samples, with the different botnets being IRC botnets, Spam-bots, ClickFraud, Port Scan,
distributed denial-of-service (DDoS), FastFlux, HTTP botnets, and then synthetic botnets created
and controlled by the creators of the CTU-13 dataset. Table 4.1 summarizes the data and labels of
each scenario. Each row provides information on a scenario of the dataset, where each scenario
incorporates different botnets from others. The columns for each scenario each provide the overall
duration the data was taken (Duration), the total number of packets transmitted (Packet #), the
number of unique bots (Bot #), the total amount of communication flows sent across the duration
(Total Flows), and then the total number of flows belonging to each class of Botnet, Normal,
Command-and-Control (C&C), and Background. The significance of this dataset not only is that it is large and real, but it captures a variety of botnets (that each may behave differently from one another) and mixes botnet traffic with normal, command-and-control, and background traffic.
Table 4.1: Data and Label Summary of CTU-13

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Duration (hours)</th>
<th>Packet #</th>
<th>Bot #</th>
<th>Total Flows</th>
<th>Botnet Flows</th>
<th>Normal Flows</th>
<th>C&amp;C Flows</th>
<th>Background Flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.15</td>
<td>71971482</td>
<td>1</td>
<td>2824636</td>
<td>39933</td>
<td>30387</td>
<td>1026</td>
<td>2753290</td>
</tr>
<tr>
<td>2</td>
<td>4.21</td>
<td>71851300</td>
<td>1</td>
<td>1808122</td>
<td>18839</td>
<td>9120</td>
<td>2102</td>
<td>1778061</td>
</tr>
<tr>
<td>3</td>
<td>66.85</td>
<td>167730395</td>
<td>1</td>
<td>4710638</td>
<td>26759</td>
<td>116887</td>
<td>63</td>
<td>4566929</td>
</tr>
<tr>
<td>4</td>
<td>4.21</td>
<td>62089135</td>
<td>1</td>
<td>1121076</td>
<td>1719</td>
<td>25268</td>
<td>49</td>
<td>1094040</td>
</tr>
<tr>
<td>5</td>
<td>11.63</td>
<td>4481167</td>
<td>1</td>
<td>129832</td>
<td>695</td>
<td>4679</td>
<td>206</td>
<td>124252</td>
</tr>
<tr>
<td>6</td>
<td>2.18</td>
<td>38764357</td>
<td>1</td>
<td>558919</td>
<td>4431</td>
<td>7494</td>
<td>199</td>
<td>546795</td>
</tr>
<tr>
<td>7</td>
<td>0.38</td>
<td>7467139</td>
<td>1</td>
<td>114077</td>
<td>37</td>
<td>1677</td>
<td>26</td>
<td>112337</td>
</tr>
<tr>
<td>8</td>
<td>19.5</td>
<td>155207799</td>
<td>1</td>
<td>2954230</td>
<td>5052</td>
<td>72822</td>
<td>1074</td>
<td>2875282</td>
</tr>
<tr>
<td>9</td>
<td>5.18</td>
<td>115415321</td>
<td>10</td>
<td>2753884</td>
<td>179880</td>
<td>43340</td>
<td>5099</td>
<td>2525565</td>
</tr>
<tr>
<td>10</td>
<td>4.75</td>
<td>90389782</td>
<td>10</td>
<td>1309791</td>
<td>106315</td>
<td>15847</td>
<td>37</td>
<td>1187592</td>
</tr>
<tr>
<td>11</td>
<td>0.26</td>
<td>6337202</td>
<td>3</td>
<td>107251</td>
<td>8161</td>
<td>2718</td>
<td>3</td>
<td>96369</td>
</tr>
<tr>
<td>12</td>
<td>1.21</td>
<td>13212268</td>
<td>3</td>
<td>325471</td>
<td>2143</td>
<td>7628</td>
<td>25</td>
<td>315675</td>
</tr>
<tr>
<td>13</td>
<td>16.36</td>
<td>50888256</td>
<td>1</td>
<td>1925149</td>
<td>38791</td>
<td>31939</td>
<td>1202</td>
<td>1853217</td>
</tr>
</tbody>
</table>
The total size of the CTU-13 dataset is 696.6 GB, making this problem infeasible on home computers, and thus the need for a high-performance computing (HPC) environment necessary. Different datasets also often have different information being tracked, where some information in one dataset is not present in another. As such, models trained on some datasets cannot work on others because they were built with specific features in mind. For the purposes of this research, however, the dataset has been modified to explore generalized traits of network communications. Information such as packet information and IP addresses are common throughout all, so trying to focus on how communication is done rather than what data is recorded allows for a more robust model to be made.

