NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

NETWORK OF TAMCNS: IDENTIFYING INFLUENCE REGIONS WITHIN THE GCSS-MC DATABASE

by

Victor G. Castro

June 2017

Thesis Advisor: Gurminder Singh
Second Reader: Arijit Das

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
	June 2017	Master's Thesis 06-29-2017 to 06-16-2017

4. TITLE AND SUBTITLE

NETWORK OF TAMCNS: IDENTIFYING INFLUENCE REGIONS WITHIN THE GCSS-MC DATABASE

5. FUNDING NUMBERS

6. AUTHOR(S)

Victor G. Castro

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)

N/A

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)*

The Global Combat Support System-Marine Corps (GCSS-MC) system enables the logistics and supply chain management system for the United States Marine Corps. This system allows leaders, operators, maintainers, and suppliers to work together on a common platform and provides transparency and situational awareness in the logistics and supply cycle. The data associated with the GCSS-MC system is stored in a database. As the size of the data increases, challenges arise in obtaining insights into large data sets and its impact on network infrastructure. Network science identifies relationships between objects and provides tools to quantitatively determine objects whose influence impacts other objects or the system as a whole. This thesis applies network science techniques to determine important Table of Authorized Material Control Numbers (TAMCNs) in the GCSS-MC database, according to their impact on other TAMCNs in the database based on degree, eigenvector, betweenness, and closeness centrality. Additionally, this thesis develops a formula to rank components from most to least important. We further develop a process to identify pertinent tables within the database and export the information to create a complex network containing multiple layers that analyze various attributes associated with GCSS-MC. We find that the methodology identifies the most important TAMCN and provides a list of TAMCNs in order of importance. We also analyze the community and core structure of the GCSS-MC complex network and identify influential TAMCN regions in the database.

14. SUBJECT TERMS	15. NUMBER OF PAGES
GCSS-MC, database, centrality, influence, graph, network science	101
	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.

NETWORK OF TAMCNS: IDENTIFYING INFLUENCE REGIONS WITHIN THE GCSS-MC DATABASE

Victor G. Castro
Captain, United States Marine Corps
B.S., United States Naval Academy, 2011

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2017

Approved by: Gurminder Singh
Thesis Advisor

Arijit Das
Second Reader

Peter Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Global Combat Support System-Marine Corps (GCSS-MC) system enables the logistics and supply chain management system for the United States Marine Corps. This system allows leaders, operators, maintainers, and suppliers to work together on a common platform and provides transparency and situational awareness in the logistics and supply cycle. The data associated with the GCSS-MC system is stored in a database. As the size of the data increases, challenges arise in obtaining insights into large data sets and its impact on network infrastructure. Network science identifies relationships between objects and provides tools to quantitatively determine objects whose influence impacts other objects or the system as a whole. This thesis applies network science techniques to determine important Table of Authorized Material Control Numbers (TAMCNs) in the GCSS-MC database, according to their impact on other TAMCNs in the database based on degree, eigenvector, betweenness, and closeness centrality. Additionally, this thesis develops a formula to rank components from most to least important. We further develop a process to identify pertinent tables within the database and export the information to create a complex network containing multiple layers that analyze various attributes associated with GCSS-MC. We find that the methodology identifies the most important TAMCN and provides a list of TAMCNs in order of importance. We also analyze the community and core structure of the GCSS-MC complex network and identify influential TAMCN regions in the database.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AAV	Assault Amphibious Vehicle
AS	autonomous system
CDF	cumulative distribution function
CSV	comma separated value
DOD	Department of Defense
EBS	E-Business Suite
fMRI	functional magnetic resonance imaging
GEXF	Graph Exchange XML Format
GCSS-MC	Global Combat Support System-Marine Corps
GSC	giant strong component
GUI	graphical user interface
HDFS	Hadoop Distributed File System
HMMWV	High Mobility Multipurpose Wheeled Vehicle
HTML	HyperText Markup Language
IDE	integrated development environment
JSON	JavaScript Object Notation
LAV	Light Armored Vehicle
MANET	mobile ad hoc network
MDR	Master Data Repository

MRI	magnetic resonance imaging
NSN	national stock number
NPS	Naval Postgraduate School
NRP	Naval Research Program
OSPF	Open Shortest Path First
PDF	probability distribution function
PFP	Positive-Feedback Preference
SQL	Structured Query Language
TAMCN	Table of Authorized Material Control Number

Acknowledgments

Professor Singh: Thank you for your cool demeanor and willingness to let me explore ideas. I really appreciate the focus you applied to this work and your encouragement and confidence that I could complete the task.

Professor Das: Thank you for your commitment to this project and the many hours you spent bouncing ideas around. NPS can be very hectic and fast-paced, and I appreciate your words of affirmation, which helped me believe I could finish. I also appreciate the energy you bring to your work.

Professor Gera: Although you are not one of my thesis advisors, I would not have been able to complete this project without your help in network science. Thank you for your willingness to help a student from another department.

Lunch Club: Thank you, Dan, Alexis, Boulat, Warren, and Tony for your friendship. You made spending many hours at NPS an enjoyable experience. In particular, I want to acknowledge Dan for his assistance with math and LaTeX.

Sybil and Paulina: Sybil, witnessing you pass through your battle with cancer with grace and a positive outlook inspired me to do better and never give up. Paulina, the love of learning that you instilled in me at a young age has not abated, and I am grateful for your ever-present support and love.

Joseph, Dominic, and Annie: Thank you, J, Dom, and Annie, for your patience as I plugged away at schoolwork not only during the day, but many times late into the night, as well. J and Dom, thank you for your boundless energy and affection. Difficulties do not seem challenging after spending time with you. Annie, thank you for always believing in me and caring for our boys, who are our most precious gifts. I love the three of you so much!

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Marine Corps Order 4400.150 [1] defines the Global Combat Support System-Marine Corps (GCSS-MC) system as "a Marine Corps specific logistics chain management system which provides cross-functional information to enhance ground supply and maintenance operations." Figure 1.1 illustrates the interactions between users and the GCSS-MC system. GCSS-MC supports users based in units in garrison, stationed abroad, and even in combat zones. Whether GCSS-MC is fielded in a large maintenance facility within the U.S. without any bandwidth constraints or in expeditionary locations in constrained communications environments, GCSS-MC sustains logistics operations, ensures asset visibility, and enables responsiveness in supply and maintenance requests.

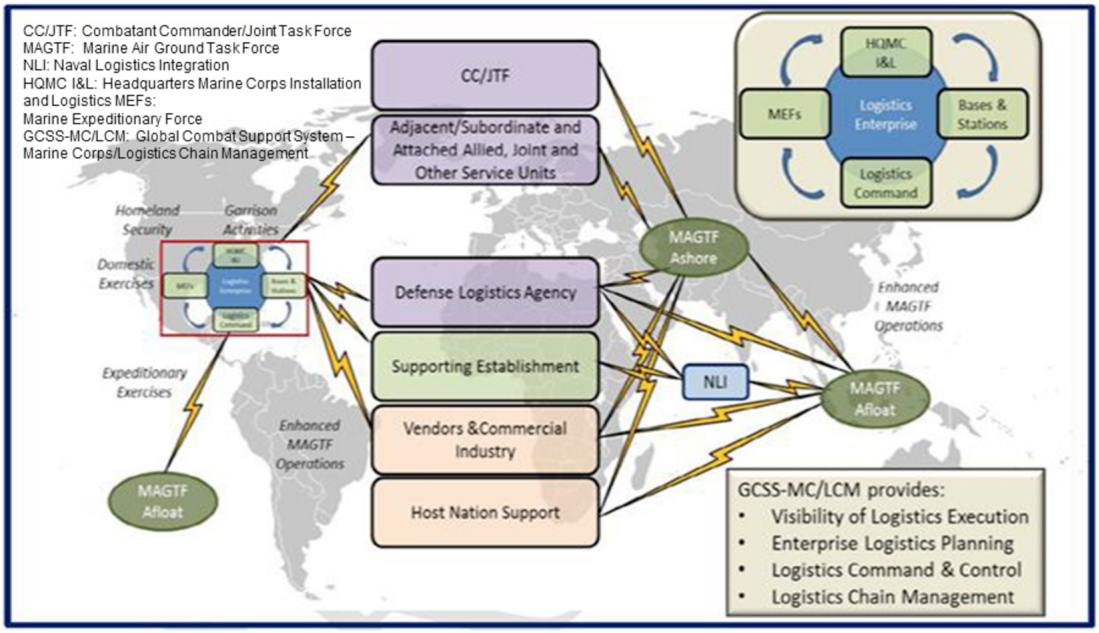


Figure 1.1. GCSS-MC High-Level Operational Concept Graphic (OV-1). Source: [2].

GCSS-MC users access the system through an Internet-based graphical user interface (GUI) built upon the Oracle E-Business Suite (EBS) database. Figure 1.2 displays an example screen found in the GCSS-MC GUI that allows a user to fill and submit a service request.

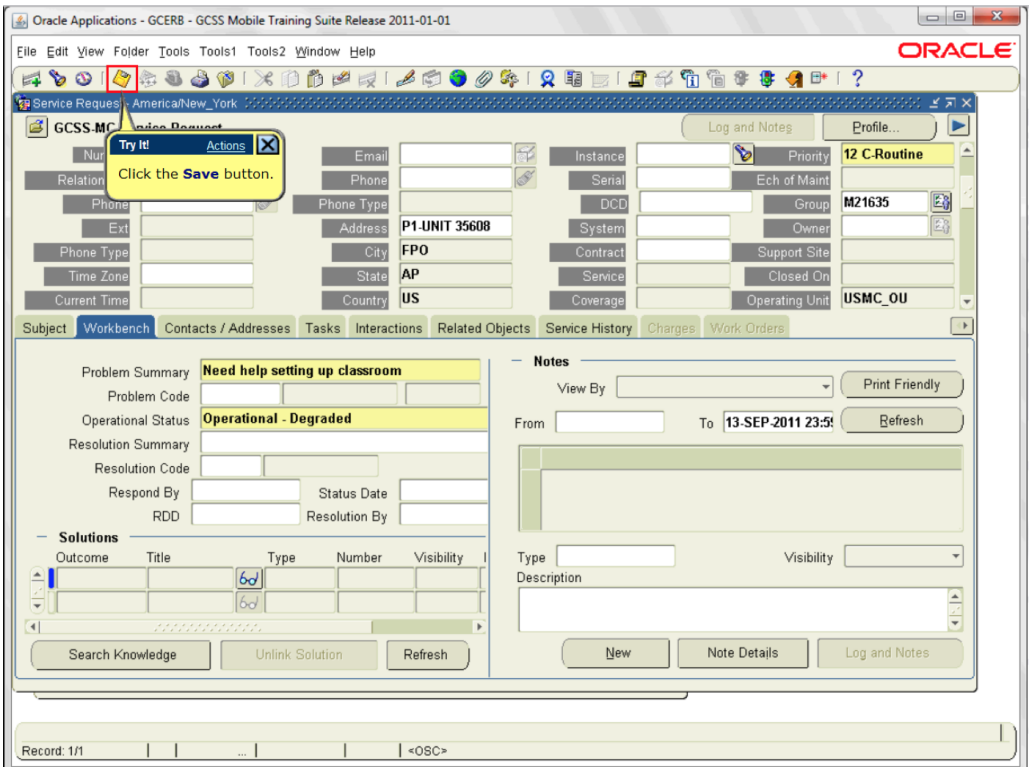


Figure 1.2. GCSS-MC Example Service Request Form. Source: [3].

