NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

PARTIAL INFORMATION COMMUNITY DETECTION IN A MULTILAYER NETWORK

by

Scott D. Warnke

June 2016

Thesis Advisor: Ralucca Gera
Second Reader: Gerry Baumgartner

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	06-17-2016	Master's Thesis 07-07-2014 to 06-17-2016

4. TITLE AND SUBTITLE

PARTIAL INFORMATION COMMUNITY DETECTION IN A MULTILAYER NETWORK

5. FUNDING NUMBERS

6. AUTHOR(S)

Scott D. Warnke

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Laboratory for Telecommunication Sciences

808 Greenmead Drive, College Park, MD 20740

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Identifying communities in a dark network is a potentially difficult task. The nature of dark networks, and their characteristic of concealing connections within the network, makes community detection an enterprise based on operations and decisions with only partial information. We take this concept of operation with only partial information, and extend it to our work by identifying communities within a dark network using only a single layer from the full multilayer network. Additionally, the concept of identification of terrorist networks within civilian populations is one of ever-increasing importance in our world today. We create a large multilayer synthetic network, and embed a known terrorist network in the larger synthetic network. We construct our synthetic network in a manner to ensure that our terrorist network is not unique, in order to make discovery of the terrorist network difficult. In this portion of our work we are concerned with identifying the entire terrorist network, not just a community within the terrorist network. We use known discovery algorithms to discover the terrorist network, and compare the results to modified algorithms introduced in this thesis and their ability to discover the terrorist network as quickly as possible.

14. SUBJECT TERMS	15. NUMBER OF PAGES
multilayer networks, partial information, network discovery, synthetic network construction, community detection, dark networks	131
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

PARTIAL INFORMATION COMMUNITY DETECTION IN A MULTILAYER NETWORK

Scott D. Warnke
Captain, United States Army
B.S., United States Military Academy, 2006

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
June 2016

Approved by: Ralucca Gera
Thesis Advisor

Gerry Baumgartner
Second Reader

Craig Rasmussen
Chair, Department of Applied Mathematics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Identifying communities in a dark network is a potentially difficult task. The nature of dark networks, and their characteristic of concealing connections within the network, makes community detection an enterprise based on operations and decisions with only partial information. We take this concept of operation with only partial information, and extend it to our work by identifying communities within a dark network using only a single layer from the full multilayer network. Additionally, the concept of identification of terrorist networks within civilian populations is one of ever-increasing importance in our world today. We create a large multilayer synthetic network, and embed a known terrorist network in the larger synthetic network. We construct our synthetic network in a manner to ensure that our terrorist network is not unique, in order to make discovery of the terrorist network difficult. In this portion of our work we are concerned with identifying the entire terrorist network, not just a community within the terrorist network. We use known discovery algorithms to discover the terrorist network, and compare the results to modified algorithms introduced in this thesis and their ability to discover the terrorist network as quickly as possible.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

BA	Barabási-Albert
CMSH	Cavalrymen Make Stuff Happen
DoD	Department of Defense
ER	Erdős-Rényi
MUDD	Multilayer Undiscovered Degree
NPS	Naval Postgraduate School
NMI	Normalized Mutual Information
NCO	Non-Commissioned Officer
RW	Random Walk
UDD	Undiscovered Degree
U.S.	United States
USG	United States government

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

The purpose of this thesis is to use partial information to identify communities within a terrorist network. We look at the Noordin Top Terrorist Network as an example of a terrorist network. This was a terrorist network that existed in Indonesia until 2009 when Noordin Mohammad Top was killed. Noordin Top was a coordinator between five separate terrorist organizations that operated in Indonesia, and it was his connections to members of these separate groups that created the Noordin Top Network. The Noordin Top Network was a multilayer network consisting of 14 individual layers that represent particular relation types between members of the terrorist organization. There are 139 terrorists represented in these 14 layers. A terrorist network is one example of dark network that attempts to conceal information about the network in order to remain clandestine. Since dark networks are networks that do not have all the possible information available for researchers to study, we explore the idea of using partial information to detect communities within these dark networks.

We use two methods to represent partial information in our terrorist network. The first method is to use only a single layer of the 14 layers available, and perform community detection on this single layer. We identify a target community that is present in the the single layer of the network as well as the full 14-layer network. After successfully identifying the target community using a single layer, we use a single layer of the network to attempt to identify the remaining communities in the network that are present in the single layer. We use all 14 layers individually to determine the community structure of the full multilayer network.

The next method of partial information representation we explore is network discovery. We discover our terrorist network using network discovery algorithms. We discover our terrorist network, after it has been embedded in a larger non-terrorist network. We will start with no information of the network, and attempt to discover the terrorists within a larger civilian population. This requires the creation of a large synthetic network. We create a large synthetic network consisting of over 7,500 vertices and nearly 200,000 edges that exists in 10 layers. We create this network using the Noordin Top Network, a random graph with inherent structure, a completely random graph, and real world social network

data. We create this synthetic network with the goal of making the Noordin Top Network difficult to detect. We do not want our terrorist network to stand out as unique, making it subsequently easier to discover. Of the ten layers in the synthetic network, three of them are based on the data from the Noordin Top Network, and we refer to these as our target layers. While terrorist and non-terrorist vertices exist in all ten layers, our target layers retain the original terrorist information, with non-terrorists attached to obscure the footprint of the Noordin Top Network.

We use four discovery algorithms to discover the terrorist network. These algorithms all use monitors, which discover a vertex and all neighbors of a vertex. One of these discovered neighbors is then selected for the placement of the next monitor in the discovery process. All four algorithms use these monitors to discover the network, but the way in which the placement of the next monitor is selected is how the algorithms differ. The first two algorithms we use are Random Walk and Undiscovered Degree (UDD). Random Walk discovers vertices in the network randomly with no bias to the amount of information a monitor will discover. Since the placement of each subsequent monitor is random, Random Walk serves as a lower bound for the rate at which we can discover terrorists within the synthetic network. UDD is an algorithm that attempts to discover the network by placing the monitor on a vertex that will discover the most previously undiscovered vertices. UDD evaluates each neighbor, and determines which neighbor has the most previously undiscovered neighbors, and places a monitor there. This has been shown to be a very effective algorithm to use for network discovery. We introduce two new algorithms that are modifications of UDD. The first modified algorithm is Multilayer Undiscovered Degree (MUDD), which is a modification of UDD allowing the algorithm to take advantage of the multilayer structure of the synthetic network. When MUDD discovers a non-terrorist vertex, it continues placing monitors with the same criteria used by UDD. When MUDD discovers a terrorist vertex, it selects the placement of the next monitor based on information in the previously selected target layers. This does not guarantee the subsequent identification of a terrorist vertex, but increases the chance of selecting a terrorist vertex, and hopefully discovering the terrorist network faster. The second modified algorithm implemented is Probabilistic MUDD, which is an attempt to degrade the perfect information available for MUDD to use in the selection of the placement of the next monitor. When we create the synthetic network, we assign a random probability between 0 and 1, from a nor-

mal distribution to each individual vertex in the synthetic network. When MUDD evaluates the number of undiscovered vertices a neighbor will discover, this count of undiscovered neighbors is multiplied by the probability assigned to that neighbor. This has the effect of degrading the information that Probabilistic MUDD uses to determine the placement of the next monitor. We implement this algorithm in the attempt to better simulate real life, in which it is unlikely that we will have all of the information available to determine where the next monitor is placed.

When we used a single layer of the network, we were able to identify a target community. This identification was limited to being successful only if the target community was highly unique. Furthermore, we found that any other conclusions about the community structure were limited by the amount of information in the individual layer being used.

When we analyzed partial information through network discovery, we generated ten synthetic networks, and ran all four discovery algorithms through all ten graphs, and compared the results of the network discovery. We were primarily concerned with the rate at which terrorist vertices were discovered by each algorithm. When we looked at the results, we saw that UDD, MUDD, and Probabilistic MUDD all out preformed Random Walk. Random Walk and UDD both performed as expected, namely that Random Walk discovered vertices randomly, and UDD discovered large hubs within the network quickly. MUDD and Probabilistic MUDD outperformed UDD in both rate of terrorist vertex discovery and amount of the terrorist network discovered. There was a clear advantage to using MUDD or Probabilistic MUDD instead of UDD. Surprisingly, Probabilistic MUDD performed as well as MUDD, despite the degradation of the information used by Probabilistic MUDD.

We were able to demonstrate that it is possible to identify a target community from a multilayer network, using only one layer of the network. We demonstrated that any conclusions about the total community structure of the full network, are limited by the amount of information contained within that single layer. We were successful in creating a large synthetic network that had a much smaller terrorist network embedded within. We were able to modify algorithms to take advantage of the multilayer structure of the synthetic network, and discover terrorist vertices faster than algorithms that do not account for the multilayer network.

We feel that while we were successful in the goals of this work, there is still much more that can be done in the future that is related to this work.

Acknowledgments

First I would like to acknowledge God. It is only through His grace that I am able to accomplish anything. I am truly blessed to know Him, and thankful every day for the gifts He gives.

My thesis advisor Ralucca Gera has been a fantastic mentor through this entire process. She has helped guide my research on a daily basis. Her knowledge, expertise, and patience are proof of her professionalism. It has been a true joy to learn from her and work with her on this thesis. Without her, none of this would have been possible.

Gerry Baumgartner has served as both my second reader and sponsor for this research. His insight and analysis have been incredibly beneficial and have made this work both rewarding and successful. His ability to pose thought-provoking questions during the research process have truly pushed me, and inspired this thesis greatly.

Craig Rasmussen, the chair of the Applied Mathematics Department at the Naval Postgraduate School, has been a great help to me. His review and critique of my work has made it a better, and more professional document. I truly appreciate his hard work.

I would also like to thank Riqui Schwamm and Karl Schmitt. I absolutely would not have been able to accomplish any of this work without their assistance. The hard work and review of complicated code made this thesis possible. I would not be able to do any of this without their expertise and analysis.

I have been lucky enough to have learned from the wonderful faculty of Applied Mathematics Department at the Naval Postgraduate School for the last two years. I have learned from them each and every day. They are the reason I have been able to accomplish this work. There are many of them that have gone far beyond what is required of a professor to assist me and other students. They have fostered an amazing environment within the department; that is the primary reason for the success of so many of their students.

My fellow students, especially Karoline Hood and Ryan Miller: Thank you for the help with everything for the past two years. From our first classes together, until the

completion of this thesis, you have helped me tremendously every step of the way.

I have have spent the first ten years of my career working with some of the best soldiers, NCOs, and officers the Army has to offer. They have taught, developed, mentored, and inspired me constantly. It is an honor so serve next to them, and I will be indebted to them forever.

Finally, and most importantly, I want to thank and acknowledge my amazing family. I have wonderful parents who made me the person I am today. I grew up with the best brother a person could hope to have. I am lucky enough to call Lauren my wife. She is the most beautiful person I know, and raising our children together brings new joys to my life every day. She does so much for our family. I would be lost without her.

CHAPTER 1:
Introduction and Motivation

1.1 Introduction

Network Science is a new and growing field. A network can be used to represent practically everything from the Internet to a human brain. Networks can be as simple or complex as one desires. We can also model terrorist organizations using a network. Perhaps through continued study and modeling of terrorist or enemy behavior as networks, we can be more effective in disrupting enemy or terrorist networks. There is a lot of research that attempts to model the current operating environment of the U.S. military as a network. One example is the work in [1], modeling the members of Afghan villages to improve the military's non-lethal targeting, or influence, in the region.

Networks can be very large, and the discovery of the network can be a daunting task. If we want to discover the network, discover the individuals and who they are connected to, there are a few ways we can go about accomplishing this task. Obviously, if we have all of the network to start with, there is no need to discover the network. We can simply look at the network, and see which terrorists work or communicate with each other. But when dealing with drug cartels, terrorist, or enemy combatant networks, this is not usually the case. These can be looked at as dark networks [2]. In fact, these networks will actively work to keep their structure hidden or secret from outsiders [3]. It might be very difficult to imagine the enemy making their network difficult to discover. One need only look at the example of the recent terrorist attack in San Bernardino, California, in December 2015. During the investigation, the FBI attempted to access the information on one of the terrorists' locked iPhone [4]. The FBI wanted to be able to learn more about the terrorist's network. Who supported them? Who helped supply them? Were the terrorists being directed by some terrorist organization? These are all questions the FBI wanted to be able to answer. But, answering this question was very expensive. One need only look at the vast amount of press coverage regarding this [5]–[7], to see that this was costing the FBI. Besides this cost, there would be a higher cost that the population would pay, as Apple said. That is, since once

such technology was invented, there is always a way for it to find its way into the wrong hands and be maliciously used.

Network discovery might be very expensive or difficult. For this reason, we desire to discover the network as efficiently as possible. We might try to analyze one of these dark networks, yet we are given partial information. That might be due to the fact that it is expensive to discover the network, or because it is difficult or infeasible to have a complete picture of the network. We still desire the ability to perform some level of analysis, even if we only have some of the information on hand. In the military, we do not wait around to have a complete picture of the enemy situation. We will execute the mission with the information we have on hand. This information might be updated later on, and then we can alter our original plan.

1.2 Motivation

We will motivate this thesis by starting in Afghanistan. The following story is a real experience from Afghanistan, it is not unique to the author, but is something that many U.S. military members experienced.

Place yourself on patrol in Southern Afghanistan. It is March 2010, and your Army Battalion is located in Helmand Province. On this particular day, you are on a dismounted patrol. You and about 30 other U.S. Soldiers are walking along a large irrigation canal. You are located in a large city, but this portion of the city is largely agrarian. There are wheat and poppy fields all around you. Along the various foot paths there are small hamlets. They aren't quite large enough to be considered villages, but there are a handful of mud adobe homes grouped together every few hundred meters. If you had to guess, in each hamlet there were a few dozen men, women, and children. After patrolling for a few hours, you begin to notice something. As you approach each hamlet, some of the chimneys begin to smoke. Not every chimney, but at least one in each hamlet. As you leave a hamlet, the chimneys stop smoking. And a few chimneys in the next hamlet you are approaching begin to smoke as well. It does not take long to realize what is happening. Somebody is signaling your location to everyone in the area. There are hundreds of people in the area, and some undoubtedly are Taliban. Others may not be fighters, but they might be sympathizers. There are certainly others still that do not support the Taliban. Yet someone is signaling

your location. They are communicating, through the use of smoke, to each other exactly where you and your Soldiers are located. There is potentially a very large population in this area, but only some of them are enemy. The majority of the population are peaceful farmers.

If we look at this situation, there is a large network in this area. All the members of the population are connected. They might live in the same hamlet, or attend the same mosque, or shop at the same markets. Some will communicate with others, and some are even family members. The more we look at this, the network of the area grows in size and complexity. We know that the Taliban is communicating through the smoke signals, but we can not simply capture the residents of the homes with chimneys that smoke at our approach. The residents might be sympathetic to the U.S., but are under Taliban pressure to light a fire when we approach. In the U.S. Army, the Cavalry has a saying: "Cavalrymen Make Stuff Happen (CMSH)." Cavalrymen traditionally operate in front of friendly lines. They are forced to make do, with limited time, intelligence, and resources. We have some information, the smoke signals used as communication, we need to make stuff happen with what we have. We do not have the luxury of waiting to further develop the situation. We need to use this partial information about the network to figure out something about the structure of enemy's network.

This situation described above was one that many Soldiers and Marines faced in Afghanistan. It is not unique only to the identification of Taliban networks in Afghanistan. Perhaps our partial information is radio traffic the enemy uses. Our partial information could be email traffic that drug smugglers use in their operations. Regardless of the network, or the partial information we have, there is a desire to analyze the network with the information on hand. We have some of the information, and it is time to Make Stuff Happen.

1.3 Define the Problem

Given a terrorist network, our first goal is to identify a specific target terrorist community. We will limit ourselves the partial information to find this target community. This will consist of the smoke signals we saw on our dismounted patrol. Using just this partial information, we will try to identify the target community in the network.

The second part of the research is an extension of the first. We now consider a discovery method to identify the terrorists within a much larger network. We will construct this synthetic network, and ensure it is composed of both terrorists and non-terrorists, and introduce enough randomness, structure, and copies of different layers of the terrorist network. This is done so that the terrorist network will not easily stand out if embedded in a random network. Once we have constructed this synthetic network, we will then use several algorithms to discover the terrorists in the network. We will compare previously established discovery algorithms, and modified algorithms that attempt to take advantage of the multilayered structure of our networks.

CHAPTER 2:
Background

The field of Network Science is a broad field with a lexicon and jargon of its own. In this chapter, we will introduce many of the definitions, terms, concepts, and equations that will be used through out this thesis. This chapter will allow us to accurately describe the networks used in this work, and provide detail in the methodology used to research the specifics of the networks in subsequent chapters.

2.1 Graph Theory and Network Science

In its simplest form, a graph is defined as an ordered pair of sets. The first is a set of vertices, called the vertex set, and the second is a set of edges, called the edge set [8].

According to Gary Chartrand and Ping Zhang [8], a graph is formally defined as follows:

Definition 2.1.1 *A graph G consists of a finite nonempty set V of objects called vertices (the singular is vertex) and a set E of 2-element subsets of V called edges.*

The edge set is composed of elements that link two vertices together. A graph is very powerful tool for studying relational data, and displaying the information in the data. The primary difference between a network and a graph is that networks traditionally have labels and information/metadata/weights associated with vertices and/or edges. The vertices are uniquely labeled, and the edges are representations of connections between the vertices [9], and the information associated with them is given by the attributes that the data contains.

2.1.1 Complex Networks, Monoplex and Multiplex networks

The field of Network Science is predominately focused on studying large or complex networks that model the world we live in. These networks are often too large to study visually; they contain so much information looking at an image can be misleading. Mark Newman [9] highlights as an example the Internet, which is ever changing and growing.

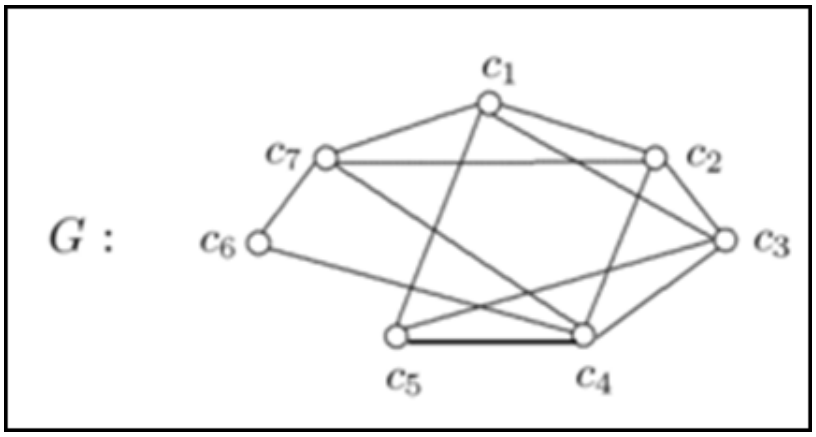


Figure 2.1: This is a simple graph, G. It consists of edges and vertices. Source: [8].

Further more, despite the fact that the Internet is built by individual humans without centralized control, not even experts have a complete picture of its structure. Figure 2.2 is an image of a visualization of the Internet. What makes complex network problems interesting is that even if this image accurately depicted the complete structure of the Internet, the Internet would change before the image could be produced [10].

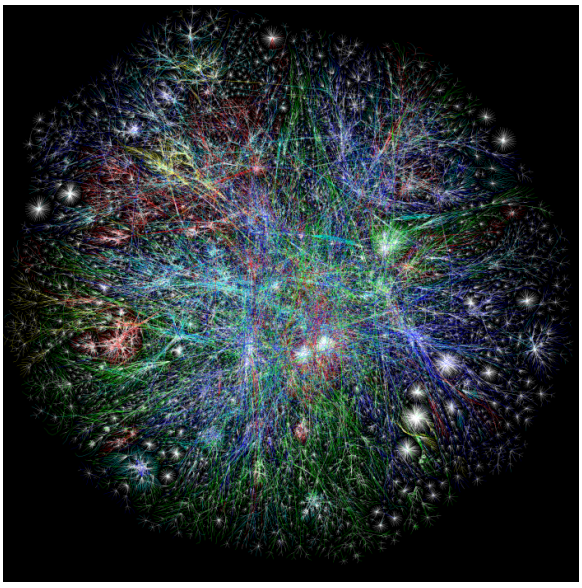


Figure 2.2: Visual representation of the Internet from November 2003. It consists of over 5 million edges. Source: [11].

We will be discussing social networks, where individuals are represented as vertices and

connections between two individuals are represented as an edge between two vertices. This is a standard way to represent a social network, as illustrated in Newman's text [9]. These networks show connections between actors. But connections between actors are seldom identical or equivalent. In practice, these connections represent relations between actors that can be of multiple types and carry different weights.

A Multiplex is a network that shows multiple types of connections between vertices, based on the attributes associated with that relationship (in the case of communication, the relationship can be email exchange, or SMS text or phone calls). As described by De Domenico et al. [12], use of a Multiplex allows for more information to be preserved in the representation of the network, and therefore a more accurate depiction of the network is given. We will be using a Multiplex to represent different types of connections between terrorists. This is not limited to multiple connections of a single type, but multiple types of connections. If terrorists have communicated with each other, there will be an edge between those two vertices in the communication layer. If the same two terrorists have trained together, there will be a separate edge between the same two vertices in the training layer. In the Multiplex network of this example, these two vertices would have have two edges between them. One edge would represent the connection from the communication layer, the other would represent the connection from the training layer.

The Multiplex networks are useful for displaying and maintaining information about the layers of the network. Another alternative is a Monoplex, which is a single layered network [12]. Monoplex in this thesis will capture how many edges exist between each pair of nodes without differentiating the types of edges. We can take a Multiplex with its multiple edges representing multiple types of connections, and condense these into a collection of parallel edges or a single edge with weight. The more connections in the Multiplex, the higher the weight in the Monoplex. Using a Monoplex is sometimes desired because there are some algorithms that will not perform on a Multiplex. Conversely, other algorithms perform better on a Multiplex since they capture the details of the data through the multilayer.

It is important to provide clarity about what these terms will represent in this work. A Multiplex network will represent different types of connections between terrorists. A Monoplex network will be the representation of all connections between terrorists, but we will represent them as a network with weighted edges. When we use a Monoplex, we will combine

the multiple edges between two vertices in the Multiplex as a single weighted edge between the vertices. Additionally, we will label layers of the terrorist network. Layers correspond to a representation of all of the connections of a single type and the nodes associated with them. For example, we will refer to the communication layer. This layer will have connections representing all of the terrorists that communicate pairwise. In general, there could be weight associated with the strength or frequency of the relation. In our data, there is no weight associated with the frequency or type of communication, simply an edge representing that two terrorists have that particular relationship.

2.1.2 Dark Networks

Specific to this work, we will be analyzing terrorist networks. Terrorist networks are a special type of social network [13]. The network we are focusing on is made up of people and the connections between them, the hallmark of a traditional social network. Terrorist networks, however, are a special type of social network, a dark network. A dark network is network that is illegal and covert [2]. Since dark networks are illegal and covert, members of the network are actively trying to hide information about the network. It is this deceptive nature that makes information about the network difficult to obtain and presents an incomplete network at best. If the network were not dark, it might be easy to obtain complete the information necessary for comprehensive analysis on the network. These captured dark networks are usually quite sparse. Furthermore,as highlighted in [2], even the connections that do exist are often infrequently used. For example, two members of a network might coordinate or train together, creating an edge in the network. Subsequently, these individuals might not coordinate or train together for a long time, essentially allowing this edge in the network to go dormant. This makes it difficult to analyze the network, since we might not be capturing the information when the relationship is active, thus missing it completely. In Baker and Faulkner's work [13], they show that dark networks will attempt to maximize concealment even at the expense of efficiency.

The concept that terrorist networks are covert and attempt to conceal information may seem intuitive, and not altogether important. Yet the fact that they are covert makes analysis of a dark network particularly difficult [2]. As Raab and Milward [2] point out, trying to draw overarching conclusions about the structure of all dark networks is a difficult and probably

fruitless task. However, there are some techniques that will be highlighted in subsequent chapters that do allow analysis of dark networks.

2.2 Network Metrics and Comparison

The size and complexity of most networks make analyzing them visually difficult if not impossible. If one wants to analyze a graph, sometimes it is simple enough to just visually look at the graph in order the analyze its structure. In doing this, there is much that can be learned about the graph. For example, we present the topic of isomorphism. For a small enough graph, one way to determine one graph is isomorphic to another, is to visually show it or identify the node bijection that preserves adjacencies. An example of this is shown in [8]. In Figure 2.3, we see an example of two graphs that are isomorphic.

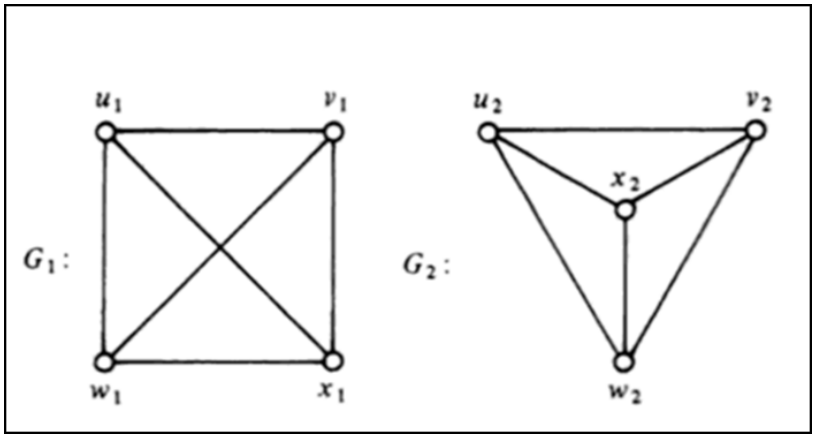


Figure 2.3: Graph G_1 and G_2 are both graph with 4 vertices and 6 edges. Furthermore, these two graphs are isomorphic to each other. Source: [8].

According to [8], Graph Isomorphism is formally defined as follows:

Definition 2.2.1 *Two (labeled) graphs G and H are isomorphic (have the same structure) if there exists a one-to-one correspondence ϕ from $V(G)$ to $V(H)$ such that $uv \in E(G)$ if and only if $\phi(u)\phi(v) \in E(H)$.*

This concept is simple enough with graphs the size of what is displayed in Figure 2.3. It is impossible, however, to visually analyze networks on larger scales. In fact, seemingly

simple concepts like isomorphism can be very difficult to extend from Graph Theory to Complex Networks. Isomorphism is such a difficult problem, that there is no polynomial time algorithm for the graph isomorphism decision problem [14].

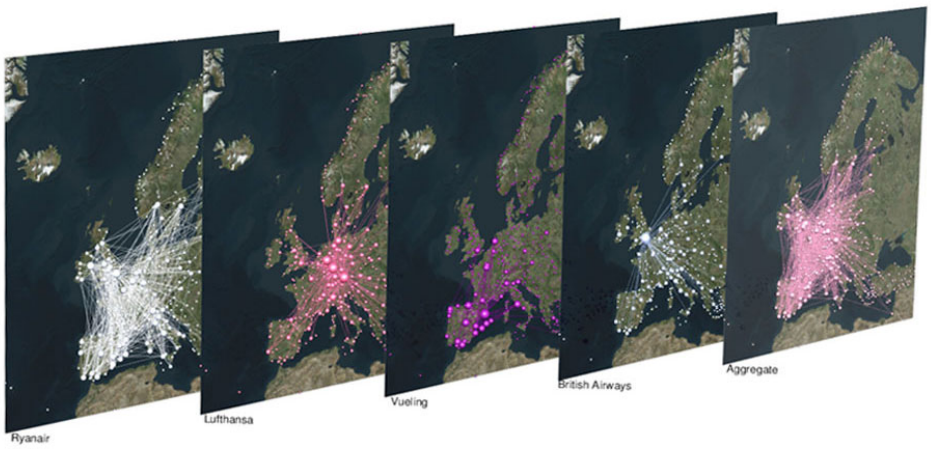


Figure 2.4: Visual representation of various European Airlines, and the airports they serve. Source: [15].