4.5 Methodology
4.5.1 Features

For each scenario, pings for each source IP address were clustered together into 0.5 minute intervals with accompanying rates of change in the out-degree and in-degree (the out-degree of a given IP address is the number of all IP addresses the given IP address communicates with, while the in-degree is the number of all IP addresses that communicate with the given IP address), change in the out-degree and in-degree weight (the weight corresponds to the total number of packets sent), changes in communication frequency and duration, and the change in betweenness centrality, where the first interval places these values at 0 by default. The betweenness centrality for an IP address is the number of times the IP address falls on a shortest path between two other IP addresses, and it is calculated as $BC(i) = \sum_{u\neq v\neq i \in N} \frac{\sigma_{uv}(i)}{\sigma_{uv}}$, where $u, v, i$ are nodes in the node space $N$, $\sigma_{uv}$ is the total number of shortest paths from node $u$ to node $v$, and $\sigma_{uv}(i)$ is
the total number of shortest paths that include node $i$. These seven calculations are, collectively, our primary features for detecting anomalous communications. As graph-based features, they are not concerned with the specifics on how IP addresses communicate with other IP addresses. As rates of change, they can (theoretically) pinpoint when communications become anomalous. An example of the importance of rates of change is with the ClickFraud botnet. Many advertisements pay revenue based on clicks on their ads; ClickFraud botnets force computers to rapidly click these advertisements, draining revenue from the advertisers. Simply considering the total number of clicks to advertisements can be misleading—the advertisement itself may have embedded itself in a website, and a user may have accidentally clicked this. If many users click the advertisement, this number naturally increases, and it may be the case that many users went to this advertisement over a period of time. However, if there is very little change in the number of clicks to this advertisement, followed by a very large increase, and this is then followed by no real change, it is likely that a ClickFraud botnet has been deployed and is active. Another example comes from DDoS attacks, where the popularity of an issue or a person has caused many people to engage with some websites. Legitimate increased traffic assumes a continuously increasing amount of traffic that occurs gradually as word-of-mouth spreads. A DDoS attack, however, sees a sharp spike in activity (representing a high change) followed by continuous activity (with no real change from the spike). The total amount of activity may be the same, but the gradients that show when the activity occurs may be very different.

To summarize, our model will consider the changes to the out-degree, $\Delta O$, the changes to the in-degree, $\Delta I$, the changes to the out-degree weight, $\Delta OW$, the changes to the in-degree weight, $\Delta IW$, the changes to the communication frequency $\Delta F$, the changes to the communication du-
ration $\Delta D$, and the changes to the betweenness centrality, $\Delta BC$, over 0.5 minute intervals as the feature vectors. The intuition in using these features comes from a demonstrably proven fact about botnets: botnets rely on making rapid, large-scale attempts at communication to other nodes in “bursts” of time, and they make fewer of these communications to smaller degrees as time continues. This result is a highlight of [120]. Thus, tracking what occurs as time changes, or the rates of change, is an intuitive approach to identifying botnet traffic amid normal or background traffic. This problem will be solved using a deep learning framework.

4.5.2 Loss Function

The standard loss function for a binary classification is the binary cross-entropy function, which is

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^{N} y_i \log(p(y_i)) + (1 - y_i) \log(1 - p(y_i))$$

(4.1)

where $y_i$ is the binary label for a given node $i$ (in our case, 1 for botnet, 0 for non-bot) and $p(y_i)$ is the predicted probability that $i$ is a botnet. This loss function examines each point separate from the others, which is typically enough in most binary models. In botnet communication, however, botnets have no desire to communicate with other botnets; the bot-master does this. We can exploit this fact and modify the loss function to improve accuracy by considering two new sets of nodes: the forward and reverse star. The forward star $V_i^+$ of a given node $i$ is the set of nodes $i$ communicates with. Similarly, the reverse star $V_i^-$ of a given node is the set of nodes that communicates with node $i$. 