Nearly every action undertaken in GCSS-MC revolves around the aspect of the service request. Figure 1.2 illustrates the GUI users interact with to submit a service request. Users can submit service requests for services such as requesting repairs, ordering parts, and requesting equipment. GCSS-MC service requests even support non-traditional tasks such as assistance setting up a classroom as seen in the Problem Summary field of Figure 1.2. Once a service request is submitted to the system, anybody with the proper access and authority can view and modify the request.

For example, a Marine may submit a service request desiring repairs to a radio. The radio repairman may access the service request, view the maintenance history of the item, read notes concerning the problem, and note any maintenance completed. This example illustrates interactions between users and the system. Any complexity is abstracted out of the user interaction to create a friendly user experience. Figure 1.3 depicts an example work queue containing all of the open service requests for some unit. The figure reveals some of the underlying complexity of the system which lies in the Oracle EBS. Notice that the work

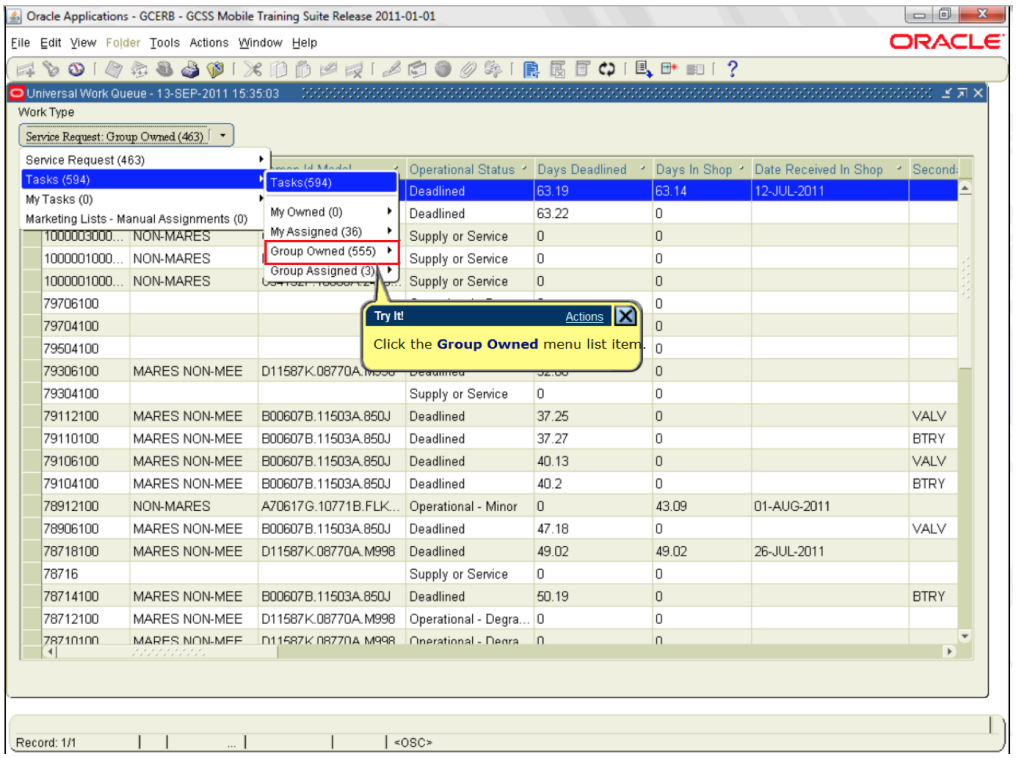


Figure 1.3. GCSS-MC Example Work Queue. Source: [3].

queue resembles a database. All of the information entered in a service request, as depicted in Figure 1.1, is incorporated into a database schema that controls the information and can be accessed as seen in Figure 1.3.

Naturally, the GCSS-MC database contains data that includes information pertaining to Marine Corps equipment. This presents two areas of interest. First of all, every Marine unit (that has been granted permission) accesses the same physical Oracle servers to access GCSS-MC. The size and frequency of the data retrievals vary from unit to unit. Generally speaking, the greater the amount of data, the longer the time it takes to retrieve it. As the data in the GCSS-MC database is growing rapidly, existing infrastructure may not handle the demand for the data in a timely manner. Techniques should be developed that address data growth through software without having to invest in costly additions to infrastructure. Additionally, the current most organized view of the database exists in the work queue. This solution may be sufficient for single units, but observations can be more difficult to perceive as GCSS-MC aggregates data from multiple into one giant database.

1.1 Problem Statement

GCSS-MC treats every Table of Authorized Material Control Number (TAMCN) equally; it does not distinguish TAMCNs as it provides content to users. Since it does not distinguish among TAMCNs, data retrieval from the database can be slow. New infrastructure may obscure the speed of a database pull, but the underlying issue remains and as the data grows, hardware improvements may cost more than a software solution. The fundamental function of GCSS-MC remains providing the user with a situational awareness of the data in a timely manner. The data is the key component of this function.

Network science analyzes relationships in information. Since the priority is data, network science can identify influencers in the data that can be optimized to provide faster performance while maintaining the same network infrastructure.

1.2 Research Questions

Network science may provide insight to the relationships found within the GCSS-MC database which can help optimize it for access. This thesis seeks to answer the following questions:

- Can network science techniques identify influential portions of the GCSS-MC database?
- What methods are best suited for finding influence in a database?
- Does GCSS-MC data model real life systems?
- What can be done to optimize the performance of the GCSS-MC database?

1.3 Relevance to Department of Defense (DOD)

This research benefits the Marine Corps by providing insight into methods that can be used to improve system performance. This study can also be applied to other entities within the DOD since other services also need to manage logistics and supply systems. Although this research applies to a logistics system, fundamentally this research seeks to improve database performance in general. Therefore, this research has the potential to improve databases throughout the DOD. Additionally, the methodology described in this thesis

may provide DOD entities with additional data analysis tools that enhance understanding of patterns and trends within databases and can assist in the decision making process.

1.4 Thesis Organization

We organize this thesis in the following format. Chapter 2 begins with a coverage of prior work conducted on the GCSS-MC database and an introduction to graph theory principles found in this thesis. We also discuss degree, eigenvector, betweenness, and closeness centrality and introduce well known synthetic models. We conclude by introducing other network properties that characterize networks. Chapter 3 describes our methodology, which identifies tables in the GCSS-MC database that contain information used to construct the network, export the tables to a usable file format, and construct and analyze the network. Chapter 4 applies the methodology described in Chapter 3. We also provide an analysis of the data and describe the formula we developed to rank data within the database. Chapter 5 presents the findings of this thesis and provides recommendations for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background

This chapter provides background information on network science. We also address previous work and analysis conducted on GCSS-MC as it pertains to this thesis. We begin by introducing topics in graph theory that are used in constructing a complex network. Once the reader obtains an overview of graph theory principles and an understanding of the importance and functionality of complex networks, we discuss methods for analyzing networks. Specifically, we introduce four network centralities commonly used to determine influence in a network. For each centrality, we discuss cases where they are normally used and research associated with them. After providing an example that encompasses all the centralities we introduce research that combines centrality measures. We also discuss three widely used synthetic models used to model networks. We conclude the chapter by introducing other important concepts of network science used in this thesis.

2.1 GCSS-MC Research

Although we introduce GCSS-MC in Chapter 1, we focus our attention on TAMCNs. In this section, we also discuss previous research conducted on GCSS-MC. Refer to Section 2.1 in Bitto's [4] thesis for further information about the architecture of the GCSS-MC database. Marine Corps Order 4400.150 [1] identifies the TAMCN nomenclature by the commodity, item number, and major class and subclass of Marine Corps inventory. We discuss establishing relationships between TAMCNs in further detail in Chapter 3.

2.1.1 Finding Redundant Entries in GCSS-MC Database

Previous studies [4], [5] on performance issues affecting GCSS-MC analyzed the growth of data associated with GCSS-MC data tables. Das [5] received data tables representing a 2014 snapshot of GCSS-MC tables. Bitto [4] loaded four gigabytes of data into an Oracle 11G single instance database and focused on long-running Structured Query Languages (SQLs) that involve multi-table joins. The research examined moving these SQLs to a 10-node, 100-terabyte Hadoop Distributed File System (HDFS) cluster and using Java code to achieve the same result.

A follow-on research effort [5] continued to analyze the data integrity of the tables. If there is duplicate or non-existing information in the database, the time needed to execute a query increases. Duplicate information includes rows in a table loaded multiple times with the same data. Non-existing information is an entry in a table that is blank. The study conducted the analysis on all 3174 tables in the database. About half of those tables are actually Oracle system tables. The script written in R analyzes every table, generates a statistical summary for each table, and outputs a .txt file with the analysis.

```
     AAC                TAMCN               NSN             NOMENCLATURE
Length:8345         Length:8345         Length:8345         Length:8345
Class :character    Class :character    Class :character    Class :character
Mode  :character    Mode  :character    Mode  :character    Mode  :character


CLASS_OF_SUPPLY      RECORD_KEY            ID_NO              MODEL_NO
Length:8345         Length:8345         Length:8345         Length:8345
Class :character    Class :character    Class :character    Class :character
Mode  :character    Mode  :character    Mode  :character    Mode  :character


ITEM_EXIT_DATE                       DATA_SOURCE     SOURCE_FROM
Min.   :1950-01-01 00:00:00      Min.   :1.000        Length:8345
1st Qu.:2009-12-01 00:00:00   1st Qu.:1.000        Class :character
Median :2016-07-01 00:00:00   Median :2.000        Mode  :character
Mean   :2016-03-09 02:28:47   Mean   :1.848
3rd Qu.:2021-10-01 00:00:00   3rd Qu.:2.000
Max.   :2036-07-01 00:00:00    Max.   :4.000
```

Figure 2.1. Example MDR Table Output. Adapted from [5].

Every .txt file represents an analysis of a unique table. Refer to Figure 2.1. The analysis includes the number of rows in the table, the name of every column, the minimum value of each column, the values of the columns split into quartiles, and the maximum value of the column. The study recommended methods to investigate the data by first determining the number of rows to ensure the amount of data on hand matches the amount that should actually be there. Then they examine the form of the data. For example, if the input data is a character, but it was originally a number, the SQL performance would be affected since it needs to map the number to the character type. The system then identifies potentially problematic data entries. An example of a potentially problematic data entry is a date decades ago when the entries referring to it did not exist or one in the future. Consider an entry adding a computer to the database in 1920 although computers were not invented until

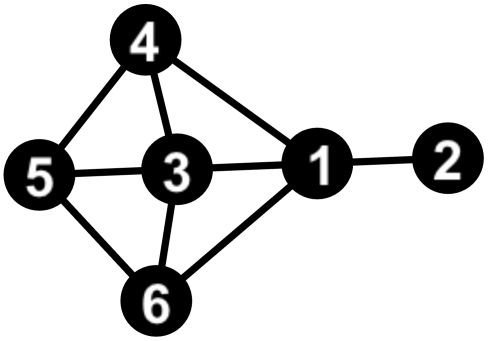
later. Also, consider an entry for a computer in maintenance in 2037. A computer currently in maintenance cannot have a current date two decades in the future. It then examines the correctness of the data distribution. The data distribution is correct if, for example, 75% of dates are earlier than 2021 and 25% of dates are earlier than 2009. Lastly, it runs all possible types of SQL queries and further investigates slow queries.