The image in Figure 2.4 is a large and complex network. Clearly, we require more than just visual cues to evaluate networks of interest in this work. We will now discuss and define some of the network metrics, allowing us to better analyze large complex networks.

2.2.1 Degree Sequence

As mentioned, some networks are very large and it is difficult to conduct analysis of these networks visually. One of the frequently used techniques to summarize information about the graph is to look at the degree sequence of the the network. This is simply a list, usually ordered, of the degrees of all of the vertices in the network. While it doesn't capture the network completely, it summarizes edge information. Another way to summarize this for much larger graphs is with the degree distribution. The degree distribution is a histogram

or other visual depiction of the degree sequence [16]. An example of a degree distribution is shown in Figure 2.5.

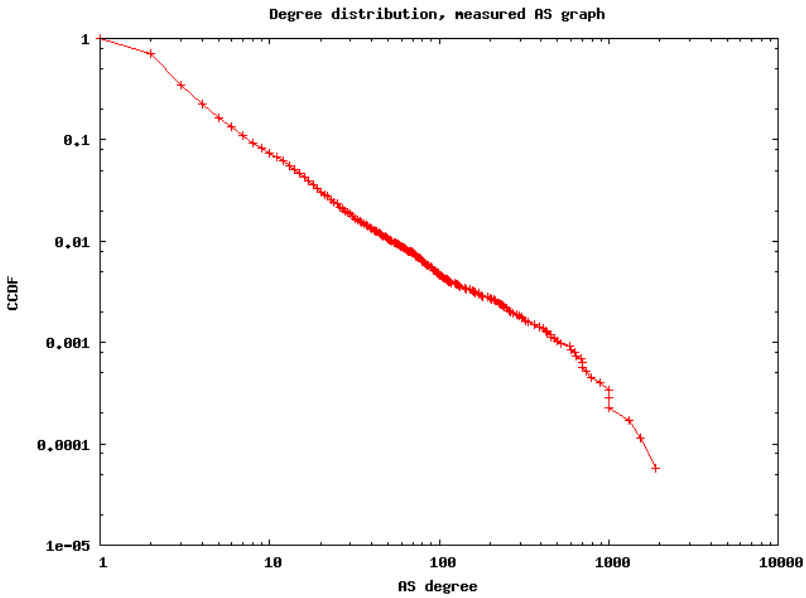


Figure 2.5: Degree distribution of the CAIDA network. The CAIDA network is a network of Autonomous Systems (AS) that make up part of infrastructure of the Internet. This distribution displays the degree of the vertices on the x axis, and the number of vertices on the y axis. These axises are both logarithmic plots, giving the degree distribution a linear appearance. This network displays the power law property. Source: [17].

This does not show the vertices uniquely, but it does allow us to observe a great deal about the network. Of particular interest to us is the fact that most social networks follow a power-law distribution. This power-law structure is noted as scale free [16]. As stated, the plot in Figure 2.5 is not of a social network, so the power-law property is not unique to social networks. The power-law property is however, something we would expect to see in a large social network. This quality might not be present in smaller networks. Social networks have many vertices with a low degree, and a few vertices that act as hubs with high degree.

2.2.2 Subgraphs

Due to the large size and complexity of most networks, sometimes it is convenient to only analyze portions of a network. These smaller portions of the full network are called sub-

graphs [8]. Gary Chartrand and Ping Zhang formally define subgraph as:

Definition 2.2.2 *A graph H is called a subgraph of a graph G, written $H \subseteq G$ if $V(H) \subseteq V(G)$ and $(H) \subseteq E(G)$. [8]*

In definition 2.2.2 $V(G)$ and $E(G)$ are the vertex and edge sets of graph G, and $V(H)$ and $E(H)$ are the vertex and edge sets of graph H. When we analyze the subgraph of the full network, there are many advantages. Some of these advantages are that we potentially have the ability to analyze the subgraph visually. Additionally, another advantage is that we can compare different subgraphs of the network to each other, and in doing so gain potentially important insight about different parts of our dark network.

2.2.3 Clustering Coefficient

The first metric we will discuss is the clustering coefficient. Newman [9] defines the clustering coefficient as:

Definition 2.2.3 *the average probability that two neighbors of a vertex are themselves neighbors.*

There are two ways to calculate the clustering coefficient. The first is the local clustering coefficient. The local clustering coefficient is a measure of how many pairs of neighbors of a given vertex are themselves adjacent, out of how many could be adjacent. The numerator can be thought of as counting triangles about a given vertex. The equation for calculating the local clustering coefficient is shown in Equation 2.1.

$$C_i = \frac{\textit{(number of pairs of neighbors of i that are connected)}}{\textit{(number of pairs of neighbors of i)}}. \tag{2.1}$$

While this is an effective method for determining locally how connected the neighborhood of a vertex is, it does not evaluate the entire network. For that reason, there is a generalization of the local clustering coefficient.

In his book Newman also describes how to calculate such a generalization, sometimes calle the global clustering coefficient. The equation is shown in Equation 2.2.

$$C = \frac{(number\ of\ triangles) \cdot 3}{(number\ of\ connected\ triples)}. \tag{2.2}$$

This (global) clustering coefficient for the full network is not limited to evaluation of a single vertex. It is a single value for the entire network. We will use this clustering coefficient, and any subsequent calculation of the clustering coefficient using Equation 2.2.

This metric is important to us, because social networks have been shown to have a higher clustering coefficient than random networks [9]. Newman discusses that the clustering coefficient of the given network is sometimes compared to the one of a random network with an identical degree sequence. The edges in this network occur randomly, creating a network comparable to the network being measured. If the network in question has higher clustering than the random network, the clustering coefficient shows that there is a higher density of triangles present [9]. This suggests that portions of the network are close groups of individuals. These close groups might be an indication of community structure, a concept we will discuss shortly.

2.2.4 Density

Another metric we will use is the density. A compete graph is a network in which every node is adjacent to every other node in the network. This represents the densest possible network for any given network size, and represents an upper bound for the density value in a network. Conversely, the least dense graph possible of a given network size, would be a network without any edges. This is trivial, but it does represent a lower bound for density measure. Newman formally defines density as:

Definition 2.2.4 *The maximum possible number of edges in a simple graph (i.e., one with no multiedges or self-edges) is* $\binom{n}{2} = \frac{1}{2}n(n-1)$. *The density of a graph is the fraction of these edges that are actually present.*

The density is a measure that shows where a network falls below the maximum possible number of edges [9]. Newman gives the equation for density as:

$$\rho = \frac{m}{\binom{n}{2}} = \frac{2m}{n(n-1)} \tag{2.3}$$

In this equation, ρ is the density of the network, m represents the number of edges, n represents the number of nodes [9].

When evaluating the density of a network, this formula will always return a value between 0 and 1. Values closer to one indicate a denser graph. Values that are closer to zero indicate a more sparse network. We are analyzing dark networks, so based on the dark network literature we would anticipate these networks to not be very dense. Again, it will be difficult to compare our networks to others, because of the nature of dark networks. But there is potential to compare subgraphs within a network. By doing this, it is possible to find portions of these dark networks that are more interconnected.

2.2.5 Clique

We have been discussing subgraphs of a network, and how they can be helpful in analyzing the network. Newman defines a clique as:

Definition 2.2.5 *a maximal subset of the vertices in an undirected network such that every member of the set is connected by an edged to every other [9].*

This means a clique is a subgraph of vertices that are adjacent to every vertex in the subgraph. Newman explains the cliques in social networks indicate highly cohesive subgroups. In these cliques, it is likely that all of the members of the clique know and/or interact with each other. Further, since the definition of a clique requires that every vertex be connected to every other vertex, this definition can be relaxed. Dense subgraphs can be said to be a relaxation of a clique [9], which is very useful to our research. Since we are dealing with dark networks, it is unlikely that we have all of the information. Therefore, an assumption is that we do not have all of the edges between the existing nodes in our networks. With this understanding of the network, it is unlikely we will encounter any true

cliques. We may, however, encounter clique like subgraphs. And with this, we can draw the reasonable conclusion that all members of the clique-like subgraph know each other and participate in activities together.

2.2.6 Community

When analyzing a social or dark network, the association among vertices can lead to interesting insights. In order to determine the associations that are the most naturally occurring, we will look at subdivisions of the network so its sets are densely connected internally and sparsely connected externally. These subdivisions are referred to as communities [9]. Lancichinetti formally defines a community as:

Definition 2.2.6 *groups of nodes, called communities or modules, with many links connecting nodes of the same group and comparatively few links joining nodes of different groups [18].*

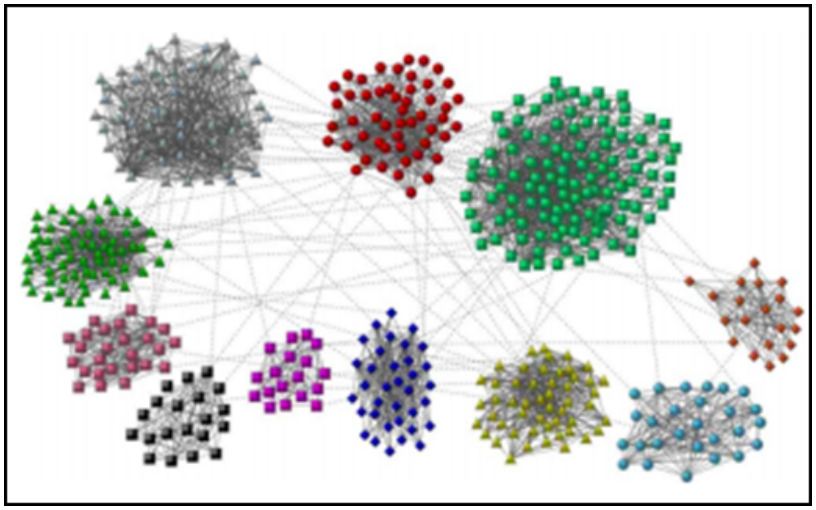


Figure 2.6: Illustration of community structure. There are more connections within the colored groups, than between the colored groups. Source: [18].

Simply stated, when subdividing a network into communities, we want there to be many edges within a community, and few edges outside of the community. This would mean that all the vertices in a community are more closely related to each other than they are to vertices outside of the community. Community subdivisions can be done in multiple ways,

but we will concern ourselves only with subdivisions that do not allow a vertex to be in more than one community, namely a partition of the vertex set into communities. Some community detection algorithms allow for vertices to be in more than one community; we do not examine those detection algorithms here.

The Louvain Method is an optimization algorithm for community detection, which tries to optimize modularity defined in Subsection 2.2.7. The Louvain Method has two phases. In the first phase, the Louvain Method uses a "fast-greedy" approach. This places each vertex in its own community. Then, one at a time, the neighbors of communities are merged with that community, and the modularity score is recomputed and compared. If higher than before, the node gets added to that community; if not, the node is rejected. This is performed till modularity doesn't increase anymore. In the second phase, the communities found based upon the scores of the first phase are merged into community-vertices. This new network is then processed through the same algorithm till that modularity is stable, and the nodes of the original network are partitioned according to the communities found in the second step [19]. The original work introducing the Louvain Method referred to it as the "local greedy approach," and highlighted its speed and accuracy [20]. This is the algorithm implemented by the "NetworkX" library in Python, the Igraph package in R, and Gephi, and it is what we will use in this work.

We expect the Louvain Method to return modularity scores between -1 and 1. A random graph would have a modularity of 0. Any positive modularity results indicate community structure that is more likely than random. Negative scores indicate community structure that is less likely to occur than a random graph.

2.2.7 Modularity

In the absence of ground truth of the communities on network, researchers have used modularity. The modularity measures the tendency of vertices with similar attributes to connect. If there are more edges between vertices, the community is said to have high modularity [9]. The underlying assumption is that similar nodes tend to create adjacencies, thus forming the communities that modularity measures. For a particular partition into communities, modularity can be thought of as how strong that particular community structure is in a network. Newman gives the equation used to calculate modularity in Equation 2.4:

$$Q = \frac{1}{2m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})\delta(c_i c_j). \tag{2.4}$$

where m is the number of edges, δ is the Kronecker delta between the vertices in class c_i and in class c_j, A_{ij} are the entries of the adjacency matrix associated with the network, and k_i and k_j are the degrees of vertices i and j [9]. Values for Q will be between -1 and 1. Newman details that values above zero indicate a positive assortative mixing in the network, while negative values indicate a disassortative mixing [9].

2.2.8 Cluster Adequacy

Modularity provides a powerful metric to measure a given community structure of networks. UCINET introduces the concept of Cluster Adequacy [21]. Cluster adequacy is described and used by Mathais Seimes in his work with Network Based Taxonomy of Law Systems [22]. Cluster adequacy, $Q^{'}$, is a way to normalize the modularity of a network. This is done by dividing the measured modularity by the theoretically best modularity for a give number of communities in a network. The equation for cluster adequacy is given by:

$$Q^{'} = \frac{Q}{1 - \frac{1}{m}}, \tag{2.5}$$

where Q is the measured modularity and m is the number of communities detected [22]. Using this equation, the ideal modularity would occur if the number of communities found in the network were to equally partition the vertices. For example consider a network of 10 vertices. If modularity determined there were two communities in the network, the ideal community structure would consist of five vertices in each community, returning a denominator in equation for cluster adequacy of 0.5. If the measured modularity, Q, was also 0.5, the cluster adequacy, $Q^{'}$, would be 1.

Everton recommends the use of Cluster adequacy when qualifying a community detection [23]. Cluster adequacy serves as a way to attempt to normalize modularity. This makes the assumption that an ideal partition is an equal partition.

2.2.9 Normalized Mutual Information

The Louvain method provides us with community structure, and with modularity we can see how good our partition is. If we need to compare our partition to something, what do we compare it to? In some cases, we have a ground truth, a real community structure, available for use. When ground truth is available, we should compare our partition to the ground truth using Normalized Mutual Information (NMI). NMI compares two partitions by comparing the communities of the two partitions of the same network. The equation used to calculate NMI is

$$\mathrm{NMI}(P^a, P^b) = \frac{-2\sum_{i=1}^{k_a}\sum_{j=1}^{k_b} n_{ij}^{ab} \log\left(\frac{n_{ij}^{ab} n}{n_i^a n_j^b}\right)}{\sum_{i=1}^{k_a} n_i^a \log\left(\frac{n_i^a}{n}\right) + \sum_{j=1}^{k_b} n_j^b \log\left(\frac{n_j^b}{n}\right)}. \tag{2.6}$$

Equation 2.6 compares the two partitions P^a, P^b of sizes n_a, n_b respectively [24]. One partition is considered the ground truth, and the other partition is considered the comparison partition. The way that NMI compares the two partitions is through the construction of a confusion matrix. The columns of the confusion matrix are labeled according to the communities of the ground truth partition. The rows are subsequently labeled according to the communities of the comparison communities. NMI then considers the community each vertex is placed into in both the ground truth and comparison partitions. A small example of the partitions (Comparison and Ground Truth) and the construction of the confusion matrix are displayed in Figures 2.7 and 2.8 and Table 2.1.

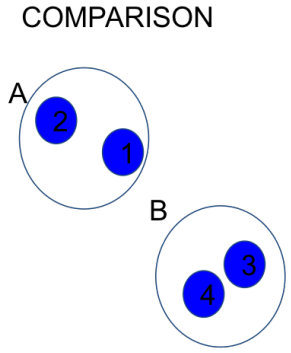


Figure 2.7: A community partition of an inferred network

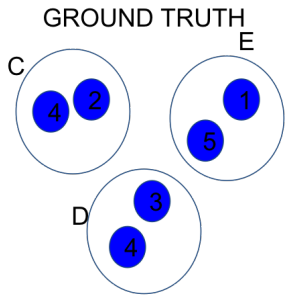


Figure 2.8: Ground Truth Communities

This illustrates how the confusion matrix is constructed, and subsequently used in the calculation of NMI. NMI does not require that the two partitions being compared be the same size. This provides some flexibility in the implementation of NMI. We can compare networks that are different sizes. The common vertices of the two networks must have the same labels in order to be compared. However this flexibility can also cause some issues in analysis.

If the two partitions being compared have different numbers of vertices, NMI does not account for the vertices that are not in the intersection of the two networks. In the Equation 2.6, only the partition of the vertices that have the same labels (i.e., the intersection of the two networks) is being used. An example of how this might potentially cause misleading results is shown in Figure 2.9.

We can see there are vast differences between Ground Truth α and Ground Truth β. If we ran community detection on the network in the Comparison partition from Figure 2.9,

	C	D	E
A	1	0	1
B	0	2	0

Table 2.1: This is the confusion matrix for the two partitions in Figure 2.8. The Ground Truth community partitions are represented in the labels of the columns of the matrix. The Comparison community partitions are represented in the labels of the rows of the matrix. For each node that is in Community i of the Comparison and Community j of the Ground Truth, an entry is added into the corresponding position in the confusion matrix. This matrix is then used in conjunction with Equation 2.6.

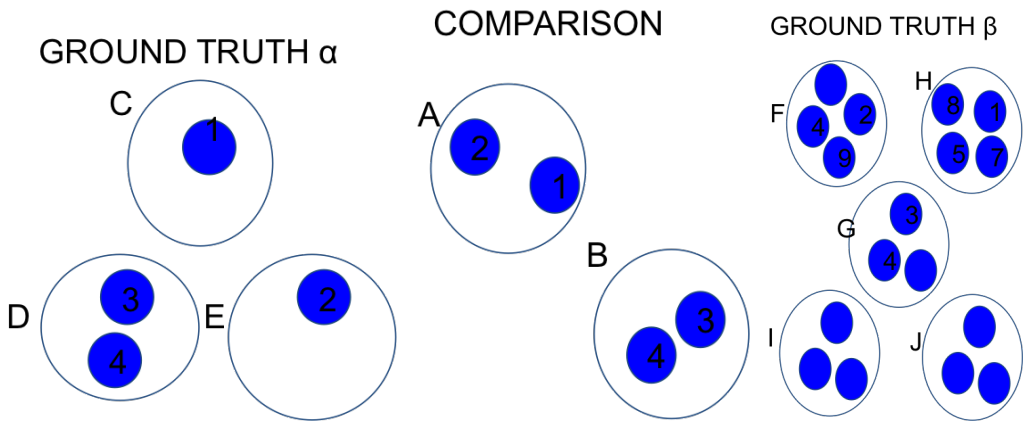


Figure 2.9: Two different ground truth partitions and a comparison partition.

and attempted to use NMI to determine how well our community partition reflected the community structure in the Ground Truth, we might see misleading results. If the actual ground truth looked like Ground Truth α or Ground Truth β in Figure 2.9, our intuition would tell us they have very different NMI values. Yet, each confusion matrix will return the same value for NMI. The confusion matrices are given in Tables 2.2 and 2.3. This illustrates a potential limitation with NMI if the sizes of the networks being compared are vastly different. The limitation can arise if the size of the vertex list differs greatly between the comparison and the ground truth. This is because any vertices that do not appear in both partitions are not accounted for in the confusion matrix.

Our work is primarily focused on examining dark networks and analyzing these networks with only partial information. Clearly, only using partial information, there is a high likelihood that our network comparisons will be different sizes. However, there are some ways to adjust for this potential issue.

First, when we are comparing networks of different size, and our partial information con-

	C	D	E
A	1	0	1
B	0	2	0

Table 2.2: Confusion matrix for comparison and ground truth α from Figure 2.9.

	F	G	H	I	J
A	1	0	1	0	0
B	0	2	0	0	0

Table 2.3: Confusion matrix for comparison and ground truth β from Figure 2.9.

sists of only using one layer of the multiplex network, we can insure all vertices that appear in the full network are also in the single layer. Even if the vertex has no connections in that particular layer, we will still have all vertices appear in each layer. These vertices with no incident edges in a particular layer will have a degree of zero. This forces the two partitions being compared to have vertex lists that are the same size. Doing this will allow us to avoid having networks of vastly different size, a potential issue illustrated in Figure 2.9.

2.3 Networks Used in this Work

In this work we will reference many types of networks. These will consist of both real world networks and synthetically generated networks. Where applicable, we will use actual networks that exist in the real world. Since we are primarily concerned with terrorists, we will use terrorist networks. There are benefits to using large real world networks, the most obvious of which is the fact that they are based on the real world. Social media provides an excellent opportunity to generate or collect large real world networks [25], and we will use this when possible. We also want to include synthetic networks. This will allow us to introduce some random qualities in the network we generate, allowing us to construct multiple different networks that we can test our discovery algorithms against.

2.3.1 Real Networks

Noordin Top Network

Specific to this work, we will be analyzing the Noordin Top Terrorist Network. This is a terrorist network that operated in Indonesia. It is not a single group, but is made up of many smaller terrorist organizations. Noordin Mohammad Top served as a coordinator between the multiple groups [26]. Noordin Mohammad Top was killed by Indonesian law enforcement during a raid in 2009. These organizations do not work towards a single purpose, but they are willing to assist each other and cooperate when properly coordinated. This is the role that Noordin played in this network. We will refer to this network as the Noordin network, but it is actually made of of multiple terrorist networks. While the Noordin network consists of these many terrorist organizations, there are five primary organizations represented. These organizations are Darul Islam (DI), KOMPAK, Jemaah Islamiyah (JI), Ring Banten Group (DI), and Al Qaeda [23]. Noordin himself was a member of JI [26], but

he would reach out to members of other networks when he needed. He was considered a coordinator between these organizations, and was also considered the most wanted man in Indonesia [26]. It is Noordin network that we will analyze in this work. We will discover more about the Noordin Network as we analyze it and examine the results in Chapter 4. All of our data for the Noordin Top Network was taken from the CORE Lab at the Naval Postgraduate School [27].

Facebook dataset

We will use a subgraph of the Facebook network to build a larger synthetic multilayer network. We want to use this Facebook data as a way to introduce a real world example of a network into our synthetic network. This data is provided by the Standford Large Network Dataset Collection [28]. This is a large anonymous subgraph of Facebook. It contains over 4,000 vertices and 88,000 edges. Using this network will greatly increase the size of our synthetic network we will create in Chapter 3. The Facebook network is a large dense network with many hubs, typical of a social network. We will discuss our reasoning for using this in Chapter 3.

2.3.2 Synthetic Networks

Erdős-Rényi Graph

We have mentioned specific real world networks like Facebook and Noordin Top, but we will also use random networks in the current thesis. Random networks will allow us to generate synthetic networks that include some specified attributes while including some randomness, thus achieving a mix of randomness and structure. We will make use of two synthetic networks in this work: Erdős-Rényi and Barabási-Albert. The first random network we will use is the an Erdős-Rényi random graph. Erdős-Rényi graphs are named after Paul Erdős and Alfréd Rényi. These two authors were the first to study random graphs in the 1950s and 1960s [29].

A random graph is referred to as $G(n,m)$ with n vertices and m edges in the graph, G. Newman states that n and m are fixed parameters, and the graph G is one of the random graphs that could exist with the given parameters. Specifically, given the n vertices, the m

edges are placed randomly in the graph [9]. Formally, Newman defines a random graph as:

Definition 2.3.1 *the network is created by choosing uniformly at random among the set of all simple graphs with exactly n vertices and m edges [9].*

Newman is using the term network in this definition, but as previously discussed, this will hold for graphs as well. Another choice of obtaining random graphs is by using the probability, p, of an edge to appear in the graph (instead of the number, m, of edges) in a random graph. This means given n vertices, we would place an edge between any given pair of vertices with a probability p. Additionally, it is usually more common not to think of a random graph or network as a single graph. Instead the random network is one outcome out of the entire population of all possible networks with a given set of parameters [9]. These random graphs are sometimes referred to as Erdős-Rényi graphs or Poisson random graphs. Newman gives the probability $P(G)$ of selecting any given Erdős-Rényi graph from all of the possible choices as

$$P(G) = p^m(1-p)^{\binom{n}{2}-m}, \tag{2.7}$$

where n is the number of vertices, m is the number of edges, p is the probability of an edge existing between two vertices, and $P(G)$ is the probability of choosing a particular graph G. This is under the assumption that we are concerned with only simple graphs, and therefore the probability of selecting a non-simple graph is zero.

While this will provide us with a random graph, there are some limitations when using a Erdős-Rényi graph [9]. Primarily for us, it is highly likely that if the probability p is low, we will have a network with more than one component. This means that our graph will not be connected. This process is random, so there is no guarantee that there will be a path connecting all of our vertices. This could be potentially problematic when analyzing social networks [30]. Our social networks, and the algorithms we use to analyze them are predicated on the fact that the network is connected, or at least the majority of the vertices are connected. This disconnection is something we would like to avoid when constructing

our synthetic network.

Barabási-Albert Graph

While Erdős-Rényi graphs are an excellent example of random graphs, our work deals with social networks. As previously stated, social networks have a power-law or scale free property. We would want our synthetically generated networks to have a similar scale free property. Barabási-Albert Graphs are used to generate these types of graphs, constructed through a method of preferential attachment [31]. The default preference (and the one used here) is for high degree in the existing graph, but that preference can be generalized. A Barabási-Albert graph is created from a small seed network, by adding more vertices, one at a time to the network. With a fixed degree for all incoming vertices, each vertex is added to the preexisting vertices based on the preference of attachment to existing nodes of high degree. This means that as vertices are added to the network, they are more likely to attach to vertices that already have higher degree, creating hubs. Barabási and Albert define the probability that a vertex will attach as it is added to another vertex with the probability Π:

$$\Pi(k_i) = \frac{k_i}{\sum_j k_j}, \tag{2.8}$$

where each node i has degree k_i [32]. This preferential attachment creates scale-free networks that more closely recreate networks that evolve through preference, unlike Erdős-Rényi graphs. These preferential attachment networks have few hubs with very high degree, and many vertices with low degree, namely, they present the power law degree distribution.

We will use both types of synthetic graphs, Erdős-Rényi (ER) and Barabási-Albert (BA), in this work. By using both type of random graphs, we will maximize the benefit of randomness and structure ensuring our synthetic networks have characteristics that are similar to a real world social network.

2.4 Partial Information

The very nature of our dark network contains the idea of partial information. It is slightly paradoxical, but one fact we know for sure, is that we do not know everything about the

network. Beyond the concept that we do not have all the information, we want to further explore what we can determine and analyze with an incomplete picture of our dark network. We will approach this in two different manners. We will assume the dark network represents the ground truth, and then look at part of the network, or partial information. We will attempt to analyze a dark network with only partial information. We will spend more time discussing partial information in Chapter 3.