98
Under the key assumption that a botnet most desires to communicate with non-bots, then the entropy for the forward star for a botnet is

$$H_p(q_{V_i^+}(y_i)) = -\frac{1}{|V^+_i|} \sum_{j=1}^{|V^+_i|} (1 - y_j) \log(1 - p(y_j))$$

(4.2)

The entropy for the reverse star for a botnet is

$$H_p(q_{V_i^-}(y_i)) = -\frac{1}{|V^-_i|} \sum_{j=1}^{|V^-_i|} y_j \log(p(y_j)) + (1 - y_j) \log(1 - p(y_j))$$

(4.3)

which is the same as Equation 4.1 but replacing the appropriate sizes. This is because anything can communicate with a botnet regardless of desire (and thus accounts for the situation where botnets, despite not wanting to, may communicate with other botnets). With regards to non-bots, their forward star is the same as Equation 4.3, as is their reverse star (anything desires communicate with non-bots). We can define a new loss function as

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^{N} y_i \log(p(y_i))[B_i] + (1 - y_i) \log(1 - p(y_i))[N_i]$$

(4.4)

where $B_i = H_p(q_{V_{i}^+}(y_i)) + H_p(q_{V_{i}^-}(y_i))$ and $N_i = H_p(q_{V_{i}^+}(1 - y_i)) + H_p(q_{V_{i}^-}(1 - y_i))$. Thus, we fully account for the communication behaviors of botnet and non-bot traffic, as desired.

4.5.3 Neural Network Architecture

The deep learning framework used here is a sequential model that consists of three sets of three dense layers, where dropout occurs between the second and third layers and between any two sets (thus 5 instances of dropout). Thus, our network is 9 layers in depth. The first dense layer relies
on a rectified linear unit (RELU) activation, one of the most popular activation methods, and has seven neurons, one for each feature. The second layer expands to fourteen neurons, one per feature per classification label, and uses a sigmoid activation. The final layer considers only two neurons, one for each classification, also using sigmoid activation. The model uses the Adam optimizer for computational efficiency; this optimizer requires little fine-tuning and is popular for benchmarking deep learning frameworks. This optimizer had a learning rate of 0.001, no fuzz factor, no decay rate, $\beta_1 = 0.9$, and $\beta_2 = 0.999$, all of which are the default settings for the Adam optimizer in Keras.

4.6 Experiments and Results
4.6.1 Experimental Setup

Because of the size of the dataset, an HPC environment is necessary, and the Onyx HPC environment was used. Table 4.2 summarizes the Onyx HPC specs.

<table>
<thead>
<tr>
<th>Specification Type</th>
<th>Onyx HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Type</td>
<td>Intel Xeon E5-2699v4 Broadwell</td>
</tr>
<tr>
<td>Number of Cores</td>
<td>44</td>
</tr>
<tr>
<td>Processing Speed (GHz)</td>
<td>2.8</td>
</tr>
<tr>
<td>Dedicated Memory/Accessible (GB)</td>
<td>128/122</td>
</tr>
<tr>
<td>Environment</td>
<td>Cray Linux</td>
</tr>
<tr>
<td>Computational Time (hours)</td>
<td>120</td>
</tr>
</tbody>
</table>

Training was performed on 90% of the dataset, with validation and testing on 10% of the data. The splitting was performed using 10-fold cross validation, a common approach for machine learn-
ing models. The way 10-fold cross-validation works is that first the dataset is shuffled randomly and then split into 10 groups. For each unique group, one group is treated as a test dataset with the remaining groups as training dataset. Our neural network is fit on the training set (9 groups) and evaluated on the test set (the remaining 1 group), generating a sub-model. The evaluation score is retained, and the sub-model is discarded, with the process repeating for the remaining sub-models. Once all sub-models are evaluated, the skill of the model is summarized via evaluation scores; specifically, the mean of the 10 results is provided. This procedure helps estimate the model’s ability to perform on unseen data and effectively gives us our desired 90%/10% split. The model was coded using Python 3.6, TensorFlow 1.10, and Keras 2.2.4. We use the following accuracy rates:

- $FPR = \frac{FP}{TN+FP}$
- $TPR = \frac{TP}{TP+FN}$
- $TNR = \frac{TN}{TN+FP}$
- $FNR = \frac{FN}{TP+FN}$
- $Precision = \frac{TP}{TP+FP}$
- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- $ER = \frac{FN+FP}{TP+TN+FP+FN}$
- $F_1 = 2 \cdot \frac{Precision \cdot TPR}{Precision + TPR}$

Here, $F$ stands for false, $T$ stands for true, $P$ stands for positive, $N$ stands for negative, $R$ stands for rate, $ER$ is the error rate, and $F_1$ is the traditional $F$-score for a test’s accuracy. Thus we define, in order, a false positive rate, a true positive rate, a true negative rate, a false negative rate, the precision, the accuracy, the error rate, and the $F$-score. Defining the null hypothesis $H_0$ that an IP
address is a botnet, we define the true positive when a botnet is correctly identified as a botnet; we define the true negative as when normal traffic is detected to be normal; we define a false positive when normal traffic is detected as botnet; and we define a false negative when a botnet is detected as normal traffic.

### 4.6.2 Results and Analyses

Table 4.3 shows the results for each scenario.

<table>
<thead>
<tr>
<th>ID</th>
<th>FPR</th>
<th>TPR</th>
<th>TNR</th>
<th>FNR</th>
<th>Prec</th>
<th>Acc</th>
<th>ER</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.332</td>
<td>97.591</td>
<td>99.668</td>
<td>2.409</td>
<td>99.661</td>
<td>98.630</td>
<td>1.370</td>
<td>98.615</td>
</tr>
<tr>
<td>2</td>
<td>0.434</td>
<td>97.733</td>
<td>99.566</td>
<td>2.267</td>
<td>99.558</td>
<td>98.650</td>
<td>1.350</td>
<td>98.637</td>
</tr>
<tr>
<td>3</td>
<td>0.172</td>
<td>98.217</td>
<td>99.828</td>
<td>1.783</td>
<td>99.825</td>
<td>99.023</td>
<td>0.977</td>
<td>99.015</td>
</tr>
<tr>
<td>4</td>
<td><strong>0.069</strong></td>
<td>97.382</td>
<td><strong>99.931</strong></td>
<td>2.618</td>
<td><strong>99.929</strong></td>
<td>98.657</td>
<td>1.343</td>
<td>98.639</td>
</tr>
<tr>
<td>5</td>
<td>0.430</td>
<td>95.971</td>
<td>99.570</td>
<td>4.029</td>
<td>99.554</td>
<td>97.771</td>
<td>2.229</td>
<td>97.730</td>
</tr>
<tr>
<td>6</td>
<td>0.179</td>
<td>98.127</td>
<td>99.821</td>
<td>1.873</td>
<td>99.818</td>
<td>98.974</td>
<td>1.026</td>
<td>98.965</td>
</tr>
<tr>
<td>7</td>
<td>0.703</td>
<td>94.595</td>
<td>99.297</td>
<td>5.405</td>
<td>99.262</td>
<td>96.947</td>
<td>3.054</td>
<td>96.872</td>
</tr>
<tr>
<td>8</td>
<td><strong>0.119</strong></td>
<td>98.159</td>
<td><strong>99.881</strong></td>
<td>1.841</td>
<td><strong>99.879</strong></td>
<td>99.020</td>
<td>0.980</td>
<td>99.012</td>
</tr>
<tr>
<td>9</td>
<td>0.167</td>
<td><strong>99.317</strong></td>
<td>99.833</td>
<td><strong>0.683</strong></td>
<td>99.833</td>
<td><strong>99.575</strong></td>
<td><strong>0.425</strong></td>
<td><strong>99.574</strong></td>
</tr>
<tr>
<td>10</td>
<td>0.185</td>
<td><strong>99.058</strong></td>
<td>99.815</td>
<td><strong>0.942</strong></td>
<td>99.814</td>
<td><strong>99.437</strong></td>
<td><strong>0.563</strong></td>
<td><strong>99.435</strong></td>
</tr>
<tr>
<td>11</td>
<td>0.890</td>
<td>97.280</td>
<td>99.110</td>
<td>2.720</td>
<td>99.093</td>
<td>98.195</td>
<td>1.805</td>
<td>98.178</td>
</tr>
<tr>
<td>12</td>
<td>0.148</td>
<td>96.920</td>
<td>99.852</td>
<td>3.080</td>
<td>99.848</td>
<td>98.386</td>
<td>1.614</td>
<td>98.362</td>
</tr>
<tr>
<td>13</td>
<td><strong>0.034</strong></td>
<td><strong>98.613</strong></td>
<td><strong>99.966</strong></td>
<td><strong>1.387</strong></td>
<td><strong>99.965</strong></td>
<td><strong>99.289</strong></td>
<td><strong>0.712</strong></td>
<td><strong>99.285</strong></td>
</tr>
<tr>
<td>Avg</td>
<td>0.297</td>
<td>97.613</td>
<td>99.703</td>
<td>2.387</td>
<td>99.695</td>
<td>98.658</td>
<td>1.342</td>
<td>98.640</td>
</tr>
</tbody>
</table>