2.2 Principles of Graph Theory

West [6] defines a graph G as a triple consisting of a vertex set $V(G)$, an edge set $E(G)$, and a relation that associates with each edge two vertices (not necessarily distinct) called its endpoints. In this thesis, we refer to a vertex in $V(G)$ as a node. Figure 2.2 represent two main types of graphs, directed (digraph) and undirected graphs. Rosen [7] defines a digraph as consisting of a nonempty $V(G)$ and a set of directed edges in which an edge is an ordered pair (u,v) where the edge starts at node u and ends at node v. A loop in a graph connects a node to itself. Rosen [7] states an undirected graph does not have any edges with direction. Rosen [7] also states that a simple digraph does not have repeated directed edges or loops. A directed multigraph can have multiple edges either to other nodes in $V(G)$ or to itself. Rosen [7] further states that a multigraph is an undirected graph that contains multiple edges but no loops. West [6] defines the order of a graph, $n(G)$ as the total number of nodes in $V(G)$ and the size of a graph as the total number of edges, $e(G)$.

Rosen [7] defines the degree of a node v in an undirected graph as the number of edges incident on it, except that a loop at a node contributes twice to the degree of that node. West [6] states an edge e is incident to a node u if u is an endpoint of e and the degree of some node u belonging to $V(G)$ is written as $d(u)$. For example, in Figure 2.2 (a), the degree for node 3 is $d(3) = 4$. Every node of a digraph will have a degree. This degree is partitioned in an in degree and an out degree, annotated as $d^+(u)$ and $d^-(u)$ for some node u, respectively. The sum of $d^-(u)$ and $d^+(u)$ equals $d(u)$. In Figure 2.2 (c), $d^-(3) = 1$ and $d^+(3) = 3$. So, $d^-(3) + d^+(3) = 4$.

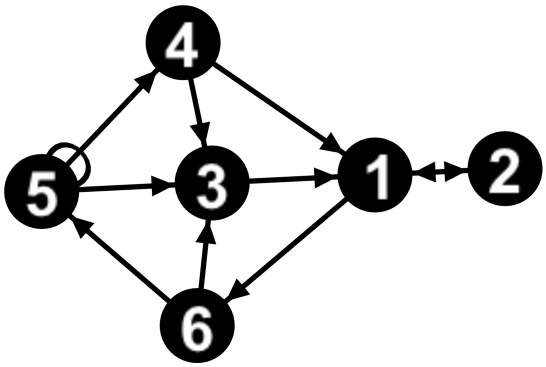
Rosen [7] states that two nodes u and v are considered neighbors, also referred to as adjacent, if u and v share an endpoint in an edge. West [6] defines a neighborhood of some node u as the set of all nodes adjacent to u, written as $N(u) = 1...n$. In both (a) and (b) of Figure 2.2, $N(3) = \{1, 4, 6, 5\}$. There are various ways to represent a graph. This thesis utilizes



(a) Undirected Graph

	1	2	3	4	5	6
1	0	1	1	1	0	1
2	1	0	0	0	0	0
3	1	0	0	1	1	1
4	1	0	1	0	1	0
5	0	0	1	1	0	1
6	1	0	1	0	1	0

(b) Adjacency Matrix for (a)



(c) Directed Multigraph

	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	0	0	0	0
3	1	0	0	0	0	0
4	1	0	1	0	0	0
5	0	0	1	1	1	0
6	0	0	1	0	1	0

(d) Adjacency Matrix for (c)

Figure 2.2. Example Graphs and Corresponding Adjacency Matrices

adjacency matrices as the main graph representation to compute necessary calculations. West [6] defines the adjacency matrix, $A(G)$, as the n-by-n matrix in which entry $a_{i,j} = 1$ if an edge exists in G with endpoints $\{v_i, v_j\}$. Figure 2.2 (b) and (c) are matrix representations of their respective graphs. For Figure 2.2 (b), the adjacency matrix is created by inserting a 1 into the matrix if the two nodes share an edge and 0 otherwise. The nodes are listed in the first row and column of the matrix. The summation of any node's row or column is equal to the degree of that node. The adjacency matrix for Figure 2.2 (d) is similarly constructed. Since the edges have direction in a digraph, the entry corresponding to a pair reflects that. In Figure 2.2 (d), the degree of any node is equal to the summation of that node's column and the summation of its row.

West [6] states that a path is a simple graph whose nodes can be ordered so that two nodes are adjacent if and only if they are consecutive in the list. Some graph G is capable of possessing multiple paths between two nodes. The shortest path between two nodes is the path with the fewest edges between end nodes, also known as a geodesic. For example, in Figure 2.2 (a), $(1, 4, 3, 5)$ is a path between nodes 1 and 5 so (a) contains a $(1, 5)$ path. Additionally, since a path exists between 1 and 5, $(1, 5)$ is said to be connected. Figure 2.2 (a) contains three shortest paths from 1 to 5 including the path $(1, 3, 5)$. Moreover, if for any nodes $u, v \in G$, a (u, v) path exists, G is considered to be connected. Both Figure 2.2 (a) and (b) are connected graphs.

2.3 Complex Networks

In this section, we discuss the concept of complex networks. We explore the definition of a complex network and provide some examples. Additionally, we introduce multilayer networks. In social network analysis, centrality is defined as the position of some person relative to other persons within that social network [8]. We use this definition in general to discuss central, or important, nodes in a network. We introduce different centrality measures such as degree, eigenvector, betweenness, and closeness along with some examples.

2.3.1 Introduction to Complex Networks

Section 2.2 introduces concepts found in Graph Theory. Complex networks are similar to traditional graphs but deal with a much larger scale. Boccaletti et al. [10] define complex networks as "networks whose structure is irregular, complex and dynamically evolving in time, with the main focus moving from the analysis of small networks to that of systems with thousands or millions of nodes, and with a renewed attention to the properties of dynamical units." For example, consider Figure 2.3, created by Burch and Cheswick [9], depicting a map of the Internet. Other examples of complex networks include power grids, road networks, the brain, airline routes, and terrorist networks. These examples illustrate the variety associated with complex networks. They are found across scientific disciplines such as biology, engineering, and sociology.

Due to their size and complexity, complex networks can be difficult to understand and study. Strogatz [11] describes six possible complications associated with complex networks:

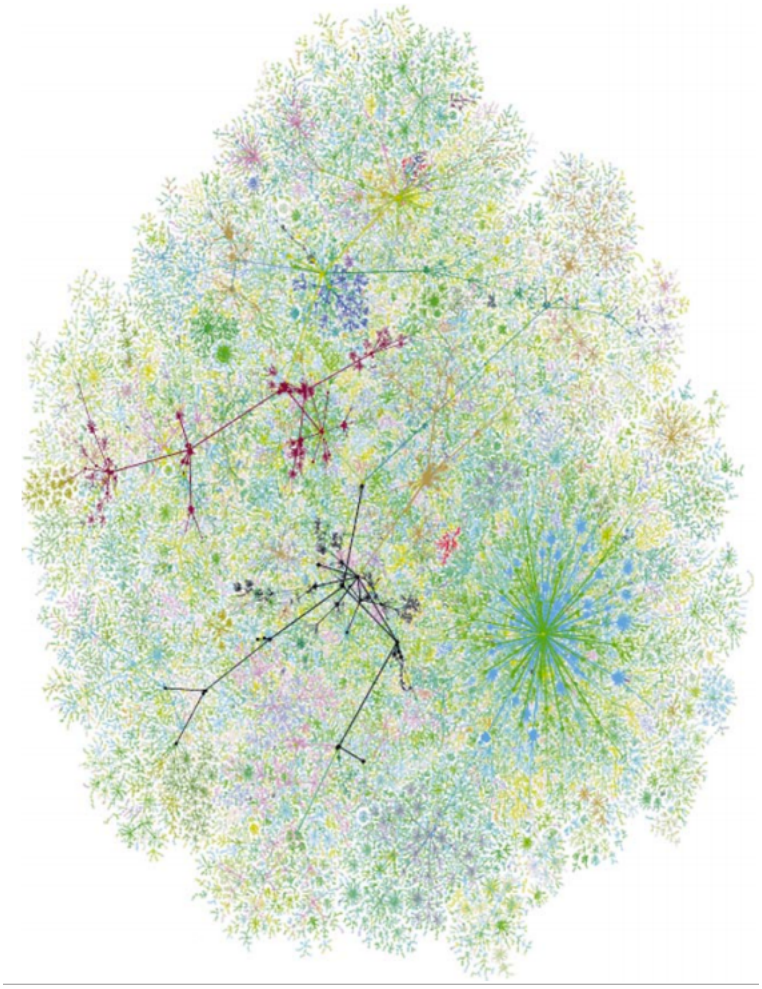


Figure 2.3. Complex Network Depicting Shortest Paths from a Test Host to an Announced Network. Source: [9].

structural integrity, network evolution, connection diversity, dynamical complexity, node diversity, and meta-complication. Structural integrity can be an issue with a lack of understanding of how the system works. Complications with network evolution can be observed with the Internet. Consider Figure 2.3. If Burch and Cheswick [9] wrote *Mapping the Internet* today, the resulting Internet mapping would appear completely different since the Internet evolves and changes frequently. Connection diversity entails the variability of edges in a network with regard to aspects such as weight and direction. Nodes that are nonlinear can exhibit dynamical complexity. Kohn [12] studied cell cycle regulatory networks by organizing known interactions in the form of a diagram, map, and/or database

and developed a molecular interaction map with intricate wiring. The interactions between the diverse types of nodes in this map illustrate complications that can arise from node diversity. Attempting to map the Internet raises meta-complications. This occurs because the physical and logical wiring for the Internet increases the complexity of the structural integrity for an Internet topology and the Internet constantly changes, which affects the network evolution. Meta-complications occur when complications influence one another.

2.3.2 Degree Centrality

In Section 2.2, we defined the degree of a graph. In this section, we extend this principle to a measure of centrality within a network. Each node in a network has a degree centrality. Freeman [13] describes the degree of a node as the count of the number of other nodes adjacent to it and therefore in direct contact to it. Opsahl et al. [14] formalize this measure as:

$$C_D(i) = \sum_{j}^{N} a_{i,j} \tag{2.1}$$

where C_D is the degree centrality, i is the node being measured, j represents all other nodes, and a is the adjacency matrix in which the cell $a_{i,j}$ is 1 if i is connected to j and 0 otherwise. For example, $C_D(3) = 4$ for node 3 in Figure 2.2 (a). While degree centrality is measured per node, a graph can have an average degree. Average Degree, Avg_G, for some graph G is measured as:

$$Avg_G = \frac{\sum_i^N C_D(i)}{n(G)}, \tag{2.2}$$

where n(G) is the order of the graph. For example, the average degree for Figure 2.2 (a) is 3.