2.4.1 Single Layer Information

The first way we will consider partial information is by taking advantage of the fact that our Noordin Top Network is a mulitplex. It is each these layers that will provide partial information. Our network is comprised of 14 different layers. There are 139 known terrorists in the Noordin network, and all 139 will be present in each of the 14 layers. There may not be any connections for a particular individual in a given level. These 14 layers are used to create the multiplex of the Noordin network. Each of the 14 layers represents a particular real life type of connection between individuals. The 14 layers in this network are: Classmates, Communication, Friendship, Kinship, Logistical Function, Logistical Place, Meetings, Mentor-ship Ideology, Mentor-ship Supervisory, Mentor-ship Technology, Operations, Recruiting, Soul mates, Training [23]. Some of the meanings behind these layers are self explanatory. Others require some discussion. For example, adjacency in the Communication layer, means that there has been some sort of documented communication between two individuals. Kinship indicates there is a familial relation between two individuals. Some of the less intuitive layers are Operations, in which adjacent individuals have conducted terrorist operations together in the past and Soul mates, which indicates whether two individuals attended the same mosque during the same time frame. We will not focus on the labels or type of layers in the network. For a complete explanation of all of the layers, refer to Everton's book [23].

It is important to note that there is a great deal of research supporting the idea that some layers will be more important than others [2]. In [2], any form of trust is highlighted as being important within dark networks. This is because members of dark networks are attempting to conceal their connections, and are willing to make connections only with individuals they trust. For our purposes, we will avoid focusing too much on the meaning

of any of the layers. We wish to focus instead on using the information contained in the layer as a representation of a portion of the information contained in the network.

For the construction of the layers, we will look at the Communication layer as an example. In a case where two individuals have communicated, there will be an edge present between the two individuals in the communication layer of the Noordin network. All of these communication edges will be aggregated to construct the communication layer. In Table 2.4, we can see there are 318 edges in the Communication layer. This is how all 14 layers are constructed. When we aggregate all 14 layers, we have the full Noordin Top Multiplex network.

These layers provide one of the types of partial information we will use. A single layer represents a "slice" of the network. When we take this slice, we get all of the information available. Again, this is equivalent to having the ability to capture all radio transmissions of the enemy. Or perhaps gaining access to an adversary's email. This does not represent full knowledge of the enemy network, only partial information. But it is all the information of a particular type. It is important to note that there is a great deal of research supporting the idea that some layers will be more important than others [2]. In [2], any form of trust is highlighted as being important within dark networks.

2.4.2 Network Discovery

Another way to consider partial information of a network is through network discovery. This is the idea that initially there is no knowledge of the network. A single vertex is discovered, and from this point, more knowledge of the network is gained by discovering neighbors of the first vertex. In this manner, our knowledge expands outward.

2.5 Network Discovery Algorithms

When analyzing large complex networks, it is common to only deal with a portion of the network in order to simplify the problem. One way to do this is do run a network discovery algorithm for a specified amount of time, and analyze that portion of the discovered network. An obvious question is what is the fastest way to discover the network. There are some more terms we will address that are related to network discovery.

Layer	Edges
Communication	318
Meeting	88
Operations	469
Friendship	154
Mentorship (Supervisory)	51
Soulmates	17
Mentorship (Technology)	13
Recruiting	24
Kinship	49
Logistical Function	554
Mentorship (Ideology)	15
Training	263
Logistical Place	97
Classmates	205

Table 2.4: Table displaying the number of edges in all 14 layers of the Noordin Top Network. All layers have 139 vertices.

The first term is monitor. Davis et al. [33] defines monitor and how it discovers the network with the following definition

Definition 2.5.1 *We say that a monitor on node i detects a node j if $d(i,j) \leq 1$. Also, then the monitor on i discovers the label of j and the* $\deg j$. *A monitor on node i detects an edge ij if i and ij are incident, and i detects the label ij of the edge (i.e., that the monitor discovers the label of the other end node of ij) [34].*

Davis uses the term *node* in Definition 2.5.1, this is interchangeable with vertex. A monitor can be visualized on a simple subgraph as in Figure 2.10.

This type of a monitor evaluates a vertex's neighbor. When we use the term neighbor, we are specifically referring to the vertices that are adjacent to a given vertex. In Figure 2.10, we have placed a monitor on vertex v_3. This monitor has 3 neighbors. They are v_2, v_4, and v_6. They are colored red in Figure 2.10.

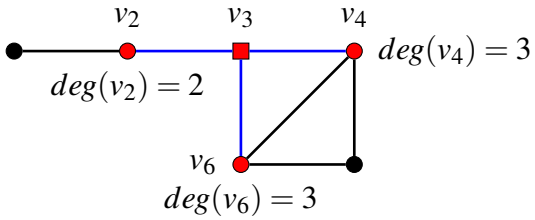


Figure 2.10: A graph and a monitor placed at vertex v_3. Source: [34].

2.5.1 Random Walk Discovery

The first algorithm we will discuss is called Random Walk. This algorithm functions exactly as one would imagine. Given a random start vertex, a random adjacent neighbor is selected. Random Walk then places the next monitor on that random neighbor. This discovers more neighbors. From that monitor, a new random neighbor is chosen. The algorithm used in Random Walk can be seen in Algorithm 1.

Input: G original graph
Output: G_{inf}, a set of vertices and edges that form the inferred graph

$NextVertex \leftarrow$ Random node from G

repeat
    $MonitorList$.add($NextVertex$)
    G_{inf}.add($NextVertex$)
    foreach $NextVertex.neighbor$ **do**
        if $NextVertex.neighbor \notin G_{inf}$ **then**
            G_{inf}.add($NextVertex.neighbor$)
        end
    end
    foreach $NextVertex.edge$ **do**
        if $NextVertex.edge \notin G_{inf}$ **then**
            G_{inf}.add($NextVertex.edge$)
        end
    end
    $NextVertex \leftarrow$ random from $NextVertex.ns$
until *50% of vertices have monitors*

Algorithm 1: Random Walk Algorithm. Source: [34].

As the name Random Walk implies, this algorithm randomly moves through the network discovering vertices and their neighbors. As expected and shown in [34], this algorithm is

not very effective in discovering the network quickly. We will treat this as a lower bound in our work.

2.5.2 Undiscovered Degree

The next method of network discover we will examine is Undiscovered Degree (UDD). This algorithm can be found on [35]. The Undiscovered Degree algorithm starts by placing a monitor on a single vertex. This vertex can be specified, or it can be selected at random. Once a monitor is placed on a vertex x, the monitor discovers all of the neighbors of x, called the neighborhood of x, and denoted by $N(x)$. Additionally, UDD creates a list of all discovered nodes in graph G, labelled $L(G)$. For each $y \in N(x)$, UDD queries y for the count $|N(y) - L(G)|$. This value $|N(y) - L(G)|$ counts the number of undiscovered nodes adjacent to y, which we call undiscovered degree of y and formally define as

$$DoUN(y) = |N(y) - L(G)|, \forall y \in N(x).$$

This allows for a choice of monitors on the nodes that will light up, through discovery, as much as possible with a single neighbor of the discovered network.

We present the algorithm for Undiscovered Degree in Algorithm 2 [34].

In [34], UDD is shown to be very effective in discovering the network quickly. This is because the algorithm specifically selects the placement of the next monitor in order to bring in as many, previously unseen vertices, as possible. We can see an illustration of the behavior of a monitor in the UDD Algorithm in Figure 2.11. In this illustration, the monitor has been place on vertex v_3. This monitor discovers v_2, v_6, and v_4, the adjacent vertices of v_3 called neighbors. All discovered vertices are placed on the discovered vertex list of the graph. The algorithm then considers each neighbor in turn, and counts the number of undiscovered vertices for each neighbor. The neighbor with the most undiscovered neighbors, v_4 in this case, is where the next monitor is placed.

UDD prioritizes discovery of large degree nodes. These are hubs in social networks, making this a desirable algorithm to use for network discovery of a dark network. We are limited in the information that is available in a dark network, so we would like to maximize our discovery within that network, and this algorithm has a lot of potential.

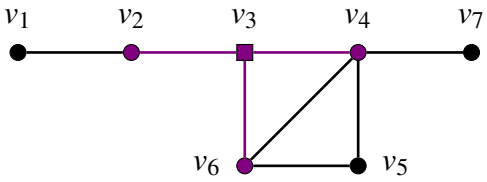


Figure 2.11: Behavior of a monitor in Undiscovered Degree. Adapted from [34].

Input: G original graph
Output: G_{inf}, a set of nodes and edges that form the inferred graph

$NextNode \leftarrow$ a random node from G

repeat

  $ListOfMonitors$.add($NextNode$)
  G_{inf}.add($NextNode$)
  $UpdateList$.add($NextNode.neighbors$)
  foreach $NextNode.neighbor$ **do**
    $UpdateList$.add(($NextNode.neighbor$).$neighbors$)
    if $NextNode.neighbor \notin G_{inf}$ **then**
      G_{inf}.add($NextNode.neighbor$)
1     $RestartCounter = 0$
    else
      $SeenCount(NextNode.neighbor) += 1$
    end
  end
  foreach $NextNode.edge$ **do**
    if $NextNode.edge \notin G_{inf}$ **then** G_{inf}.add($NextNode.edge$)
  end
  foreach $UpdateList.node$ **do**
**   **if** $node \in G_{inf}$ **then**
        $node.FakeDegree =$ length($node.neighbors \setminus G_{inf}$)
        (i.e. # of undiscovered neighbors)
    end
  end
2 **if** $RestartCounter == 2$ **then**
3   $NextNode \leftarrow$ a random node from $V(G) \setminus ListOfMonitors$
4 **else**
    $NextNodes \leftarrow$ MaxFakeDegreeNodes($G_{inf} \setminus ListOfMonitors$)
    if length($NextNodes$) > 1 **then**
      $NextNode$=node with minimum of $SeenCount(NextNodes)$
    else
      $NextNode = NextNodes$
    end
5   $RestartCounter += 1$
  end

until *50% of nodes have monitors*

Algorithm 3: Algorithm for Undiscovered Degree, Source: [34].

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Methodology

We have discussed in great detail many aspects of Network Science. Now we will use all of this information to analyze a dark network, and attempt to identify communities within a terrorist network and identify a terrorist network within a larger network.

3.1 Topology of the Noordin Top Terrorist Network

Previously we discussed the Noordin Top Terrorist Network. This is the network we will primarily use for this analysis. We can see all 139 vertices in the network in Figure 3.1,

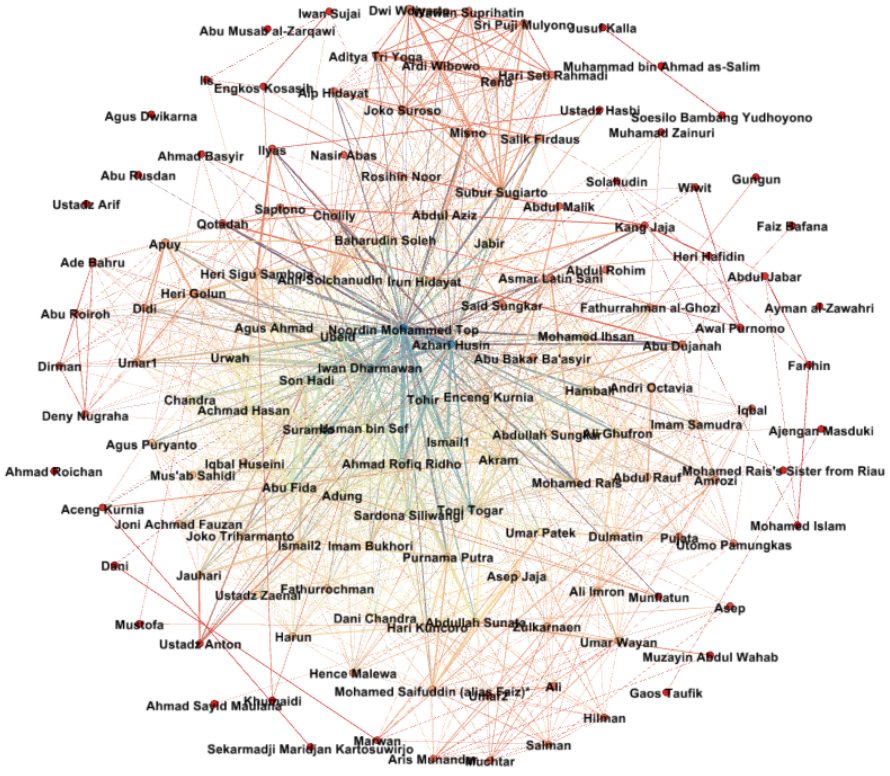


Figure 3.1: Noordin Top Terrorist Network. This image created using Gephi Network Visualization Software.

as well as the name associated with each terrorist. Vertices with higher degrees have blue edges, while vertices with lower degrees have red edges. From Figure 3.1 we can see that Noordin Mohammed Top is one of the vertices with the highest degree. While this network is not too large, 139 terrorists, we do not want to rely solely on visual analysis. We will begin by analyzing the degree distribution of the full Noordin Top Network.

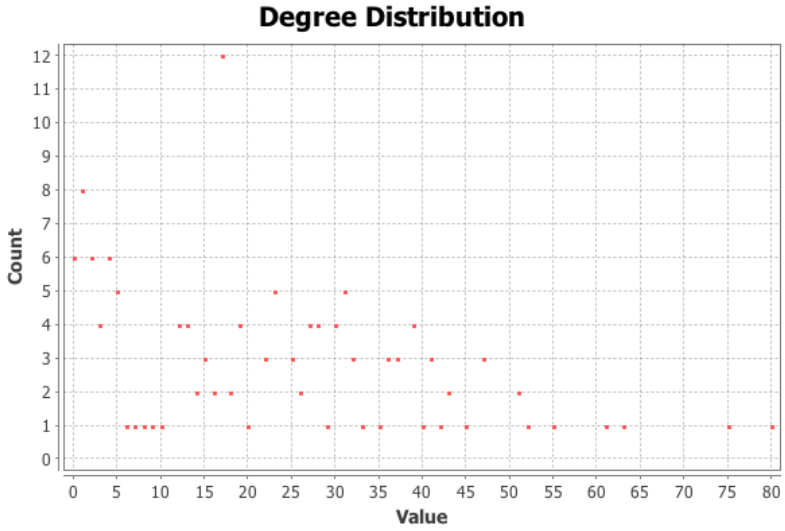


Figure 3.2: Degree Distribution for the Noordin Top Terrorist Network. This image created using Gephi Network Visualization Software.

In Figure 3.2 we see the degree distribution. We can see that this network has a general scale free distribution. It is not a perfect example of the scale-free property we expect to see in social networks, but it is important to remember this is a dark network. One of the key aspects of a dark network is that we do not have all of the information available. Additionally, these networks will evolve in a manner that is inherently inefficient. When we say inefficient, we mean the network will attempt to obscure connections between its members. Instead of passing a message directly from the source to the destination, the message will be passed through multiple intermediaries. This will skew the scale free nature of the degree distribution.

We can calculate the clustering coefficient for this network as well. We utilize the Gephi Network Visualization package to calculate the clustering coefficient. When we do this, the clustering coefficient for the Noordin Top Network is 0.708. This high value by itself

is a bit informative, but later we will compare the full Noordin Top Network to subgraphs of the network using it.

We also calculate the density of the network. The density of the full network is 0.156. This tells us that the network is rather sparse. This is not unexpected. Dark networks are typically sparse, and the Noordin Top Network is no different.

The next step in our initial analysis of the Noordin network is to run community detection on the full network. We will again use Gephi, which allows us to use the Louvain community discovery method, modularity and more importantly provides a visual output that is particularly useful. The Louvain method is not deterministic, so we will run it and its modularity multiple times to ensure we have an accurate picture of the community structure and its measure on the Noordin Network.

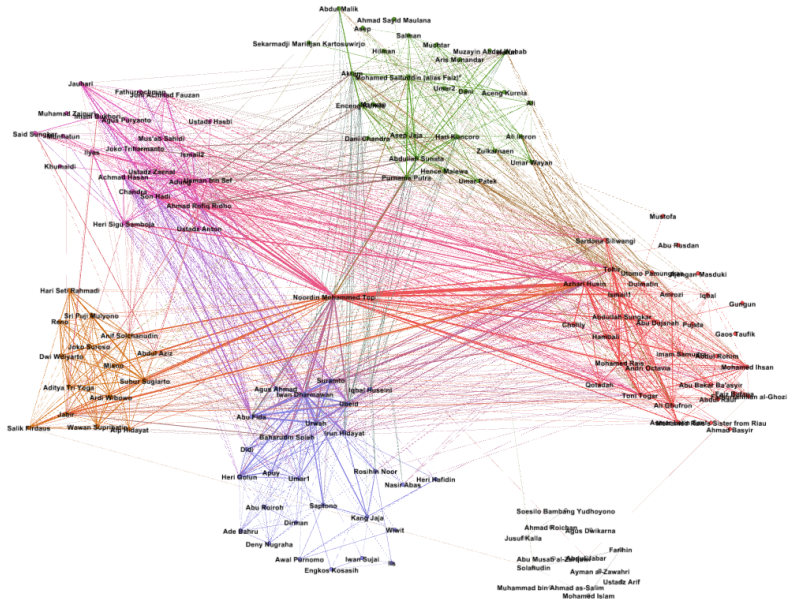


Figure 3.3: Noordin Top Terrorist Network organized by community membership using the Gephi.

While the Louvain method is not deterministic, it consistently returns a modularity value of approximately 0.35 and about the same partition each time we run it. This variation is not a property that is unique to the Louvain method; all community detection algorithms are non-deterministic. In Figure 3.3 we see the results of Louvain community detection. In this image, communities with a size of five or more terrorists were assigned a random color.

It is important to note that all vertices in this network were assigned to a community. In some cases, the community only has one or two members, we clumped these communities together and colored gray as a misfit community. We will focus on identifying a large communities (at least 5 members). We can clearly see in Figure 3.3 there are five larger communities that make up a majority of the network. Visually looking at these, we can see that there are more connections with in each community, than among the communities. We can identify Noordin Mohammed Top as the most connected member of the network. He is shown in the center of the image, as a member of the red colored community, but highly connected to other communities in the network.

We will now evaluate each community individually to find some of the previous metrics of interest already discussed. In order to accomplish this, we will consider each community as its own network. We will maintain all vertices that are members of a community, as well as all edges that connect members of a community. We will not maintain edges that connect communities. This can be thought of as “cutting and pasting" the community from the full network, into a network of its own. We will treat each community as its own subgraph. This can be seen in Figure 3.4. In this figure we see one of the Communities as its own network. We will run the same analysis of this community. The results can be seen in Table 3.1, showing that there are 34 members of this community, there are 221 edges within the community. Additionally, the density, clustering coefficient, and modularity are given.

Figure 3.4 and Table 3.1 show the results for one of the communities, when we treat that community as a subgraph. This was done for all of the significant communities in the network. Additionally, since Louvain is non-deterministic, we will implement this method multiple times to ensure we have a representative partition of the network into communities.

Vertices	34
Edges	221
Density	0.394
Clustering	0.774
Modularity	0.185

Table 3.1: Metrics for red community

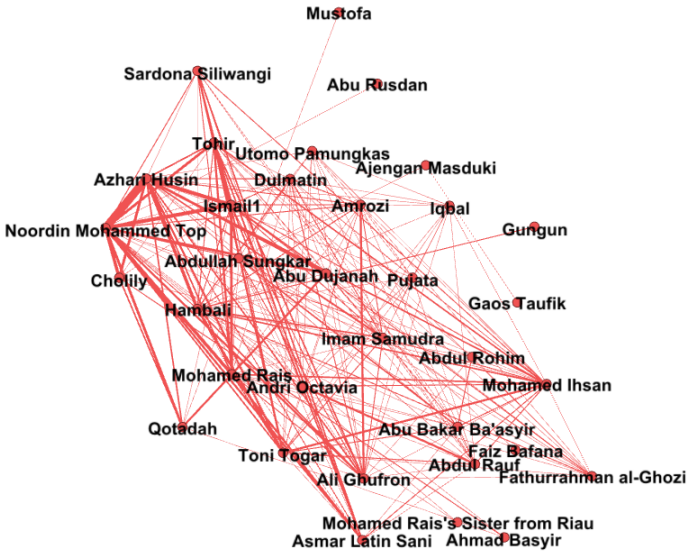


Figure 3.4: One of the Communities from the Noordin Top Terrorist Network using the Gephi.

3.1.1 Five Terror Groups within Noordin Top Network

When we introduced the Noordin Top Terrorist network in Chapter 2, we discussed that the network consists of 5 terrorist groups, and that Noordin Mohammed Top serves as a coordinator between the terrorist organizations. It is no coincidence that community detection partitions the network into 5 major communities. Additionally, Noordin Top coordinates between all five organizations. We see this in the network, in the fact that Noordin Top has so many connections outside of his community. This means that Louvain community detection could potentially identify communities that exist in real life.

3.1.2 Identification of a target community

Following the analysis of the network, we will have to identify a target community. Our goal is to find terrorists. In the first situation, we will be looking for a particular community of terrorist within the full terrorist network. We will not simply look for community structure in our partial information picture of the terrorist network. We want to find a specific community. In order for this to be possible, we will need to find a community from the

full network that stands out as unique from the rest of the communities, and then identify it from the partial information.

A single community that stands out as unique is not enough to evaluate our partial information as an accurate reflection of the full network. We would also like to see how much the partial information can tell us about the full network. We will use NMI to compare our partial information to the full network. Using NMI to compare will allow us to see how good our partition into communities is in comparison to the partition with in the full network.

3.2 Partial Information

Once we have fully analyzed the Noordin Top Network, we know enough about it to then turn our attention to partial information. As we discussed in Chapter 2, we will look look at two different types of partial information. We will consider partial information that consists of a complete single layer from the full network. We will also consider partial information that is a result of network discovery of the full network.

3.2.1 One layer of the network

In the Chapter 1, we discussed having the ability to intercept all radio traffic or cell phone conversations in a particular geographical location. This is where we will start with our attempt to identify a specific community in our network. Sticking with the scenario detailed in Chapter 1, we will start with the communication layer. The communication layer represents the scenario of capturing all enemy radio transmissions in a region. Similarly, it could represent intercepting all cellphone traffic in a particular area. The communication layer might represent hacking into a server, and tracking all email traffic through a server. The physical interpretation is not what is important in this scenario. What is important is that it represents all the available information in that particular layer.

The communication layer is one of the 14 layers that make up the Noordin Top Network. An image of this layer can be seen in Figure 3.5. We have selected the communication layer partly because it is one of the layers that has the most edges from the 14 layers of the full Noordin Top Multiplex Network. This will hopefully give us the best chance of being able

to draw conclusions about the full network, using only partial information. We will analyze the communication layer in a similar fashion to the way we analyzed the full network. We will run community detection using Louvain Modularity, and attempt to identify the target community. Additionally, we will compare the results of the community detection from the communication layer to the results of community detection from the full Noordin Network. We will make this comparison using NMI.

When we consider a single layer of the network as our partial information, we will not use only the communication layer. While the communication layer is one of the layers that has the most number of edges, and it is the layer that was used to illustrate the concept of partial information in Chapter 1, but it is not the only layer in the network. We will use all 14 layers individually in an attempt to conduct community detection using partial information. The communication layer can be seen in Figure 3.5, it has 139 vertices, but only 120 of these vertices are connected. The remaining 19 vertices represent 19 terrorists that are known to be members of the Noordin Network, yet there is no information available about their communications. Among these 120 connected vertices, there are 318 edges. This layer is the largest in the network, and it can be contrasted with other layers. Two other layers displayed in Figures 3.6 and 3.7 are Classmates and Recruiting. They each have 139 vertices, but Classmates has 205 edges, while Recruiting has 24 edges. This means that all 139 terrorists are represented in both layers, however there are only 205 and 24 connections of a particular type in each layer. We display these two layers, not because they are unique, but to illustrate there are 14 layers, and all 14 layers are different. All the layers provide some information used to construct the Noordin Top Network, but no single layer has all the information. We will use NMI to compare community detection in each of the layers to the ground truth of the full Noordin Top Network.

3.2.2 Network Discovery

We have considered partial information as a single layer of a multilayer network, but there are other ways to think of partial information. There has been work in community detection while discovering a network. In this concept, as more and more of the network is discovered, we conduct community detection. The goal of this is to be able to draw conclusions about the community structure of a given network, without having to discover an entire

Figure 3.5: Communication Layer from the Noordin Top Terrorist Network. This image was created using the Gephi Network Visualization Software.

network. Monitors that light up the network might be expensive to place on a network. Expensive in this use is not restricted strictly to monetary amounts. It is not difficult to image a situation where each monitor is expensive or difficult to emplace. In an operational environment, the monitor might represent a mission to capture a terrorist. In that example, the expense is the risk of conducting an operation. We would attempt to minimize the number of monitors required, and therefore minimize the risk Soldiers are exposed to conducting missions. Another expense might be the cost paid in political power. Again, it is easy to image a situation where law enforcement would like to gain access to an individuals locked mobile device. The monitor in this situation is being able to access the information in the locked device, and determine the contact information in the device in order to build a network. It might cost the law enforcement officials a great deal of political power in order to

Figure 3.6: This is the Classmates layer of the Noordin Top Terrorist Network. This image was created using the Gephi Network Visualization Software.

Figure 3.7: This is the Recruiting layer of the Noordin Top Terrorist Network. This image was created using the Gephi Network Visualization Software.

gain access to that locked device. The goal would be to use as few monitors as possible.

With the goal of understanding the community structure of the network as early as possible, there has been some research into this idea. Others have found that a lot depends on the structure of the network. Certain networks, such as man made power grids, do not require a large number of monitors. There are other networks, like technological and dark networks, that provide a more refined community picture with every monitor placed. This would seem to follow our intuition of dark networks. Dark networks form in a manner to hide information. This clandestine nature would make community detection more difficult. For more on this topic, one can read [36].

3.3 Embedding the Terrorist Network in a Larger Network

Up to this point, we have been looking for a target community within a terrorist network. We could say we are looking for a specific group of terrorists out of a larger group of terrorists. We could also say that we are looking for a particular group of needles in a stack of needles. This is because our terrorists do not act like other members of social networks. Our terrorists are trying to conceal their network. While searching for particular terrorists among other terrorists presents its own set of challenges and difficulties, it is not an extremely relevant situation. If we already know who all the terrorists are, we probably

are not terribly concerned with their community structure. We are much more interested in discovering terrorists that are hidden in a non-terrorist network. We want to search a civilian population, and find the most efficient method to discover the terrorists. To reuse our idea of searching for needles, we would like to find the needles in the haystack. However, we want to do this as efficiently as possible. As we previously stated, monitors might represent something that is expensive. We want to find terrorists, and we want to find them as fast as possible, using as few monitors as possible and the topology of the network.

All of this brings up a few concepts that we use in our methodology. We want to use our multilayer terrorist network, and embed it into a large non-terrorist network. We then want to search in this network to find the terrorists. This will require us to construct a large synthetic network, embed the Noordin Top Network in the synthetic network, and do this in a manner that does not make the terrorists appear unique, nor their affiliation to each other stand out. Once we have this large synthetic network, we want to search through the network, and discover the terrorists.

Evans and Lapides in [37] have already looked into embedding individual layers of the Noordin Top Network in to larger synthetic networks. They created random Bernoulli Graphs, and embedded individual layers from the 14 layer Noordin Top Network into the random graphs. They then used closeness centrality to attempt to identify the terrorist from within the synthetic network. We will take this concept of embedding the Noordin Top Network into a synthetic network, but instead of using a Bernoulli Graph, we will use multiple types of graphs. We will concern ourselves with the full Noordin Top Network, not individual layers. Finally, we will not use metrics such as closeness centrality to identify the terrorists, but will instead discover the network through the use of discovery algorithms.

