THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

SCHOOL OF SCIENCE, ENGINEERING, AND TECHNOLOGY

Effective Network Configuration Strategies for the Mitigation of Denial-of-Service Attacks

MATTHEW POST
SPRING 2022

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Computer Science
with honors in Computer Science

Reviewed and approved* by the following:

Jeremy Blum
Associate Professor of Computer Science, School of Science, Engineering, and Technology
Thesis Supervisor

Linda Null
Associate Professor of Computer Science, School of Science, Engineering, and Technology
Associate Program Chair, Computer Science & Mathematics
Honors Advisor and Second Reader

* Electronic approvals are on file.

ABSTRACT

Denial-of-service attacks are a popular mode of attack for malicious users within computer networks. No practical system can create a perfect defense protocol or strategy against this advanced persistent threat. Previous research has highlighted Colonel Blotto models that represent some kind of network attack in specific network environments.

We propose a Colonel Blotto model that simulates general network structures that can be commonly applied or seen in real-life networks. We use a regret minimization algorithm to develop attacker and defender strategies under many different parameters in the system. These Nash Equilibrium strategies we find can be used to determine effective approaches against a denial-of-service attack. We explore variations in these strategies under different conditions and report the trained agents' rationale for choosing their strategies. Furthermore, we study the parameters of the game and how a network administrator might best leverage them. Our findings show approaches that a defender should take when configuring a network to mitigate the damages caused by the attack. We found unexpected strategies in some cases and gave insights into how one might design a network when considering parameters outside the scope of this model.

TABLE OF CONTENTS

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

ACKNOWLEDGEMENTS

I would like to truly express my deepest appreciation to my thesis supervisor Dr. Jeremy Blum. This thesis would not have been possible without his help and dedication to this project. As a thesis supervisor, he has helped me tremendously to understand concepts, provided insight, and gave me the confidence to finish a project I thought to be impossible. Often, Dr. Blum went far out of his way to provide extra support and guidance to me for this project, for schoolwork, and even for career advice.

I would like to extend my deepest gratitude to my family and friends that have supported me through my college and thesis journey. Without them, I would have never even had the opportunity to be a part of this.

Furthermore, I would like to thank Dr. Linda Null for wearing a lot of hats in my college career. She was the first professor I met coming to this college and without hesitation, she agreed to be the second reader for my thesis, my academic advisor, my honors advisor, and a mentor throughout college.

I am also grateful for Dr. David Witwer for being an amazing mentor and being the catalyst for my passion as a volunteer firefighter.

Lastly, I would like to thank Dr. Niki Pissinou from Florida State University for her mentorship in the summer of 2021's research experience for undergraduates, as well as the National Science Foundation for funding towards research efforts, some of which ended up being a part of this thesis. Grant number: NSF 1851890

Chapter 1 Introduction

Network security

Computer network systems are essential in today's technologically advanced society. Network systems have a multitude of uses including civil and corporate banking, military operations, general communications, smart devices, and many more. Without computer network systems, society would look nothing as it does today. With these networks practically running modern-day society, it is critical that these systems are protected and run as intended.

There are many different reasons why a network would not be operating correctly. Some factors that would prevent a network from working are relatively uncontrollable, such as natural disasters or hardware failure. Computer networks may also be affected by factors that can be more controlled. Commonly a network may be compromised by an attacker. An attacker is a person who intentionally tries to shut down a network with a myriad of different attacks. Different strategies usually benefit the attacker in different ways, but typically have the same or similar effect on the defender of the network. An example of this would be if the defender's network was typically separated into two disjoint networks. Having an effective strategy for situations like this is vital for the health of the world's computer network systems.

Denial-of-Service attacks in the real world

Denial-of-service (DoS) attacks are persistent threats to network security. Such attacks have been around since the mid-1990s and seem to be threats that will be around for the foreseeable future. Denial-of-service attacks generalize a large category of attacks, many of which are advanced persistent threats (APT). Distributed denial-of-service (DDoS) attacks may be the most popular kind of denial-of-service attack, but DDoS attacks still hold a lot of generalization. This quote by Kaspersky Labs alludes to the relevance of DDoS attacks nowadays.

> *"When compared to Q3 2020, the total number of Distributed Denial of Service (DDoS) attacks increased by nearly 24%, while the total number of smart attacks (advanced DDoS attacks that are often targeted) increased by 31% when compared to the same period last year. Some of the most notable targets were tools to fight the pandemic, government organizations, game developers, and well-known cybersecurity publications." – Kaspersky Labs [17]*

The model in this paper characterizes, but does not limit itself, to a denial-of-service attack that overwhelms a network by jamming nodes in a network with an influx of network traffic, represented as a graph. The utility of a network is measured in terms of the connectivity present, even in the face of an attack. The attack is modeled based on a Colonel Blotto game, played between a defender and an attacker of the network. Each player develops strategies to provide the best possible outcome, with goals of high network connectivity for the defender and low network connectivity for the attacker. We posit that the strategies developed for the Colonel

Blotto game can be abstracted into higher-level strategies that could be used to defend real networks against these kinds of attacks.

The derived strategies and subsequently the high-level abstractions of such are not reactive or produce a result that immediately strengthens a network's chance against an attack. Rather, the results produce configuration methodologies and rules that one can follow to ensure the best possible worst-case performance of a network. In other words, these configurations guarantee some level of operation in the event of catastrophic failure. Pairing these kinds of strategies or network configurations with other kinds of network security measures results in an overall more robust network.

Chapter 2 Literature Review

Colonel Blotto

Overview

Colonel Blotto is a classic example in game theory, where the Colonel, Colonel Blotto, must defend against a rational enemy. The rules of Blotto are relatively straightforward. There is a theoretical map with battlefields on it in which the two players are allowed to allocate resources. Both players allocate their respective resources on these battlefields simultaneously with some specified number of resources each. Each battlefield may be worth a different value. After the players place their resources, the game is over and evaluated. Whoever has more resources on a specific battlefield will win the battlefield. A player's score is the sum of the values of the battlefields they have won, and the player with the highest score wins.

Blotto has been used to model real-world scenarios, including the electoral college system used in U.S. presidential elections. In this instance of the game, the players are the politicians, and the battlefields are the states or territories [1, 2, 4]. The value of the battlefield is the number of electors it is assigned. To win a state, a politician must allocate more resources to it than their opponent.

Strategies in Games of Imperfect Information

Blotto is a finite, two-player, zero-sum game of imperfect information. A finite game is one in which a player chooses from a finite set of possible actions [1, 6]. In a two-player, zero-sum game, one player's gain is the other player's loss. An imperfect information game is one in which players have some hidden knowledge. These qualities are important because a well-known result from game theory is that all two-player, finite, zero-sum games are guaranteed to have at least one Nash equilibrium strategy [18]. A Nash equilibrium strategy provides a guarantee of the worst outcome a player may experience.

In an imperfect information game, each player has their respective public and private information. In this game model, the total of every player's resources is public information; this means that every player knows this fact and likely will use it to determine their next action. Private information, in this simple model, would be each player's strategy [2, 4, 5]. Each player does not know what the other will do.

The outcomes of two-player, imperfect games can be represented with a payoff matrix. A payoff matrix is a visual representation of all of the games' possible outcomes. The games' outcomes in each block of the matrix are derived from each player's utility function [3]. A utility function is a mathematical function that represents each agent's score, or utility, given certain actions from each player. A well-known, two-player, imperfect-information game is the Prisoners' Dilemma, whose payoff matrix is shown in Table 1. This figure shows the first prisoner's actions in the first row and the second prisoner's actions in the first column. The cells that intersect represent the utility of each player where the first entry in the tuple is the first prisoner's utility and the second entry is the second prisoner's utility. For example, when both prisoners do not take the plea deal, both prisoners have a utility of -1.

Table 1: Prisoner's dilemma payoff matrix

	No Plea Deal	Plea Deal
No Plea Deal	(-1, -1)	(0, -4)
Plea Deal	(-4,0)	(-3,-3)

Unlike the Prisoners' Dilemma, Blotto can be modeled as a zero-sum game. A zero-sum game is a game in which the payoffs of each other players cancel each other out [3]. In Colonel Blotto, a game win could be assigned 1, a tie 0, and a loss -1. Thus, the sum of both players' utilities is always 0. The Prisoners' Dilemma is not a zero-sum game. This is evident when adding both of the prisoners' utility from any cell in Table 1. For example, when both prisoners do not take the plea deal the combined utility is -2, not 0.

Cooperative vs. non-cooperative games

Unlike in the Prisoners' Dilemma, there is no benefit in cooperation to players in a 2-player, zero-sum game, since one player's gain is another player's loss. In non-cooperative games, all agents utilize strategies that are completely self-sufficient and do not rely on the assistance of other players. On the other hand, in cooperative games, agents may strategize together to better optimize their respective utility.

A game may encourage cooperation by changing the rules. An example of this would be by changing the number of players or making it a non-constant sum game. All cooperative games may have non-cooperative counterparts, although all non-cooperative games do not necessarily have cooperative counterparts [5, 6]. This is interesting when thinking about general-purpose game models or models simulated after real-life scenarios. Specifically, when thinking about a non-cooperative game becoming cooperative. Although this is not always possible, there may be an optimal strategy with other agents.

Variants of Blotto have cooperative strategies. For example, when modeling elections, cooperative strategies become possible when there is a third party. A third party could serve as a spoiler in an election, with the promise of some reward should they help one of the parties win the election [6, 13]. The converse seems to be more intuitive and well explored. Even a game model based on something cooperative such as health care is still very well known on the non-cooperative side. For example. the healthcare industry works together to get the most optimal product [3, 6]. This is based on the general idea of competition in capitalism where competition drives better products or results.