After 1,200 epochs, we have significant results. In bold are those top three scenarios with either the lowest false positive rate (FPR; also called fall-out) and the highest true negative rate (TNR; also called specificity) or highest true positive rate (TPR; also called recall) and lowest false negative rate (FNR; also called miss rate). Scenarios 4, 8, and 13 gave us the smallest FPR.
values and the highest precision (Prec) values while Scenarios 9, 10, and 13 gave us the highest TPR values, as well as the smallest error rate (ER) values and largest accuracy (Acc) values and $F_1$ scores. Unsurprising are Scenarios 7 and 11 with the worst FPR values; these scenarios had the least duration of information available. Most surprising is the consistent results for Scenario 13, which had the second largest duration of information but only 1 bot, showcasing the power of an ANN. Scenarios 9 and 10 have the most botnets, and as these scenarios provided some of the highest TPR values, a more heavily compromised network can allow for better detection of the anomalies, which is important for cleaning a system.

The Avg row takes the average of all the values in a column to given an overall picture of this model’s performance on the CTU-13 dataset as a whole; here, our FPR is 0.297%, much better than our desired goal of 5%, while the TPR is 97.613%, much better than the desired 95%. As such, our model is very good in predicting general botnet communication behaviors. In general, however, in order to improve overall efficacy, a more robust feature selection would be necessary. In general, botnets have particular, inherent structures based on their goals. Structures can be represented via sub-graphs, and so sub-graph analysis, such as specific sub-graph shapes and features such as diameter would be beneficial to improving the efficacy of this model.

To better see how our model performed on each scenario, we include confusion matrices for each scenario. We note as a reminder that we tested on 5% of the data, thus indicating our significantly lower counts than the actual data amount in the CTU-13 dataset from Table 4.1.
### Table 4.4: Confusion Matrices for CTU-13

<table>
<thead>
<tr>
<th>Scenario</th>
<th>A:B</th>
<th>A:N</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P:B</td>
<td>P:N</td>
</tr>
<tr>
<td>1</td>
<td>1949</td>
<td>48</td>
</tr>
<tr>
<td>2</td>
<td>921</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>1314</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>84</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>33</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>217</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>248</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>8933</td>
<td>61</td>
</tr>
<tr>
<td>10</td>
<td>5266</td>
<td>50</td>
</tr>
<tr>
<td>11</td>
<td>397</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>104</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>1913</td>
<td>27</td>
</tr>
<tr>
<td>Avg</td>
<td>21381</td>
<td>257</td>
</tr>
</tbody>
</table>