3.3.1 Creation of all 10 layers

Our stated goal is to create a multilayer network, and embed the Noordin Top's Network in it, in absence of such a real network with the terrorists occupying part of it. This presents a unique set of challenges. First is the fact that the Noordin Top Network is so sparse. Then we must create a large network. We want this large network to be as realistic as possible. Then we want to embed the Noordin Top Network in the synthetic network. But we do not want the terrorist network to stand out as too unique from the rest of the network. We want

our task of discovering the terrorist vertices to be realistic and therefore relatively difficult. It would be an extremely daunting task to create a synthetic network that exactly replicated a true multiplex network. The size and type of connections could be research topics of their own. Additionally, recreating a true and realistic embedding of a terrorist network within a civilian population could be the topic of multiple research papers or fields of study. As we don't have a good understanding of this embedding from real data, we introduce a methodology for such an large network and the embedding of the terrorist network in it. We create our synthetic network and subsequently embed a true to life replication. We will focus on creating a synthetic network that contains the Noordin Top Network in a manner that makes discovery difficult and realistic. Our goal is to create the synthetic network in such a way that the Noordin Top Network's size and topological structure is not unique.

As we have stated many times, the Noordin Top Network is a multilayer network. We will attempt to maintain this quality, and replicate this quality in our synthetic network. With this in mind, first we will address the sparse nature of our dark network layers. There has been previous research into attempts to infer additional community structure in dark networks. This research shows there are methods to take existing communities, infer additional edges within communities, and in doing so construct a community structure that more accurately reflects the true community structure of the network. For an in-depth study of this method, Ryan Miller's thesis provides more information [38].

Figure 3.8 shows the method to create the three categories we will use. For our purposes, we will take these three categories to represent a more accurate depiction of the Noordin Top Network. Miller [38] used the 14 layers of the Noordin Top Network to create 3 categories. Miller creates these categories to infer more information about the network. We have shown that some of the layers are quite sparse, but we still want to include the information these sparse layers contain. He explained the process outlined in his thesis to infer more information about the dark network. The three categories he created accurately depict the structure of the Noordin Top Network. We use these categories instead of the layers, because they are informative as Miller explained, and it is a way to condense the information contained in the network. The three categories used are labeled Lines of Communication (LOC), Trust, and Knowledge. The labels of these categories and what each category represents are important. We do not want to fixate on these labels, we want to simply use these

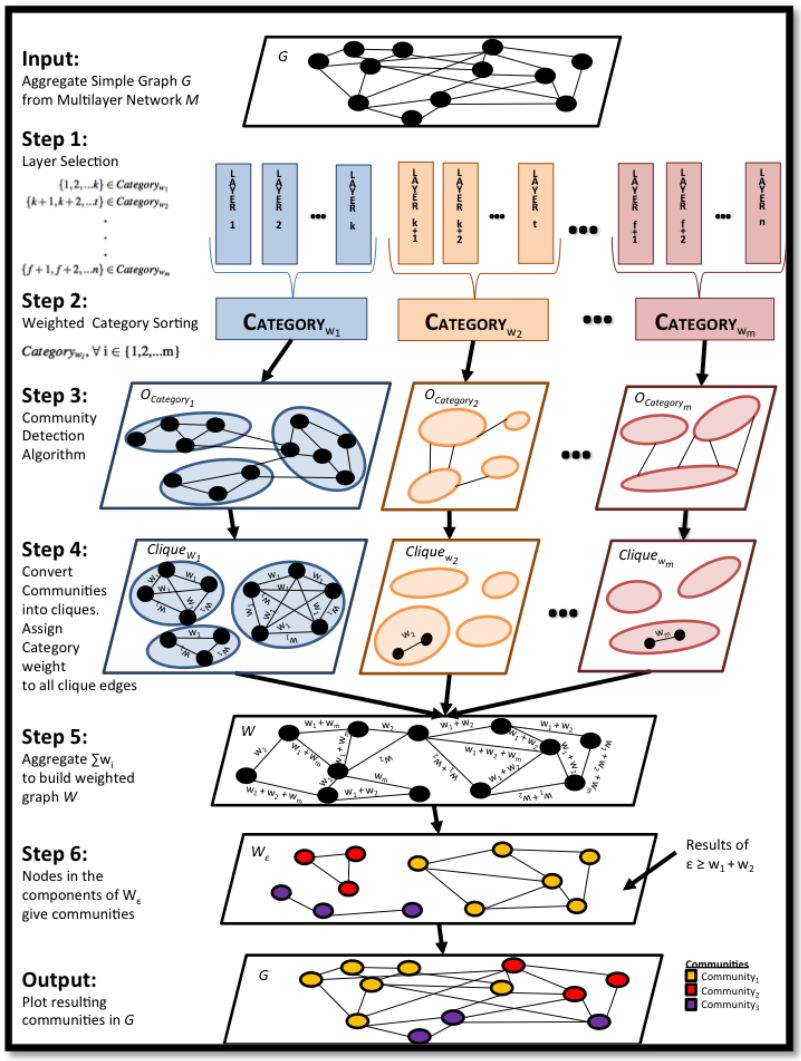
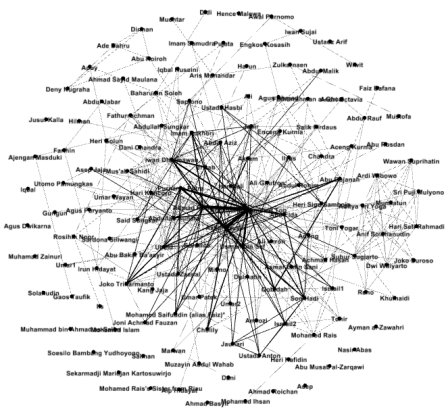


Figure 3.8: Illustration of the method used to construct categories using multilayer dark networks. Source: [38].

three categories as a way to represent the Noordin Top Network as a multilayer network, without having to use all 14 layers.

In his work, Miller describes the creation of *categories*. He creates these categories through the use of layers, the same layers from our 14 layer Mulitplex network. We will be using his categories, but we will use them as layers in our synthetic network. We will use the terms category and layer interchangeably. In Miller's work, categories and layers are different. In our work, they are the same concept, a layer within a multilayered network.

In Figure 3.9 we have displayed the LOC category and some relevant data from the LOC category.



Number of Vertices	139
Number of Edges	383
Density	0.04
Clustering Coefficient	0.569

Figure 3.9: Network and data for the LOC category of the Noordin Top Network.

The other two categories are Knowledge and Trust. The Knowledge category is displayed in Figure 3.10 and the Trust category is displayed in Figure 3.11 The data associated with each of the two categories is in Tables 3.3 and 3.2.

We will start the construction of our synthetic network by using these three categories as layers. Some important points to remember about these three layers will help with the construction of the rest of our network. All 139 of the terrorists are in each of these three categories. This does not mean that all 139 are connected in each layer, just that they are all present in the layer. We will think of each vertex as an individual terrorist, and each layer as a type of relationship. The goal is to now construct a synthetic network using these layers.

Vertices	139
Edges	418
Density	0.044
Clustering	0.655

Table 3.2: Data for the Trust category of the Noordin Top Network.

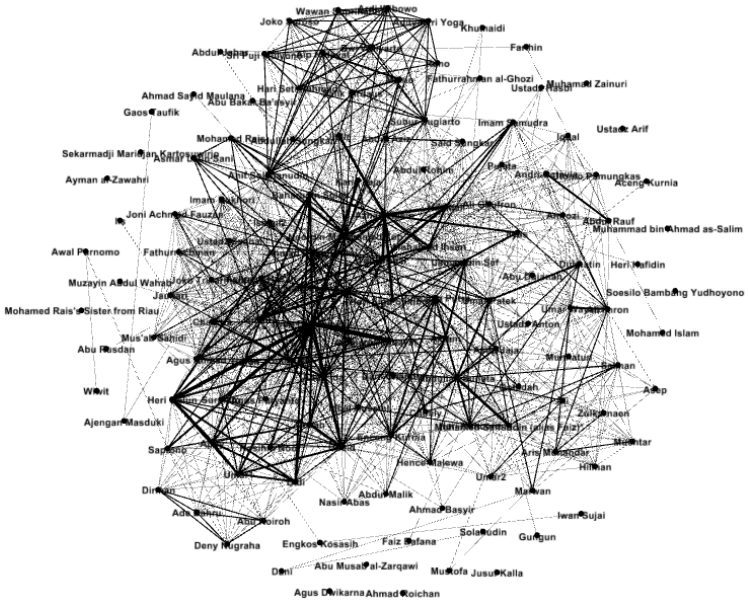


Figure 3.10: Knowledge category of the Noordin Top Network. This image created using Gephi Visualization Software.

Vertices	139
Edges	1308
Density	0.136
Clustering	0.688

Table 3.3: Data for the Knowledge category of the Noordin Top Network.

All 139 of our terrorists will be present in the network, but there will be some layers in which they do not have any connections. Similarly, later when our non-terrorist members of the network are created, they need to appear in some categories but not necessarily every category.

We will continue the military theme from the introduction in Chapter 1 in the way we distinguish terrorist vertices and non-terrorist vertices. Traditionally, in the military, friendly units are colored blue on maps. Conversely, enemy units are colored red. We will use this

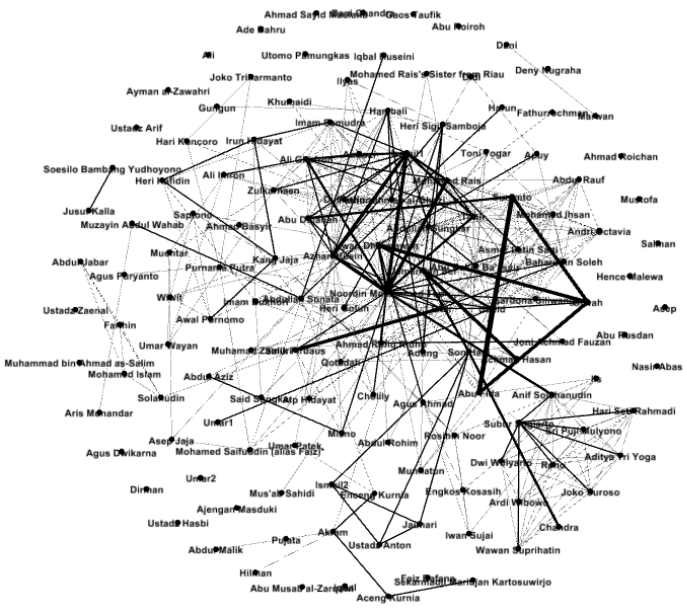


Figure 3.11: Trust category of the Noordin Top Network. This image created using Gephi Visualization Software.

in our synthetic network. All of our terrorists will now be called red nodes. Non-terrorist nodes will be called blue nodes. Currently, the LOC, Knowledge, and Trust layers are populated with red nodes. For this reason, we will consider them red layers. Eventually we will add blue vertices to these layers, and they will no longer be completely red. We will, however, refer to the layer as red.

Our goal is to embed the Noordin Top Network into a synthetic network with the purpose of ensuring the Noordin Top Network is not unique. We would like to mimic real life by making sure that it is not easy to discover our red nodes. We strongly believe this is necessary in order to construct a realistic social network embedding our terrorist network. We feel that if we instead mix our Noordin Top Network with a mathematical construction of a network based on formulas, it will then be too easy to identify the terrorist. This is based on the fact that real networks have inherent structures that our existing mathematical models cannot capture. These inherent structures would make the testing of our inference

algorithm in discovering the red network unrealistic.

For this reason, the next three layers will be exact copies of our first three layers but they are blue nodes. We will have non-terrorist copies of the LOC, Knowledge, and Trust layers. To avoid confusion and provide clarity, we will refer to our original terrorist layers as R1, R2, and R3 since they are red layers. Subsequently, we will refer to these three copies as B1, B2, and B3 since they are blue layers. The graph induced by R1, R2 and R3, and the graph induced by B1, B2, B3 are isomorphic copies of the Noordin Top Network.

Our next category to add is B4. This layer is an exact copy of the full monoplex Noordin Top Network (it has all 139 nodes and 1499 edges). We will add this category to our synthetic network to further insure that there is nothing unique or specific to the Noordin Top Network's topology that causes any algorithm to specifically target the Noordin Top Network. Another way to think of this category is as our "Soccer Mom Terrorist Network." We are including this category because we do not know if there are non-terrorist networks that develop in a manner such that they have terrorist-like topologies. There is a chance that the network that represents a Soccer Mom carpool network has a similar topology to the Noordin Top Network. When we include category B4 in our synthetic network, we avoid having to control the exact topology of the monoplex network to avoid terrorist-like qualities. We are simply constructing our network in a manner to make the Noordin Top Network not unique.

All of our categories up to this point have been based of real world networks. We will now introduce some our synthetic categories to increase the size of our real network to a desired 1 terrorist to 50 civilians. Increasing the size of the network will make detection of terrorists more difficult, and better test our discovery algorithms. Up to this point, we have attempted to construct the network so the Noordin Top Network is not unique. We will now shift our focus to introducing some randomness to our network. In order to do this we will use some of the synthetic graphs that we outlined in Chapter 2. We will construct these categories using synthetic models implemented in networkx, a python coding language network package.

Category B5 is a BA Graph. Our original Noordin Top Network has 139 vertices. We will work towards increasing the size of our synthetic network by increasing this in our BA

graph by a factor of 10. We want to expand the size of this layer, because it will help us reach the desired size of 1 terrorist to 50 civilians. BA graphs are constructed using the premise of preferential attachment, and in B5 new vertices will be attached with a degree of three. Sharma, Magnani, and Montesi used BA graphs when constructing synthetic networks [39]. While their work was primarily concerned with missing data points in social networks, we will use BA graphs as well for our goals. Our purpose for the use of synthetic graphs is to introduce some controlled structure in the networks we generate.

Our truly random category will be B6. Category B6 is constructed using an Erdős-Rényi (EA) random graph. Again we use Networkx to construct the graph. We again use a graph with 1390 vertices, and probability that any two vertices are adjacent is set at 0.005. We use such a small probability because the network is large. If we use a larger probability, the vertices in this category will have very high degrees. And consequently, when we include this category in the synthetic network, our synthetic network will not be scale free any longer.

We have ensured that the Noordin Top Network is not unique with categories B1, B2, B3, and B4. We then introduced some randomness with categories B5 and B6. Our final category is B7. Category B7 is anonymous Facebook data. This network has 4,039 vertices and 88,234 edges. Including this network as a category accomplishes two tasks. First, it introduces real world social network data into our synthetic network. Also, in Chapter 1, we referenced patrolling in real world geographic locations. This category is highly connected, so including it helps represent a very dense category based on real data rather than synthetic. It imposes almost a pseudo-geographic location quality into our synthetic network. This category can be thought of as representing neighbors in a community. Members would be adjacent in this category if they lived in the same street, or subdivision. This layer will help increase the size of our network, and force the vertices present in the synthetic network to be more highly connected.

At this point in the creation of our synthetic network, we have three red categories and seven blue categories. The 139 terrorists are represented in the red categories. Additionally, no terrorists appear in a blue category, nor blue vertices appear in a red category. We do not yet have a single network. Rather, we have many disconnected and unrelated categories. Our next task is to combine these categories into a single synthetic network.

We have the goal of creating a purple network, where we can not easily distinguish red nodes from blue nodes. In Figure 3.12, we can see the ten layers that currently exist. We want to aggregate these layers into a single purple multilayer network.

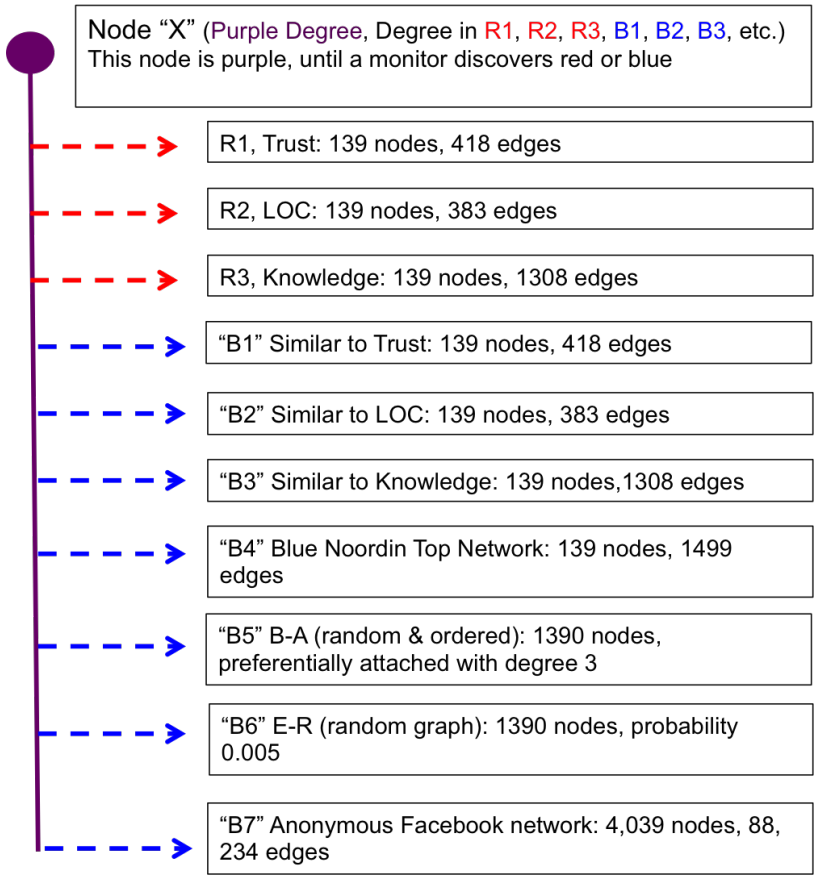


Figure 3.12: Information contained in each purple vertex in our synthetic network.

This concept of the construction so far is illustrated in Figure 3.13. As we can see, there are three red categories and 7 blue categories.

3.3.2 Cross Population and Connections between categories

Our goal, however, is to have a connected large synthetic network embedding the red nodes realistically. Since this synthetic network will be composed of red and blue layers, we can think of this network as purple. There are both red and blue vertices in our purple network, but we do not know if they are red or blue until the inference algorithm places a monitor on

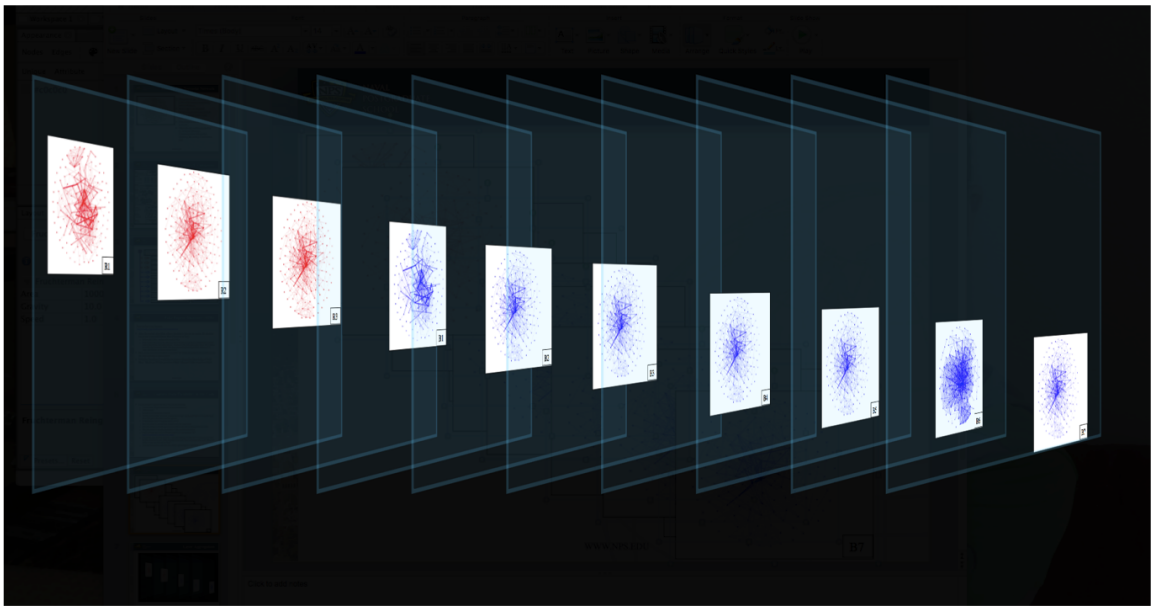


Figure 3.13: Illustration of our 10 disconnected layers of the network-to-be.

the node and queries it. We will focus on discovery shortly, but for now we simply want to construct the purple network. We want to take the layers of Figure 3.13, and aggregate the categories into Figure 3.14.

We will start this aggregation by first working with our vertex lists. We need to generate a vertex list for the purple network. We will do this by taking the union of the vertices in all the categories. Specifically, we will only consider the vertex list of one of the red categories. This is because the three red categories already represent all the red vertices, we do not want any duplicates of red vertices because they represent terrorists. So we will take the union of all of the blue vertices and red vertices. This list will now represent all members of our purple social network, and we will call it $V(PurpleNetwork)$. We will consider this vertex list purple. But if we consider our purple vertex list, each vertex in the list only has connections in a single category. Or in the case of red vertices, only in the red category. Any algorithm would be able to easily discover red vertices by simply searching through categories R1, R2, and R3. So our next task is to effectively cross-populate vertices across all of the categories.

We now take the union of the purple vertex list $V(PurpleNetwork)$ with the existing nodes

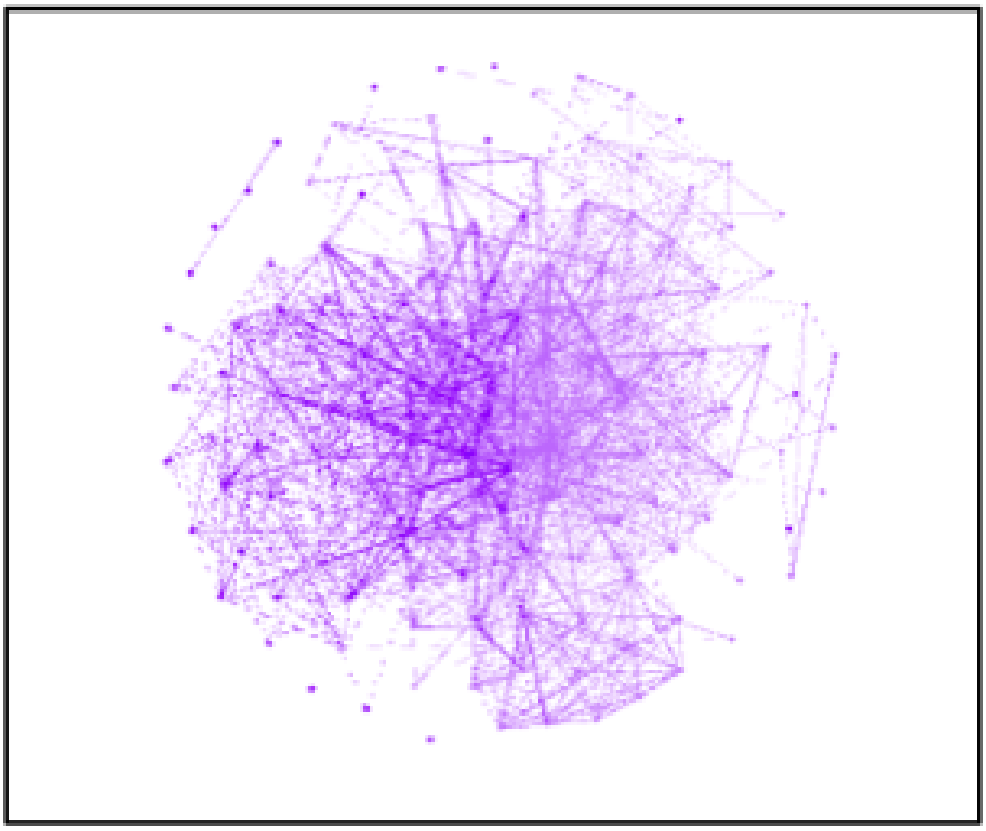


Figure 3.14: Illustration of the theoretical Purple network we want to create. This image created using Gephi Visualization software.

in every category, but adding the new nodes with a degree of zero. What we have now is a purple network with 10 categories, each of which has $V(PurpleNetwork)$ as the vertex set, much like the original Noodin Top had 3 layers and all terrorists showed in all three layers. Up to this point in each category, only the original red or blue vertices of that layer have nontrivial degree. So far, the existing edges are colored red or blue as they came from the original red or blue layers. So the edge list is not yet purple.

To cross-populate the layers of our network purple, we add edges between layers without changing the original edge list from the categories. In order to accomplish this, we will randomly attach newly added vertices from each layer to the previously connected vertices in a given category. We will do this in a manner to have each vertex attached in approximately half of the layers. Vertices that were originally red will be attached to vertices in categories B1-B7, with a probability of $p = 0.3$. Vertices that were originally blue will be attached

in any category, except the one they originated from, with a probability of $p = 0.5$. These probabilities will be applied independently in each category. When a vertex is attached into a category, it will be added with degree 3. This will be conducted with preferential attachment, the same process outline in Barabási-Albert random graphs. This process will have the effect of taking our previously red or blue edges, and turning them purple. This happens because in every category, there will now be red and blue vertices that are adjacent to each other, regardless of the category from which the vertices originated.

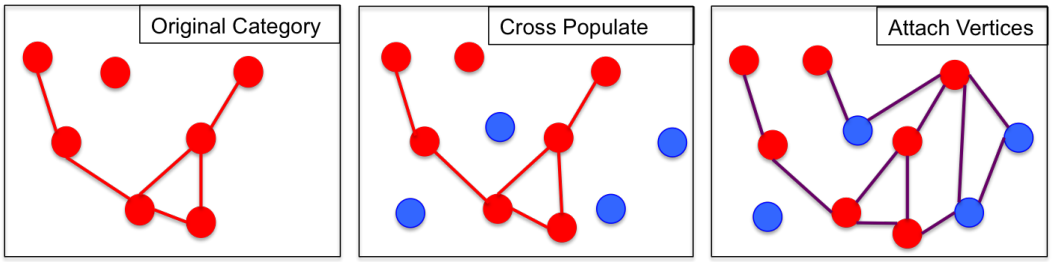


Figure 3.15: This illustrates the process of cross populating vertices from one layer to another, and attaching them. Doing this will create a purple category for later use in the purple network.

We see an illustration of this cross population and connection in each category in Figure 3.15, showing an example of the process in an originally red category. All original vertices and edges are red. We then cross populate blue vertices into the category. There are no new attachments between red nodes since we do not alter the original terrorist data in any way. While all vertices are added to a layer, only some are attached. In using this method, we have changed the edges from red to purple. An identical method can be used in a blue category. The only difference would be that the incoming cross populating vertices would be both red and blue.

The last step of our aggregation of these ten layers into a single large purple network is performed by first taking our purple vertex list $V(PurpleNetwork)$, and using the adjacency list from each layer to construct our purple multilayer network. This will create a network that has all vertices from all of the layers. There is unique adjacency list information that originates from each layer. These layers are made up of both red and blue vertices. Additionally, we do not know if the edges originated from the original layer construction, or from the cross population process. This has the effect of making all of the edges in our network purple. We have cross populated and attached in a manner to avoid creating

any new red vertex to red vertex edges, maintaining the original structure of our terrorist network. Our synthetic network is now completely purple, with the Noordin Top Network embedded in the purple network in a non-unique fashion.