Nash Equilibrium

The Nash Equilibrium, often abbreviated as NE, is arguably the most important concept in game theory. The Nash Equilibrium, introduced by John Nash, describes a situation in a game in which all rational players have no incentive to deviate from their strategy [3, 6].

There are two types of Nash Equilibria in game theory. There is a pure strategy equilibrium and a mixed strategy equilibrium. A pure strategy equilibrium is a strategy where it is in every player's best interest to use one non-changing action. In comparison, a mixed strategy equilibrium is where it is in every player's best interest to choose a strategy from a fixed probability distribution of possible alternatives. In other words, a mixed strategy has assigned probabilities for choosing each possible action such that there are at least two actions with nonzero probability.

If, for example, both players in a two-player zero-sum game do not have a pure equilibrium strategy that will guarantee their choice is always best for them, then they should try to find a mixed strategy Nash Equilibrium. After calculating the probability of their moves, each player can both come to a mixed equilibrium strategy, which gives them the best outcome given the information they have on each other [6]. It is important to note that a game may or may not have a pure strategy equilibrium, but all finite games have one or more Nash Equilibrium strategies, typically a mixed strategy equilibrium [3, 6].

Colonel Blotto has a Nash Equilibrium, as it is a finite game. Furthermore, traditional Blotto games have been proven to have one or more pure Nash Equilibria, but only under explicitly defined conditions [7, 8, 9].

Models

Overview

Colonel Blotto provides a very flexible, generic model that has been extensively adapted. These adaptations include sequential models, non-two player variants, models with unique resource allocation, unique resources, and models based on jamming attacks. The categories are listed in a hierarchy of most general to most specific. Every category has its purpose, and some may less representative than others [4, 5]. Our model is a simultaneous two-player topology-specific Blotto game with a utility representing the node average node degree.

Sequential models

A sequential game is a game where one agent decides on a strategy while knowing what the other agents have done previously. Juxtaposing this type of game is a simultaneous game. In a simultaneous game, each agent decides their strategy without knowing any private information about their strategy. An example of a simultaneous game would be the Prisoner's Dilemma, as each agent only takes one turn then the game is over. Games like this usually are simpler to model.

A Colonel Blotto game can be represented both as a simultaneous game and as a sequential game. The simultaneous game would have the users allocate their resources without any of the other agents' information and a winner could be chosen after both agents decide. Our model is a simultaneous one as we are focused on the optimal configuration of a network prior to any attack. This differs from most research that focuses on the model's creation rather than what it produces under given parameters. An example of a simultaneous Colonel Blotto game is described below.

We assume there are two players, Colonel Blotto and Enemy Edward. Both players have 10 units to allocate to 3 battlefields B1, B2, and B3. Colonel Blotto's strategy is to allocate his units 5, 3, and 2 respectively, whereas Enemy Edward allocates his units 4, 5, and 1 simultaneously. After both players deploy their strategy, Colonel Blotto wins because he has more units on battlefields B1 and B3. Colonel Blotto lost his battle at B2, but he still wins that game as his utility is 1, the number of battles won minus the battles lost, whereas Enemy Edward's utility is -1. Notice that the sum of the two players' utility is 0. This is an example of Colonel Blotto as a simultaneous zero-sum game.

Researchers such as Fuchs and Khargonekar [8] made a Colonel Blotto model representative of a deterministic sensor network. Their model has Player A receive data from sensor networks looking over regions. Player B does not observe the sensor's initial readings. The authors describe their game as "a deterministic threshold sensor, s_i, monitors region, R_i, along the battlefield [8]." Additionally, they keep the threshold in which the sensors change readings to be some universal constant τ. The overall outcomes of the output of each sensor are public knowledge. Each player uses a multi-strategy approach to optimize their utility; the utility of each player is represented by $\frac{1}{N}$, where N represents the finite sum of battlefields in the game. This utility allows for simple interpretations and constitutes an easily defined zero-sum game.

The solutions, or equilibrium strategies, to this game is different for either player. Player A has an advantage because they know how the sensor networks will be laid out before Player B allocates their resources. This information can become crucial in later turns in the game, possibly building on itself. The authors further talk about the possibility of Player B having a pure strategy approach under certain conditions [8]. Our Colonel Blotto model may similarly have one agent sometimes play a pure strategy given a specific topology with some kind of bottleneck or centralized node.

The model described has many areas where it can be expanded. For example, the utility functions are made purposely simple, the authors leave the door open for any interpretation of using multivariate probability distributions, and the players are constrained relatively loosely relatively to their model and its representation. The researchers also have an entire solution revolving around action theory in a "first-price all pay action" [8]. This flexibility and openness to interpretations make this model very appealing to build off. The sensors and their thresholds can embody a lot of other theoretical models. In particular, the authors state, "the basic

machinery developed in this paper can potentially be used for analyzing resource allocation problems in adversarial communication and information network blocking and jamming problems [8]."

The correlation between the two is extremely apparent and extremely applicable. A DoS attack may extend on their model in many ways. An attacker could have this mutable threshold τ, a Colonel Blotto subgame in a node when representing a DoS attack, or an option of allocated resources to a specific node or edge depending on its importance to the network structure. Solving these subgames could be an extension of our model for more intricate networks.

Modeling Jamming Attacks with Blotto

These jamming models are subsets of the Blotto model representing denial-of-service attacks. Guan *et al.* [13] developed a Blotto-based model to represent internet security, rumor spreading control, and communication timeliness of IoV (Internet of Vehicles) [9, 13]. The internet security section describes how the model will apply to a computer network. Under their broad model, one of the possible actions for the attacker is a DDoS attack. Similarly, Labib and Ha [1] presented a model in which they represented a jamming attack on the Internet of Things (IoT) networks.

Like in our work, these previous works created a small-scale network scenario, although with different rationales. In [13], the authors chose small-scale examples because the researchers' model was representative of different real-world applications and posed it simple as a proof of concept that their model works. We use the same topologies as [13]. Our regret minimization algorithm guarantees convergence to an equilibrium strategy, as opposed to Guan's genetic algorithm which cannot. Labib and Ha choose small networks to analyze the effect of a fusion center, which could defend against incoming jamming attacks [1, 13].

In these studies, computational reasons also limit the size of networks that were considered. Wang et al. [13] noted for example that "computational complexity raises rapidly with the increase of network scale." The explosion in computational complexity limits the modified genetic algorithms these authors used. We did not investigate further information about the more complex genetic algorithms, as we used smaller, less general, sample sets for our Colonel Blotto game. This echoes the process of other models representing DDoS attacks [1, 10, 11, 12, 13]. This paper's model will have similar computation limits due to the regret

minimization algorithm. The strategies reported in the results section were on slightly smaller networks with up to five times as many training iterations for each strategy.

Guizani *et al.* [14] also proposed models to represent antijamming attacks on IoT networks. Their model ended up being representative of higher and lower power jammers. In this model, an attacker can choose between high and low power jamming. Having a strategy that applies to both power levels may be better because of that extra protection [11, 14, 15].

Limitations

Overview

All game theory models have limitations. There is always some condition that the model assumes or constraints to create its data. It is important to identify these limitations to any model that is made. Without knowing the model's limitations, the model may be misused or misrepresented. If such an event occurred, it may invalidate results that are valid or conversely valid results that are invalid. This is critical to establish a proper model.

Common limitations

Limitations are often computational or come from the designed constraints. In models such as the one presented in Hajimirsadeghi's "Internetwork dynamic spectrum allocation via a Colonel Blotto game" and Labib's "A Colonel Blotto game for anti-jamming in the Internet of Things" there exists computational limitations [1, 10]. The authors address this in their respective research stating that one of their posed algorithms or equations has computational limitations. In some cases, the algorithm may have a time complexity that is growing rapidly with large or even medium-sized data sets. In these situations, the authors typically pose a different solution veering away from the computationally intensive one, or they may simply state the limitation and leave it to the reader to use it appropriately. It is critical to mention this in a written report as mentioned before.

Limitations may also come in the form of a constraint set by the author. These constraints are often laid out in the title, abstract, or introduction of the works because of their importance. An example of a constraint set by an author is an agent A_n may only be allowed to allocate x number of resources on a territory t_i in a Colonel Blotto game. Similarly, if a game were representing an election, there might be a limitation that only y number of candidates, or agents, are allowed to run for election.

Conclusion

Game theory itself is extremely versatile when it comes to replicating and simulating real-life scenarios. Because of this, representative models can be exhaustively studied by applying different ideas from game theory and mathematics. This allows researchers to find conclusive approaches for real-life scenarios. Whether it be in the form of an equilibrium strategy with cooperation between parties or an equilibrium strategy for a party only concerned with themselves. Having this statistical data behind the models compels the representative agents to follow the preferred or effective strategies. This can ensure effective decision-making in the real world.

In conclusion, the Colonel Blotto game model has a varying amount of use cases when used to represent jamming attacks. When extruding ideas from researchers such as Labib, Guan, Fuchs, and Hajimirsadeghi, a theoretical model of Colonel Blotto used to represent DoS attacks involving resource allocation on networks, edges, and servers, nodes, is entirely feasible and likely will create some or many equilibria in which the defender of the attacks could have an optimized approach to keeping their networks secure. This model should assist network administrators in achieving their goals of network security and be utilized as a learning tool to fend off ill-intentioned attackers.

Our proposed model uses average node degree as the utility function. This utility function has not been used in a Blotto game to represent an attack on a network. This metric objectively represents the value a server would be to an overall network. Additionally, our regret minimization algorithm produces strategies that are verifiably equilibrium strategies. This differs from many previous works, where they approximate equilibrium strategies using genetic algorithms. The advantage of these models is that they can represent bigger networks more

efficiently and even have infinite actions for either player. Our model gives exact strategies under given constraints. This allows us to determine the approaches players should opt to take under similar scenarios. This insight is different from past research where they strictly created similar models without diving into what strategies the players choose and why.