Ortiz-Arroyo [8] and Scott and Carrington [15] describe degree centrality as a local measure because given any node u in a network, $C_D(u)$ is confined to its neighborhood $N(u)$. For example, node $2 \notin N(3)$ in Figure 2.2 (a). $C_D(3)$ in Figure 2.2 provides a measure of importance in the graph but does not take into account any nodes outside of its neighborhood. However, [8] asserts nodes with high degree centrality have a higher probability of affecting the overall network since they have a relation to every node connected to them and can quickly transmit information and communicate with other nodes in their neighborhood.

Laxe et al. [16] compared port hierarchies from 2008 and 2010 to determine the effect of changes to maritime port policy following the reaction to the financial crisis of 2008, as documented in [17]. After the crisis in 2008, maritime companies modified their policies to choose different ports. Laxe et al. collected data and constructed a network where nodes represented ports and edges represented shipping lanes between ports. Laxe et al. utilized degree centrality and a form of betweenness centrality p where $p_i = \sum_{s \neq v \neq j \in \{pi\}} \frac{\sigma_{st}(v)}{\sigma_{st}}$ and $\sigma_{st}(v)$ is the number of shortest trajectories between s and t to rank ports based on how influential they are for both 2008 and 2010. They observed that the degree centrality did not vary as much as the p centrality for the hierarchy between 2008 and 2010. Laxe et al. used these comparisons to conclude that the combination of both centralities can precisely determine port hierarchies and that the throughput for the transport network of cargo has contracted. They also conclude mediation carried out by Indonesian ports with respect to movement along the pendulum line of the East of Asia-Northern Range seems to have been consolidated, and the relative weight of emergent port regions located at the entrance and exit of the Panama Canal may have an effect on the potential enlargement of the channel.

Opsahl [14] claims that degree is a basic indicator and is often used as a first step when studying networks. The study of Laxe et al. [16] exhibits this behavior and helps establish a comparison with the p centrality used in their paper. This thesis also begins with a degree centrality analysis as a first step in Chapter 3.

2.3.3 Betweenness Centrality

In Section 2.2, we defined the shortest path of a graph. Betweenness focuses on shortest paths in its measure of centrality. Shortest paths can have a significant role in complex networks. Open Shortest Path First (OSPF) is a common routing protocol used to route Internet traffic. In this routing protocol, as the name states, a packet is routed from some node u to some node v using the shortest path. If t is a node and the path from (u, v) is (u, t, v) in a graph representing a routing network where $E(G) = \{(u, t), (t, v)\}$, then intuitively t has a high betweenness value. Freeman [13] describes a node such as t as a node that controls communication between the two other nodes. Pioro et al. [18] explain in a larger network where there are multiple shortest paths to choose from, OSPF will evenly distribute traffic among those paths. Nodes along these paths have different values of betweenness. Freeman [13] expands on his explanation of betweenness in a larger network. When multiple

shortest paths exist between two connected nodes, the nodes exhibit a partial betweenness meaning nodes along those paths possess some control in the transfer of information between the connected nodes. Freeman [13] discusses nodes with partial betweenness as having the potential to control the information flow as a probability. For nodes $u, v \in G$, the probability of using any one shortest path is $\frac{1}{g_{uv}}$, where g is a geodesic (shortest path) from (u, v). If t is a node in a (u, v) path, $g_{uv}(t)$ is equal to the number of shortest paths t is a member of from (u, v). So the betweenness b of node t on a (u, v) path is:

$$b_{uv}(t) = \frac{1}{g_{uv}} \cdot g_{uv}(t) = \frac{g_{uv}(t)}{g_{uv}} \tag{2.3}$$

The probability introduced earlier is $b_{uv}(t)$ for t because it is the possibility that t will be a member of a randomly chosen (u, v) path. Freeman [13] continues and determines a universal betweenness centrality C_B for any node k taking into account all nodes and paths in G by taking the summation of all of the partial betweenness values of k:

$$C_B(k) = \sum_{i<j}^{n} \sum^{n} b_{ij}(k) \tag{2.4}$$

where i and j are connected points and $i \neq j \neq k$ for all unordered pairs in G, and n is the order of the graph.

Table 2.1. Graph Geodesics for Figure 2.2 (a)

	1	**2**	**3**	**4**	**5**	**6**
1	-	1	1	1	3	1
2	-	-	1	1	3	1
3	-	-	-	1	1	1
4	-	-	-	-	1	3
5	-	-	-	-	-	1
6	-	-	-	-	-	-

In our example, we use C_B as formalized by Boccaletti et al. [10]:

$$C_B(i) = \sum_{j,k \in V(G), j \neq k} \frac{n_{jk}(i)}{n_{jk}}, \tag{2.5}$$

where i is the node whose betweenness is measured, j and k are nodes, n_{jk} are the number of shortest paths from (j, k), and $n_{jk}(i)$ are the number of shortest paths from (j, k) containing i. Table 2.3.3 lists the number of shortest paths found in Figure 2.2 (a). Since Figure 2.2 (a) is an undirected graph, the shortest path (i, j) for any nodes i and j is the same as (j, i). Table 2.3.3 reflects that quality of undirected graphs and lists the values only once. Node 3 appears only once in each of the shortest paths for paths $(1, 5)$, $(2, 5)$, and $(4, 6)$. The $C_B(3)$ equals:

$$\begin{aligned} C_B(3) &= \sum_{j,k \in V(G), j \neq k} \frac{n_{jk}(3)}{n_{jk}} \\ &= \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1. \end{aligned} \tag{2.6}$$

Zhou and Mondragón's [19] study on modeling the Internet topology at the autonomous system (AS) level explores which model models the Internet most accurately and introduces their own model. First, they use an actual AS level graph to set a basis for comparison for various measures including betweenness centrality. They study an Interactive Growth and a nonlinear preferential attachment model. An Interactive Growth model creates a random graph and adds nodes through preferential attachment. We discuss preferential attachment in further detail in Section 2.3.8. The network adds nodes per time step with probability $p \in [0, 1]$ and probability $1 - p$. The network adds a new node with probability $p \in [0, 1]$ connecting it with a host that exists in the network and adds two edges from the existing host to two other peers in the network. The network adds a new node with probability $1 - p$ and connects it with two host nodes and adds a new edge between one of the hosts who received the new node and an existing host in the network. Nonlinear preferential attachment favors high degree nodes, and [19] implements this model similarly to Interactive Growth except it uses nonlinear preferential probability. Zhou and Mondragón combine these two models and introduce the Positive-Feedback Preference (PFP) model. In PFP, the network adds a node by attaching it to an existing host and also adds an edge between that host and a peer with probability $p \in [0, 1]$. With probability $q \in [0, 1 - p]$, the networks attaches a new node to an existing host and adds two edges between that host and two peers. With probability $1 - p - q$, the network attaches to two existing host nodes and adds an additional edge between one of those host nodes and a peer. They find that the maximum value of

C_B for the AS and PFP graphs are significantly larger than for the Interactive Growth and nonlinear preferential attachment graphs. Zhou and Mondragón conclude that PFP more accurately reproduces AS level measurements.

2.3.4 Closeness Centrality

Closeness centrality is similar to betweenness centrality in the sense that they both use shortest paths in their centrality measure. While betweenness centrality measures a node based on the number of shortest paths it is a member of, closeness centrality measures the shortest distance from a node to every other node in the graph. Specifically, Freeman [13] describes closeness as the inverse of the sum of the shortest distances from a node to every other node in the graph. The inverse is necessary because the value increases as the nodes are farther apart. Borgatti [20] describes how closeness centrality affects a graph with regard to network flow. Nodes with low closeness centrality in a graph signify that the node is relatively close to other nodes in the graph and can receive flows sooner. Conversely, it can also reach other nodes faster. Costanbeder and Valente [21] relate closeness to a social network. Nodes with low closeness centrality measures are able to more efficiently contact other nodes within the network.

First, Freeman [13] defines the number of edges between two nodes as $d(i, k)$ where k is the node whose closeness is being measured and i is some node in $V(G)$. The closeness centrality C_C for k is:

$$C_C(k)^{-1} = \sum_{i=1}^{n} d(i, k), \tag{2.7}$$

where i is a node connected to k and n is the order of the graph. Freeman describes how this value can be normalized by removing the impact of the size of the graph by taking into account the average distance between k and all other nodes in the graph by subtracting 1 from n. For some node k whose neighborhood is $N(V(G) - k)$, meaning it is connected to all other nodes except itself. $n - 1$ is the minimum sum of the distance to all the nodes. As the normalized value of C_C shrinks, the average distance between a node to any other node grows. Freeman cites the normalized closeness measure as

$$C_C(k) = \frac{n - 1}{\sum_{i=1}^{n} d(i, k)}. \tag{2.8}$$

The closeness centrality for node 3 in Figure 2.2 (a) is

$$\begin{aligned} C_C(3) &= \frac{n-1}{\sum_{i=1}^{n} d(i,3)} \\ &= \frac{6-1}{d(1,3)+d(2,3)+d(4,3)+d(5,3)+d(6,3)} \\ &= \frac{5}{1+1+1+2+1+2} \\ &= \frac{5}{6} \approx 0.833. \end{aligned} \tag{2.9}$$

Ma and Zeng [22] study metabolites in metabolic networks to analyze the connectivity of genome-based metabolic networks. They find the genome based metabolic networks for 65 fully sequenced organisms and perform a connectivity analysis on the entire network. If a sub-network is fully connected, they refer to it as a strong component. The largest strong component in the network is the giant strong component (GSC) containing 274 nodes. The GSC seems to model a scale-free network and resembles the bow-tie structure of the internet. We discuss scale-free networks in further detail in Section 2.3.8. Ma and Zeng use closeness centrality to characterize the connectivity of the GSC. Through closeness centrality, they find the top 10 central metabolites in the E.coli metabolic network. Using this information, they find that eight of these metabolites are in the central metabolism. They also introduce the term overall closeness centralization index to find correlations with the average GSC path length.

2.3.5 Eigenvector Centrality

We discuss degree centrality in Section 2.3.2. While computing degree centrality, every node is considered equal in terms of importance. If a uniform weight is applied, then the weighted degree can be computed and the ranking of the nodes (based on degree centrality) stays the same. Eigenvector centrality takes into account the individual importance of a node based on the importance of its neighbors. For example, according to [23] and [20], a node with a high degree centrality demonstrates an example of an important node in the network. For a node, the eigenvector centrality value increases if that node connects to important nodes, and the converse is true. Bonacich [23] also points out that unlike degree centrality, which is a local measure, eigenvector centrality takes into account the

entire network because indirect connections also affect the eigenvalue of a node. Bonacich describes eigenvector centrality in two equal ways:

$$Ax = \lambda x, \qquad \lambda x_i = \sum_{j=1}^{n} a_{ij} x_j, \tag{2.10}$$

where A is the adjacency matrix, λ is the largest eigenvalue of A, x is the eigenvector associate with λ, n is the order of the graph, and a_{ij} is an entry in the adjacency matrix.