Additionally, by tracking the information contained in each category, we have created a multiplex network. Each vertex in the category is purple. We do not know if it is red or blue until we place a monitor on it. To each vertex we associate a 10-tuple containing the information from the 10 categories used to build the purple network. A view of our network that includes information used to construct each category can be seen in Figure 3.12. The obvious goal is to then discover the terrorist vertices within this purple network, but first we will discuss some discovery research of others, and introduce our new algorithm that will discover the terrorists in our Multiplex network.

3.4 Network Discovery Algorithms

We have created a large synthetic network. We have embedded a real terrorist network within this large synthetic network. Our goal now is to create a discovery algorithm that targets the terrorist network. In Chapter 2, we discussed the Undiscovered Degree Algorithm (UDD), as a fast method to discover a network. We will now modify that algorithm to target red nodes, rather than just greedy on all nodes.

3.4.1 Multilayer Undiscovered Degree

As described in Chapter 1, in our scenario, we assume that monitors are very expensive to place in the network. This provides the motivation to discover red vertices as fast, and therefore as inexpensively, as possible. We also know that our synthetic network is a multiplex, a characteristic that we want to take advantage of. We will modify UDD and implement an algorithm we call Multilayer Undiscovered Degree (MUDD).

MUDD will operate similarly to UDD. The primary differences will be how much information monitors discover about neighbors, and what the algorithm does when a red vertex is discovered. Our network is purple. We do not initially know if a vertex is red or blue. When a monitor is placed on a vertex, the monitor discovers if the occupied vertex is red or blue. Consequently, since we do not know if the edge connecting to vertices is red, original

to the Noordin Top Network, or blue, original to one of our blue layers, we only know edges to be purple. The monitor also discovers all of the immediate neighbors, but it only discovers the neighbors as purple and what their degree is (again a purple degree).

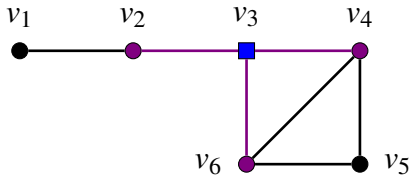


Figure 3.16: MUDD when the monitor discovers a blue vertex. In this scenario, the algorithm continues to use the full purple network as shown in Figure 3.17. Adapted from [34].

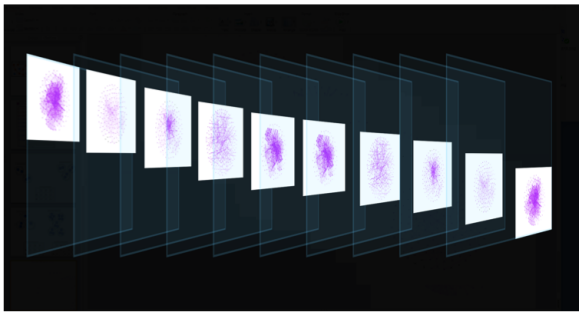


Figure 3.17: Full purple multilayer network. When a MUDD monitor discovers a blue vertex, it will continue to evaluate this network.

In Figure 3.16, the monitor has discovered v_3 as a blue vertex. This monitor discovers v_2,v_4, and v_6 as purple vertices. It evaluates the neighborhoods of each v_i counting the number of undiscovered neighbors of v_i using the full 10 layer purple network. This is similar to the standard UDD algorithm.

The primary difference between MUDD and UDD is when a red vertex is discovered. We see the discovery of a red vertex in Figure 3.18. The MUDD algorithm continues to evaluate the neighborhood of each adjacent vertex, except it considers on the neighborhood in the target layers, shown in Figure 3.19.

This reflects a realistic scenario: if the monitor represents capturing a terrorist, by capturing the terrorist we would find out who the terrorist is connected to. Not necessarily everyone that a terrorist is connected to are terrorists; a non-terorist may live next door to a terrorist. Similarly, a person my have gone to school with a terrorist, and this also does not mean

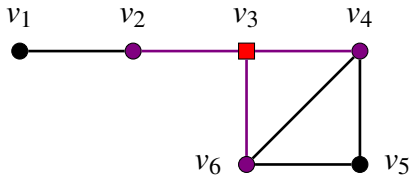


Figure 3.18: MUDD when the monitor discovers a red vertex. Once the red vertex is discovered, the algorithm now evaluates the network in the three target layers, as shown in Figure 3.19. Adapted from [34].

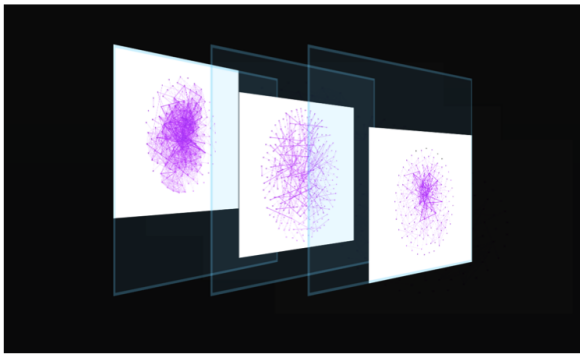


Figure 3.19: MUDD algorithm evaluating the target layers only. When a MUDD monitor discovers a red vertex, it will evaluate the target layers of the multilayer network.

he/she is a terrorist. A terrorist may even call somebody on the phone for a completely innocuous reason. The fact that individuals talked on the phone to a terrorist does not make them terrorists. However, if an individual calls a terrorist, went to school with a terrorist, and lives next door to a terrorist, there is a good chance that the individual is also a terrorist. In fact with all of those connections to a terrorist, it is probably worth the expensive monitor to place a monitor on that vertex in the network. We will use this concept to build our "equation for a terrorist", which will in turn guide our MUDD inference.

In our MUDD algorithm, the user has the flexibility to select which categories will be used in an "equation for a terrorist." We have purposely built our synthetic network so that categories R1, R2, and R3 are our equation for a terrorist. MUDD can be modified for use on general multilayered networks. Perhaps in one region, say Iraq, the important layers are Sunni vs Shia Mosque, Classmates, and Communication. In a different region, like Afghanistan, the important layers could be Tribe, Training, and Travel. MUDD has the flexibility to be modified by the user to take advantage of the Multilayered construct of the

network of interest, and differences in the key factors of the population that makes up the network.

Once the monitor discovers a red vertex, it uses a different set of criteria than UDD uses to select the placement of the next monitor. MUDD no longer uses the undiscovered degree of neighbors for red vertices. Instead, MUDD only considers neighbors in all three layers, R1, R2, and R3. Among the neighbors, MUDD then selects the neighbor with the highest degree as the sum of the degrees in the three target layers. In a sense, once MUDD finds a red vertex, it only chooses a neighbor that has neighbors in the layers of interest as this neighbor is likely to be a red node as well. Note that blue nodes can have high degree as the sum of the degrees in the three target layers. Therefore, we are more likely to get red nodes since we guide our inference with the layers of interest, but as we will see, there are many blue nodes that have a high degree as the sum of the degrees in the three target layers.

Since MUDD focuses on targeting red vertices, we expect MUDD to find red vertices faster than UDD, thus outperform UDD when the measure is count of red vertices discovered.

3.4.2 Probabilistic MUDD

MUDD is predicated on a similar concept as UDD, namely that we have the ability to know the undiscovered degree of all neighbors of a given vertex. With this assumption, UDD is a very effective algorithm to use in order to discover a network quickly. But this assumption might not hold in the real world, nor in dark networks. It is probably not realistic that we have perfect knowledge of all neighbors and their degrees.

Furthermore, if the network is a dark network, the members of the dark network are actively trying to conceal their connections. For these reasons, it is unlikely that we will have perfect knowledge of their neighbor's degree. We would like to evaluate our network in a more real-world environment. In order to do this, we need to degrade our perfect knowledge with in the network. This degradation of knowledge within the network is a potentially incredibly complex endeavor. Our approach will assign to each vertex in the network a random probability. This probability will be assigned from a normal distribution between 0 and 1. We will use this probability to degrade the perfect knowledge Probabilistic MUDD uses to choose the placement of the next monitor. When the Probabilistic

MUDD algorithm counts the number of undiscovered neighbors for each neighborhood of v_i, the count will be multiplied by the probability assigned to the vertex, v_i. This will have the effect of degrading the information the algorithm uses to determine placement of the next monitor. This can be seen in Figure 3.20. For example, when the algorithm evaluates v_4, there are two undiscovered neighbors, v_5 and v_7. But this count is multiplied by the probability assigned to v_4, 0.1. This results in a value of 0.2. When the algorithm evaluates the neighborhood of v_2, it discovers only one new vertex, v_1. This count of 1 is degraded by the probability assigned to v_2, 0.9. So while MUDD would select v_4 for the next monitor, Probabilistic MUDD would select v_2.

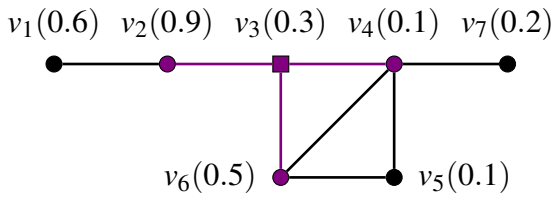


Figure 3.20: A monitor evaluating the count of undiscovered neighbors in the Probabilistic MUDD algorithm. Adapted from [34].

Other algorithms, such as Random Walk, UDD and MUDD, will not utilize this probability, so it will not affect their performances. Probabilistic MUDD will account for this probability. Probabilistic MUDD will calculate each vertex's probabilistic count of undiscovered neighbors. Since this probability is less than 1, each count will decrease. This will have the effect of degrading information across the whole network.

Using a Probabilistic MUDD will provide a more realistic application of MUDD. We would expect MUDD to out perform Probabilistic MUDD, because there is imperfect information being used in Probabilistic MUDD. An interesting result will be the outcome of comparing Probabilistic MUDD to UDD. Does Probabilistic MUDD's ability to take advantage of the Multilayer Networks outperform standard UDD, even when UDD uses perfect knowledge? This comparison will be interesting, because UDD does not differentiate between red and blue vertices, it is focused on discovering the network as quickly as it can by bringing in the most undiscovered neighbors. While MUDD Probabilistic attempts to focus on vertices with a higher likelihood of being red, but doing so with imperfect information about the structure of the network.

3.5 Comparing the Algorithms

The process of constructing a Multilayered Synthetic network was designed to avoid our Noordin Top Network as appearing uniquely in the network. We have also discussed multiple discovery algorithms. We have the goal of discovering red vertices as fast as possible. In order to test our new algorithm, MUDD, and its ability to discover red vertices, we will compare it against preexisting algorithms.

First we wil construct our synthetic multilayer network. We will then run 10 iterations each of Random Walk, UDD, MUDD and Probabilistic MUDD. Each iteration will start at a different randomly selected start vertex. We will capture the label of the vertices that each monitor is placed on, primarily focusing on whether the monitor discovers red or blue vertices. We will calculate the rate of red vertex discovery for each iteration.

Since each iteration starts at a different start vertex, we will average the discovery rates of each iteration for a given discovery algorithm. Additionally, we will calculate the standard deviation of each average discovery rate to analyze how much variation is present in each discovery algorithm. We will compare the rates of red vertex discovery of our four algorithms in the synthetic network.

Our synthetic networks have both random creation in some of the Blue layers (B5 and B6), and random assignment of probabilities in the degradation of information used by Probabilistic MUDD. For these reason, we will generate 10 versions of our synthetic network. We will compare our discovery algorithms in the same manner highlighted earlier. We will run 10 iterations of each algorithm, and compare the rates of red vertex discovery in each of the 10 synthetic multilayer networks.

Through this process we hope to gain insight in the ability of our algorithms, MUDD and Probabilistic MUDD, to take advantage of the multilayer network and discover red vertices quickly.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Analysis and Results

4.1 Results of Using a Single Layer to Identify a Target Community

In Chapter 3, we discussed conducting community detection in the Noordin Top Network. We used the Louvain method, highlighted in Chapter 2, and discovered 5 major communities. We show the results of the community detection in Figures 4.1 and 4.2.

Figure 4.1: An iteration of Louvain community detection on the full Noordin Top Network. This image was created using the Gephi Network Visualization Software.

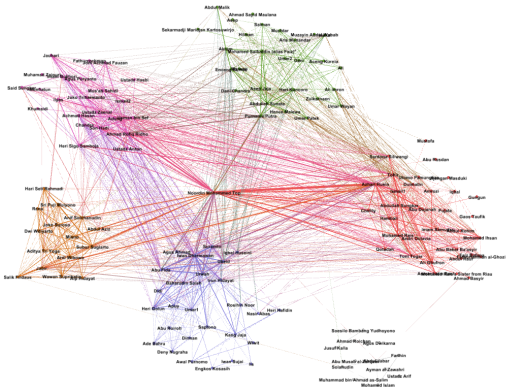


Figure 4.2: An iteration of Louvain community detection on the full Noordin Top Network, identifying the names of the individuals.

The colors assigned to the communities in any of the figures are randomly assigned by the visualization software, and serve solely for identification of communities purposes. We conducted this community detection multiple times, and the most consistent result is displayed in Figures 4.1 and 4.2. Both of these figures display the most common community structure we find. When we conducted different iterations of Louvain method community detection, there were not many differences. There are slight differences in the vertices that are not included in the five largest communities, which result in very consistent findings when analyzing the five largest communities. The only significant change we found was when Noordin Top was placed in a different community than the one shown in Figures 4.1

and 4.2. The reason Noordin Top is sometimes placed in other communities is because he is connected at such a high degree. Noordin Top has many connections, so sometimes the Louvain method would assign him to a different community. We choose to use the iterations where he was assigned to the same community repeatedly. In Figures 4.1 and 4.2, it is the community colored red. In Figure 4.2 we see the names of the terrorists associated with each vertex displayed. We see that Noordin Top is the most connected member of the network, and that he is connected to all of the communities. Again, this reflects the supporting literature [26], namely that Noordin Top is a member of one of the groups, Jemaah Islamiyah (JI), yet acts as a coordinator among all five groups.

When we look closely at the community structure of the network, we see that some of the communities have members that are leaves of a community. When we call a vertex a leaf, we mean that it has degree 1. This is best seen in Figure 4.1, when looking at the green and red communities on the right of the image.

We apply the same Louvain method on the Communication layer of the Noordin Top Network. This time, there are not five major communities, but six. There is no one-to-one correspondence of communities in the Communication Layer to the communities in the full network. The results of the community detection in the Communication Layer are shown in Figures 4.3 and 4.4.

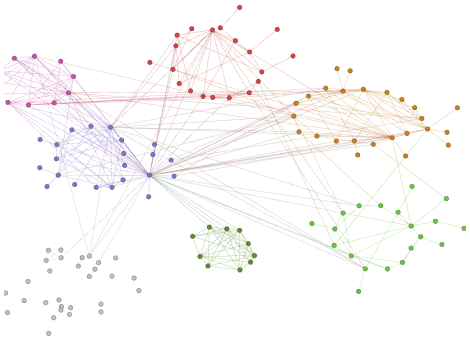


Figure 4.3: One of iteration of community detection on the Communication Layer. This image was created using the Gephi Network Visualization Software.

Figure 4.4: One of iteration of community detection on the Communication Layer. This image was created using the Gephi Network Visualization Software.

When we analyze these layers, again we see most communities have leaves. There is

one community in each iteration that does not have any leaves. Observe, in Figure 4.3 the dark green community at the bottom of the image, and in Figure 4.4 the dark blue community at the top of the image. In both cases, these communities do not have any leaves. Additionally, these communities seem to be highly connected to each other, almost forming a clique. Finally, these communities have comparatively few connections outside the community to other vertices. When we look at the names associated with these vertices in these communities, we find that it is the same individuals that make up this community that has no leaves. We conducted this community detection ten times, and our results are identical. The same ten individuals are in the same community in our Communication layer.

We will consider this community without leaves our target community, and attempt to identify the target community in the full network, using only the communication layer. We will use the technique, highlighted in Chapter 3, of treating each community as a sub-graph. We used the metrics mentioned in Chapter 2, and found the following results. We have isolated each community as a sub-graph, and Gephi assigned an arbitrary color label to the community. We do this in both the Communication Layer and in the full network. The results of the topological metrics of the isolated sub-graphs are displayed in Tables 4.1 and 4.2.

Full Network	Red	Green	Purple	Pink	Orange	
Vertices	32	29	25	23	15	
Edges	220	152	127	126	105	
Density	0.417	0.374	0.423	0.498	1	
Clustering	0.781	0.788	0.829	0.863	1	
Communication	**Red**	**Green**	**Purple**	**Pink**	**Orange**	**Blue**
Vertices	26	22	20	20	12	10
Edges	47	47	24	33	28	37
Density	0.145	0.203	0.126	0.174	0.424	0.822
Clustering	0.589	0.657	0.355	0.562	0.766	0.833

Table 4.1: Metrics for the community sub-graphs. The top portion of the table corresponds to the full network, the bottom portion corresponds to the communication layer.

We conducted community detection, and isolated the communities as sub-graphs multiple times, but the results are very consistent. We can see in Tables 4.1 and 4.2 that there is one community that has a comparatively higher clustering coefficient and is denser than the rest

Full Network	Red	Green	Purple	Pink	Orange	
Vertices	34	28	21	19	15	
Edges	184	151	85	114	105	
Density	0.328	0.399	0.405	0.667	1	
Clustering	0.751	0.794	0.844	0.891	1	
Communication	**Red**	**Green**	**Purple**	**Pink**	**Orange**	**Blue**
Vertices	27	26	18	17	10	9
Edges	48	64	28	20	37	19
Density	0.137	0.197	0.163	0.147	0.822	0.528
Clustering	0.506	0.612	0.410	0.447	0.833	0.786

Table 4.2: Metrics for the community sub-graphs for another iteration of community detection. The top portion of the table corresponds to the full network, the bottom portion corresponds to the communication layer.

of the communities, in both the full network and in the Communication layer. This pattern of a single community being denser with a higher cluster coefficient is present in all of the iterations of community detection that we conducted. We were able to identify this community from the Communication layer, and it is displayed in Figure 4.5. We have displayed the names associated with this community from the Communication layer. These same ten individuals appear every time we conduct community detection in the Communication layer.

This target community in the Communication Layer is almost a clique, which is the reason for the high clustering coefficient and the density values displayed in Tables 4.1 and 4.2. We have also displayed the names from the target community in the full network in Figures 4.6 and 4.7. We can observe that the same ten individuals in the target community from the Communication layer are all present in the target community in the full network. Depending on the iteration of community detection in the full network, there are five or six individuals present in the target community that are not part of the target community in the communication layer. This means that while the Communication layer has only approximately 20% of the of the edge list that the full network has, we can use community detection in a single layer of the network to find over 60% of our target community.

It is interesting that we are able to identify this target community consistently, but perhaps somewhat surprising is what this target community represents. The chosen target community is a specific terrorist cell responsible for an attack in Bali in 2005 [26]. This terrorist

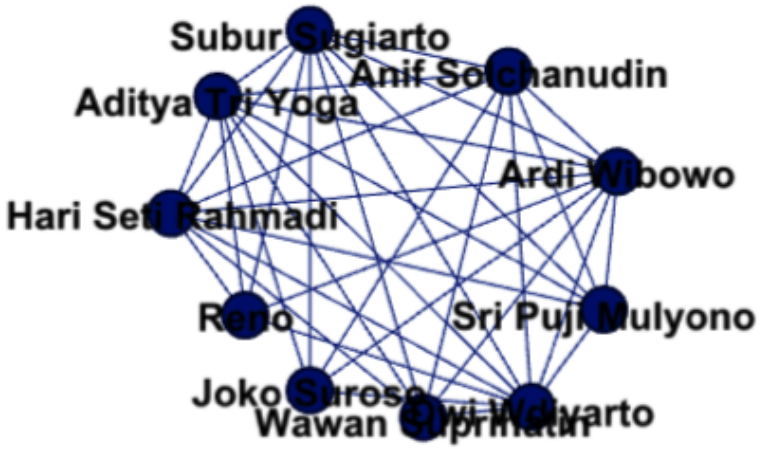


Figure 4.5: Target community discovered in the Communication Layer. This image was created using the Gephi Network Visualization Software.

cell attacked three cafes in Bali with suicide bombers in October 2005. We were able to conduct community detection on a single layer of the network, and identify a terrorist cell that has carried out attacks in the past. Of further note, the three suicide bombers were Salik Firdaus, Misno, Aip Hidayat. These are three of the individuals that appear in the target community in the full network, but not in the target community in the Communication layer. This highlights the process of recruiting suicide bombers by one of the terrorist groups in the Noordin Top Network, described in [26]. These individuals are not as closely associated with the rest of the community, and when we conduct community detection in only one of the layers, we do not include them in our target community. This is an indication that not only are we able to identify a specific terrorist cell, but when we compare the results of a single layer with the results from the full multilayer network, we can identify members of the terrorist cell that have different attributes than the majority of the members of the terrorist cell. In this case, the suicide bombers are not part of the original terrorist cell, they are recruited in for a specific operation. This differentiation among the members of the cell can be detected by using specific layers of the multilayer network.

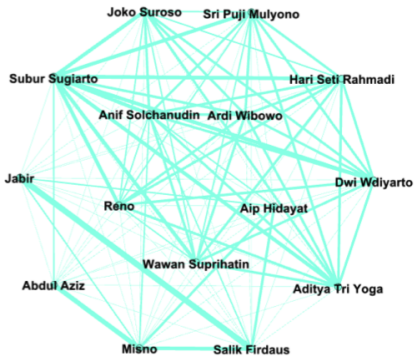


Figure 4.6: Target community discovered in the Communication Layer. This image was created using the Gephi Network Visualization Software.

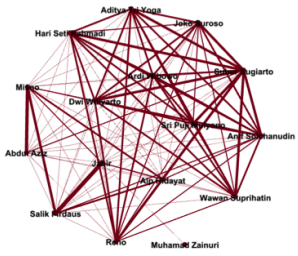


Figure 4.7: Another version of the target community discovered in the Communication Layer. This image was created using the Gephi Network Visualization Software.

4.2 Using each of the 14 Layers in Community Identification

We are able to identify a target community in the full network, using only one layer of information, but this might be more difficult for arbitrary communities. In our previous example, the target community had a significantly higher clustering coefficient and it was much denser than the other communities. This allowed us to identify it. There is, however, no one-to-one correlation for communities present in the full network to communities present in a single layer. So we cannot simply take the next denser community in Communication layer, and find the next denser community in the full network. We will evaluate all 14 layers from the multiplex, to see how well each layer can be used to determine the community structure of the full network. We will use the cluster adequacy described in Chapter 2, to evaluate how strong the community structure is in each layer. We use Normalized Mutual Information (NMI), from Chapter 2, to see how similar the community structure in a particular layer is to the community structure in the full network.

In Table 4.3, we see the results of the conducting community detection in each layer, the strength of the community structure, as well as the resulting NMI values after comparing the communities in a layer to the ground truth communities in the full network. When we look at the NMI values of the various layers, all layers have an NMI value between 0.6184 and 0.4879. This tells us there is not a significant difference between the commu-

Layer	Edges	Communities	NMI	Cluster Adequacy
Communication	318	25	0.5558	0.557
Meeting	88	110	0.5409	0.334
Operations	469	85	0.6184	0.519
Friendship	154	65	0.5494	0.720
Mentorship (Supervisory)	51	103	0.5501	0.537
Soulmates	17	129	0.5546	0.656
Mentorship (Technology)	13	130	0.5511	0.343
Recruiting	24	117	0.5712	0.759
Kinship	49	110	0.5589	0.875
Logistical Function	554	93	0.4879	0.280
Mentorship (Ideology)	15	126	0.5561	0.702
Training	263	88	0.6032	0.589
Logistical Place	97	112	0.5312	0.210
Classmates	205	101	0.5459	0.348

Table 4.3: Table displaying the number of edges, the number of communities, NMI, and Cluster Adequacy in all 14 layers of the Noordin Top Network.

nity structures of any individual layer compared to the structure of the full network. All of these individual layers have similar NMI values, even when they have very few edges. These similar values are due to the way that NMI is computed. We highlighted examples in Chapter 2 that showed potential limitations if there was a large difference in the number of vertices present in the two community structures being compared. While we have accounted for this limitation in the comparisons highlighted in Table 4.3, many of the vertices in layers with a small number of edges are not connected to the main component of the graph. Only vertices with attached edges contribute to the community structure, and there by impact the value of NMI we compute. These layers with small edge counts do contribute to building the community structure of the multiplex, they are just limited to contributing only with vertices that are attached. We can think of this concept in reference to network discovery, we cannot expect to get information about individuals that we have not discovered. In the case of these layers, when we look at a layer such as Soulmates, there only 17 edges. The community structure in this layer only contains information about the vertices that are included with these edges, while we can think of the remaining vertices as not having been discovered. We have not discovered who the soulmates of these

unconnected vertices are, so we cannot expect to get a complete picture of the community structure of the Mulitplex network from this layer, because there are so many soulmates still undiscovered.

For example, The Mentorship (Ideology) layer and the Communication layer only have a difference in NMI of 0.0003, despite the fact that the Communication layer has 318 edges and the Mentorship (Ideology) layer has only 15 edges. So while these two layers have a very different number of edges, they offer similar comparisons to the community structure of the full network. So while both layers offer similarly good pictures of the community structure, the structure of Mentorship (Ideology) is incomplete. We have not discovered enough of the relationships to give a full picture of the community structure. Mentorship (Ideology) does not give a bad picture of community structure, just a limited one.

When we look at the Cluster Adequacy, we are seeing a normalized modularity value for the communities in a layer. This represents how strong the community structure is, if there are many edges between communities the community structure is not as strong. In looking at the values for the Cluster Adequacy, we see that some have relatively high values. But this is slightly skewed by the number of communities present in the layer. For example, the highest Cluster Adequacy value is 0.875, for the Kinship layer. This might lead one to have a high confidence in the communities in the Kinship layer. Upon closer inspection, we see there are 110 communities in that layer. Specifically, there are 9 communities with 2 vertices, 1 community with 3 vertices, 2 communities with 4 vertices, and 3 communities with 5 vertices each. This means there are 95 vertices in this layer that are in a community by themselves. Of the 139 vertices in the network, 95 of them are not truly assigned to any community. This demonstrates the potential weakness of using Cluster Adequacy for any conclusions we might attempt to draw from the community structure of an individual layer.

Both the results from the NMI and Cluster Adequacy from an individual layer leave us with the same conclusion. Namely, we can use the information contained within a single layer to conduct community detection that reflects the community structure of the full network. This reflection, however, is limited by the amount of information contained in a single layer. This is not to say that the community structure of a single layer is wrong, but it might be limited. We can use the partial information contained in a single layer, but we are only getting some of the picture of the community structure of the full network.

4.3 Topology of Synthetic Network

In Chapter 3, we discussed the construction of our synthetic multilayer network, with an embedding the Noordin Top Network in the synthetic network. The network that we created is large enough and dense enough, that an image of this network will not show us anything that we can use. The Synthetic network we created has 7514 vertices and 184,875 edges. This means that roughly 1 in 54 of the vertices is one of our terrorists from the Noordin Top Network. The network is composed of a single component, so we have created a network that is connected.