Chapter 3 Model

Overview

Our model represents a general wireless network represented as a mathematical graph. Each node in the graph functions as a Colonel Blotto battlefield. The model has two agents, an attacker, and a defender. Both agents are given symmetric resources and they both decide their resource allocations to some graph simultaneously. As shown in Figure 1, the model is separated into three distinct phases. First, there is the initialization phase where the game is formulated given some parameters and the payoff matrix is filled. Secondly, the training phase iterates through rows and columns of the payoff matrix in order to evaluate the effectiveness or ineffectiveness of certain actions. Lastly, the verification phase takes the trained agents' equilibrium strategies and verifies them as such.

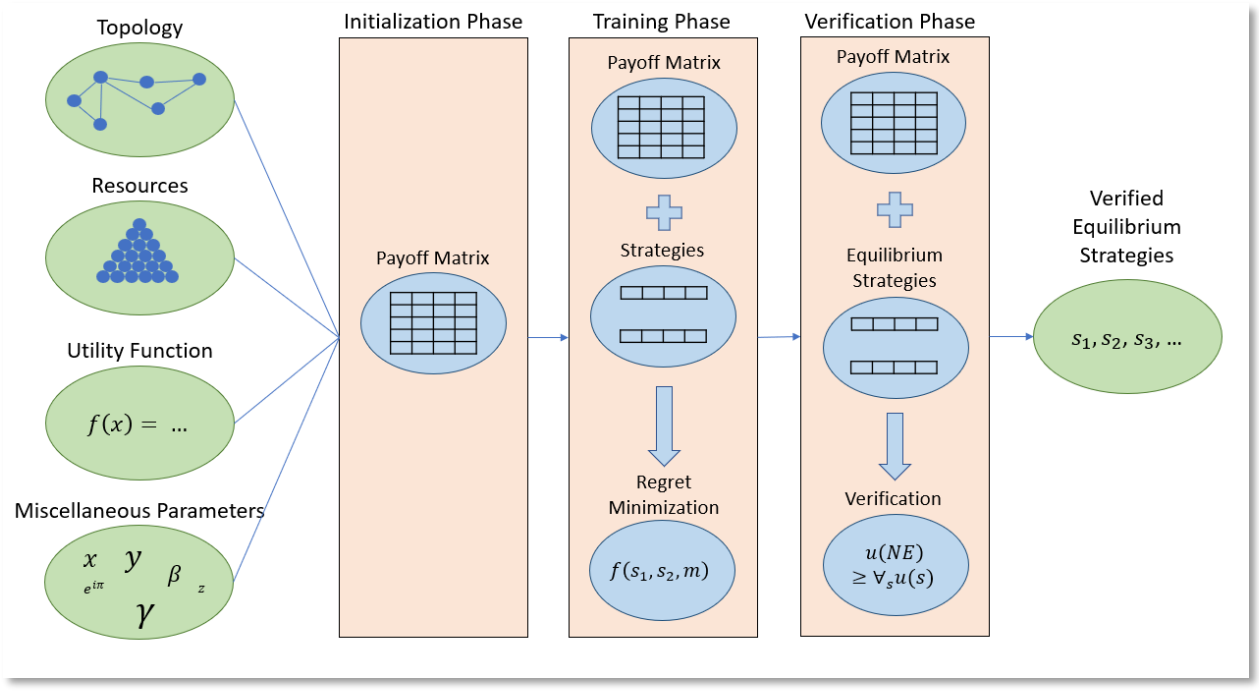

Figure 1: A visual representation of the phases in the modeling process

Initialization phase

The initialization phase instantiates the game. The model is created by taking many parameters, most notably the network topology, the number of nodes for both players, and the utility function of the game. Changing the parameters can easily result in dramatically different, yet interesting results. In addition to the parametrized game model, the codebase has functionality for further analysis, plotting, and thorough testing functionality.

The adaptive model allows for many situations to analyze, which is limited by the computational complexity of the algorithm. Two major computational costs come from the initialization and training phase. The cost of the initialization phase is mainly comprised of constructing the payoff matrix. We construct the payoff matrix by pre-calculating the utility for every possible game configuration. This is done by evaluating a game for every possible action configuration between the attack and the defender. Both players have symmetric resources and the same topology, therefore their action sets are the same. The payoff matrix is *m* by *m,* where *m* represents the cardinality of the action set.

Due to these constraints, we empirically determined that the network being tested should have less than seven nodes and less than ten resources to be able to run with approximately 100,000 iterations of training and converge to equilibrium strategies. The running time of the model is dependent on the size of the payoff matrix, which is dependent on the action set length. The action set length is determined by the equation below, where *n* is the number of nodes and *r* is the number of resources.

$$\binom{n+r-1}{n-1}$$

This is to say for each resource at each node once minus one, we choose from the remaining $n-1$ nodes. In order to not count one resource on a node twice, one was subtracted. Note that 10 nodes and 10 resources create 92,378 possible actions per player, which is over 8.5 billion games to initialize the payoff matrix. Initializing this matrix would take too long. Moreover, we would need to sample with a sufficient number of outcomes in each entry in the matrix to converge to the equilibrium strategy.

If we fix the number of resources, we can analyze the growth in the number of actions for each player as the network size grows. Then we can determine the complexity as a function of the number of resources available to a player, as shown in Figure 2.

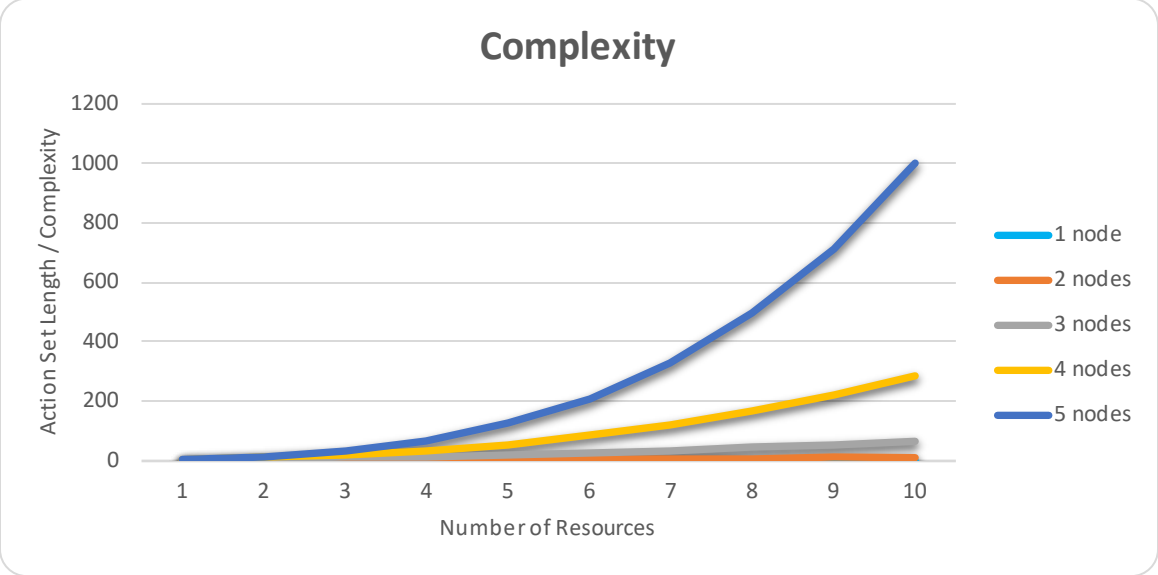

Figure 2: The complexity of the model

With the small sample size shown in Figure 2, it is apparent that the complexity grows much more quickly when the number of nodes increases rather than the number of resources. Equation 1 shows that complexity is in $O(n^s)$ where n is the number of nodes, s is a constant number of resources, and $T(n)$ represents the running time complexity of our model with input of length n.

$$T(n) = \binom{n+s-1}{n-1}$$

$$\frac{(n+s-1)!}{(n-1)!\,(n+s-1-n+1)!}$$

$$\frac{(n+s-1)!}{(n-1)!\,(s!)}$$

$$\frac{(n+s-1)(n+s-2)\ldots n(n-1)!}{(n-1)!\,(s!)}$$

$$\frac{(n+s-1)(n+s-2)\ldots n}{(s!)}$$

$$(n+s-1)(n+s-2)\ldots n \leq (s!)(n^s)$$

$$T(n) = O(n^s)$$

Equation 1: Model's big-Oh notation complexity

Utility

The utility used for our analysis is the average degree of each node in the network. This is objectively a good representation of a network's relative connectivity, as the more nodes one is connected to the more dependable a network may be. This may represent how multiple sites work in a system as the average degree has a direct relation to how valuable a system would value a site.

Other utility functions were considered when creating the model. The giant component of a network was considered for the utility; this is the largest number of nodes that can be accessed from any node within the component. This metric was ruled out because the relatively small scale did not produce interesting results. The other heuristic that was considered was average node connectivity. Average node connectivity is the number of nodes reachable from any given node. This metric yielded similar results to the average degree but typically required more analysis and was not always as clear as the average degree. Thus, the average node degree was used as the model's utility function.

Parameters

The model was created with a number of parameters. Parameters, including different network topologies and a different number of resources, are utilized but one of the most interesting parameters was changed which agent wins the tie if one occurs. In reality, both metrics are possible in a real-life network because if the defender wins the tie, it is equivalent to if the sites in the network were at full capacity but still operational. If the attacker wins the tie is to say that the defender lost because the defender's site cannot process any more legitimate traffic. When training the agents, typically 100,000 games were emulated to converge to an equilibrium strategy.

Training phase

The training phase uses the payoff matrix to converge to some equilibrium strategy. The training phase instantiates both players' strategies as uniformly distributed stochastic vectors. Hart and Mas-Colell's regret minimization algorithm is used to update the strategy each iteration to converge closer to an equilibrium strategy [16].