Seary and Richards [24] describe that eigenvalues are affected and related to other network features such as diameter, cycles, number of triangles in a graph, and graph clustering.

To find the eigenvector centrality for Figure 2.2 (a):

$$x_i = \sum_{j=1}^{n} a_{ij} x_j = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \\ 3 \\ 4 \\ 3 \\ 3 \end{pmatrix}. \tag{2.11}$$

Notice that $x(1)$ in Equation 2.11 is the degree centrality vector. It is distance 1 from each node. Continuing with the equation:

$$x(2) = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 1 \\ 3 \\ 4 \\ 3 \\ 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 4 \\ 8 \\ 14 \\ 10 \\ 10 \end{pmatrix}. \tag{2.12}$$

$x(2)$ in Equation 2.12 represents a weighted degree centrality with a distance of 2 or less.

We continue with $x(4)$ which yields the weighted degree centrality with a distance 4 or less:

$$x(4) = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 36 \\ 11 \\ 25 \\ 45 \\ 32 \\ 29 \end{pmatrix} = \begin{pmatrix} 110 \\ 36 \\ 76 \\ 142 \\ 99 \\ 93 \end{pmatrix}, \tag{2.13}$$

normalizing $x(4)$, according to Newman [25], yields the desired eigenvector. We use Networkx (which we discuss in further detail in Section 3.1) to determine the eigenvector:

$$\begin{pmatrix} 0.4504 \\ 0.1367 \\ 0.5138 \\ 0.4167 \\ 0.4089 \\ 0.4167 \end{pmatrix}. \tag{2.14}$$

The eigenvalue corresponding to node 3 in Equation 2.14 is 0.5138. It is the highest eigenvalue in the graph. It is also a member of four triangles, which is the highest in the graph. This correlates with Seary and Richard's [24] statement about the relationship with eigenvector centrality and number of triangles, namely that eigenvector centrality increases as the number of triangles increases.

Lohmann et al. [26] apply eigenvector centrality to analyze connectivity in functional magnetic resonance imaging (fMRI) data associated with the human brain. They apply eigenvector centrality to fMRI data for the first time. Previous fMRI studies using Network Science focus on other centrality measures such as betweenness. Lohmann et al. decide to use eigenvector centrality because it is less computationally intensive than betweenness and therefore can be applied to larger areas of the cerebrum. The study applies an eigenvector measure to each voxel in the brain. Yuhas [27] defines a voxel as a 3-d section of the brain containing about one million brain cells. This paper analyzes a network of about 40,000 voxels, which belabors the necessity of using eigenvectors to cover a larger portion of the brain than betweenness. Voxels with high eigenvector centrality are connected and

correlated with other central voxels in the brain. Lohmann et al. [26] collect data by conducting two experiments. Both experiments ask subjects to fixate on a cross on a screen while magnetic resonance imaging (MRI) scanners scan the subject's brain. One group is in a state of hunger while the other is in a state of satiety. This study finds that the left and right thalamus possess higher eigenvector centralities in the experiment conducted on sated subjects, suggesting that the subjects may have fatigued as the experiment continued. Lohmann et al. cites research proposing that the thalamus affects mediating attention and arousal to back the claim. The study also finds high eigenvector values in cortical and subcortical areas.

2.3.6 Centrality Summary

In this section, we summarize the centralities discussed from Section 2.3.2 through Section 2.3.5 by examining a network that utilizes these measures created by Beveridge and Shan [28]. Beveridge and Shan create a social network to analyze characters from George R. R. Martin's *Game of Thrones: A Storm of Swords*. In the social network, nodes represent characters from the book and edges connect nodes if names appear within 15 words of one another. Edges in this network are weighted and if there already exists an edge between two nodes, the edge weight is incremented.

Figure 2.4 portrays the *Game of Thrones* social network graph. The size of the node represents degree centrality. The larger the node, the greater the degree. The greater the degree centrality for a node, the more nodes that node is connected to. According to the study and by visual inspection, Tyrion has the highest degree Centrality. Tyrion also has the lowest closeness and highest eigenvector centrality. Tyrion's low closeness centrality means that his average shortest paths to other nodes are shorter than others. Tyrion can more quickly spread information through the network. High eigenvector centrality signifies that Tyrion is connected to other influential nodes. Examining both the book and Figure 2.4 illustrate that Tyrion indeed has many interactions with other main characters of the book. The only other character that has some centrality higher than Tyrion is Jon. Jon has the highest betweenness centrality. Beveridge and Shan [28] note that this signifies that as information spreads through the graph, Jon is positioned to interact with multiple communities sending information.

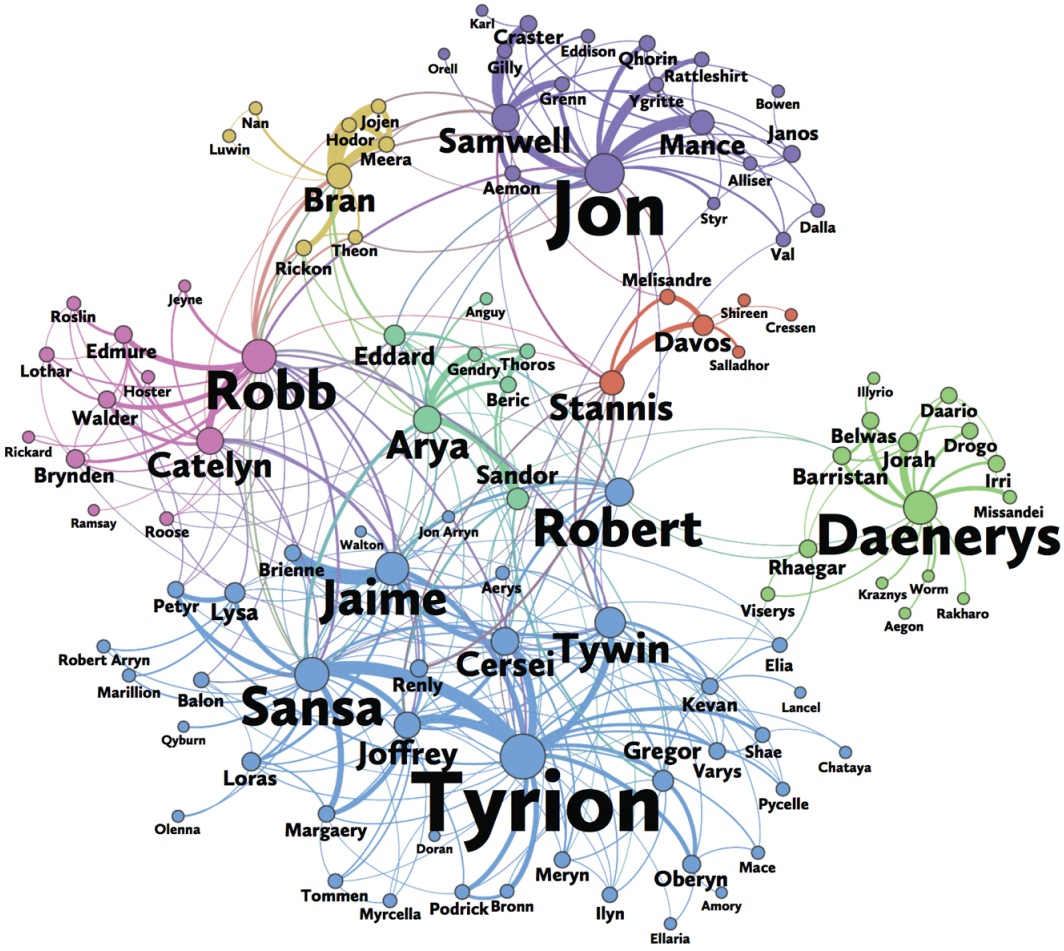


Figure 2.4. *Game of Thrones* Social Network. Source: [28].

Beveridge and Shan [28] also discuss the importance of using various measures of centralities because they complement each other since they assign influence based on different parameters. We apply this logic in Chapters 3 and 4 as we determine the most influential TAMCN.

2.3.7 Combining Centrality Measures

In this section, we discuss methods used to combine centrality measures. Each of the centrality measures highlights importance in different ways. Some measures are more important than others depending on the circumstance. Daly and Haahr [29] and Hui et

al. [30] apply social network analysis concepts to their respective research and use multiple measures of centrality to determine overall importance in their respective networks.

Daly and Haahr [29] apply social network analysis techniques in their study of mobile ad hoc networks (MANETs). They introduce aspects of MANETs that demonstrate the difficulty in delivering messages such as the inconsistent topology of the network and the possibility of many disconnected components. They frame these complications as an information flow problem within social networks. They predict that information flow in a MANET can be achieved using betweenness centrality and similarity and tie strength calculations. Similarity and tie strength calculations are used since a MANET changes. They use similarity to examine the common nodes between the node measured against the destination node and calculate it by summing the overlapping neighbors between the two nodes. This calculation assists in ranking the adjacencies established and predicting future relations. They aggregate various metrics to establish tie strength between connections including frequency, intimacy/closeness, and recency. Frequency calculates how often a node is encountered. Intimacy/closeness measures how long two nodes remain connected. Recency measures how recent two nodes encountered each other. Tie strength is the summation of frequency, intimacy/closeness, and recency. They use betweenness, similarity, and tie strength to develop utility measures for each one that are used to determine which node is the best carrier for some message. Each utility measure is assigned to a node u with a destination node d compared to some other node m. The utility values have equal importance and the total utility value is the sum of the utilities, which they call the SimBetTSUtil value. This is also the name of the routing protocol using these utility measures. They measure each of the utilities and simulate SimBetTS routing, named after the SimBetTSUtil value, and compare it to Epidemic Routing and PRoPHET routing. They find that the betweenness utility provides the best overall performance in delivering messages although it causes greater congestion in central nodes. Compared to PRoPHET routing, SimBetTS performs better. SimBetTS yields similar performance results to Epidemic routing except that its message delivery contains less overhead.

Hui et al. [30] apply social network techniques in their study of delay-tolerant networks. First, they present the problem of forwarding data in networks that develop in an ad hoc manner such as those containing smart devices that connect to networks intermittently. They cite previous methods of dealing with ad hoc networks using routing tables that update base

on the current topology of the network but it is cost ineffective since the network changes frequently and the routing tables only partially captures the structure of the network at any given time. Hui et al. propose using network centralities found in an ad hoc network to design the routing algorithm. Specifically, they focus on community detection methods and general centrality measures. We discuss community detection in further detail in Section 2.3.9. The general centrality measure used in this paper is similar to degree centrality in a digraph in regard to the degree of the node except that this centrality only counts unique neighbors connected within a certain time frame. They develop the BUBBLE algorithm in which some node u sending a message to some destination d traverses a global hierarchical ranking tree based on the general centrality previously discussed until it locates a node within the same community. Then a local ranking system sends the message until the message either reaches d or the hop count associated with the message expires. They implement BUBBLE in an ad hoc network and compare its performance to PRoPHET and SimBetTS routing. They find that BUBBLE routing performs similarly to both PRoPHET and SimBetTS but uses fewer resources.