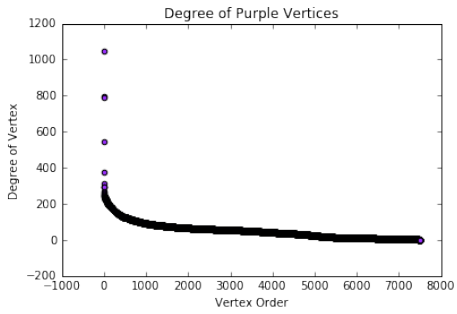


Figure 4.8: Degree sequence of the synthetic network. Specifically, the purple degree of each vertex, not distinguishing between red or blue vertices.

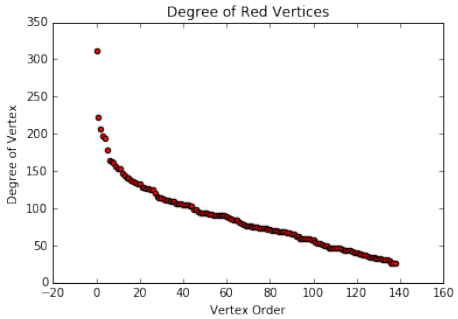


Figure 4.9: Degree sequence of the red (Terrorist) vertices in the synthetic network.

We display the purple degree sequence in Figure 4.8 and the red degree sequence in Figure 4.9. In these figures we have ordered the vertex list in an decreasing order with respect to the degree. The highest degree vertex from each network is the first listed, and decreasing in degree to the right, to the vertex with the lowest degree. These degree sequences exhibit the qualities of the Noordin Top Network that we wanted to retain, mainly that there are many vertices with low degree, and relatively few with a high degree. The red vertex with the highest degree is Noordin Top, consistent with what we observed in the original Noordin Top Network. Originally, Noordin Top has a degree of 233. After the creation of the network, Noordin Top has a degree of 362. In the original network, Noordin Top had a degree of 233, which is higher than the number of vertices in the network. This is possible, because there is a multiplicity of edges. There are some individuals that Noordin Top is connected to in multiple layers. We maintain this multiplicity of edges because it accurately depicts how strong a connection is between two edges. Subsequently, these

multiple edges have a chance to show up in the different categories used to construct the first three layers of our synthetic network, R1, R2, and R3. If Noordin Top is connected to an individual in multiple layers, and these connections show up in all three target layers of the synthetic network, we maintain this strong connection between the two individuals.

A comparison of the red vertices before the construction of the synthetic network and after the construction can be seen in Figures 4.10 and 4.11. In both of these figures, Noordin Top has the highest degree. Noordin's Degree has increase from 233 in the original terrorist network, to a degree of 362 in the synthetic network. While Noordin Top has a large degree, he doesn't have the maximum degree in the synthetic network. We took particular care in the construction of the synthetic network to ensure that no new edges were created between red nodes, so these additional 132 edges attached to Noordin Top originate from blue vertices. Another observation from the red degree sequences are the increase in connections among the less connected vertices, i.e., the vertices on the right side of each sequence. In the original network, there were many vertices that had a very low degree. Six had degree 0, eight with degree 1, and 35 vertices in total with a degree of 5 or less. This represents approximately 25% of the network, while there were a lot of vertices that were hubs with a high degree, such as Noordin Top. After we have constructed our synthetic network, There are no vertices with a degree lower than 10. As we have already stated, this means all of the new connections to previously poorly connected vertices are connections coming from blue vertices. The hubs that were previously present in the Noordin Top Network are still hubs after the construction of the synthetic network. Previously, there were only 3 vertices with degree higher than 150, now there are many with a degree of 150 or higher.

When we analyze Figure 4.8, there are a few vertices with very high degree, more than 250. Noordin Top is one of these, but the remaining hubs in the purple network are not red vertices. In fact, some of these other hubs in purple network are blue copies of the Noordin Top vertex, originating from blue layers $B1, B2, B3$, or $B4$. When we analyze the purple degree sequence, there are no portions that jump out as out of place. We see a smooth line with no easily identifiable portion of the graph that stands out as unique.

In addition to the degree sequence we looked at the community structure of the synthetic network. A chart displaying the size of each community can be seen in Figure 4.12. We can observe that there is one community significantly larger than the rest. In Figure 4.12, this

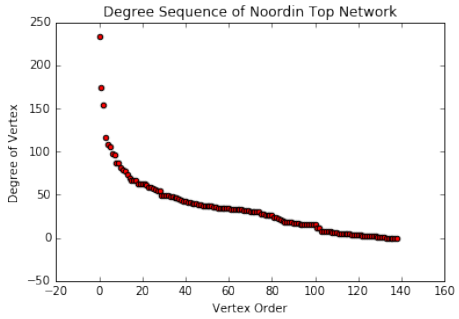


Figure 4.10: Degree sequence of the Noordin Top Network before we constructed the synthetic network.

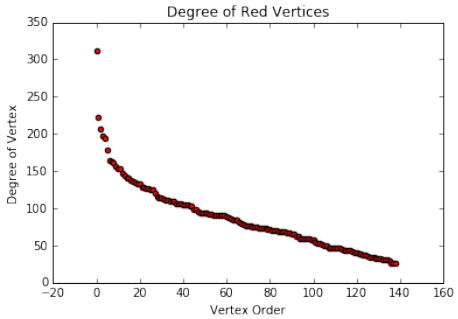


Figure 4.11: Degree sequence of the red vertices in the synthetic network, after the construction of the synthetic network.

Figure 4.12: Communities present in the synthetic network. The x axis, Modularity Class, is each community. The y axis is the size of each community. This image was created using the Gephi Network Visualization Software.

is identified here as community 2. This community has nearly 4,000 of the vertices from the network. More importantly, this community has all of the red vertices. The terrorist network was embedded into the network well enough that, when we conduct community detection, the terrorists do not stand out as unique nor as a community of the synthetic network.

These observations about the topology of the synthetic network lead us to conclude that we have successfully created a multilayer network that has a terrorist network hidden non-

trivially within it. The terrorists are connected to enough blue vertices, and connected in such a way that we can not simply identify the terrorist network as a distinct portion of the synthetic network. The Noordin Top Network does not stand out as unique, like the target community stood out as unique in the Communication layer. While our synthetic network is not an attempt to create a model of the real world, and the connections within a real world network; we have created a network in a realistic manner that makes our terrorist network not unique, and difficult to find. We want the discovery of our terrorist network to be as difficult as possible in order to mimic reality.

4.4 Four Discovery Algorithms Discovering Red Vertices in a Synthetic Network

We introduced two algorithms in Chapter 2: Random Walk (RW) and Undiscovered Degree (UDD). We also introduced two modified versions of UDD in Chapter 3, Multilayer UDD (MUDD), and Probabilistic MUDD. We will run each discovery algorithm through the synthetic network 10 times. Each iteration of the discovery algorithm will start on a randomly selected node. We will capture the rate at which each algorithm discovers red vertices.

In Figure 4.13 we see the results of 10 iterations of red vertices discovery using Random Walk. Each iteration starts at a randomly selected start vertex. The x axis has the ordering of the monitors as they placed in the network. The y axis shows the percentage of red vertices discovered, not keeping track of the blue ones. The red discovery is tracked for each monitor placed in the network, up to half of the node count in the network. In the case of our synthetic network, we have over 7,500 vertices, so the algorithm will place nearly 3,750 monitors in the discovery process.

From this chart, we can see that after placing monitors randomly in the network, we discover approximately half of the red vertices present in the network when we place monitors on about half of the total number of nodes. This is what we would expect to find if there was no inherent bias in selecting a red vertex compared to selecting a blue vertex. This reinforces our conclusion that we have generated a synthetic network that has effectively embedded our terrorist network among non-terrorist vertices. When we look at Figure 4.14,

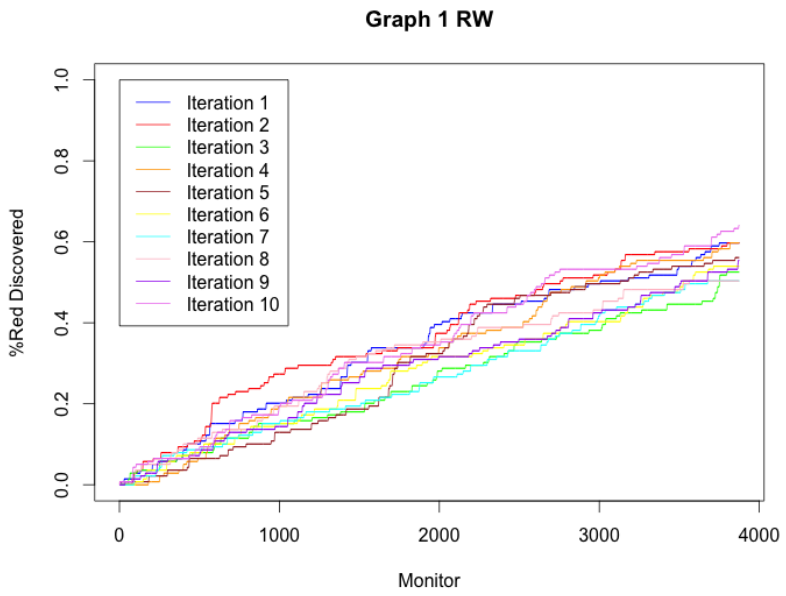


Figure 4.13: 10 iterations of RW discovering red vertices in the synthetic network (Graph1).

we see the average rate of red vertex discovery. We have included our average plus and minus one standard deviation. This shows us how much variation there is in the rate of red vertex discovery with respect to which randomly selected start vertex we initiate our discovery algorithm from. We will use this average later when we compare all four discovery algorithms. In Table 4.4 we highlight the rate of red vertex discovery using Random Walk in the first synthetic network we generate. The algorithm is limited to using 3,750 monitors, and this restriction does not allow RW to discover more than half of the red vertices present in the network.

% Red Found	Monitor
50%	3300
70%	NA
80%	NA
90%	NA

Table 4.4: Table showing how many monitors required for Random Walk to discover a given percentage of red vertices.

We used the discovery algorithm Undiscovered Degree (UDD), in the same manner. We

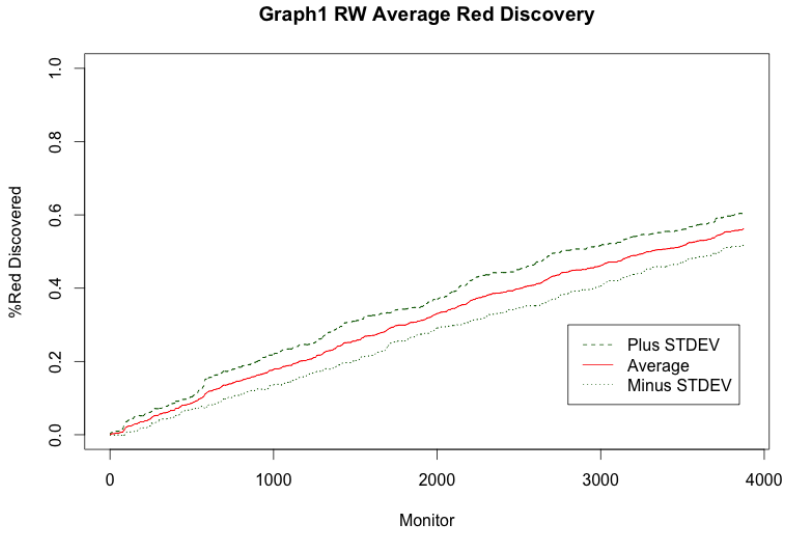


Figure 4.14: Average rate of red vertex discovery, as well as plus and minus one standard deviation, for the 10 iterations of RW in the synthetic network (Graph1).

started UDD from a randomly selected start vertex, and tracked the red vertex discovery for 10 iterations. We captured this discovery rate, as well as the standard deviation for the discovery rate. These results are displayed in the Figure 4.15 and Figure 4.16.

When we look at the results in Figure 4.15, one of the first things we observe is how similar the rate of red discovery is, regardless of the start vertex and of the choices of next monitor. This is further highlighted in Figure 4.16, when we display the Standard Deviation and the average. These three curves are nearly identical, highlighting how little variation we see in the rate of red vertex discovery.

The UDD algorithm is only evaluating each vertex as purple, and does not seek out red vertices. Despite this disregard for vertex color, UDD does very well discovering red vertices quickly. UDD discovers 50% of the red vertices when it places the 236^{th} monitor in the network. This might seem to be very fast, but as a reminder, many of the vertices in the Noordin Top Network are hubs with high degrees. In [34], UDD is shown to discover hubs faster than other vertices. The use of UDD in our synthetic network shows similar results. This high rate of red vertex discovery continues, with UDD discovering 70% of the red vertices with the placement of the 501^{th} monitor. The rate of discovery does sig-

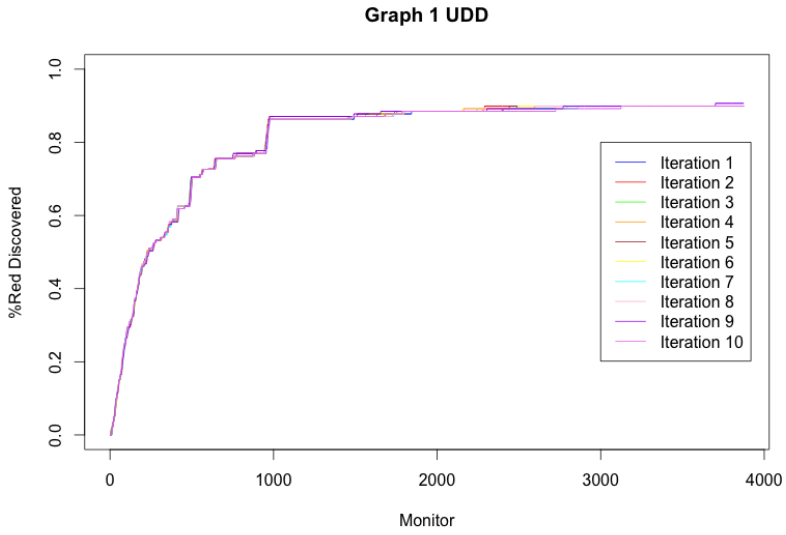


Figure 4.15: 10 iterations of UDD discovering red vertices in the synthetic network (Graph1).

nificantly slow down after the placement of the 978th monitor, discovering 86% of the red vertices. This corresponds to discovery of 120 of the 139 terrorists present in the synthetic network. These are the terrorists that had very low degree from the original Noordin Top Network. When we look at the original network, there are 20 vertices that have a degree of 2 or less. These are the vertices that UDD has not found after 978 monitors. We highlight these results in Table 4.5.

% Red Found	Monitor
50%	236
70%	501
80%	956
90%	3701

Table 4.5: Table showing how many monitors are required for UDD to discover a given percentage of red vertices.

We have tested the two previously documented algorithms now we will use our modified algorithms, and see their ability to discover red vertices quickly. First we will use the Multilayer UDD (MUDD) algorithm, 10 times in our synthetic network. This algorithm can only evaluate neighbors as purple, but attempts to specifically targets red vertices once it has discovered a red vertex.

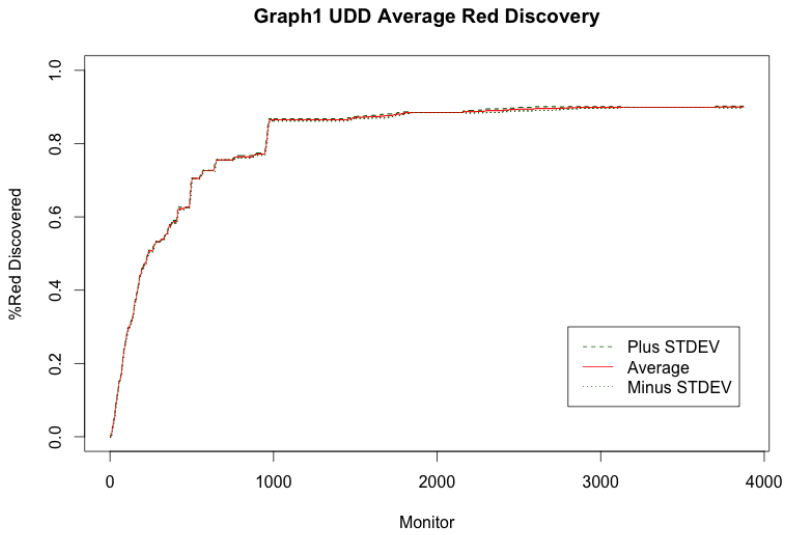


Figure 4.16: Average rate of red vertex discovery, as well as plus and minus one standard deviation, for the 10 iterations of UDD in the synthetic network (Graph1).

We display the results of the 10 iterations of MUDD in Figures 4.17 and 4.18. These results seem to be very similar to the results of UDD. There is very little variation in the rate of discovery, regardless of where the algorithm starts. Again, the algorithm seems to discover red vertices very fast, due to the fact that there are a lot of hubs in the Noordin Top Network. And finally, there seems to be a point at which the algorithm's rate of red discovery slows down greatly.

Specifically, MUDD discovers 50% of the red vertices with the placement of the 257^{th} monitor, 70% with the placement of the 478^{th} monitor, and slows significantly with the discovery of the 127^{th} red vertex (92%) with the placement of the 1454^{th} monitor. We highlight these results in Table 4.6.

Our final algorithm used to discover red vertices in our synthetic network is Probabilistic MUDD. This is the algorithm that we use to better approximate a real world situation, in which we do not have perfect information about all of the neighbors of a particular vertex when we are selecting a candidate for the placement of the next monitor. The probabilities assigned to the vertices are randomly selected, of note in this synthetic network (Graph 1), Noordin Top was assigned a probability of 0.6.

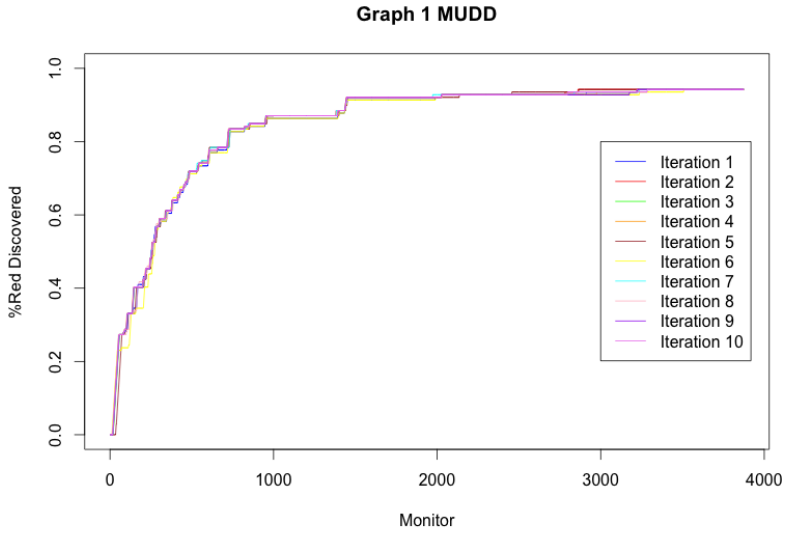


Figure 4.17: 10 iterations of MUDD discovering red vertices in the synthetic network (Graph1).

% Red Found	Monitor
50%	257
70%	478
80%	721
90%	1440

Table 4.6: Table showing how many monitors are required for MUDD to discover a given percentage of red vertices.

We have displayed the results for 10 iterations of red vertex discovery in our synthetic network (Graph 1) using Probabilistic MUDD in Figures 4.19 and 4.20. Probabilistic MUDD seems to have similar results to UDD and MUDD. It is able to discover the large degree red vertices quite quickly. There is a higher amount in variance in the discovery rate, when compared to UDD and MUDD due to the use of probabilities, but there is still very little variance in the rate of red discovery with respect to the start vertex. Specific to Probabilistic MUDD discovering red vertices in Graph 1, 50% of the red vertices are discovered with the 167^{th} monitor, 75% are discovered with the 532^{nd} monitor, and the discovery slows once 91% of the red vertices are discovered. Once again, we highlight these results in Table 4.7.

There seems to be a higher amount of variation, when compared to UDD or MUDD due to

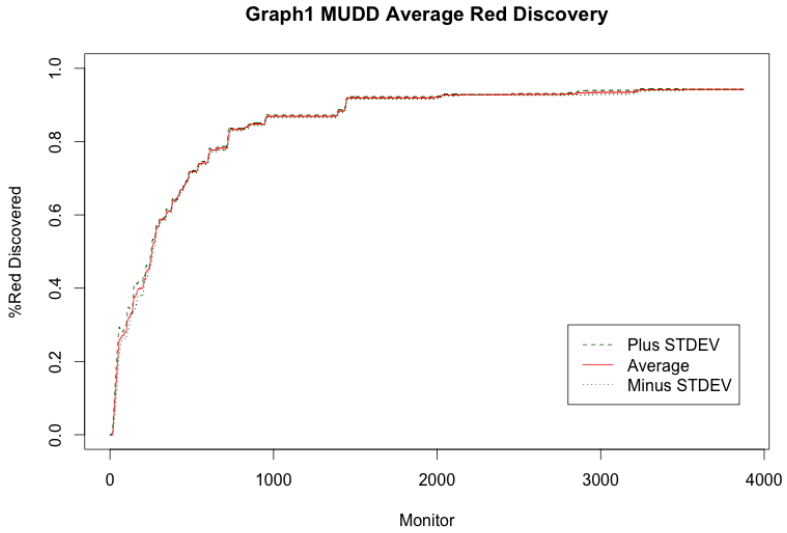


Figure 4.18: Average rate of red vertex discovery, as well as plus and minus one standard deviation, for the 10 iterations of MUDD in the synthetic network (Graph1).

% Red Found	Monitor
50%	167
70%	457
80%	731
90%	1452

Table 4.7: Table showing how many monitors are required for Probabilistic MUDD to discover a given percentage of red vertices.

the probabilities assigned to each vertex, not the vertex the algorithm starts on. There is still significantly less variation than Random Walk. Of the four algorithms we have observed, the only one that has any significant variation due to the start vertex is Random Walk.

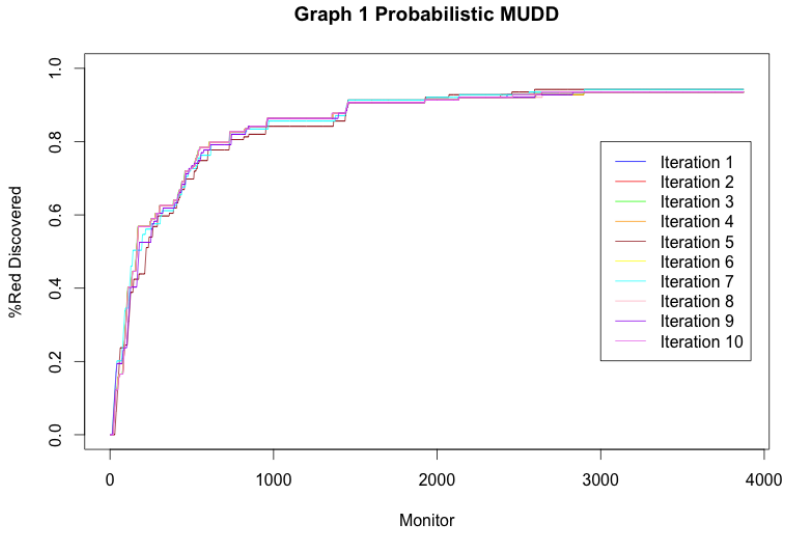


Figure 4.19: 10 iterations of Probabilistic MUDD discovering red vertices in the synthetic network (Graph1).

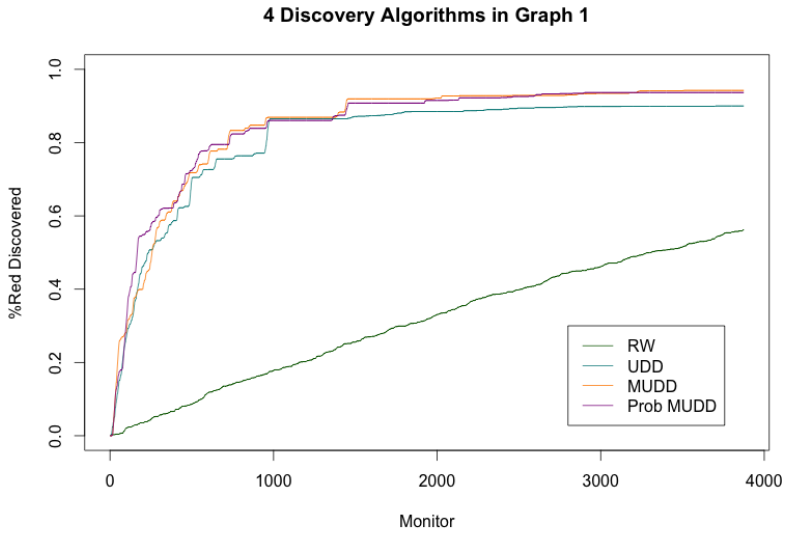


Figure 4.21: Average rate of red vertex discovery for all four algorithms (RW, UDD, MUDD, and Probablistic MUDD) in the synthetic network (Graph1).

With all four algorithms run through the synthetic network 10 times, we can compare the average rate of red discovery between the four algorithms. This is shown in Figure 4.21.

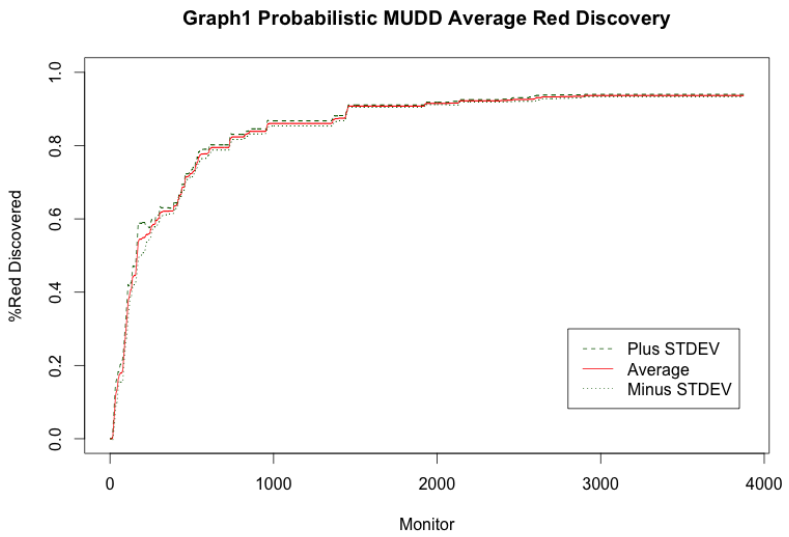


Figure 4.20: Average rate of red vertex discovery, as well as plus and minus one standard deviation, for the 10 iterations of Probabilistic MUDD in the synthetic network (Graph1).

When we look at the comparison of the four discovery algorithms, we see that UDD, MUDD, and Probabilistic MUDD all significantly out perform RW. It appears that UDD, MUDD, and Probabilistic MUDD all have a similar rate of discovery. We can see that MUDD and Probabilistic MUDD both out perform UDD. Furthermore, we see that once the algorithms have discovered approximately half of the network, Monitor 3700, MUDD and Probabilistic MUDD have discovered more red vertices than UDD.

4.5 Results of Red Vertex Discovery Across 10 Synthetic Networks

We ran each algorithm 10 times through this network, and saw little variation with regard to the start vertex for the algorithm. Our synthetic network, however, also has some random qualities in some layers, randomness in the edges between vertices when we cross populated vertices from layers, and random probabilities assigned to each vertex for use by the Probabilistic MUDD algorithm. For these reason, we study 9 additional synthetic networks. We run each discovery algorithm through each synthetic network to see if there are any significant impacts from the randomness of our synthetic network.