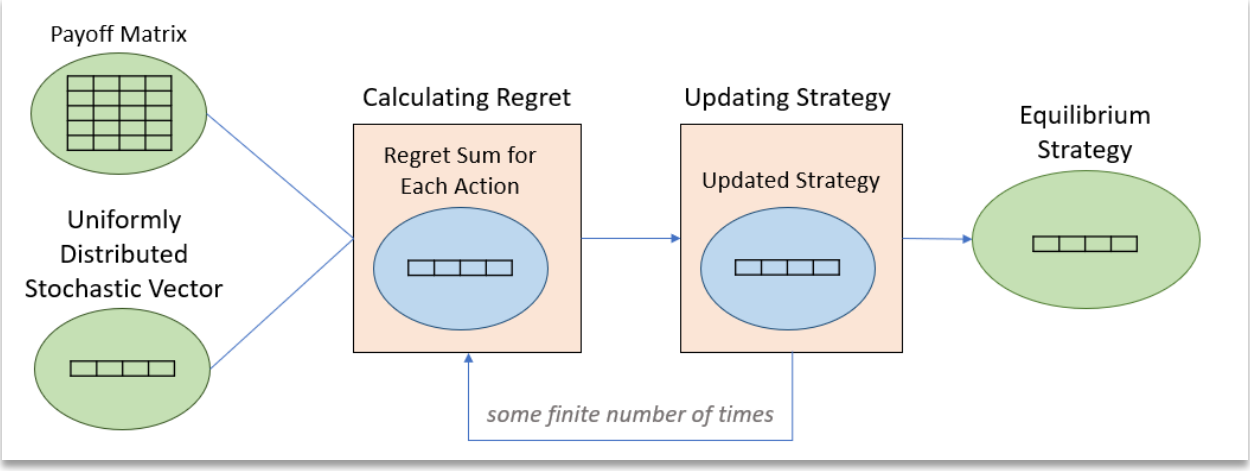

Figure 3: A visual representation of the regret minimization algorithm

The algorithm is broken down into two major steps. The first step calculates the agent's regret for not picking some other action. Regret is the difference between the possible utility of another action and the utility of some game outcome. The regret is added to a regret sum vector for each player. The regret sum vector tracks each player's regret for not playing some action. After comparing the outcome to every action, the algorithm updates the agents' strategies by turning their regret sum vectors into stochastic vectors that represent an agent's strategy. After this is repeated some finite number of times, the average of all the updated strategies is returned.

Hart and Mas-Colell's start by calculating the difference in a player's average payoff to a certain point. Let j and k be possible actions for player i, τ be some point in time such that $0 \leq \tau \leq t$ and t is some arbitrary time, u^i be the utility function for player i, and s_τ is the strategy at

the time τ. Similarly, s_τ^i represents player i's strategy at time τ. Note we only have two agents, so i will represent one of the players and $-i$ will represent the other; typically, $-i$ represents all players besides i in an n-player game. The difference in a player's average payoff is

$$Difference_t^i(j,k) = \sum_{\tau \le t : s_\tau^i = j} \left[u^i\left(k, s_\tau^{-i}\right) - u^i(s_\tau)\right].$$

Equation 2: Difference in a player's average payoff

The average regret at some time t is defined as

$$Regret_t^i(j,k) = max\left(Difference_t^i(j,k), 0\right).$$

Equation 3: Average regret for a player

Our equilibrium strategy output is defined as follows

$$equilibrium(j,k) = \begin{cases} p_{t+1}^i(k) = \frac{1}{\mu} Regret_t^i(j,k), & \forall_k (k \neq j) \\ p_{t+1}^i(j) = 1 - \sum_{k \in s^i : k \neq j} p_{t+1}^i(k). \end{cases}$$

Equation 4: Equilibrium strategy for a finite *n*-player game

for some probability distribution of the actions, p, and some set number $\mu > 0$ that guarantees $p_{t+1}^i(j) > 0$, where $p_{t+1}^i(k)$ is the probability distribution for player i to play next given k was just played. In other words, there is always some positive probability of choosing either row in $equilibrium(j,k)$ [16].

Note that computational costs are tied to the size of the payoff matrix. Each training iteration references one n-length row and one n-length column of the payoff matrix, where n is the number of actions for each agent. Recall that the ten nodes and ten resource example creates 92,378 actions per player. To run one iteration of the training algorithm, we need to sample 92,378 actions for each player. A more formal description of the regret minimization algorithm is

shown in Figure 4.

```
# Update agent
normalizing_sum = 0
for i in range(self.AGENT_ACTION_LENGTH):
    possible_util = self.payoff_matrix[i][other_agent_action_idx]

    # add regret
    difference = possible_util - util
    self.agent_regret_sum[i] += difference

    # calc normalizing sum
    if self.agent_regret_sum[i] > 0:
        normalizing_sum += self.agent_regret_sum[i]

# Update strategy
for i in range(self.AGENT_ACTION_LENGTH):
    if normalizing_sum > 0:
        if self.agent_regret_sum[i] > 0:
            self.agent_strategy[i] =
            self.agent_regret_sum[i] / normalizing_sum
        else:
            self.agent_strategy[i] = 0
    else:  # if no positive sum -> 1 / n
        self.agent_strategy[i] = 1 / self.AGENT_ACTION_LENGTH
```

Figure 4: A formal description of the regret minimization algorithm

As previously mentioned, this algorithm will compare an agent's action to every other possible action they could have taken. The agent's strategy changes according to this regret and an equilibrium strategy is calculated when all of the played strategies are averaged together. The algorithm's running time is in $O(ni)$ where n is the number of actions and i is the number of iterations.

Verification phase

The function verifies the strategies as equilibrium strategies by calculating its expected utility and comparing it to the expected utility of every possible pure strategy an agent can play. The technique can tell whether the proposed strategy is or is not an equilibrium strategy. A strategy is deemed an equilibrium strategy if given an opponent's strategy, the player cannot earn a higher utility by deviating from their current strategy [3]. To verify this, the verification phase calculates the expected utility of each player when playing any pure strategy against the other player's posed equilibrium strategy. If the utility from playing any pure strategy does not exceed the utility from the posed equilibrium strategy, then this strategy is an equilibrium strategy. This method was used to verify all proposed equilibrium strategies.

Chapter 4 Results

Overview

In this study, we looked at seven different simple network topologies for their most commonly picked actions from their respective equilibrium strategies. Most notably we analyze the strategies from the line topology, bowtie topology, ring topology, and fully connected topology. All the network topologies can be seen in Figure 5.

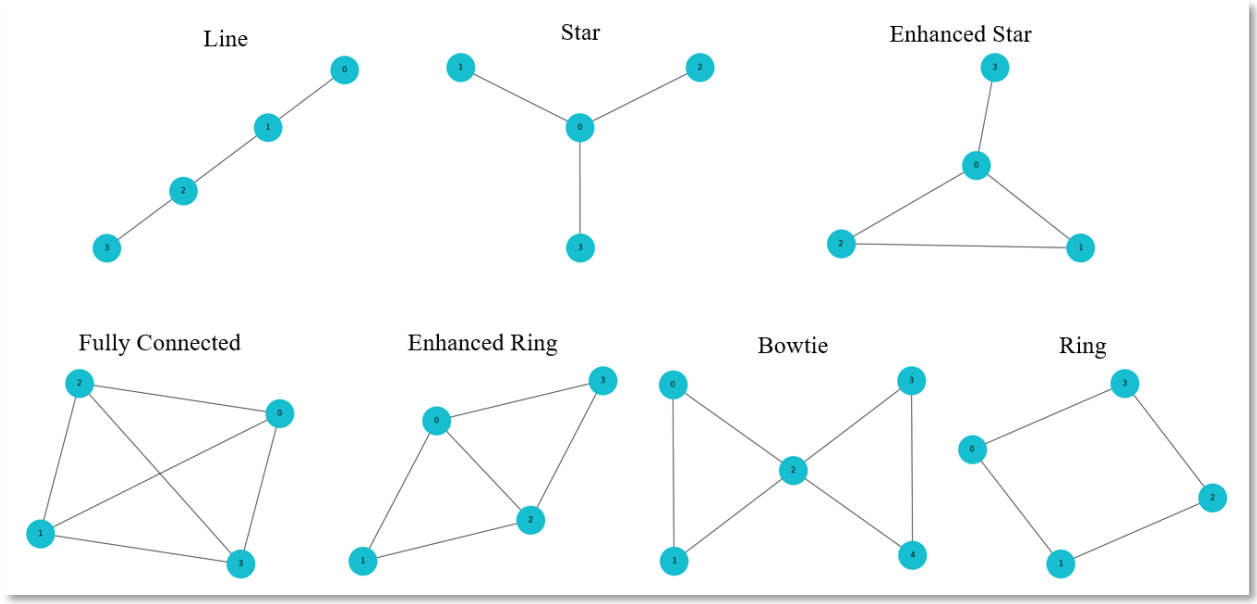

Figure 5: A visual representation of all studied topologies

For all analyses, we used a utility function representing the network's average node degree. A bowtie topology was added to the original set posed by Guan [2]. The line topology was used to represent an extremely simple network with predictable results to ensure the model is working correctly. In doing so, we identified the weak points in the network, explored variations of this small network, and identified intuitive and unintuitive strategies behind some commonly structured network topologies. Through this analysis, we discovered possible network

configurations that could be used to ensure the best worst-case performance of any network. The analysis covers different sets of parameters for the model and discusses how these changes may be crucial for network reliability.