2.3.8 Synthetic Models

In this section, we study various methods used to create synthetic networks. Modeling a network enables researchers to potentially quickly and easily study real life phenomena. For example, infectious diseases can spread quickly and their impact is substantial on society. Synthetic models are relevant to the study of infectious disease because its growth can be studied without the necessity of observing a live outbreak. Other phenomena can be expensive to reproduce. For example, modeling the network structure of the Internet or a power grid would be cost prohibitive if actual equipment were used. We review random graphs as formalized by Erdős and Rényi, small-world graphs as formalized by Watts and Strogatz, and preferential attachment as formalized by Barabási and Albert.

Erdős and Rényi [31] discuss the creation of a random graph in terms of the probability that edges form given a fixed $V(G)$ for some graph G and a probability p. They begin by describing the number of graphs that can be created from $V(G)$. If $E(G)$ for a random graph is empty, then the probability to create edges in G is 0. They establish that there are $\binom{\binom{n(g)}{2}}{e(G)}$ total possible number of graphs created with $n(g)$ nodes and $e(g)$ edges and where $\binom{n(g)}{2}$ is the possible number of edges between $n(g)$ nodes. The number of edges is an outcome

of creating the random graph. They prove properties relating to random graphs such as the probability of creating a completely connected graph, the probability that the greatest connected component (the subgraph of G with the greatest amount of connected nodes) has $n - k$ nodes where $k = 0, 1, 2, ...$, the probability that a random graph contains $k + 1$ connected components, and the probability that as edges are added to a random graph in its construction, edges that are not in the graph have the same probability to be selected as the next edge. Erdős and Rényi [32] conduct a follow on study and examine the probable structure of a random graph. They expect the number of edges in a random is $\binom{n(g)}{2}p$, where this is actually the mean number of edges since the graph is random and the total number of edges can vary for each graph with probability p. Then,

$$p = \frac{e(g)}{\binom{n(g)}{2}}, \tag{2.15}$$

where $e(g)$ is the size desired for the random graph and $n(g)$ is the order of the graph.

Watts and Strogatz [33] define a small-world network as a network that possesses a high clustering coefficient such as a lattice and small average path length similar to random networks. Watts and Strogatz create a small-world network by beginning with a regular graph. A regular graph is a graph of order n where each node is connected to its nearest k neighbors. At this point the graph is highly structured. They introduce a probability p as they rewire edges in the regular graph. A regular graph represents $p = 0$. As p approaches 1, more edges between nodes are randomly rewired to other nodes until $p = 1$ produces a random unstructured graph with a low average path length. We note that the number of edges does not increase or decrease as p changes, but rather the structure of the graph. In other words, $e(G)$ remains constant while $E(G)$ changes. Watts and Strogatz continue that with real life examples such as power grids, p creates a graph with a higher clustering coefficient which stems from more structure and a low average path length allowing nodes to reach each other on shorter paths. In this power grid, we suspect the high clustering coefficient can be attributed to a city power grid while the low average path length can be attributed to connections between cities.

Barabási and Albert [34] continue research on small-world networks by exploring scale-free networks which are a type of small-world network. First, they discuss the interaction between nodes in a network where the interaction is probabilistic and decays with a power-

law distribution. In a network following a power-law, high number of nodes have small degree centralities, which depict the connections to other nodes, and the number of nodes with high degree centrality decreases exponentially. Barabási and Albert study small-world and random networks and observe that those networks fix the order of the graph without modification. They continue with the comparison to real-life networks, namely that they generally do not possess a constant number of nodes. Usually, real life networks change as they grow by adding nodes to the network. Barabási and Albert incorporate preferential attachment through the equation:

$$\Pi(k_i) = \frac{k_i}{\sum_j k_j}, \tag{2.16}$$

where $\Pi(k)$ is the probability that a node k introduced into the network will connect to some existing node i depending on the connectivity of i. In other words, for every node introduced into the graph, it will find a node it prefers based on some connectivity and connect to it. They contrast this distribution to the Poisson distribution commonly found in random graphs. In random graphs, the tail of the distribution contains fewer high value members than a tail with a power-law. This signifies that there are fewer high degree centrality nodes in random graphs. They expand the scaling attributes of power-law distributions to a network incorporating preferential attachment. Since this model follows a power-law distribution, its growth is scale-free. Scale-free growth signifies that the shape of the distribution does not change as nodes are introduced to the graph.

2.3.9 Other Network Properties

In this section, we discuss other network properties used to analyze complex networks. Although the centralities covered in Sections 2.3.2 through 2.3.5 identify importance in a network, other properties are used to gain an understanding of network behavior. We examine community detection, assortativity, and k-core identification in further detail.

Newman and Girvan [35] describe community structure as a network in which densely connected components are connected through a sparse number of edges. These densely connected components are also known as clusters. Community detection methods strive to identify and quantify this structure. For example, Figure 2.5 colors the nodes of the graph based on their modularity class as defined by Blondel et al. [36]. Newman and Girvan [35]

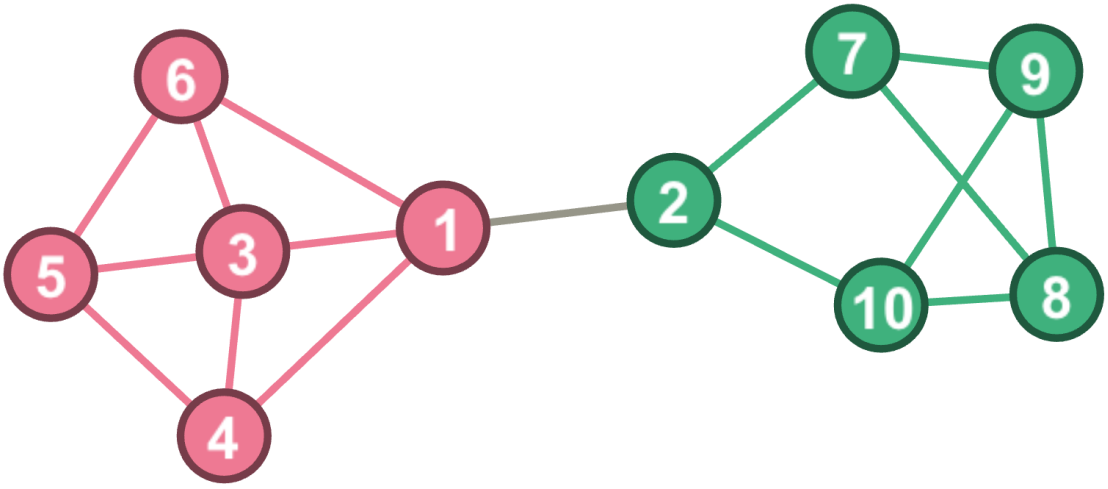


Figure 2.5. Graph Demonstrating Modularity

describe modularity as a quality index for a graph clustering between −1 and 1. They find community structure by assigning betweenness values to all edges in the graph, locating the edge with the highest betweenness and removing it, recalculating betweenness for the remaining edges, and then repeating. Blonde et al. [36] introduce an algorithm named the Louvain Method that finds high modularity clusterings within the graph and partitions the nodes into communities. The algorithm they use is divided among two phases that repeat iteratively. The initial phase partitions the nodes into as many communities as there are nodes and then examines the neighborhood of each node by comparing the node to its neighbors and determining whether the modularity would improve by placing the node in the community of its neighbor. The second phase builds a graph where the nodes are placed in the communities determined in the first phase by summing the edge weights between nodes in the communities. The edge weight in an undirected graph such as in Figure 2.5 remains 1. Afterward, the two phases are iteratively repeated. The modularity for Figure 2.5 is equal to 0.436 and the coloring shows two communities identified by the algorithm.

While modularity partitions nodes based on their membership in a community with relation to the density of that cluster, according to Newman [37], assortativity (also known as homophily) analyzes nodes based on similar connections among nodes. In another paper, Newman [38] describes the importance of assortativity and presents an algorithm for determining the assortativity coefficient of a network. Assortativity recognizes connections that exist based on similar relationships. Newman uses example from social networks to illus-

trate the point but also conjectures that the same phenomenon occurs in nonsocial networks as well. For example, he describes assortative mixing with relation to language. Considering a country with multiple languages, assortative mixing can partition communities based on a particular language which demonstrates that people tend to communicate with other people who speak the same language. Newman [38] discusses assortative mixing by vertex degree in an undirected graph using the equation:

$$r = \frac{\sum_{jk} jk(e_{jk} - q_j q_k)}{\sigma_q^2}, \tag{2.17}$$

where r is the standard Pearson correlation coefficient. The value of r is contained in the range $-1 \leq r \leq 1$. When $r = 1$, the graph is said to be perfectly assortative. Conversely, when $r = -1$, it is perfectly disassortative. The graph resembles a random network when $r = 0$. For a variable of interest such as language as introduced earlier, e_{jk} is the fraction of all edges in the network which joins two nodes with degree j and k. For the degree distribution of a graph G where p_k is the probability that a node chosen randomly has degree k and z is the mean degree of G,

$$q_k = \frac{(k+1)p_{k+1}}{z}. \tag{2.18}$$

The value q_k is the fraction of edges whose endpoints start or end at nodes with values of degree k. The value σ is the standard deviation for q_k.

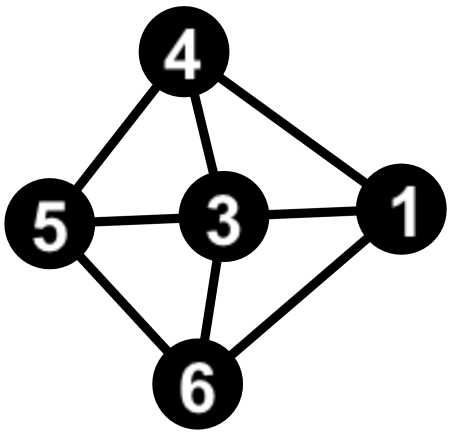


Figure 2.6. k-Core for Figure 2.2 (a)

Studying subgraphs of some graph can be more beneficial in identifying network structures. Consider scale-free networks that possess power law degree distributions as discussed in Section 2.3.8. These networks have many low degree nodes with exponentially decreasing higher degree nodes. Finding the k-core of the graph may bring more insight to the network. For example, Bader and Hogue [39] analyzed yeast protein interactions using k-cores and found that essential proteins exhibit a higher connectivity than non-essential ones. Seidman [40] uses node degrees of subgraphs to define a k-core. Specifically, let H be a subgraph of a graph G and $\delta(H)$ is the minimum degree of H. Then by definition, every node in H will be adjacent to at least $\delta(H)$ other nodes in H. If H is maximally connected and $\delta(H) \geq k$, then, per Seidman, we consider H a k-core of G. Figure 2.6 is the k-core for Figure 2.2 (a). Figure 2.6 is a k-core of 3. This means that $\delta(H) = 3$. Every node has a minimum degree of at least 3 and the graph is maximally connected.