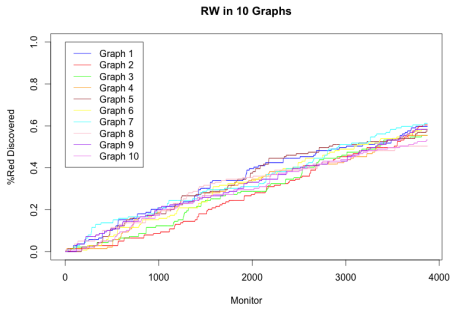


Figure 4.22: Rate of red vertex discovery for Random Walk through all 10 synthetic networks (graphs).

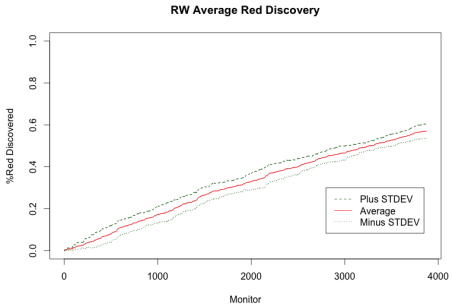


Figure 4.23: The average and standard deviation for RW's rate of red vertex discovery through all 10 synthetic networks (graphs).

First we will look at Random Walk discovery through all 10 synthetic networks (graphs), as well as the average rate of red vertex discovery. We see the results plotted in Figures 4.22 and 4.23, showing very little variation for the discovery rates even in the case of a random walk.

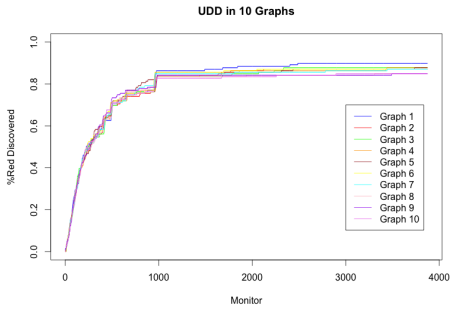


Figure 4.24: Rate of red vertex discovery for Undiscovered Degree through all 10 synthetic networks (graphs).

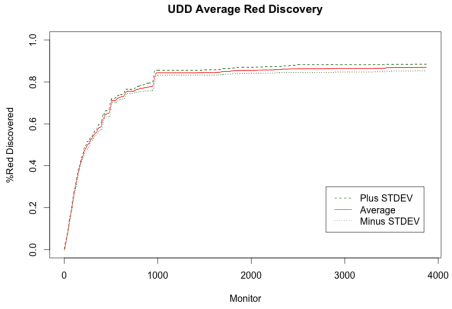


Figure 4.25: The average and standard deviation for UDD's rate of red vertex discovery through all 10 synthetic networks (graphs).

We conduct the same process for UDD, plotting the discovered red vertices in 10 different synthetic networks. We see the results in Figures 4.24 and 4.25, also showing little variation in the rate of discovery. We do see some difference in the total number of red vertices discovered; these are the red vertices that have low degree (2 or less). We see this after 2000 monitors have been placed. This difference is most likely due to how connected the previously poorly connected vertices were connected in the larger synthetic network. But

there is still a relatively fast discover of red vertices, this again is due to the number of highly connected vertices in the Noordin Top Network.

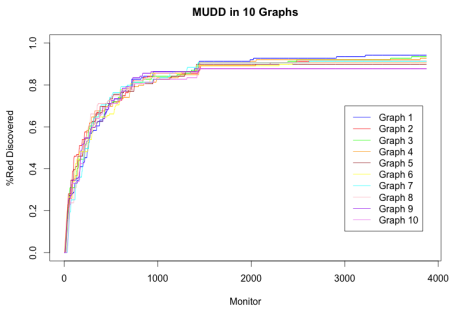


Figure 4.26: Rate of red vertex discovery for Multilayer Undiscovered Degree through all 10 synthetic networks (graphs).

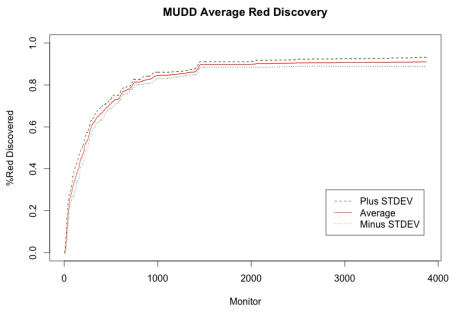


Figure 4.27: The average and standard deviation for MUDD's rate of red vertex discovery through all 10 synthetic networks (graphs).

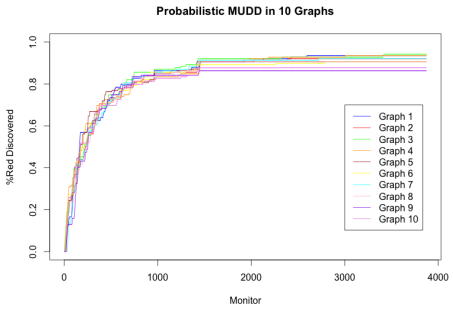


Figure 4.28: Rate of red vertex discovery for Probabilistic MUDD through all 10 synthetic networks (graphs).

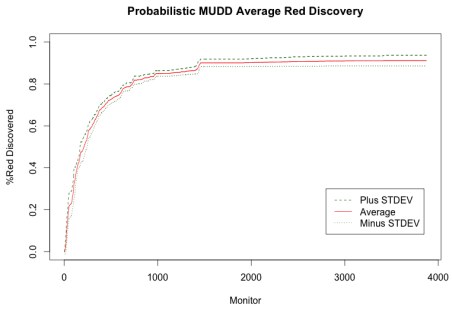


Figure 4.29: The average and standard deviation for Prob MUDD's rate of red vertex discovery through all 10 synthetic networks (graphs).

The results of the modified discovery algorithms are displayed in Figures 4.26 and 4.27 for MUDD, and in Figures 4.28 and 4.29 for Probabilistic MUDD. In these two sets of figures, we again see discovery rates that are similar to UDD, but both MUDD and Probabilistic MUDD discover red vertices at a faster rate than UDD. There is more variance in the Probabilistic MUDD discovery rates, showing us that there is indeed some impact on the algorithm's ability to discover red vertices.

We will compare all four algorithms with each other, by looking at the average rate of red vertex discovery through all 10 synthetic networks. This comparison is shown in Fig-

ure 4.30. In this graph we can see that Random Walk is outperformed by all of the other algorithms. We also see that while UDD does discover red vertices quickly, MUDD and Probabilistic MUDD are able to discover the red vertices at a higher rate. MUDD and Probabilistic MUDD have very similar discovery rates, and once averaged across all then synthetic networks, there are only a few differences between the two. Probabilistic MUDD is an attempt to degrade the use of perfect knowledge. Seeing so little difference between MUDD and Probabilistic MUDD is surprising, and perhaps an indication that our degradation of perfect knowledge was not as effective as we intended.

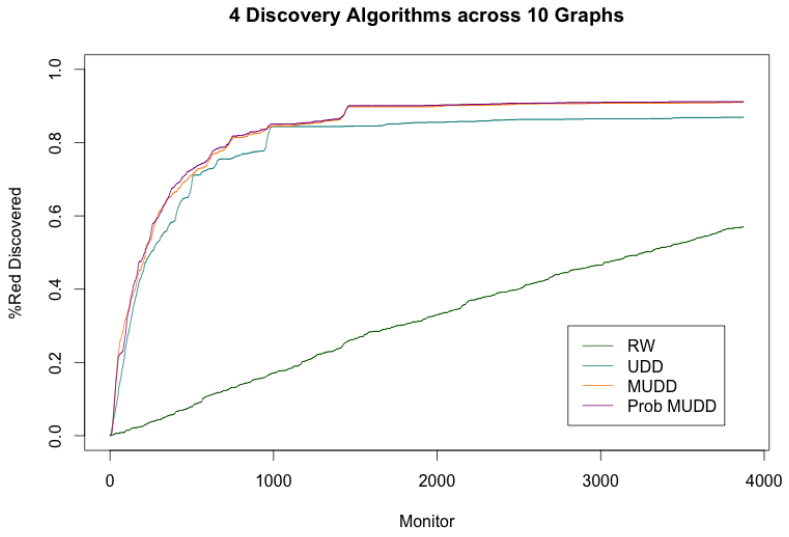


Figure 4.30: This chart shows the average rate of red vertex discovery for all four algorithms (RW, UDD, MUDD, and Probabilistic MUDD) across all 10 synthetic networks.

We show a comparison of some discovery rates in Table 4.8. This table compares each algorithm's ability to discover a given percentage of the red vertices present in the network. A few notes from this table, Random Walk takes a long time to discover half of the red vertices, and after discovering half of the network, the algorithm has only discovered half of the red vertices. This reinforces our construction of the network. If we saw a higher or lower rate of discovery, it would indicate some bias in our embedding of the terrorist network. We see that UDD is out preformed by both MUDD and Prob MUDD, which was our hope in creating the modified algorithms. We would like to point out that, on average, UDD will not discover 90% of the red vertices. Our algorithms all default to discovering

half of the vertices available in the network, this limits the number of monitors we can place. Of course, if we used one monitor for every vertex, we would eventually discover all red vertices. Additionally, we see a very slight difference between MUDD and Probabilistic MUDD. There appears to be a large distinction in each algorithm's ability to reach 90% of red vertices discovered, but this is not very significant. There is a difference of a single vertex, and it just takes more monitors for MUDD to find this additional vertex.

% Red Found	RW	UDD	MUDD	Prob MUDD
50%	3263	250	217	213
70%	NA	503	475	435
80%	NA	957	731	727
90%	NA	NA	2025	1457

Table 4.8: Table showing how many monitors it takes each algorithm to discover a given percentage of the red vertices in a synthetic network.

In Chapter 1 we discussed the idea of monitors being expensive, and the desire to discover the network as fast, and therefore, inexpensively as possible. For that reason, we will also focus on the first 250 monitors of the discovery process. All three of our more efficient algorithms discover 50% of the terrorist network by this point.

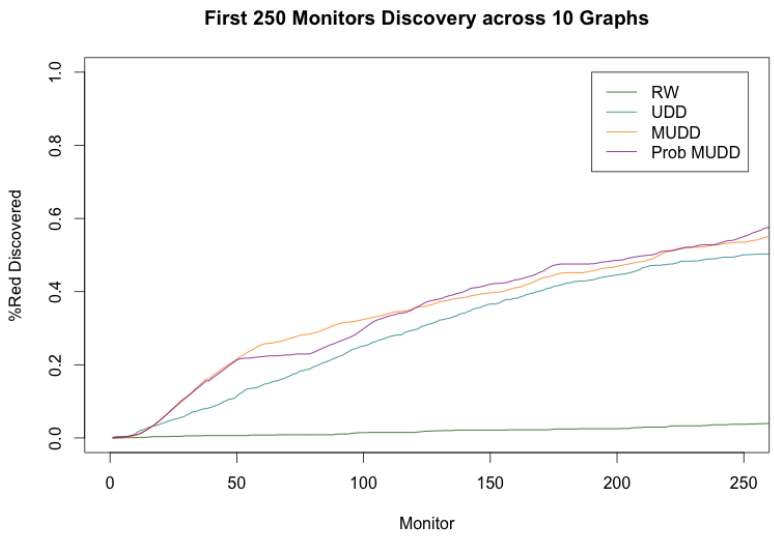


Figure 4.31: This chart shows the average rate of red vertex discovery for all four algorithms (RW, UDD, MUDD, and Probabilistic MUDD) in the first 250 Monitors.

We see the results of focusing on the first 250 monitors in Figure 4.31. After 250 monitors, Random Walk has barely discovered any of the red vertices. UDD, which focuses on vertices with a high degree, has found approximately half of the terrorist network. We can also see that MUDD and Probabilistic MUDD both discover red vertices faster than UDD. There is some difference between MUDD and Probabilistic MUDD, but each out performs the other at different points. This oscillation between the two is due to the probability associated with Probabilistic MUDD.

Something else that comes to light when we examine the first 250 monitors of discovery is the the fact that on average, none of the algorithms start discovering red vertices immediately. This is because they all start on a randomly selected vertex. If there are more than 7,500 vertices, and fewer than 150 are red, it is unlikely that we start an algorithm on a red vertex. In reality, however, we are not likely to start randomly exploring a network hoping to find a terrorist. It is more likely we will begin searching the network starting with a red vertex. For that reason, we will start examining some of the specific results from our iterations to gain additional insight into the effectiveness of our algorithms.

4.6 Specific Results of Interest

Since most of our algorithms' iterations started on a blue vertex, we will examine one of the iterations that started on red vertex. Starting a red vertex is the most likely way an algorithm like ours would be used. Once an individual is discovered or captured, and identified as a terrorist, we are then interested in discovering the rest of the terrorist's network. This is represented in our experimentation, with an iteration that starts with a red vertex. One such example is displayed in Figure 4.32. We have detailed results in Table 4.9.

% Red Nodes Found	Number of Monitors	Red Nodes Found
50%	165	70 Red Nodes
60%	273	84 Red Nodes
70%	366	98 Red Nodes
80%	731	112 Red Nodes
90%	1420	126 Red Nodes

Table 4.9: Detailed results of starting Probabilistic MUDD on a red vertex.

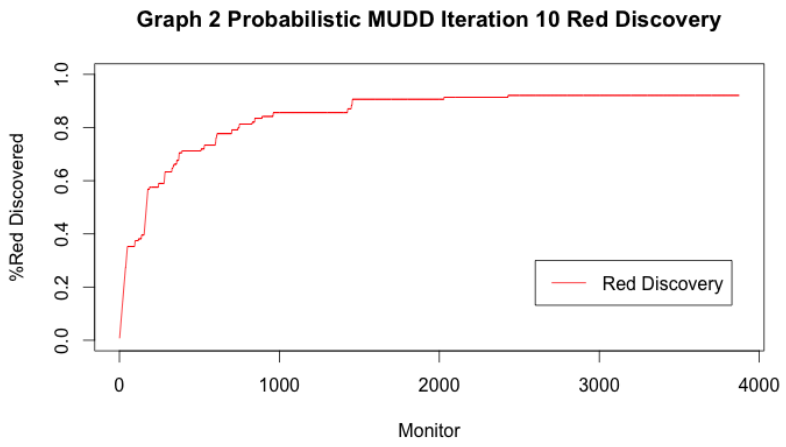


Figure 4.32: Discovery results when Probabilistic MUDD starts on a red vertex.

This algorithm starts on a terrorist vertex, and is able to discover 70 other terrorists within 165 monitors. This is a discover rate of one in three, from a network that only has one red vertex in fifty four vertices from the full network. This iteration uses the Probabilistic MUDD algorithm, and Noordin Top in this version of the synthetic network has been assigned a random probability of 0.3. So even when the most highly connected red vertex in the network has been heavily degraded, the algorithm is still very effective at discovering red vertices.

Each time we ran a discovery algorithm through a synthetic network, the output reported the color label of the vertex the monitor was placed on, the label of the vertex, and a cumulative count of the number of red vertices discovered up to that point. We give two examples of the output in Figures 4.33 and 4.34.

In these output files shown in Figures 4.33 and 4.34, the vertex color is the far left column of each figure. The cumulative red vertex count of terrorist vertices discovered up to that point in the algorithm is on the far right of each figure. The center portion of each figure is the vertex on which the monitor is placed. Every vertex in the synthetic network has a unique label.

We can easily identify the terrorist vertices because they have names, while our blue vertices are labeled with respect to which layer they originated when we created the network.

Blue	B4:129_8_1_3_1_0_2_0_1_0_0_0.6	39
Blue	B4:8_8_3_2_1_0_0_0_0_0_0_0.9	39
Blue	B4:73_3_1_1_0_0_0_0_0_1_0_0.9	39
Red	R:Ahmad Basyir_15_5_4_1_1_2_0_0_0_0_0.3	40
Red	R:Cholily_30_7_10_0_0_0_0_0_0_0_0.7	41
Red	R:Qotadah_21_11_11_1_0_1_0_0_0_0_0.7	42
	Samboja_28_8_11_0_0_0_0_0_0_0_0	
Red	R:Heri Sigu .2	43
Red	R:Agus Ahmad_46_9_7_0_0_1_0_0_0_0_0.0	44
Red	R:Achmad Hasan_41_7_8_0_0_0_0_0_0_0_0.3	45
Red	R:Chandra_40_4_7_0_1_1_0_0_0_0_0.4	46
Red	R:Imam Bukhori_39_12_12_1_1_0_0_0_0_0_0.6	47
Red	R:Ustadz Zaenal_32_0_12_0_0_0_0_0_0_0_0.9	48
Red	R:Joko Triharmanto_32_4_10_0_0_0_0_0_0_0_0.1	49
Red	R:Joni Achmad Fauzan_28_5_3_0_0_0_0_0_0_0_0.9	50
	Faiz)*_27_7_7_0_0_0_0	
Blue	B7:3437_2_0_0_0_0_0_0_0_0_0_0.7 _0_0_0_0.5	50
Blue	B7:3486_1_0_0_0_1_0_0_0_0_0_0.5	50
Blue	B7:862_2_0_1_0_0_0_0_0_0_1_0.4	50
Blue	B4:68_6_2_1_0_0_0_1_0_0_0_0.2	50
Blue	B4:116_6_1_3_1_0_0_0_0_0_0_0.9	50
Blue	B3:24_7_0_2_0_3_0_0_1_0_0_0.2	50
	Ba'asyir_25_21_3_0_0_0_0_0_0_0_0.	
Red	R:Abu Bakar 8	51
G_final2:	1/1: 1 complete	51
Red	R:Abdul Rohim_22_5_7_0_0_0_0_0_0_0_0.7	52
Red	R:Fathurrahman al-Ghozi_28_24_6_0_0_0_0_0_0_0_0.1	53
Red	R:Abdul Jabar_7_5_3_0_0_0_0_0_0_0_0.5	54
Red	R:Farihin_6_6_2_0_0_0_0_0_0_0_0.7	55
Red	R:Mohamed Islam_7_6_2_0_0_0_0_0_0_0_0.3	56
Blue	B4:39_0_0_0_0_0_0_0_0_0_0_0.3	56
Blue	B4:15_7_3_2_1_0_0_1_0_0_0_0.2	56

Figure 4.33: Output of the MUDD algorithm through Graph 2.

Blue	B7:2412_0_0_0_0_0_0_0_0_0_0_0.7	4
Blue	B7:2114_0_0_0_0_0_0_0_0_0_0_0.1	4
Blue	B7:3861_1_1_0_0_0_0_0_0_0_0_0.2	4
Red	R:Ustadz Anton_6_3_6_0_0_0_0_0_0_0_0.8	5
Blue	B1:34_8_0_3_1_0_1_0_0_0_0_0.9	5
Blue	B1:62_0_0_0_0_0_0_0_0_0_0_0.8	5
Blue	B1:12_8_0_2_1_1_0_0_0_1_0_0.1	5
Blue	B7:2702_0_0_0_0_0_0_0_0_0_0_0.1	5
Blue	B7:2675_1_0_0_0_0_0_0_0_0_0_0.3	5
Blue	B7:2833_0_0_0_0_0_0_0_0_0_0_0.3	5
Blue	B7:3406_0_0_0_0_0_0_0_0_0_0_0.4	5
Blue	B5:22_1_0_0_0_0_0_0_0_0_0_0.1	5
Blue	B4:46_1_0_0_0_0_0_1_0_0_0_0.6	5
Blue	B6:502_3_0_0_0_0_1_0_1_0_0_0.6	5
Red	R:Abdul Rohim_22_5_7_0_0_0_0_0_0_0_0.7	6
Blue	B4:120_5_1_1_1_0_0_0_0_0_0_0.3	6
Blue	B4:49_3_1_1_0_0_0_0_0_0_0_0.4	6
Blue	B5:478_2_0_1_0_0_0_0_0_0_0_0.9	6
Blue	B7:3399_1_0_0_0_0_1_0_0_0_0_0.2	6
Blue	B7:3915_3_0_0_0_1_1_0_0_0_0_0.7	6
Blue	B7:2246_0_0_0_0_0_0_0_0_0_0_0.3	6
Blue	B5:947_4_0_1_0_1_0_0_0_0_0_0.8	6
Blue	B4:69_4_0_1_1_0_0_1_0_0_0_0.5	6
Red	R:Imam Samudra_33_18_5_0_0_0_0_0_0_0_0.3	7
Blue	B3:62_5_3_2_0_0_0_0_0_0_0_0.2	7
Blue	B5:1385_2_2_0_0_0_0_0_0_0_0_0.6	7
Blue	B5:346_5_2_2_0_0_0_0_0_0_0_0.3	7
Blue	B5:213_4_1_2_0_1_0_0_0_0_0_0.2	7
Blue	B4:15_7_3_2_1_0_0_1_0_0_0_0.2	7
Blue	B5:9_5_2_2_0_0_0_0_0_0_0_0.1	7

Figure 4.34: Output of the Random Walk algorithm through Graph2.

For example, the first vertex in Figure 4.33 is labeled B4:129, this means it is the 129^{th} vertex from the $B4$ layer. The left column of this first vertex listed in Figure 4.33 reads "Blue," indicating the color of the vertex discovered. When we observe the right column of Figure 4.33, we see the number 39, indicating that up to that point in the algorithm, MUDD had discovered 39 red vertices. Continuing down the right column of Figure 4.33, we see the red vertex count does not increase until Ahmad Basyir is discovered a few monitors later.

When we inspect the labels in Figure 4.34, the output associated with Random Walk, we see red and blue vertices appear in a random order. Further, we see that blue vertices are discovered from all different originally blue layers. This is further evidence that we have cross populated and connected our vertices effectively, truly creating a synthetic multilayer network. This is contrasted by the results seen in Figure 4.33, associated with the output from our MUDD algorithm. We can observe that red vertices are discovered in groups. Once the algorithm discovers a red vertex, it continues to discover more red vertices. This is the exact characteristic we wanted our algorithm to posses.

When we look at Figure 4.33, we see that our algorithm discovers the red vertices in groups,

this brings us to our next observation. Our algorithms not only discover terrorist vertices, but our algorithms discover the terrorist vertices in their communities from the original Noordin Top Network. In most iterations of MUDD and Probabilistic MUDD, we discovered the following Community shown in Figure 4.35. This is our target community from the Noordin Top Network. We display the results of the Probabilistic MUDD output in Figure 4.36.

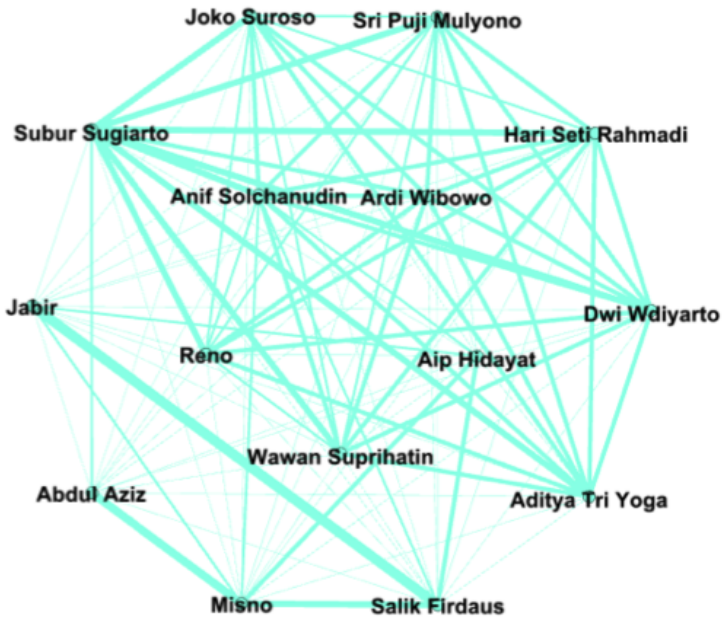


Figure 4.35: Community that is discovered together using MUDD or Probabilistic MUDD. This is our target community from the Noordin Top Network.

When we examine the output file from Probabilistic MUDD, we see that Subur Sugiarto is the 56^{th} red vertex the algorithm discovers. The following 13 vertices discovered are not only all red, but are all members of the target community we attempted to identify using only the Communication layer from the Noordin Top Network. This is 13 of the original 15 terrorists in the target community (some iterations of Louvain identified 16 terrorists in this communities). The algorithm is not targeting the community structure of the network, but it is discovering the communities none the less. This iteration is not unique, but is a characteristic we see in most of the discovery outputs, further pointing to our target community being unique.

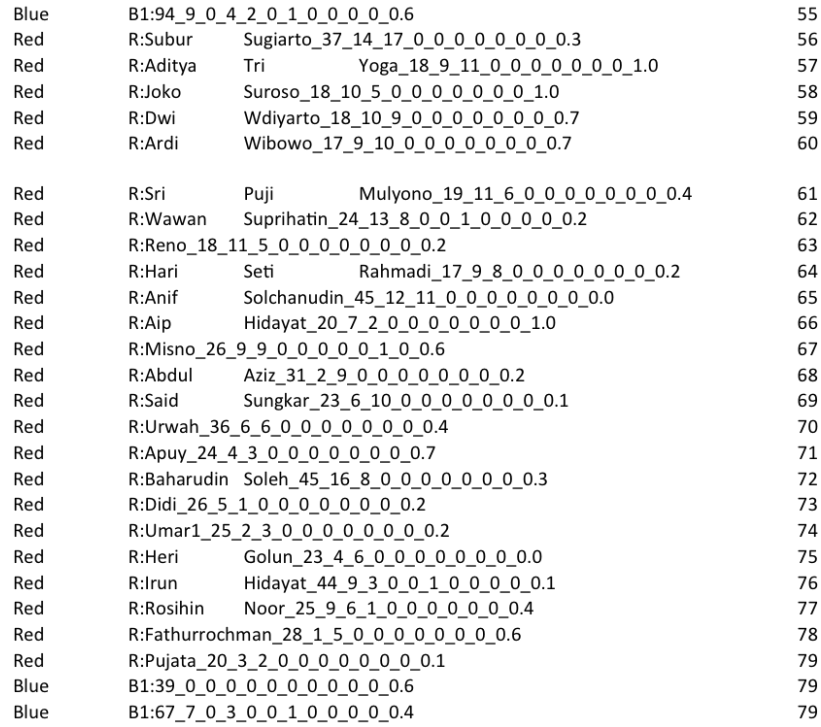


Figure 4.36: Output from Probabilistic MUDD discovering vertices in synthetic network 2. This is our target community from the Noordin Top Network.

Other communities are discovered together by our algorithms, but it is less clearly defined as a unique grouping. This is because the other communities in the network are loser tied together, with multiple edges between communities. When we look at the community structure shown in Figures 4.1 and 4.2 we see many edges between the communities, with significantly fewer edges from our target community to other communities. This point is highlighted more effectively when we inspect the community structure from the Communication layer. In Figures 4.3 and 4.4, we see our target community is connected to only one vertex outside of the community: the Noordin Top vertex. So our algorithms tend to detect most members of a community before jumping to a blue node or to another community. This is more defined in our target community because the other communities are more interconnected.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusions and Further Direction

5.1 Conclusion

In this thesis we have shown (1) that it is possible to detect and identify a specific target community from a multilayer network using a specific layer, (2) that we can successfully create a large synthetic network and embed a terrorist network with in the larger network in such a way that the terrorist network is not easily identifiable, and (3) we can take advantage of the multilayer nature of our network by using multilayer discovery algorithms that can target our terrorist network and discover the terrorists at a faster rate than single layer discovery algorithms.