Line topology

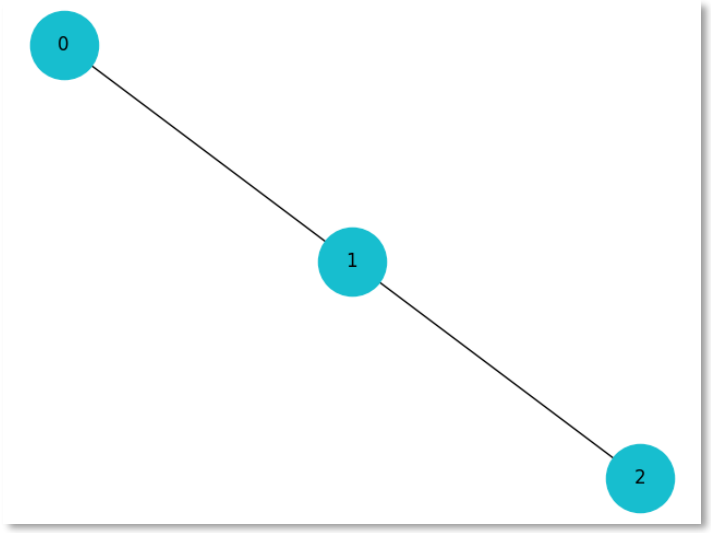

Figure 6: A line topology

We start with a simple three-node line topology as can be seen in Figure 6. With this simple configuration, we start by allowing each player to allocate one resource in the network. Table 2 shows the payoff matrix of such a game where the rows represent the defender's actions, the column represents the attacker's actions, and the intersection point is the resulting utility of either player playing their respective actions. Note that the utility is from the defender's point of view, though because of the zero-sum nature of the game, it represents both players accurately. After playing out a few scenarios, some equilibrium strategies emerge quickly.

Table 2: Payoff matrix of 1 resource 3-node line topology

	[0, 0, 1]	**[0, 1, 0]**	**[1, 0, 0]**
[0, 0, 1]	1.33333	0.00000	0.66667
[0, 1, 0]	0.66667	1.33333	0.66667
[1, 0, 0]	0.66667	0.00000	1.33333

The model identifies a strategy where the defender places its resource at node 1, the central node, 60% of the time and each outside node 20% of the time. The attacker plays an inversely similar strategy, placing the resource 20% at the central node and 40% on each of the outer nodes. Both agents seemed to identify the most valuable node in the network. The defender keeps that node secure a majority of the time, but still protects the edge nodes to prevent the attacker from always getting utility from taking them out. Likewise, the attacker sometimes allocates its resources to the center node to keep the defender agent honest, in order to capitalize on the edge node that they can get more often. The aforementioned verification method was used to verify that this is an equilibrium strategy. All the strategies discussed have been verified.

Adding more resources to the 3-node line topology, where the defender wins the tie yields similar, more general results. The defender places a majority of the resources in the middle always, secondly, the defender tries to save its edge nodes by allocating some of the resources to one of the edge nodes. For the lesser probable actions, the defender is placing more and more resources on the edge nodes, eventually at the end abandoning the middle node and splitting the edge nodes notably, unevenly. Note that this only happens for the smallest probable actions of the defender's strategy. The actions, as well as their respective probabilities, are shown in Table 3. Note that there are very slight rounding errors due to floating-point precision and inherent properties of probability. The deviation could possibly be caused by the strategies not converging yet.

Table 3: 3 vs 4 resource actions for a 3-node line topology

3 Resources		4 Resources	
Defender	**Attacker**	**Defender**	**Attacker**
[0, 3, 0] 33.3%	[2, 0, 1] 22.5%	[0, 4, 0] 27.8%	[2, 0, 2] 21.1%
[1, 1, 1] 22.0%	[0, 3, 0] 22.4%	[2, 0, 2] 13.3%	[0, 4, 0] 19.0%
[0, 2, 1] 11.7%	[1, 0, 2] 21.9%	[1, 2, 1] 11.5%	[1, 2, 1] 13.9%
[1, 0, 2] 11.2%	[1, 2, 0] 11.3%	[0, 3, 1] 9.3%	[0, 3, 1] 9.3%
[2, 0, 1] 11.0%	[0, 2, 1] 11.1%	[1, 3, 0] 9.2%	[1, 3, 0] 9.0%
[1, 2, 0] 10.9%	[1, 1, 1] 10.8%	[0, 2, 2] 7.6%	[1, 1, 2] 7.1%
		[1, 1, 2] 7.3%	[2, 1, 1] 6.9%
		[2, 1, 1] 7.3%	[3, 0, 1] 6.9%
		[2, 2, 0] 6.7%	[1, 0, 3] 6.8%

Interestingly the attacker, again, adopts a strategy that is a mirror image of the defender's strategy. Although, with extra nodes, the attack mainly allocates all to the middle or mostly to the middle than on both ends. which counteracts the defender's most common actions. This means the attacker prioritizes the end nodes for the majority of the time and then the lower probable, but not the lowest, has the attacker go back to the middle node, as statistically, the defender will give up its middle node at this time.

The results for this simple example seem to check out. We have a defender who prioritizes protecting their central, most valuable, node, though either agent does not solely focus on that center node. The defender's most common action guarantees a positive utility no matter what the attacker plays. This is a trivially desirable strategy. The defender will allocate more resources to the end nodes slowly with lesser probability as more get allocated. This can be seen in Table 4. Conversely, the attacker slowly allocates more towards the center for larger resource allocation. The only deviation from this comes from the attacker. The attacker actually veers away from this pattern a little. Instead of always trying to do the inverse of the defender, the first two actions are always putting one resource on each end node and the rest on the center node and

secondly putting all of the resources in the middle. This is in spite of the fact that the defender plays mostly in the center and then favors one end node as its second most probable action. This can be justified by the attacker not wanting to capitalize on the defender by not necessarily focusing on either the central or the ends nodes at the same time. Instead, the attacker adopts this mixed approach while the defender stays discrete, which ends up in a greater utility for the attacker.

Table 4: Line topology resource allocations for 4, 5, and 6 resources

4 Resources	5 Resources	6 Resources
[0, 4, 0] 27.8%	[0, 5, 0] 24.5%	[0, 6, 0] 21.9%
[2, 0, 2] 13.3%	[3, 2, 0] 8.8%	[2, 4, 0] 11.0%
[1, 2, 1] 11.5%	[0, 2, 3] 8.6%	[0, 4, 2] 10.6%
[0, 3, 1] 9.3%	[0, 3, 2] 8.5%	[0, 5, 1] 7.6%
[1, 3, 0] 9.2%	[1, 4, 0] 8.2%	[1, 5, 0] 7.2%
[0, 2, 2] 7.6%	[0, 4, 1] 8.0%	[3, 0, 3] 6.1%
[1, 1, 2] 7.3%	[1, 3, 1] 7.8%	[0, 3, 3] 6.0%
[2, 1, 1] 7.3%	[2, 3, 0] 7.8%	[0, 2, 4] 5.5%
[2, 2, 0] 6.7%	[2, 1, 2] 7.5%	[4, 2, 0] 5.5%
	[2, 0, 3] 5.3%	[3, 3, 0] 5.3%
	[3, 0, 2] 5.0%	[3, 1, 2] 3.8%
		[1, 4, 1] 3.7%
		[2, 1, 3] 3.7%
		[2, 0, 4] 1.0%
		[4, 0, 2] 1.0%

When the game parameters change, and the attacker wins the tie in the network this game becomes trivial. If the attacker allocates all of their resources to the central node, they will always win this node, which disconnects the entire network. Trivially, an attacker's equilibrium strategy is to play all of their resources on the central node. Though this is an equilibrium strategy, it is not always the one our model found for the attacker. The model ended up playing all of the nodes in the middle only 50% of the time. The other 50% percent of the time the attacker plays most of most of their resources in the middle but puts a resource on each edge

node. Initially, this seems like a mistake, but this is also an equilibrium strategy, likely a more practical and realistic one too.

When the attacker plays this strategy, it is still impossible for the defender to get any utility. In this case, the defender can play all of their resources in the middle and still maintain control of the central node, though there is still no utility there. The attacker will always get the edge nodes resulting in no utility for the defender. The equilibrium strategy is easily verifiable even without full formal verification. As previously mentioned, this initiative equilibrium strategy the model found may be more applicable than the trivial one. This approach shows that with some kind of advantage an agent can diversify how they play sometimes to gain some more real-life advantages. For example, there is a realistic scenario where it is more beneficial to take out two nodes rather than just the central one.

Expected utility

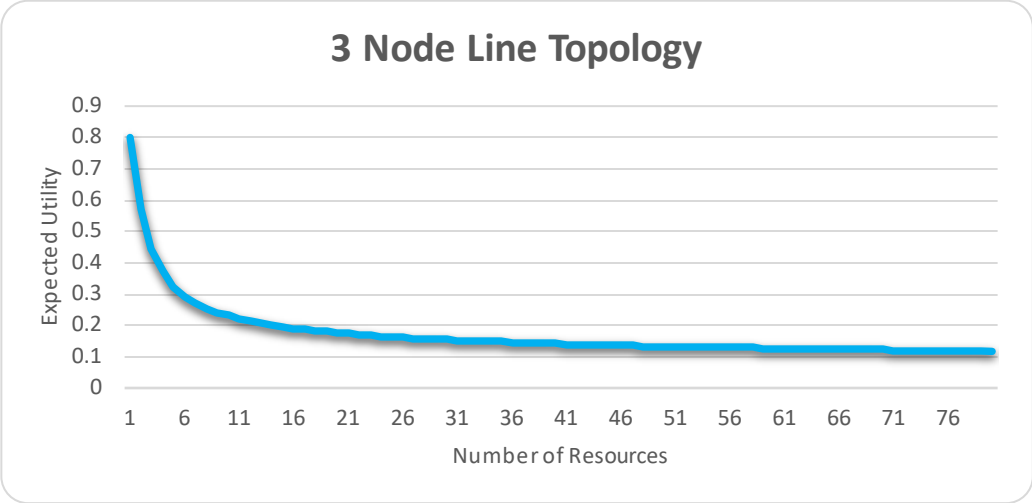

Figure 7: Expected utility of a 3-node line topology with increasing resources

The expected utility for a three-node line topology is plotted against the number of resources in the game in Figure 7. The expected utility trends downwards as the number of resources increases. Therefore, the attacker gains some advantage with more resources in play. This can be reasoned by going from the one resource to the two-resource case. With only one node, the attacker must almost settle for the edge nodes most of the time because they are unlikely to take that central node. Though when the agents have two resources, the defender has to defend against a more dynamic attacker with more possible actions to defend against. The same logic would hold as the number of resources, and therefore more actions, increase past two.