2.4 Summary

In this chapter, we discussed previous studies conducted on the GCSS-MC database. We also introduced network science principles. We examined degree, eigenvector, closeness, and betweenness centralities through real-world applications and examples. Additionally, we reviewed synthetic models, community structure, core structure, and assortativity. In Chapter 3, we set forth the methodology used to create a complex network from information within a database.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Identifying the Most Influential TAMCN

In this chapter, we introduce the methodology created to identify the most influential TAMCN in the GCSS-MC database. Considering this is one of the rare network-centric studies conducted on the GCSS-MC database, we also developed a methodology to extract the pertinent information from the GCSS-MC data. Figure 3.1 outlines the general concept of the methodology proposed in this Chapter. First, we developed a program that searches through the data collected by Das [5] and Bitto [4] mentioned in Section 2.1.1 for keywords associated with the GCSS-MC terms selected as attributes in the complex network. Once we locate the tables associated with the search terms, we use Oracle SQL Developer to manually analyze the tables. Using SQL Developer, we select the columns containing the information needed to determine influence and export the table to a comma separated value (CSV) file.

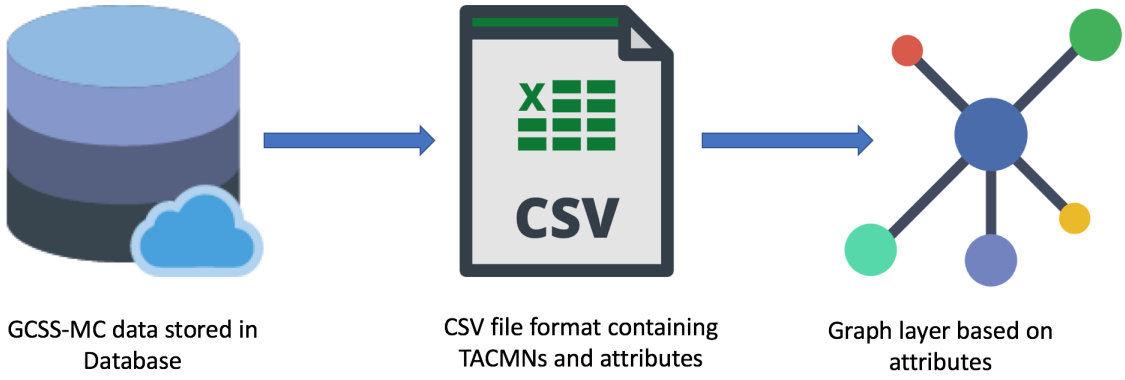


Figure 3.1. Methodology Overview

We downloaded icons created by Madebyoliver [41] and Freepik [42] on www.flaticon.com to create this figure.

We use the CSV files to construct a complex network. Each attribute represents a layer in the network that must be built. Each attribute is a graph. The collection of these graphs comprises the GCSS-MC complex network. We developed a program that reads a CSV file and constructs an adjacency matrix for any given layer. Afterward, the program reads the

adjacency matrix and builds a graph. Once the graph is constructed, we can use network visualization software to visualize the network.

After the graphs have been constructed, we analyze each centrality to determine the importance of the nodes in the network. Additionally, we compare the network to well-known synthetic models in order to match the layers in the network to the models. The goal of this methodology is to determine the centrality measures discussed in Section 2.3 and list every TAMCN in order from most to least important. As Figure 3.1 illustrates, we locate relevant tables in the GCSS-MC database and export it to a CSV file format that we parse and input into a custom built class in order to create a complex network that can be analyzed.

3.1 Tools

In this section, we discuss the tools used throughout the thesis. We use Oracle SQL Developer to examine GCSS-MC tables and export information. To build graphs and complex networks we use Networkx [43]. We use Gephi [44] to visualize graphs and complex networks and to analyze them.

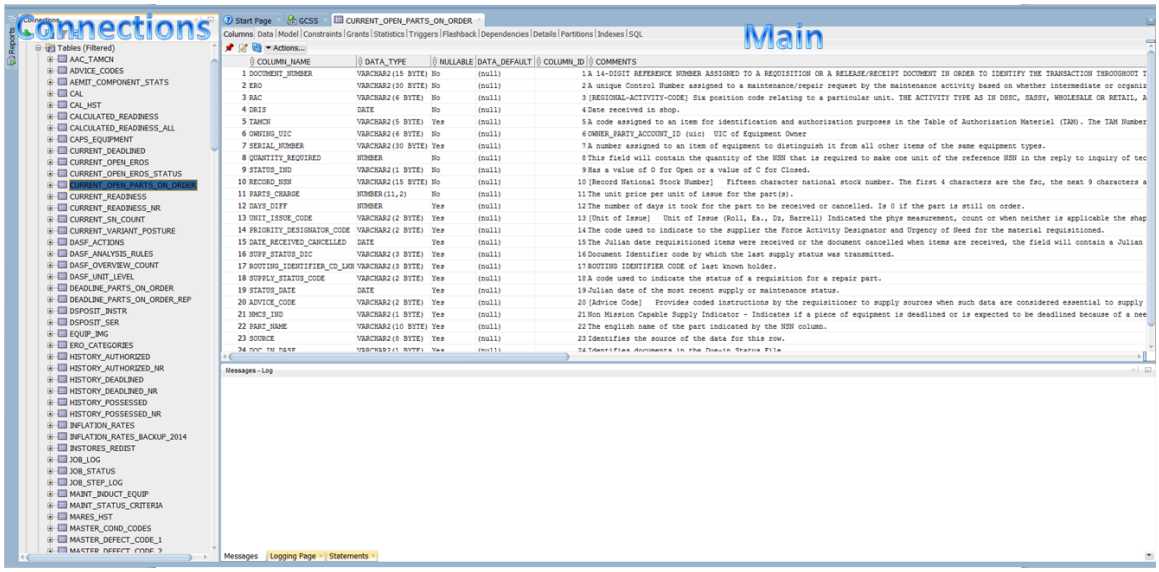


Figure 3.2. Oracle SQL Developer Example

Oracle SQL Developer is a free integrated development environment (IDE) that executes SQL queries and scripts [45]. SQL Developer allows a user to visualize the tables in a database using multiple windows in a GUI as seen in Figure 3.2. This allows a user to use

both SQL queries and buttons from the GUI to find and use data. Through SQL Developer, a user can export selected views and tables to different file formats such as CSV, HyperText Markup Language (HTML), JavaScript Object Notation (JSON), and others.

Networkx is a software package for the Python programming language that was publicly released in 2005 which enables a user to construct, manipulate, and analyze graphs and complex networks [43]. For example, we used Networkx to construct Figures 2.2 (a) and (c) and 2.5. Networkx provides tools to construct graphs in various ways. Nodes can be individually added or added through the addition of edges, and both nodes and edges can be added in batches. We demonstrate how to add edges to a graph to create Figure 2.2 (a) using Networkx:

```
import networkx as nx
graph1 = nx.Graph()
graph1.add_edge(1,2)
graph1.add_edge(1,4)
graph1.add_edge(1,3)
graph1.add_edge(3,5)
graph1.add_edge(3,6)
graph1.add_edge(5,6)
graph1.add_edge(1,6)
graph1.add_edge(4,3)
graph1.add_edge(4,5)
nx.write_gexf(graph1, "graph1.gexf")
```

After importing the Networkx module as `nx`, the variable graph1 instantiates a graph object that includes properties such as nodes and edges. We add edges one by one until the graph is populated. As each edge is added, if the nodes that the edge connects to do not exist, they are added to the graph. Networkx can write the graph to various file formats. In this thesis we use the Graph Exchange XML Format (GEXF) and Gephi file formats. The last line of code demonstrates writing the graph to a GEXF file with the name "graph1.gexf".

Although Networkx possesses the ability to visualize graphs through an implementation of MatPlotLib, we use Gephi. Gephi [44] is a graph visualization and analysis tool that allows a user to interact with graphs to intuitively find patterns, manipulate graph structures, and

run statistics. For example, as previously stated, we constructed Figures 2.2 (a) and (c) and 2.5 using Networkx but visualized them using Gephi. Gephi reads many different file formats and converts them to a Gephi file. In this instance, our GEXF was converted to a Gephi file. Within Gephi, we are able to interact with the graph in various ways such as varying node sizes and colors according to different attributes such as degree centrality or modularity. Gephi also provides pre-built visualization layouts that spatialize the data. For example, we applied the Force Atlas 2 layout for Figures 2.2 (a) and (c) and 2.5. Jacomy et al. [46] describe their Force Atlas 2 algorithm as having nodes repulse and edges attract. The repulsion and attraction is based on distance in the graph. The closer the nodes are, the less likely they are to repulse each other and vice versa. Jacomy et al. mention that their algorithm is scalable up to about 10,000 nodes. Figure 2.5 demonstrates Gephi's ability to quickly run and visualize statistics. After constructing Figure 2.5 using Networkx and visualizing it with Gephi using the Force Atlas 2 layout, we applied the Louvain method of community detection, as described in Section 2.3.9, and colored the nodes based on their modularity class using Gephi. Figure 2.6 illustrates Gephi's ability to find the core of a graph. We applied the same techniques to obtain Figure 2.6 as we did for Figure 2.5 except for the application of community detection. Instead, we filtered the graph by k-core until we found the core. We reach the core in Gephi by incrementing $\delta(H)$ until $\delta(h) + 1$ yields an empty graph. Alternatively, Networkx provides a function to calculate $\delta(H)$ such that $\delta(H) \geq k$.

3.2 Locating and Exporting Tables in GCSS-MC Database

In this section, we introduce the methodology to find which attributes to use in the formation of the complex network and locate where they are referenced in the database. We develop a program to assist in locating columns containing the desired search terms. After finding the table containing the column, we manually examine the table in order to decide which columns to export. We use an SQL query to create a view with only the needed information and then export it to CSV.

3.2.1 Deciding on Attributes

Every layer in the complex network is based on attributes. In this thesis, we define an attribute as a characteristic that is shared by nodes within the network. In each layer of

this network, the nodes represent a distinct TAMCN. The relationship between the nodes in the graph is represented through edges shared between nodes. The layer represents the relationship being examined and an edge represents a specific instance of that relationship occurring between the two nodes. Before we determine what the specific overall layers are, we must identify columns that contain data to establish relationships.

We focus on three broad fields pertaining to GCSS-MC to begin narrowing the attributes each layer represent: procurement, maintenance, and equipment transfers. We developed a Python program named File Extractor to help identify tables that contain the desired information. We introduce previous research pertaining to GCSS-MC in Section 2.1.1. Using the File Extractor script, we search through the .txt files from [5]. Figure 2.1 illustrates an example of the content of one of the .txt files. The .txt files contain the names of each column located in the GCSS-MC table associated with the .txt file name.