5.1.1 Single Layer Target Community Identification

The detection and identification of a specific target community in a multilayer network using a specific layer is not a statement that can be extended to all layers or all communities in a network. If the target community has unique characteristics, we can identify it within the same characteristics from the single layer. The target community we were concerned with was significantly denser, and with a higher clustering coefficient, than the other communities in the multilayer network. We were able to take advantage of this topology, and identify the target community from the Communication layer. While we did not identify all members of the community from the full network, we were able to identify a majority of the members. Perhaps more importantly, we had no false positive community membership identifications. All members we identified from the Communication layer were present in the full network. We say this is potentially more important, because, using the motivation from Chapter 1, if we can capture all radio transmissions or email communications and perform community detection there is a high likelihood that community membership in this single slice of the network will tell us a great deal about the community structure of the full network. Most of the members of the community not included in the communication layer were the suicide bombers, and they were not originally members of the terrorist group. It makes sense that they were not identified using the communication layer.

We are limited by what we can determine from using only a single layer. We were not able to uniquely identify subsequent communities in the full network, that we found in the Communication layer. We are able to identify the target community, in part, because it is so unique. If the target community were less unique, more similar to the other communities in the network, we would not be as successful in identifying the target community. In their work, Lapides and Evan [37] attempted to identify communities using closeness centrality using only a single layer from the Noordin Top Network. They encountered similar limitations with the information available in a single layer.

5.1.2 Single Layer Community Structure Identification

We also used the single layer of information to attempt to discover the community structure of full network based of the information contained in in the community structure of a single layer. We used the standard measure for community comparison, the Normalized Mutual Information (NMI), to measure how similar the community structure of each individual layer was to the community structure of the full multilayer network. We found that each individual layer can help marginally with depicting the community structure of the full network, but this is dependant on the amount of information contained in the edge list for each layer. The individual layers do not depict the community structure poorly, they just have a limited depiction of the community structure. This follows some of our intuition. Namely, since the full network is comprised of the aggregation of all layers, using a single layer would only give part of the community structure. We can think of this as concluding that partial information, in the form of a single layer, only depicts part of the community structure. Generally, aggregating more than one layer will give a better idea on the communities as shown in [40]. The work in [40] also looked at combining layers with different weights in order to gain a better picture of the community structure in a dark network.

5.1.3 Creation of a Synthetic Network

We introduce a general methodology to produce a synthetic multilayer network so the Noordin Top Network would not stand out as a unique featured sub-network within the network. We saw that a unique community was easy to identify when we interpreted partial information as using single layer from the network. We want to avoid creating a synthetic network with this same quality, making the identification of the terrorist network easy. We

created a very large network, with more than 7,500 vertices and more than 180,000 edges. We analyzed this network and it had the properties we desired, specifically being composed of a single component and having a scale free degree distribution. We were able to determine that all of our terrorist vertices were part of the largest community within the synthetic network, and community detection alone would not identify our terrorist network.

We analyzed the properties of the terrorist portion of the network after the synthetic network was created, and compared these properties to the original Noordin Top Network. We were able to see that the Noordin Top Network retained key properties like having Noordin Top as the most connected member and maintaining a similar portion of the Noordin Top Network as hubs with in the network before and after embedding. Other aspects of the network showed how well the terrorists had been embedded in the synthetic network, specifically how much the degree increased for of all of the poorly connected members of the Noordin Top Network. It is important to keep in mind that while the overall connection did increase for all red vertices, the increase was similar to that of the connections of the synthetic network. When we looked at the degree sequence of the synthetic network, we did not observe any portions of the network that stood out as unique.

We have effectively created a large synthetic multilayer network that has the Noordin Top Network embedded within it. The Noordin Top Network is not unique, and is difficult to find and distinguish from the rest of the network. The synthetic network is comprised of many layers that bring in elements of randomness (Erdős-Rényi random graphs), structure (Barabási-Albert graphs), and real world examples (anonymous Facebook network).

5.1.4 Discovery of Red Vertices Using MUDD

We modified an existing algorithm, Undiscovered Degree Discovery (UDD), to take advantage of the multilayer property of our synthetic network. We used this algorithm, Multilayer Undiscovered Degree Discovery (MUDD), to discover red vertices at a faster rate than UDD. We were able to do this, because our algorithm takes advantage of the layers in the network. Our network has layers that are more likely to have connections between red vertices. This is our "equation for a terrorist." Even though our algorithm is limited by the fact that it can only discover neighbors as purple, it can take advantage of the discovered red vertex. Once a vertex is discovered as red, the algorithm only evaluates neighbors from

layers that are part of our "equation for a terrorist."

It is important to note that evaluating neighbors in the target layers only, does not guarantee the discovery of a red vertex with the next monitor. We have purposely added connections to blue nodes that also belong to the target layers, which is realistic and it makes it difficult for algorithms to find red vertices. Despite this fact, the introduced algorithm out preforms UDD in the discovery of red vertices.

Specifically, MUDD out performs UDD in the first 250 monitors placed in the network. This is important, because we want the algorithm to discover red vertices quickly, not just through exhaustive measures by discovering the full network. Furthermore, as we introduced in Chapter 1, we want to use as few monitors as possible. In our experimentation, monitors are assumed to be cheap and easy to place in the network. This is not always the case in the real world. We want to consider the monitors to be both expensive and difficult or dangerous to place in the network. This makes any advantage MUDD provides important for us, regardless how small the advantage may seem.

MUDD also outperforms UDD in the long run, as the algorithms discover 50% of the network. The last 35 red vertices represent approximately 25% of the terrorist network. These are the poorly connected vertices from the original Noordin Top Network. The fact that these 25% of the vertices are poorly connected does not mean that we are less interested in discovering them. We highlighted the suicide bombers Salik Firdaus, Misno, Aip Hidayat. These three individuals were recruited into one of the five terrorist organizations within the Noordin Top Network, there were not as connected to the network as other members. The ability of MUDD to discover more of the network than UDD in the long run, specifically these more poorly connected vertices, gives us a higher chance to discover more of the terrorists in this synthetic network.

The performance of MUDD demonstrates that there is an advantage to using a multilayer network to represent these networks. There is additional labor, processing, and storage requirements to maintain a multilayer network.

5.1.5 Discovery of Red Vertices Using Probabilistic MUDD

We also used a version of MUDD to better approximate real world knowledge. This Probabilistic MUDD algorithm uses degraded knowledge of the number of neighbors for each vertex. We did this to replicate the concept that in reality we would not know everyone an individual is connected too. We expected Probabilistic MUDD to not perform as well as MUDD. What we actually found was that MUDD and Probabilistic MUDD had very similar discovery rates. This is due to the way in which Probabilistic MUDD degrades knowledge.

The Probabilistic MUDD algorithm evaluates all of the undiscovered neighbors of each vertex using perfect knowledge. That is to say, the degradation takes effect after the undiscovered neighbor count is established by reporting back a percent of the real count of undiscovered neighbors. This count is degraded, but not the connections from the vertex in question. So a vertex might be evaluated as not having a large number of undiscovered neighbors, and therefore not a likely candidate for subsequent monitors, but the connections are preserved. Therefore, once the monitor is placed, all the connections are discovered, not just the reported proportion. We have potential concepts for altering these connections presented later in this chapter.

Additionally, the information is degraded for all vertices in a similar manner, regardless of the fact that they are terrorists or not. The probabilities assigned vertices were allocated randomly from the same distribution. There was no discrimination between red vertices and blue vertices, the probabilities were assigned to purple vertices. So we were effective in degrading the information used to evaluate vertices within the target layers, but we degraded all vertices similarly. This is seen in the oscillation during the first 250 monitors of discovery between MUDD and Probabilistic MUDD. Sometimes MUDD out preformed Probabilistic MUDD, sometimes it did not. When we ran Probabilistic MUDD through all 10 synthetic networks, we saw a greater amount of variance, when compared to the MUDD discovery through all 10 synthetic networks. Yet in the long run, they both did about as well as each other in the discovery of red vertices. This similarity is seen when the algorithms discover the same amount of the terrorist network after discovering half of the synthetic network.

5.1.6 Discovery of Communities as a group

When we closely analyzed the output results from our algorithms, we able to notice that communities that were present in the full Noordin Top Network were discovered in sequence by out algorithms. This was surprising, because our algorithms are not designed to discover communities. The ability to discover communities as a group is due to the fact that once the algorithms discover a red vertex, the selection of the next vertex for discovery is from the target layers of the synthetic network. These target layers are comprised of blue and red vertices, but the red vertices from that layer represent the community structure from the Noordin Top Network. There are blue vertices that have been added to these target layers, but any red vertex selection will most likely be from the same community from the community the monitor vertex originated from. The reason the next vertex is most likely from the same community is based on the concept of a community. Namely, that vertices within a community have more edges connecting to other members of the community, and fewer connections outside the community. So when an algorithm is selecting a vertex for the next monitor, it is more likely to select along an edge that connects inside the community.

We saw this behavior when we highlighted the discovery of the target community from the single layer community detection, that was discovered as a group in the MUDD and Probabilistic MUDD algorithms. In either MUDD or Probabilistic MUDD, once a member of this community is discovered by a monitor, the algorithm starts to select the subsequent vertex for monitor placement from candidates in the three target layers. We we analyzed the target community, we noticed there were relatively few connections outside the community. In the case of the Communication layer, the only connections outside of the community were connections from a member of the community to Noordin Top himself. This means that once the algorithm discovers a red vertex, if the next vertex is red as well, it is highly likely that the newly discovered vertex is also in the same community. When the community in question has a high clustering coefficient and is dense, this pattern of discovering the community in a grouped is more prevalent. When there are more connections outside of the community, as in the other communities from the full Noordin Top Network, this grouping is not as prevalent.

5.2 Future Work

We have gained a great deal of insight into the use of multilayer networks, and partial information in this work. We have designed a multilayer synthetic network that effectively makes discovery of an embedded terrorist network difficult. Additionally, we have developed modified algorithms that take advantage of the multilayer structure of our network to discover terrorists faster than algorithms that do not account for the multiple layers. There is room with all of these concepts to extend this work in other directions.

5.2.1 Use other Terrorist Networks

In our work, we concerned ourselves solely with the Noordin Top Network, however, our processes should work with other dark networks. The FARC is a terrorist organization in Columbia and Venezuela that is organized with multiple layers. Boko Harram is a terrorist organization in Africa that also has been studied as a multilayer network. Either of these could be used to test the effectiveness of both the synthetic network creation process and the MUDD red vertex discovery process.

Another interesting extension would be to see if the age of the dark network has any impact. One could use an older, more established organisation, and compare the results to a newer, emerging network, as it is in the collection of the three networks mentioned.

The type of dark network might impact the results as well. We concerned our research and experimentation with terrorist organizations, but there are other dark networks that are covert in nature. One could use data from a criminal gang or drug smuggling network, and compare the results with a terrorist network of similar size.

Additionally, the size of the dark network, when compared to the rest of the synthetic network, might impact the results. We had a synthetic network with 1 in 54 vertices being a member of the terrorist network. We could alter the size of the terrorist network, larger or smaller, and see if the algorithms were any more or less effective at discovering red vertices.

Finally, we could alter the structure of the terrorist network itself. We used real data to create the terrorist network. We could create a synthetic terrorist network, with community

structure, and simply color the vertices red. Then create a larger multilayer synthetic network in the process outlined here, embedding a synthetic network within a larger synthetic network. This would be a very interesting way to test our algorithms' ability to discover the red vertices.

5.2.2 Different embedding of Noordin Top Network Within Synthetic Network

The synthetic network we created was very effective at concealing our terrorist network. Further work could be done to make this concealment more realistic and more difficult. However, we lack real data to mimic this process.

Our main interest was to embed it in a way that is not unique and easily identifiable. One of the ways to extend it, would be to attach vertices within a layer with greater care. Currently, any vertex that is added to a layer, is added with a degree of three. This random attachment to three random vertices is effective, but could be improved. We adopted this view for its simplicity, which is desired for a first attempt at the problem. We could use the layer's original degree distribution as a guide. We could "bin" the degree distribution, and determine different values to attach vertices. These different values could match the "bins" proportionally. For example, if 10% of the original layer had a degree of 7 or less, 10% of the vertices being attached will attach to the layer with a degree of 7 or less. This could be continued for all "bins," in this case bins representing 10% of the original degree distribution. This would ensure that the original degree distribution of a layer is not altered too much.

When we cross-populated and attached the vertices from the different layers, we did so in the same manner for each layer. Our only stipulation was that no new edges were added between red vertices in the target layers, in order to not alter the red information in the network. This was also based on the fact that terrorist try to cover their relatinships, so there was no need for creating additional ones. We might alter the behavior of the red vertices in the network. Perhaps the terrorist network is trying to embed itself deeply in the civilian population. The red vertices might attempt to attach to blue vertices at a higher degree in the target layers. Conversely, the terrorist network might not integrate into the civilian population, and as a result, there are fewer blue vertices attached in the target layers.

We based the construction of our synthetic network using three layers as our target layers. In his work, Miller [38] uses the same three layers for community detection. He also uses each layer weighted differently. We have our algorithm weigh all three layers equally. Perhaps weighting layers according to an importance might yield better results.

5.2.3 More Realistic Synthetic Network

The synthetic network we created used seven layers that were originally blue layers, with no terrorist vertices. Four of these layers were copies of layers form the original Noordin Top Network. Two layers were synthetic layers that introduced randomness and structure. The final layer was a real world network, consisting of anonymous Facebook data. All of these layers were selected for specific reasons, in order to attempt to create a synthetic multilayer network that embedded the Noordin Top Network as a non-unique portion of the full network. There are many alterations we could make to this process to make the synthetic network more realistic, and make discovery of terrorist network more difficult.

We could alter the first four blue categories or layers. These are the blue copies of the three categories from the Noordin Top Network, and a copy of the Noordin Top Network as well. We could increase the number of copies present in each layer. This would create more hubs within the network that are blue. It would have the effect of making more non-terrorist versions of the Noordin Top Network in the synthetic network. These additional vertices with high degree would probably cause UDD to decrease it's rate of red vertex discovery, and it would be interesting if MUDD and Probabilistic MUDD would be able to discriminate the red hubs from the blue hubs as effectively as they are able to discriminate in the current network.

In our current synthetic network, we have one layer that is derived from a real world social network, the Facebook layer $B7$. We could simply create additional layers in the synthetic network based on other social networks. These would be more real world influences in our network. For example, we might want to add in LinkedIn data to create a layer that represents a workplace or workplace like relation. Additionally, we could create different cross population and attachment criteria for these new layers. Any additional real world influence in our synthetic network would be welcomed as making the challenge for the algorithms to discover red vertices more realistic.

Additionally, none of our layers have connections between layers. There are no edges connecting vertices in separate layers. All of our edges exist in one layer at a time. There might be ways to interconnect vertices between layers. These inter-layer connections could be representative of many different relationships. Of particular interest might be to consider each layer as a snapshot in time. Inter-layer edges might represent a change in a vertex from one time period to the next. This multilayer network might be able to show the evolution of a network through time, as opposed to our network that represents different types of relationships among the same group of individuals.

5.2.4 Degradation of Perfect Knowledge

We attempted to degrade the perfect knowledge the algorithms used when we implemented Probabilistic MUDD, but this seemed to have little impact on Probabilistic MUDD's ability to discover red vertices. It would be interesting to alter the way this happens in the network. A few approaches to alter this would be to degrade individual layers of the synthetic network differently. It would make sense to degrade the information in the red layers, R1, R2, and R3, more than the blue categories. This could represent the tendency of dark networks to hide their connections because of their clandestine nature. Subsequently, we would not degrade the information in blue categories as much. We could implement this, by assigning the random probabilities to each layer before we aggregate the layers to form the purple synthetic network. We could still use a normal distribution for each layer, but center the mean at a value less than 0.5 for the red layers, and at a value higher than 0.5 for the blue layers. These probabilities would be used when aggregating the layers to determine the purple degree of each vertex in the network. This degradation of knowledge would affect each monitor's ability to determine the undiscovered neighbors in the algorithm. It would not affect the edges between vertices, and it would not reflect any difference between red vertices and blue vertices.

Another way to degrade the perfect knowledge would be to attempt to hide some of the connections in the network. This would also impact the monitor's ability to evaluate the list of undiscovered neighbors, but this method would alter the structure of the network. This alteration would include the terrorist vertices, so potentially this is something that drastically changes the results. A way to implement this would be to assign to every edge

(not to every vertex) a random probability from a possibly uniform distribution. Next we determine a threshold value. Any edge that has a probability less than our threshold value is not considered when our algorithm attempts to determine the list of undiscovered neighbors. This might reflect a real world situation in which we are not able to determine all of the connections a person has to other individuals in a given network. Again, this would affect red and blue vertices in a similar fashion.

Perhaps a more realistic way to degrade the knowledge would be to account for the vertex color when we degrade the information contained in our network. In this work, and in the two previous examples, vertices are treated identically, regardless of color. In reality, a terrorist might behave differently than a non-terrorist. We could degrade red vertices more than blue nodes, and potentially get a more realistic result. Additionally, we could establish a more detailed degradation criteria for vertices in each layer, based on vertex color, and get an even more realistic result still.

Vertex behavior could be accounted for in another way. In the real world, we expect some terrorists to be better at hiding within the civilian population. Additionally, a person might have some indicators of terrorist behavior, but in reality they are not a terrorist. We could account for this with false positive and false negative color identification when a monitor discovers the color of a vertex. For example, there could be a small probability that a color is misidentified by the monitor. A false positive would be a blue vertex that the monitor identifies as red, while a false negative would be when a monitor identifies a red vertex as blue. This potential for altering the vertex color reporting could provide interesting results.

5.2.5 Further Modifications to MUDD

We have modified UDD, and our new algorithm MUDD outperforms UDD, in the discovery of red vertices. We feel there is room for more improvement still. Currently, when MUDD discovers a red vertex, it places the subsequent vertex after evaluating the target layers. This process then starts over, regardless of the color vertex the subsequent monitor discovered. An improvement would be to test the color of the subsequent vertex. What we mean by this, is if the subsequent monitor discovers a blue vertex, go back to the previous red monitor, and choose the next best candidate. This would have the effect of testing if the monitor selected a red vertex as the next placement. A few points to consider with this modification

are how many tests should the algorithm allow, and what happens if there are no neighbors that are red. First, how many tests should the monitor allow? Too many tests, and the algorithm will simply not discover anything except the red network. It will find a red vertex, and test each neighbor, only discovering red vertices. This modification would never discover a blue vertex after the first red vertex is found. It will, however, never find red vertices that are not connected to the initial red vertex it discovers. The second situation is once the algorithm discovers a red vertex that is a leaf from the terrorist network. When testing the neighbors of this leaf, all neighbors will be blue with the exception of the monitor that lead to the discovery of the leaf. The algorithm will be stuck on this leaf, not allowed to discover any additional red vertices. For all of the reasons listed above, there should be some sort of limit on the number of tests a monitor can preform before the algorithm resorts to evaluating the purple degree of the neighbors in order to discover more of the network. Adjusting the number of tests the algorithm is allowed could provide some interesting results.

Another modification to MUDD that we would like to implement would be to give the algorithm a memory. What we mean by this is to remember which previously discovered vertices are red, and have this memory influence the placement of the next monitor. The concept of this memory, is to evaluate candidates for the placement of the next monitor not just on the number of undiscovered neighbors, but which previously discovered red vertices are directly adjacent to the candidates. Simply put, if a vertex is immediately adjacent to multiple red vertices, it should be a prime candidate for the next monitor. This has the real world scenario of a person knowing one terrorist does not mean he or she is a terrorist. However, if a person knows multiple terrorists, it is more likely that he or she is a terrorist. The implementation of this would be difficult, but could potentially lead to a very fast rate of red vertex discovery.

This work has been exciting and very thought provoking. There remains a great deal to be done in both improvement of our methods, and modifications leading in future directions. Hopefully this work has sparked insight and interest in to the field of dark networks.

List of References

[1] B. W. Hung, S. E. Kolitz, and A. Ozdaglar, "Optimization-based influencing of village social networks in a counterinsurgency," in *Social Computing, Behavioral-Cultural Modeling and Prediction*. New York, NY: Springer, 2011, pp. 10–17.

[2] J. Raab and H. B. Milward, "Dark networks as problems," *Journal of Public Administration Research and Theory*, vol. 13, no. 4, pp. 413–439, 2003.

[3] J. Xu and H. Chen, "The Topology of Dark Networks," *Communications of the ACM*, vol. 51, no. 10, pp. 58–65, 2008.

[4] D. W. Opderbeck and J. G. Hurwitz, "Apple v. FBI: Brief in Support of Neither Party in San Bernardino Iphone Case," *Draft Amicus Brief*, Mar. 22 2016. Available: http://ssrn.com/abstract=2746100.

[5] E. Lichtblau and K. Benner. (2016, Feb. 18). Apple Fights Order to Unlock San Bernardino Gunman's iPhone. *The New York Times*. http://www.nytimes.com/2016/02/18/technology/apple-timothy-cook-fbi-san-bernardino.html?_r=0. Accessed: 2016-04-18.

[6] C. A. Rubin, J. and R. Winton. (2016, Feb. 16). Apple opposes order to help FBI unlock phone belonging to San Bernardino shooter. *The Los Angeles Times*. http://www.latimes.com/local/lanow/la-me-ln-fbi-apple-san-bernardino-phone-20160216-story.html. Accessed: 2016-04-18.

[7] E. Nakashima. (2016, Feb. 16). Apple vows to resist FBI demand to crack iPhone linked to San Bernardino attacks. *The Washington Post*. https://www.washingtonpost.com/world/national-security/us-wants-apple-to-help-unlock-iphone-used-by-san-bernardino-shooter/2016/02/16/69b903ee-d4d9-11e5-9823-02b905009f99_story.html. Accessed: 2016-04-18.

[8] G. Chartrand, *Introduction to Graph Theory*. New York, NY: Tata McGraw-Hill Education, 2006.

[9] M. Newman, *Networks: An Introduction*. Oxford, UK: Oxford University Press, 2010.

[10] M. Lad, D. Massey, and L. Zhang, "Visualizing Internet Routing Changes," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 6, pp. 1450–1460, 2006.

[11] B. Lyon *et al.* (2003). The Opte Project. The Opte Project. [Online]. Avaialable: http://www.opte.org/.

[12] M. De Domenico, A. Solé-Ribalta, E. Cozzo, M. Kivelä, Y. Moreno, M. A. Porter, S. Gómez, and A. Arenas, "Mathematical Formulation of Multilayer Networks," *Physical Review X*, vol. 3, no. 4, p. 041022, 2013.

[13] W. E. Baker and R. R. Faulkner, "The Social Organization of Conspiracy: Illegal Networks in the Heavy Electrical Equipment Industry," *American Sociological Review*, vol. 58, no. 6, pp. 837–860, 1993.

[14] R. C. Read and D. G. Corneil, "The Graph Isomorphism Disease," *Journal of Graph Theory*, vol. 1, no. 4, pp. 339–363, 1977.

[15] M. De Domenico, M. A. Porter, and A. Arenas, "MuxViz: A Tool for Multilayer Analysis and Visualization of Networks," *Journal of Complex Networks*, 2014. Available: http://comnet.oxfordjournals.org/content/early/2014/10/12/comnet.cnu038.abstract

[16] M. E. Newman, "The Structure and Function of Complex Networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.

[17] K. Claffy, "CAIDA: Visualizing the Internet," *Internet Computing Online*, p. 88, Jan 2001.

[18] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark Graphs for Testing Community Detection Algorithms," *Physical Review E*, vol. 78, no. 4, p. 046110, 2008.

[19] G. K. Orman, V. Labatut, and H. Cherifi, "On Accuracy of Community Structure Discovery Algorithms," *arXiv preprint arXiv:1112.4134*, 2011.

[20] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast Unfolding of Communities in Large Networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008. Available: http://stacks.iop.org/1742-5468/2008/i=10/a=P10008

[21] S. P. Borgatti, M. G. Everett, and L. C. Freeman, *Ucinet for Windows: Software for Social Network Analysis*. Analytic Technologies, 2002.

[22] M. M. Siems, "A Network-Based Taxonomy of the World's Legal Systems," *Durham Law School Working Paper March 2014*, 2014. Available: http://ssrn.com/abstract=2387584 or http://dx.doi.org/10.2139/ssrn.2387584.

[23] S. F. Everton, *Disrupting Dark Networks*. Cambridge, UK: Cambridge University Press, 2012, no. 34.

[24] L. Ana and A. K. Jain, "Robust Data Clustering," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. IEEE, 2003, vol. 2, pp. II–128.

[25] N. Blenn, C. Doerr, B. Van Kester, and P. Van Mieghem, *Crawling and Detecting Community Structure in Online Social Networks using Local Information*. New York, NY: Springer, 2012.

[26] Terrorism in Indonesia: Noordin's Networks. (2006, May). International Crisis Group. [Online]. Available: http://www.crisisgroup.org/en/regions/asia/south-east-asia/indonesia/114-terrorism-in-indonesia-noordins-networks.aspx

[27] N. Roberts and S. F. Everton., "Terrorist data: Noordin top terrorist network." Available: https://sites.google.com/site/sfeverton18/research/appendix-1, June 2011.

[28] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," Available: http://snap.stanford.edu/data, June 2014.

[29] B. Bollobás and O. M. Riordan, "Mathematical Results on Scale-Free Random Graphs," *Handbook of Graphs and Networks: From the Genome to the Internet*, pp. 1–34, 2003.

[30] M. E. Newman, D. J. Watts, and S. H. Strogatz, "Random Graph Models of Social Networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. suppl 1, pp. 2566–2572, 2002.

[31] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[32] R. Albert and A.-L. Barabási, "Statistical Mechanics of Complex Networks," *Reviews of Modern Physics*, vol. 74, no. 1, p. 47, 2002.

[33] B. Davis, R. Gera, G. Lazzaro, B. Y. Lim, and E. C. Rye, "The Marginal Benefit of Monitor Placement on Networks," in *Complex Networks VII*. Springer, 2016, pp. 93–104.

[34] R. Gera, N. Juliano, and K. Schmitt, "Optimizing Network Discovery through Intelligent Walks," submitted for publication.

[35] R. Gera, "Network Discovery Visualization Project: Naval Postgraduate School network discovery visualization project," Available: http://faculty.nps.edu/dl/networkVisualization/, July 2015.

[36] S. Chen, R. Gera, B. Greunke, N. Sharpe, and S. Warnke, "An Evolution of Community Detection through Network Discovery," submitted for publication.

[37] J. Lapides and D. Evans, "Private Communication with Jeffery Lapides of Lapides Consulting, LLC under management of Dan Evans of Storm King Analytics, working with the Network Science Center at West Point," Private Communication, Apr. 2016.

[38] R. Miller, "Purpose-Driven Communities in Multiplex Networks: Thresholding User-Engaged Layer Aggregation," Master's thesis, Monterey, California: Naval Postgraduate School, 2016.

[39] R. Sharma, M. Magnani, and D. Montesi, "Missing Data in Multiplex Networks: A Preliminary Study," in *Signal-Image Technology and Internet-Based Systems (SITIS), 2014 Tenth International Conference on*. IEEE, 2014, pp. 401–407.

[40] B. Crawford, R. Gera, R. Miller, and B. Shrestha, "Community Evolution in Multiplex Layer Aggregation," submitted for publication, Apr. 2016.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California