When expanding the line topology to four nodes, there is similar behavior when discussing the expected utility. This is shown in Figure 8 for the same reason as before.

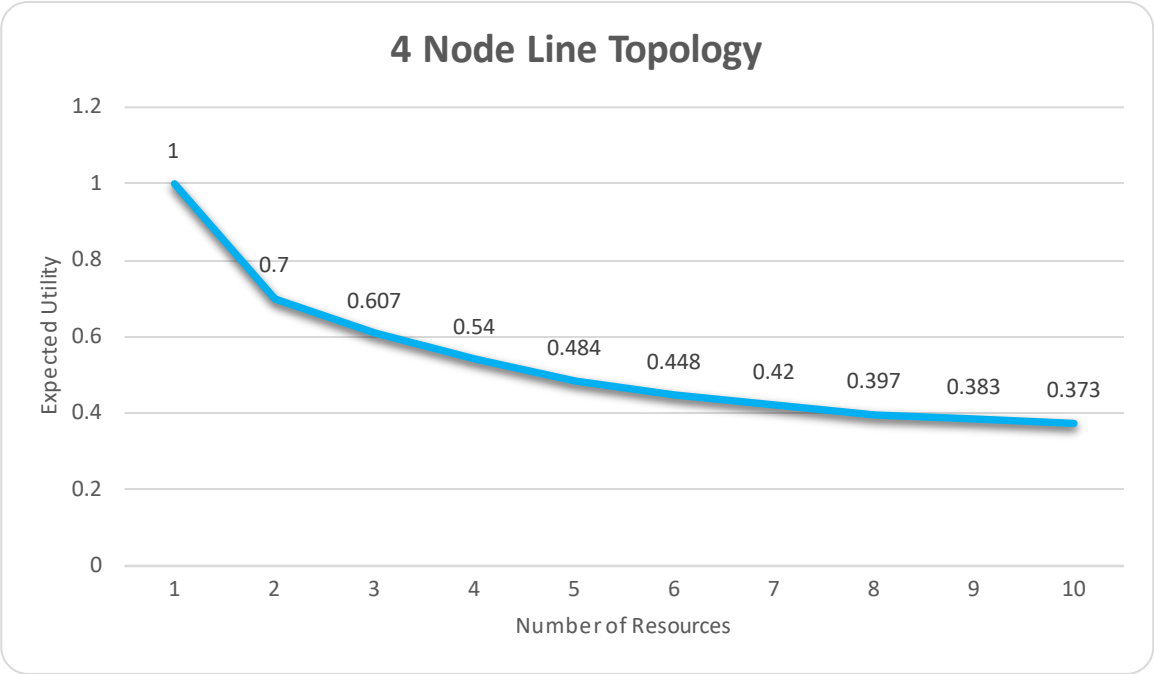

Figure 8: Expected utility of a 4-node line topology with increasing resources

Other topologies

Expanding this logic further to different topologies yields comparable results. The same comparison can be had for different topologies. Six different network topologies were compared. The network topologies were carried over from Guan's paper [2]. Namely the line topology, fully connected topology star topology, ring topology, enhanced star topology, and enhanced ring topology.

The same relationships are observed across multiple topologies due to the same reasoning as before. Figure 9 shows a very similar relationship when comparing the different topologies and their expected utility, likely due to the same reasoning.

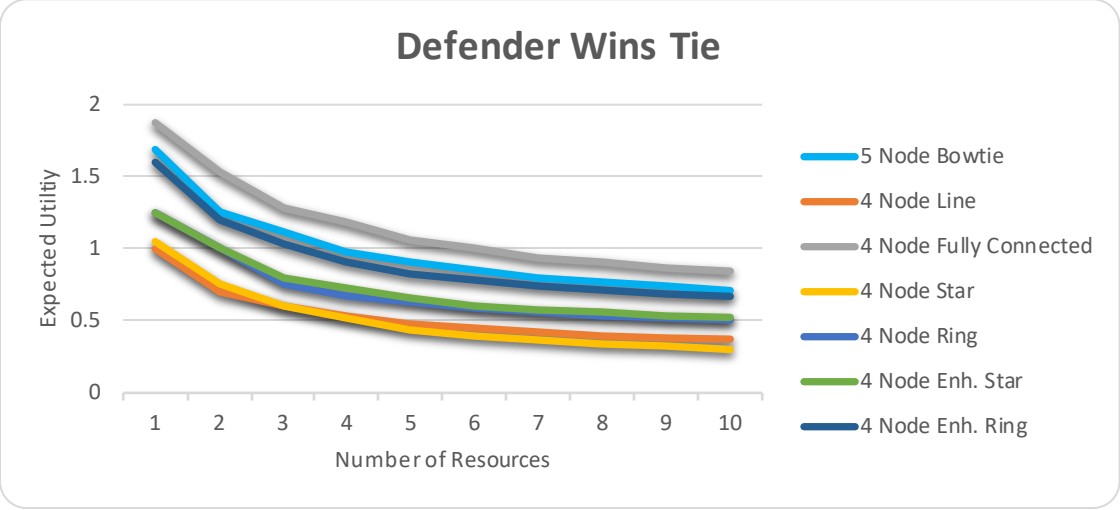

Figure 9: Expected utility of topologies with increasing resources and the defender wins ties

Again, more interesting results come from when the game favors the attacker. Figure 10 shows the same relationships graphed; except this time the attacker wins the tie in the games.

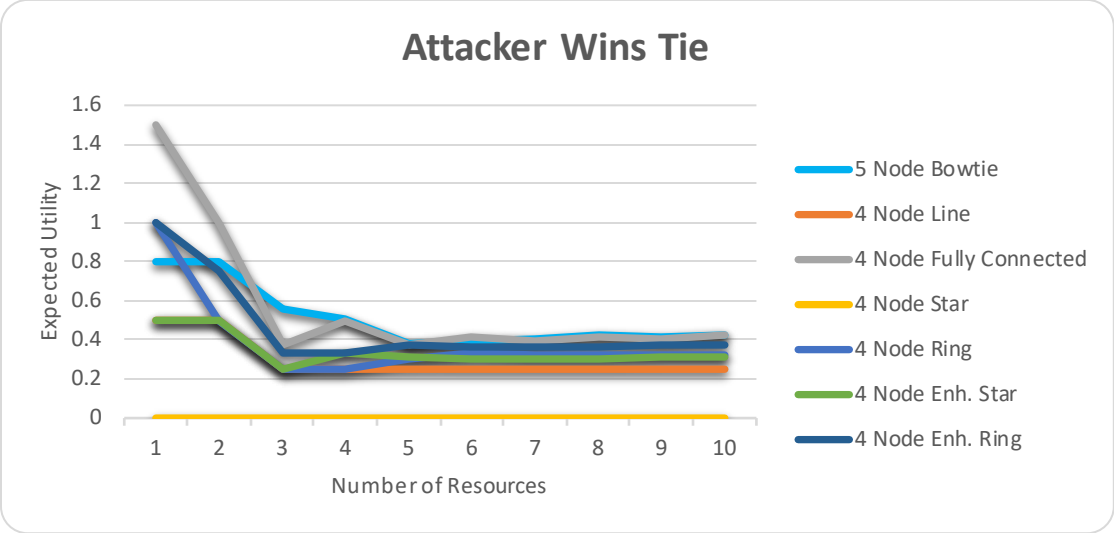

Figure 10: Expected utility of topologies with increasing resources and the attacker wins ties

This yields interesting results. The star type topology has a similar fate to the previously discussed 3-node line topology, where the defender has no strategy resulting in a positive utility against an intelligent attacker. There is an obvious odd behavior with most if not all of these topologies. The expected utility seems to fluctuate as the resources increase. This is extremely dramatic with one to four resources. The pattern continues but to a much smaller scale. Though it is hard to see, there still is a fluctuating pattern for six or more resources.

Starting with the fully connected topology, the attacker gains some large advantage with one, two, and three resources but when there are four resources that seems to go away. The advantage for the defender, if any exists, comes when the number of resources is equal to the number of nodes. When this happens, the attacker tends to play their most spread-out actions the most. In other words, the attacker has a stronger emphasis on hitting every node in the topology.

Bowtie topology

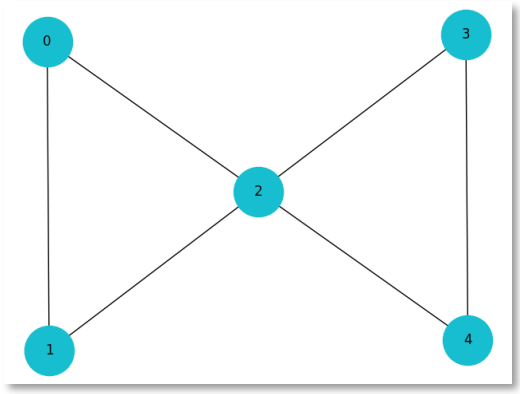

Figure 11: A bowtie topology

The bowtie topology is an extension of the line topology. The bowtie topology has a central node that holds the most value, labeled with the number 2 in Figure 11. This node has a degree of four whereas all of the other nodes have a degree of two. This larger degree is extremely valuable to the defender. One might think it is best to allocate all of their resources to the central node; though this is a naïve strategy, it does not have desirable outcomes for either agent.