For each table, the number in the top-left corner represents the scenario number. P:B refers to the predicted botnet flows, P:N refers to the predicted non-botnet flows, A:B refers to the actual botnet flows, and A:N refers to the actual non-botnet flows. The immediate result from all of these tables is a lack of consistency in narrowing down total botnets; although the proportions are similar (thus explaining the consistent rate percentages above), the model should instead be able to more consistently constrain the actual numbers. In Scenario 9, for example, although 61 botnet flows incorrectly identified as normal traffic amid 8933 correctly identified botnet flows seems reasonable, this is still a very high number in its own right, meanwhile other scenarios have fewer...
overall digits. Furthermore, in Scenario 7, which only had 2 total botnet flows, and none were actually mislabeled. Ideally, there would be a higher variance in the FPR and FNR values which would make these matrices show values more consistent in size (where appropriate; 2 botnet flows compared to 8994 is not a reasonable comparison). However, in order to better constrain the actual numbers, a stronger understanding of important features would be necessary. It may be the case that some scenarios reacted very strongly to the provided features (such as Scenario 5 with its low false positive count of 1) and others only did so well as to be “consistently good.” While additional features might not improve Scenario 5’s 1 mislabeled botnet flow, they may significantly improve Scenario 9’s 61 mislabeled botnet flows.

Another possible improvement of the model is more training, but for our model’s features this approach would not work. Figure 4.1 shows the average, accuracy, precision, and error rates over time across all of CTU-13, and we include more epochs than our established 1,200 to show that our model began to suffer from over-fitting when trained any longer. Figure 4.2 shows the loss across CTU-13, though we only provide instances of every 100 epochs to better illustrate the end trend. For the same reason, we also omit the first 300 epochs in Figure 4.2.
Figure 4.1: Trends of Accuracy Metrics Across all of CTU-13
In both figures, trends after 1200 epochs showed an increase or decrease where neither were desired, indicating the model was over-fitting. The precision actually increased briefly, but this was just noise. Prior to 1200 epochs, however, trends were where they were desired, and thus our reasoning for training for only the indicated amount.

4.6.3 Comparison to Other Work

Unfortunately, to our knowledge, few other works have considered the CTU-13 dataset from a deep learning perspective and in classifying the actual flows. The one we found had promising results [71]. We compare our methodology (labeled DL-9) to theirs. However, they considered their framework on two datasets; one similar to ours that encompasses all of CTU-13, and one in
the partition as was originally recommended [38]. They also only report the FPR, TPR, Precision, and $F_1$ scores. As such, we compare ours to their results that used the entire dataset and report the same scores, and we label their methodology SAD. To further illustrate how powerful our method is, we also compare our method to a traditional support vector machine (SVM) using a radial basis function (RBF) kernel, which we label SVM. We also consider a smaller, 3-layer ANN that has 3 dense layers with dropout between the second and third layers, following the same setup as in Section 4.5.3, and we label this DL-3. Finally, we use our full 9-layer ANN on the raw data, where we omit the two non-numeric features as they are both categorical and limited in different types yet span a very large dataset. We label this DL-Raw. Table 4.5 shows the results.

<table>
<thead>
<tr>
<th></th>
<th>FPR</th>
<th>TPR</th>
<th>Precision</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAD</td>
<td>18.74</td>
<td>99.34</td>
<td>81.26</td>
<td>89.40</td>
</tr>
<tr>
<td>DL-3</td>
<td>23.52</td>
<td>58.77</td>
<td>71.42</td>
<td>64.48</td>
</tr>
<tr>
<td>DL-9</td>
<td>0.297</td>
<td>97.61</td>
<td>99.70</td>
<td>98.64</td>
</tr>
<tr>
<td>DL-Raw</td>
<td>21.21</td>
<td>88.98</td>
<td>80.75</td>
<td>84.67</td>
</tr>
<tr>
<td>SVM</td>
<td>11.40</td>
<td>88.65</td>
<td>88.61</td>
<td>88.63</td>
</tr>
</tbody>
</table>