Starting with one resource for each player the results are the same as the line topology. The defender mostly plays in the middle node and the attacker mostly avoids the middle node. This is exactly the same as how the agents played in the three-node line topology. Similarly, when the game parameters change so that the attacker wins the tie, the attacker strictly plays their resource in the middle, as they are guaranteed to cause significant damage to the network no matter what the defender does. For the rest of the analysis of the bowtie topology, we do not focus on when the attacker wins the tie.

Adding more resources to a game with normal conditions, both agents' strategies change. For two and three resources, there are not any major changes from the defender. Notably, the

attacker does start to focus on the central node when more than one resource is available. But four resources and above the defender changes their approach. Instead of allocating a majority of their resources into the central node, the defender decides to try to maintain one side of the bowtie, including the central node. This kind of logic makes sense when the number of resources approaches the number of nodes in the topology. As more resources are added, the attacker has more potential attacks or actions the defender has to worry about to capitalize on the sides of the bowtie. Settling for the ends of the bowtie allows the defender to salvage some utility without gambling all on the central node.

Overall, the results are quite in line with what the line topology produced. Nontrivial equilibrium strategies were present when the attacker wins the ties, just like the line topology. When the game's resources are greater than two the attacker limits resources allocated to the central node. The attacker only allocates at most 50% of their resources towards the central node when the defender has a central-focused strategy. These allocations additionally are typically in the less likely set of actions. So again, we can see an attacker capitalizing off a defender who prioritized their highest potential node, giving us a less trivial strategy that utilized a more distributed attack.

As more resources are added to the game sometimes the defender decides to sacrifice their central node in order to keep the ends of the bowtie network intact. This follows the logic from the expected utility section where the defender, in general, is at a disadvantage when more resources are added. Likewise, the slight oscillation in Figure 10 coincides with when the attacker has a center-focused strategy. When the attacker has a center-focused strategy, their expected utility is marginally lower than when the previous strategy was non-center-focused. It is worth noting that the attacker's focus is debatable as more resources are added. The actions

percentage played and order in which they are played gets more and more meshed together with more resources, which would explain why the oscillation noticed is extreme at the beginning of the graph but then becomes extremely minimal, if not completely unnoticeable.

Table 5: 5 resource bowtie topology actions

Defender	Attacker
[0, 0, 3, 1, 1] 11.8%	[0, 0, 5, 0, 0] 12.0%
[1, 1, 3, 0, 0] 11.6%	[1, 1, 1, 1, 1] 5.7%
[2, 2, 1, 0, 0] 7.5%	[1, 0, 4, 0, 0] 4.3%
[0, 0, 1, 2, 2] 6.6%	[0, 2, 0, 2, 1] 4.1%
[0, 0, 5, 0, 0] 6.4%	[0, 2, 0, 1, 2] 4.1%
[0, 1, 4, 0, 0] 5.0%	[2, 1, 0, 2, 0] 4.1%
[1, 0, 4, 0, 0] 4.5%	[0, 1, 4, 0, 0] 4.0%
[0, 0, 2, 2, 1] 4.5%	[2, 1, 0, 0, 2] 4.0%
[1, 1, 1, 1, 1] 4.5%	[0, 0, 4, 1, 0] 3.9%
…	…

In addition to this, as resources are added the attacker opts for actions that evenly distribute to one side of the bowtie. This can be seen in Table 5. Recall that this kind of strategy will mainly have resources allocated to the center but is more lenient when disbursing the extra nodes. This allows the attacker to get the benefit of both the non-centered and centered strategies, minimizing the oscillation. Previously we noted that for some large constant number of resources the game starts to resemble a game where real number allocations are allowed. Therefore, it is probable that for a real number-like representation of the bowtie topology, the attacker would opt for actions that most accurately represent the one-sided behavior. For some real-life distributed attack, it is highly probable that these actions will be used. The previously

mentioned nontrivial strategies are still notable as they can more accurately represent topology edge cases, specific attacks, and potentially represent subgame in a complex network.

In a bowtie network topology, a defender's first instinct likely will be to, more times than not, allocate all the resources to the central node. Our equilibrium strategy partially agreed with this idea but made many attempts to allocate resources to either side of the bowtie. This behavior suggests that a network with this limitation would want to add redundancy on either end of the central point.

Ring topology

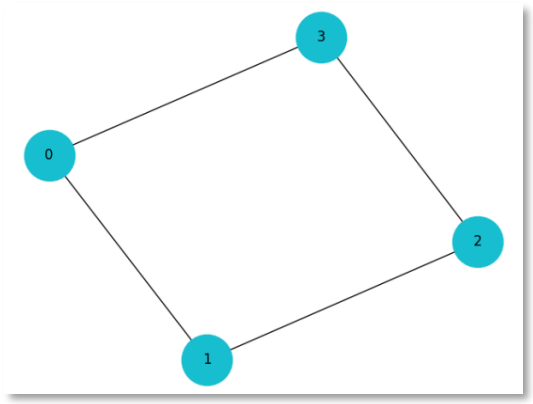

Figure 12: A ring topology

The ring topology expresses a symmetric network in which all of the nodes are equally valued in its initial state. Under normal game conditions, the attacker has always chosen to keep a strategy of mainly attacking non-adjacent nodes, as the attacker has the potential to completely disconnect the graph this way. As for the defender, when the agents are allocated fewer resources than there are nodes, they will choose to mainly defend directly against this attack by picking an inversely similar strategy. Intuitively one might think to allocate resources to nodes that are already connected. Though this turns out to yield less expected utility until more resources are added. Having fewer resources gives the defender the advantage, with a larger reward being determined on the 50/50 chance that the defender selected the correct non-adjacent nodes. Note that this preference for the non-adjacent is greatly favored for two resources and then only very slightly favored for three resources. The advantage diminishes quickly.

When the number of resources is greater than or equal to the number of nodes in the graph the defender changes their strategy to defending mainly adjacent nodes. This supports the notation that the attacker gains an advantage with more resources and the defender now must try and save some part of the network, not allowing the network to become completely

disconnected. Again, we can see the defender settling and prioritizing some subgraphs of the topology when more resources are added to the game. Either agent opts to never allocate all their resources to any one node whenever there are more than two resources. This kind of action is never desirable as the strategy for either player is to keep some connectivity in the network.

When changing the parameters of the game so that the attacker wins the ties at a node, a remarkably similar relationship can be observed. The attacker will always prioritize separating the ring topology by allocating mainly, or all, to non-adjacent nodes. The only major difference between the normal game configuration is when the number of resources is less than the number of nodes. In this case, the defender opts to mainly allocate their resources to one node. This is because when the attacker wins the tie, this is the greatest chance a defender has to stop the network from turning to two disjoint subgraphs. The defender chooses a few different actions for three resources but for two resources the defender only allocates to at most one node. This is because the attacker always places one resource on each non-adjacent node, therefore the only way the defender can get any utility is if the defender wins one of the nodes.

The ring topology showed a notable example of how the defender loses the advantage as resources increase. There is a clear point at four resources when the defender rarely places actions that would protect the whole network. The defender more often than not will allocate more resources to neighboring nodes to ensure some connectivity is left in the network.

Fully connected

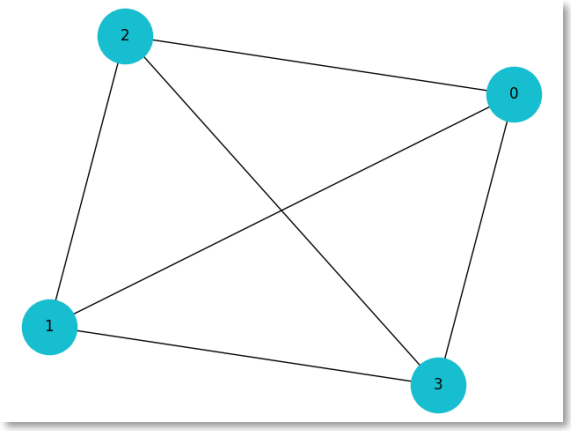

Figure 13: A fully connected topology

Similar to the ring topology, a fully connected topology expresses a symmetric network. Though a fully connected topology has this symmetric property in all states of the game. This means that if a node is lost, the resulting operational topology is a subgame of a smaller fully connected topology. Additionally, each node always holds the same value to each agent in the game as well as in the subgame produced by the attack. This would make a sequential version of this game easy to implement and study. This kind of game is well studied and may be possible to explore by modifying sequential models such as Gaun's model [13].

Due to the number of edges in this topology, many of the actions' probabilities ended up being very close to each other. All of the actions had equal probability to play their respective permutations due to the properties of this topology mentioned above. A simple example of this is if the player played all of their resources at any one node. That would be four different actions that are played equally often. Whichever agent wins in the case of a tie typically plays a more distributed strategy with their resources. They can play more distributed strategies more often because they have the advantage in more situations, allowing them the opportunity to try to win

more nodes. The loser of the tie mostly selects a few nodes to prioritize, usually putting around half of the resources on two nodes. The figure of half the resources derives from the design of the game. To any connection at all, two nodes must be connected. So, to have a chance of having any connections any action should have the intention of supporting at least two nodes. The players value these nodes highly and allocate up to half their total resources for one of them, in this topology. This kind of approach was observed in many other topologies with similar network conditions.

Limitations

Our model has limitations to the type of attacks or real-life network scenarios it can emulate. The running time complexity of the training algorithm makes it unreasonable for sizeable networks greater than 15 nodes. This is a limitation of many of the Colonel Blotto models. Typically Colonel Blotto models have a reasonable upper bound of around 20 to 30 nodes. We limit the game to only allowing for two players, but this does not remove the accuracy of a distributed denial-of-service attack. The simultaneous nature of the game is used as our focus is to get the pre-allocation of resources and resulting attack and defense strategy in a zero-knowledge spontaneous attack.