Across all metrics, applying our ANN performs significantly better than the other methodologies except in the TPR reported from SAD. However, they performed training on only 80% of the data, where ours was done with 90%. While 80%/20% training/testing splits are commonly used, it is becoming more recommended that with very large data, such as CTU-13, the training portion is larger; however, there is no optimal or rigorous split, although some efforts have been made to determine the most appropriate split for a given data size [33]. We believe the 1.7% difference TPR
can be explained by the 10% difference in training size. What is significant, however, is that our $F_1$ is larger, indicating our model can be more accurately applied to other data, though we do note that this value is more influenced by the larger Precision value which is in turn influenced by the significantly lower FPR value. While accuracy, precision, and error rate are all useful for insights, a model can perform well according to these metrics by correctly evaluating all the non-bot traffic because of the magnitude of the non-bot traffic relative to the botnet traffic. Incorrectly identifying botnet traffic would have minimal impact on accuracy and error rate. On the other hand, precision is influenced entirely by false positives and true positives, where false negatives are significant to the stability or instability of the network; misclassifying normal traffic as botnet is not as severe as misclassifying botnet traffic as normal. The $F_1$ measure considers the actual classification of botnets, or the TPR, along with the precision. The higher the $F_1$, the better overall the detection methodology.

Compared to the models, our larger ANN surpasses the 3-layer one in all aspects, though a traditional SVM also surpassed the smaller network, indicating strong evidence that both the architecture and the size of the network are important for improving overall accuracy. Further, the feature choice is also important, as our model performed better with our custom features than it did with raw data. However, the architecture itself is still valuable when comparing DL-Raw to a smaller ANN, DL-3, or the SVM.

4.7 Conclusions
4.7.1 Discussion

Reliably detecting botnets and other anomalous communication is a considerably important topic that has gained popularity relatively recently with the improvement of artificial neural net-
works (ANNs). However, most efforts, while significantly powerful, suffer in three major areas: a priori assumption on the botnet traffic, relying on small or virtual or synthetic datasets, and examining only specific botnet structures. We proposed a more general botnet detection methodology that sought only communication-based behaviors as features with only one assumption: a botnet should behave differently from normal traffic.

To this end, we deployed a 9-layer sequential ANN, meaning each layer follows directly from a previous layer, that was able to detect botnets across a very large, real dataset, the CTU-13 dataset, with a false positive rate of 0.297% and false negative rate of 2.387%; our true positive rate was 97.613%. Individually, we found that multiple bots had a higher likelihood of being found. Botnets are also easily detectable if the number of botnet communications has a significantly large amount compared to normal communications (ignoring command and control flows and background flows). However, even if botnets attempt to blend in with the surrounding communications through deception or stealth, they exhibit some underlying structures that makes them detectable when employing graph-based analytics. Such underlying structures are also the limitations of this work; we do not, for example, consider sub-graph properties, such as the general shape of a sub-graph centered around a given node. Further, we are only looking at the communication behaviors over intervals of time; consequently, our time intervals may be capturing too little information to accurately determine botnet behavior.

4.7.2 Future Work

Because we were detecting a variety of botnets without much assumption on how they actually behave, our methodology could be modified to better exploit specific behaviors that other
researchers have used to improve accuracy. Further, there are some communication features we have not considered. Botnets, when not trying to infect other nodes, absolutely only communicate with other botnets or the bot-master to give or receive instructions; they do not, for example, distribute malware to an already infected computer. So identifying times of infection and links that caused the infection as a feature by looking into the reverse star (a list of all nodes that communicated to a given node) of a given address can provide insights to improving the model. Similarly, a more thorough examination of the forward star (a list of all nodes that a given node communicates with) to see what sort of communication was made can further improve the model. Furthermore, as established, botnets are likely to “stand out from the crowd,” and so examining sub-graph features and isolating sub-graphs and examining patterns of these isolated sub-graphs could yield significant improvement because the underlying botnet structure is being captured.
CHAPTER V
CONCLUSIONS

5.1 Discussion

Wireless communication networks are complex structures, but they have become so essential to everyone’s lives that they must be maintained and kept safe as they contain both personal important information such as passwords to our bank and medical information as well as classified military information. However, it is their complex nature that makes them easy targets from a variety of sources as well as making them difficult to defend. This dissertation proposes the idea that the robustness of networks to survive attacks from a variety of attacks is just as important as research into the defense of these attacks. Networks come in several abstract layers, and this dissertation will examine three of them: the physical layer, the data-link layer, and the network layer.

At the physical layer are jamming attacks that can disrupt a network’s nodes completely, rendering them useless. We propose the optimal deployment of a new network’s access points that can maintain connectivity in the advent of an also optimal jamming attack; networks robust against these attacks both maintain connectivity and, as our experiments showed, can reveal insights into jamming efficacy on their network. However, several networks are already installed, and so we also consider both how robust common topologies are against optimal jamming attacks as well as the effects of optimally placing new nodes onto the preexisting topology and examining this new network’s robustness.
At the data-link layer, where efficient communication transmission is necessary, jamming attacks may be quickly identified, and so the denial-of-sleep (DoSL) attack comes into play. These attacks prey on the limited batteries of the network’s nodes, and they send just enough data to not warrant immediate suspicion by operators who expect their networks to continuously send information. Here, we examine the optimal deployment of these DoSL devices to examine how a network can handle such an attack by considering the network’s throughput compared against a normal throughput value. A robust network will still be able to maintain the transmission of a significant amount of throughput over the lifetime of its network, where the lifetime of the network is defined when a source node can no longer reach a sink node because all useful intermediate nodes are inactive. We have presented a model that provides insights to this problem and these kinds of attacks.

At the network layer, physical devices are no longer essential, and now the attackers are trying to gather information rather than stop their spread. Malware and botnets infect vulnerable targets and mine information or pass along the infection to more and more computers until either relevant infrastructure is infected (to gather classified data from a central hub, as an example), a significant amount of infrastructure is infected, or until the right information has been gathered, such as important passwords, documents, or bank or medical information. As these malicious users rely more on stealthy acquisition methods, detecting them becomes increasingly difficult. As such, we have presented a means of detecting these malicious users via generalized parameters, having providing a model that will do so effectively.

Ultimately, the importance of defending a network cannot be understated. This dissertation examines not just one major area but three and unites them into a multifarious robustness exami-
nation of networks in general. We examined the jamming attack and will examine the DoSL attack and botnet detection. This dissertation’s results provide valuable insights into keep networks safe at several, different levels.

5.2 Future Work

There are many directions that this work can aspire to in keeping the theme of a multi-layered robustness analysis of a wireless network. One major direction is a thorough analysis of a “throughput attack.” Here, an attacker places both jammers and DoSL devices on a given network, relocating them as needed, in order to negatively impact the amount of information the network can actually transmit. Within the legitimate information, the attacker may also send malicious communications. The defender needs both a robust network in order to mitigate the immediate attack as well as a neural network in place to detect the malicious information. Given a network topology where a source node seeks to transmit information to a sink node, where each node consumes power in order to generate and transmit information to the sink, and where some devices are placed to eliminate nodes or transmit malicious communication, a defender or network operator could determine how robust against such a multi-layered attack the network is by determining how much legitimate information has actually reached the sink compared to how much could have reached the sink without any attack.

Further, examining attacks on the remaining four layers is also a consideration. A type of attack on the transport layer could push the network bandwidth to its limits. An attack on the session layer could prevent the administrator from performing switch management functions on Telnet server software. An attack on the presentation layer could force afflicted systems to continuously restart,
halting any work being done. Finally, an attack on the application layer could force all resources to be consumed, such as processing speed and CPU load, causing a major shut-down of individual nodes in the network. Examining each of these attacks together would provide valuable insights in determining just how at-risk even secure wireless networks really are; while it is questionable as to the legitimacy of an effective seven-layered attack, understanding the role of critical infrastructure and noting inherent vulnerabilities could provide defenders valuable insights to safeguarding the most important parts of any network from any attack.
BIBLIOGRAPHY


[42] Y. Gu, Y. Ji, and B. Zhao, “Maximize lifetime of heterogeneous wireless sensor networks with joint coverage and connectivity requirement,” 2009 International Conference on Scal-
able Computing and Communications; Eighth International Conference on Embedded Comput-