In addition to our limitations, there are several limitations that almost all game models face, such as the accuracy of the simulated game environment topology, resource type, and ease of resource allocation.

With the model's limitations, we were still able to produce results regarding predetermined configurations and how to maintain a robust system against severe system failure. The simple topologies modeled resemble networks currently deployed through many different systems.

Additionally, it is worth noting that it is computationally infeasible to calculate the expected utility past a certain number of resources with three, or any reasonable amount of, nodes. Having more than a few hundred nodes would not produce an output in any reasonable time. This leaves an open question on how a function grows. The two options are converging to some positive value or converging to zero. Assuming this pattern holds, it would seem possible that the integer gets so large that the value is similar to a real number representation. This is because an extremely large integer would quantify the model the same way some decimal

number does. The other possibility is that the action set can be so large that the defender simply cannot decide the best action to pick against so many attacks. This is a clear limitation of this kind of model due to the computational costs of the algorithm in place.

Chapter 5 Conclusion

Colonel Blotto game models are succinct ways of describing wireless network construction and DoS attack simulations. Finding Nash Equilibrium strategies for these models provides effective strategies for network administrators to use to mitigate the effects of denial-of-service attacks.

Other researchers have created Colonel Blotto models designed for wireless networks. Some of these models focus on cooperative games, others on multi-layered networks, and some sequential-styled blotto games. Our focus was on effective strategies in a simultaneous two-player topology-specific Blotto game. Though games like this have been modeled and studied, no work has been done on analysis of the strategies produced under specific topologies and with average node degree as the utility function.

We identified effective strategies that figurative defenders of some networks can use to ensure worst-case performance in case of catastrophic failure of the system. We found strategies for specific types of connections commonly found in all types of networks. The corresponding attacker strategy lets us report on malicious attackers' priorities. The Blotto model exposed strategies that were nontrivial and beneficial to identify for a robust system with parameters that cannot all be represented in a model. The behavior of certain network parameters was studied and analyzed for any real-life applicable network to use. The strategies found were equilibrium strategies.

Equilibrium strategies prevent the figurative "cat and mouse" game that some network defender plays with their advisories. A game in which the defender adapts to the new attacks is presented to them dynamically and after they have occurred. When a defender can consider the range of possible attacker options, they can form a mixed strategy that prevents an attacker from

exploiting a defense that always chooses the same approach. For example, in the bowtie topology, one may choose to defend the central node all the time when an equilibrium strategy shown in Table 5 would perform better. Furthermore, the most obvious strategies in a game are not always the most effective ones. An example of this can be found with the simple 1 resource 3-node line topology. Rather than defending the middle and losing the edges, there exists an equilibrium strategy that allows the defender to occasionally allocate resources to the edges.

Future work could include an improved training algorithm, a multi-agent model, an asymmetric resource game, lossy network connections, and a sequential game focusing on the same type of parameter and strategy analysis highlighted in this paper. The accumulation of past works, our results, and future work, undoubtedly will give insight into the types of approaches network administrators could use to guarantee effective network configurations strategies in order to mitigate the effectiveness of repetitive or spontaneous denial-of-service attacks.

References

[1] M. Labib, S. Ha, W. Saad, and J. H. Reed, "A Colonel Blotto game for anti-jamming in the Internet of Things," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, San Diego, CA, Dec. 2015, pp. 1–6.

[2] S. Guan, J. Wang, H. Yao, C. Jiang, Z. Han, and Y. Ren, "Colonel Blotto Games in Network Systems: Models, Strategies, and Applications," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 637–649, 2020. doi:10.1109/tnse.2019.2904530

[3] K. Leyton-Brown and Y. Shoham, "Chapters 1 - 3," in Essentials of game theory: a concise, multidisciplinary introduction, San Rafael, Calif: Morgan & Claypool Publishers, 2010, pp. 1–30.

[4] E. M. Shahrivar and S. Sundaram, "Multi-layer network formation via a Colonel Blotto game," *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2014, 10.1109/GlobalSIP.2014.7032237

[5] M. E. Nikoofal and J. Zhuang, "Robust Allocation of a Defensive Budget Considering an Attacker Private Information," Risk Analysis, vol. 32, no. 5, 2012. dio: 10.1111/j.1539-6924.2011.01702

[6] K. M. H. R., E. Panaousis, and G. Theodorakopoulos, Decision and game theory for security: 6th international conference, GameSec 2015 London, UK November 4-5, 2015 proceedings. Cham: Springer, 2015.

[7] M. Shubik and R. J. Weber, Systems Defense Games: Colonel Blotto, Command and Control. Ft. Belvoir, VI: Defense Technical Information Center, 1978.

[8] Z. E. Fuchs and P. P. Khargonekar, “A sequential Colonel Blotto game with a sensor network,” 2012 American Control Conference (ACC), 2012. dio: 10.1109/acc.2012.6315589

[9] Y. Wu, B. Wang, and K. Liu, “Optimal power allocation strategy against jamming attacks using the Colonel Blotto game,” in IEEE Global Telecommunications Conference, 2009, pp. 1–5.

[10] M. Hajimirsadeghi, G. Sridharan, W. Saad, and N. B. Mandayam, “Internetwork dynamic spectrum allocation via a Colonel Blotto game,” in Proc. IEEE Annu. Conf. Inf. Sci. Syst. (CISS), Princeton, NJ, USA, Mar. 2016, pp. 252–257.

[11] N. Namvar, W. Saad, N. Bahadori, and B. Kelley, “Jamming in the Internet of Things: A game-theoretic perspective,” in Proc. IEEE Glob. Commun. Conf. (GLOBECOM), Washington, DC, USA, Dec. 2016, pp. 1–6.

[12] B. Wang, Y. Wu, K. R. Liu, and T. C. Clancy, “An anti-jamming stochastic game for cognitive radio networks,” IEEE J. Sel. Areas Commun., vol. 29, no. 4, pp. 877–889, Apr. 2011.

[13] S. Guan, J. Wang, C. Jiang, Z. Han, Y. Ren, and A. Benslimane, "Colonel Blotto Game Aided Attack-Defense Analysis in Real-World Networks," 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 2018, pp. 1-6, doi: 10.1109/GLOCOM.2018.8647886.

[14] M. Guizani, A. Gouissem, K. Abualsaud, E. Yaacoub, and T. Khattab, "Combating Jamming Attacks in Multi-channel IoT Networks Using Game Theory," *2020 3rd International Conference on Information and Computer Technologies (ICICT)*, San Jose, CA, USA, 2020, pp. 469-474, doi: 10.1109/ICICT50521.2020.00081.

[15] A. Gouissem, K. Abualsaud, E. Yaacoub, T. Khattab and M. Guizani, "IoT Anti-Jamming Strategy Using Game Theory and Neural Network," 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 2020, pp. 770-776, doi: 10.1109/IWCMC48107.2020.9148376.

[16] S. Hart and A. Mas-Colell, "A simple adaptive procedure leading to correlated equilibrium," Econometrica, vol. 68, no. 5, pp. 1127–1150, 2000.

[17] Kaspersky, "DDoS attacks in Q3 grow by 24%, become more sophisticated," www.kaspersky.com, 08-Nov-2021.

[18] J. F. Nash, "Equilibrium points in N-person games," Proceedings of the National Academy of Sciences, vol. 36, no. 1, pp. 48–49, 1950.

[19] Radware, "Morris Worm," Radware. [Online]. Available: https://www.radware.com/security/ddos-knowledge-center/ddospedia/morris-worm/#:~:text=According%20to%20Morris%2C%20the%20purpose,connected%20to%20ARPANET%20in%201988.

Appendix

The source code for the model can be found at the link below

https://github.com/mattpost1700/DeceptionBasedTopologies

ACADEMIC VITA

Matthew Post

Education

M.S. Computer Science / The Pennsylvania State University, Capital College **January 2021 – May 2023**

B.S. Computer Science / The Pennsylvania State University, Schreyer Honors College **August 2018 – May 2022**

Awards/Engagement:

- Computer Science Outstanding Undergraduate Student Award 2022
- Student Marshal for School of Science Engineering and Technology
- Undergraduate thesis concentrating on AI driven network security
- Invited to the American Math Competition (AMS)
- New student orientation leader
- Russian Honors Society member
- Integrated undergraduate and graduate program
- Elected freshman and sophomore honors councilman
- College tutor leader

Relevant coursework: Graduate Algorithm Design and Analysis, Advanced Topics in DBMS, Formal Methods of Software Engineering, Operating Systems, Data Structures, Secure Programming, Compilers, Computer Organization and Architecture, Android App Programming, Software Engineering Design, Formal Languages, and a capstone project

Languages: Proficient in Russian
Coding Languages: Java, Python, C#, Kotlin, C++, and React

Experience

Software Developer Intern / Sho Technology Solutions **September 2020 – Present**

- In charge of design, development, and release of multiple projects simultaneously
- Developed an effective stock market trading algorithm and application
- Developed an image classification focused mobile application
- Completed all aspects of the software engineering lifecycle including documentation
- Communicated with clients for project design and scope
- Developing APIs, bug identification and fixing for other developers

Research Assistant / Florida International University **May 2021 – August 2021**

- Research was funded by the NSF for advanced secured sensor enabling technologies
- Focused on deception-based network security via game theoretic equilibrium strategies

Manager / Colarusso's Café, Clarks Summit, PA **July 2015 – January 2022**

- Oversees all elements of fiscal management related to balancing daily ledgers, dispensing tips to employees, and handling all cash transactions and deposits
- Hiring, interviewing, and training employees; received management position in 2017

Volunteer Firefighter, Lower Swatara Fire Department, PA **June 2021 – Present**