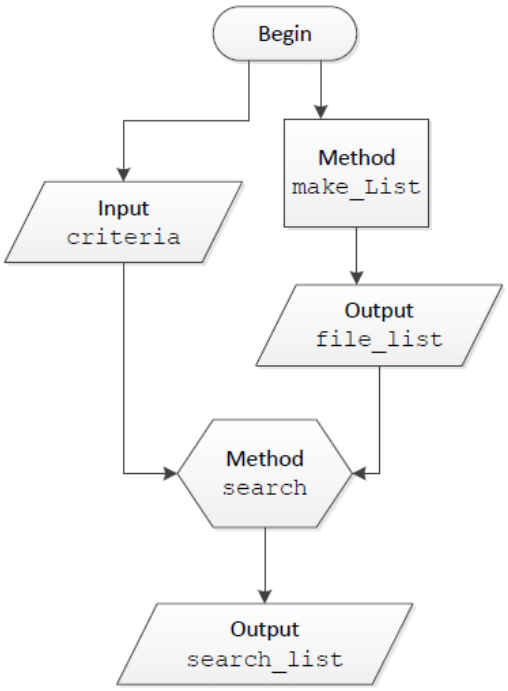


Figure 3.3. File Extractor Script Flow Chart

Figure 3.3 illustrates the logical flow of the File Extractor program. File Extractor uses Python lists as the basic data structure to manipulate data. The two main lists are `file_list` and `search_list`. The list `file_list`, once populated, is a list of every file name in the

directory being searched. The directory contains 3,174 files corresponding to each of the tables analyzed by Das [5]. The list `search_list`, once populated, is a list of each file containing column names that match our search criteria. File Extractor uses two methods to populate `file_list` and `search_list`. The `make_List` method populates `file_list` by using the `OS` library in python to iterate through every file in the directory and add the file name to `file_list`. Once completed, `file_list` serves as a search queue and one of the inputs for the second method, `search`.

Figure 3.3 depicts the method `search`, which takes in two parameters, `file_list` and `criteria`. The parameter `criteria` is a list of strings used to search in each file. We establish the `criteria` list based on our knowledge of the files in the directory. Figure 2.1 illustrates an example of a file found in the directory. We know that the column titles are the only strings in the document that are capitalized. We use this knowledge to define the `criteria` list as a list of strings we expect column titles to possess. For example, we may define the list `criteria` as the list containing the strings `['TAMCN', 'REPAIR']`. For every file in `file_list`, `search` opens the file and for every string in the `criteria` list, `search` checks if the file contains that string instance of `criteria`. If it contains each string belonging to `criteria`, then there exists column names within that file (which represents a table in the GCSS-MC data) that match information we may find useful according to our `criteria` input. Once and if `search` finds matches for all strings belonging to `criteria`, the method appends the file name to the `search_list` list.

The main execution block for File Extractor executes the two methods discussed and then prints each item in the list `search_list` along with the total number of items in `search_list`. The last 10 lines of output produced by executing File Extractor using the `criteria` parameter `['TAMCN', 'REPAIR']` are:

TFSMS.C_TFSMS_TAM_CHG_RV_BK20130518.txt
TFSMS.C_TFSMS_TAM_SS.txt
TFSMS.C_TFSMS_TAM_SS_BK20130518.txt
TFSMS.C_TFSMS_TAM_SS_REV.txt
TFSMS.C_TFSMS_TAM_SS_RV_BK20130518.txt
TFSMS.ECP2098_TAM_ATTRIBUTES.txt
TLCMDR.STARRS_TAMCN_HST.txt
WSTIAC.WST_MIMMS_HIST_EROS_EXCEP.txt

```
WSTIAC.WST_MIMMS_REPAIR_PARTS_EXCEP.txt
The size of the search list is:140
```

This represents the last nine files out of 140 files that contain column names containing both the strings `'TAMCN'` and `'REPAIR'`. Each file name listed follows the pattern `Username.Table_Name.Txt`. The `Username` part of the file name refers to the user the `Table_Name` is listed under in the GCSS-MC database that owns the schema. Now that we have a method for finding desired attributes, we discuss the methodology associated with exporting desired table content from GCSS-MC data using Oracle SQL Developer.

3.2.2 Exporting GCSS-MC Tables

In this section, we discuss locating and exporting the tables we find using the methodology found in Section 3.2.1. First we access the GCSS-MC data by providing the path to the location found in the Naval Postgraduate School (NPS) servers. Figure 3.2 displays an example of the SQL Developer layout of the different windows and tabs available to the user. The connections window of the SQL Developer GUI lists all of the tables in the `Tables` tab. Expanding the tab allows us to view the titles of the tables. We choose a table to view which then displays information in the main window. For the data set we access in this thesis, under the columns tab, SQL Developer displays a description of each of the columns. This allows us to verify information we search to help us determine whether the table contains data that should be exported. The data tab displays the data associated with the table.

The main window also contains a tab that allows users to input SQL queries. We use the select method in SQL to create a view containing the desired columns that we then export. We use the query:

```
select column_i, column_j, ... , column_k from tableName
```

where `column_i` through `column_k` are the columns we want to export and `tableName` is the table they pertain to. Running the query displays only those columns and their associated data in the Messages window. These columns are exported to a CSV file for complex network construction.

3.3 Complex Network Construction and Analysis

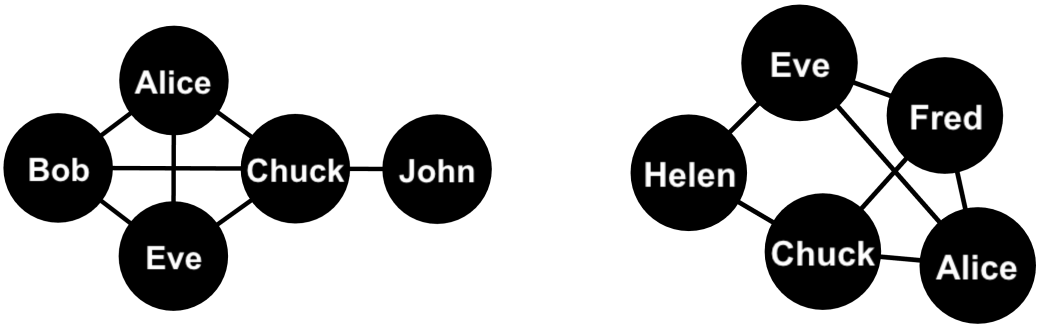
In this section, we discuss the methodology used to create a complex network. We also introduce the methods and data structures implemented to create the complex network from data retrieved from the GCSS-MC database. Then we discuss how we implement Networkx and Gephi to analyze the network. In this section, we discuss the generic methodology while in Chapter 4, we apply the methodology to the GCSS-MC data.

Table 3.1. Sample CSV Pull from a Notional Database

Name	**Friend Made**	**Class Taken**
Alice	Greg	CS101
Alice	Jill	MA101
Bob	Greg	-
Bob	Phil	-
Chuck	Greg	MA101
Chuck	Sam	PH101
Eve	Greg	CS101
Eve	Greg	CH101
Eve	Greg	SW101
Fred	-	MA101
Fred	-	CH101
Helen	-	SW101
Helen	-	PH101
John	Sam	-

First, we begin with a conceptual overview of the complex network construction. Assume Table 3.1 contains CSV entries from a notional database and that we obtained the CSV file using the methodology from Section 3.2 in a social network scenario. The `Name` column contains people that represent nodes in a graph. The `Friend Made` and `Class Taken` columns represent attributes that manifest themselves as edges between nodes.

The goal of Section 3.3.1 is to convert the nodes and attributes found in Table 3.1 to a network which represents it, such as in Figure 3.4. Figure 3.4 represents two layers of a complex network. The first layer represents friends made during some time period. The second layer represents classes taken during the same time period. In Figure 3.4 (a), an



(a) Friendship Layer

(b) Course Taken Layer

Figure 3.4. Network Formed from Table 3.1

edge connects nodes if the nodes have a common friend made. In Figure 3.4 (b), an edge connects nodes if the nodes share a common class. Notice that not all of the nodes are in both layers. Nodes missing from Figure 3.4 (a) signify that those nodes did not take any classes during the time periods they made those friends. Nodes missing from Figure 3.4 (b) signify those nodes did not make any friends during the time periods they took their respective courses. Section 3.3.1 discusses the methodology developed to apply the same concept to the GCSS-MC database.

3.3.1 Building the Network

Section 3.2.2 discusses retrieving data from GCSS-MC and exporting it to a CSV file format. Once obtained, we use the CSV Python library to open the CSV file and the `csv.reader` method to place it into a variable we call `reader`. Figure 3.5 illustrates the methodology used to process the CSV data in a flowchart and convert it to a data structure we further analyze. We convert `reader` to a Python list called `data` where every item in the list is a row of the CSV file. This row is similar to rows found in Table 3.1 in that a column contains nodes and one or more column contains attributes for that node. The CSV file may contain multiple columns of information, however, we focus on two columns at a time, the column containing the TAMCNs since they are nodes and the column containing an attribute that will form a layer. Applying this reasoning to the notional data in Section 3.3, Table 3.1 displays two columns for attributes, and Figures 3.4 (a) and (b) illustrate the layers of the graph representing those attributes.

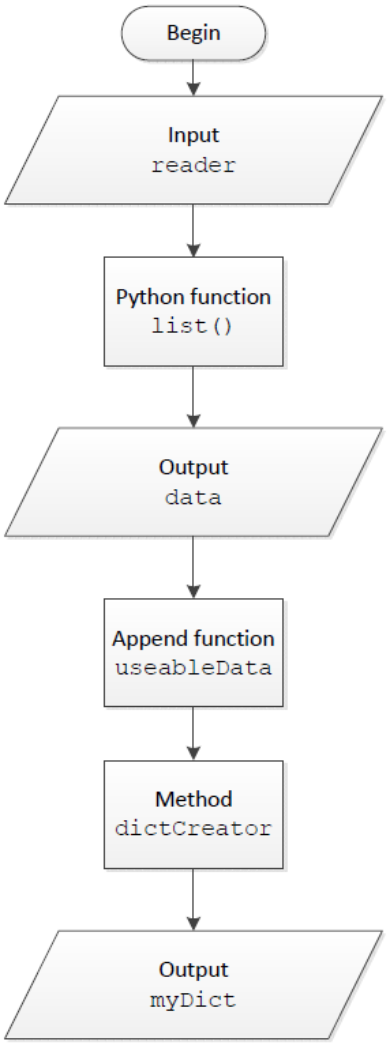


Figure 3.5. Flow Chart Describing CSV File Processing

We iterate through each item in `data` and append a list containing the TAMCN and attribute to a new list we call `useableData`. The list `useableData` contains pairs of a TAMCN and its attributes. Since a TAMCN can have multiple attributes by appearing in several rows but `useableData` lists items one pair at a time, we construct a Python dictionary to list all of the attributes for each TAMCN. We call this dictionary `myDict` and the key is the TAMCN while the value is a list of the attributes associated with that TAMCN. We construct `myDict` through a method we created called `dictCreator` by iterating through the `useableData` list. For the first appearance of each TAMCN, we create a dictionary

entry with the key being the TAMCN and the value being a list containing the attribute. For every subsequent appearance of a TAMCN that has been added to `myDict`, we append the attribute to the list contained in its value.

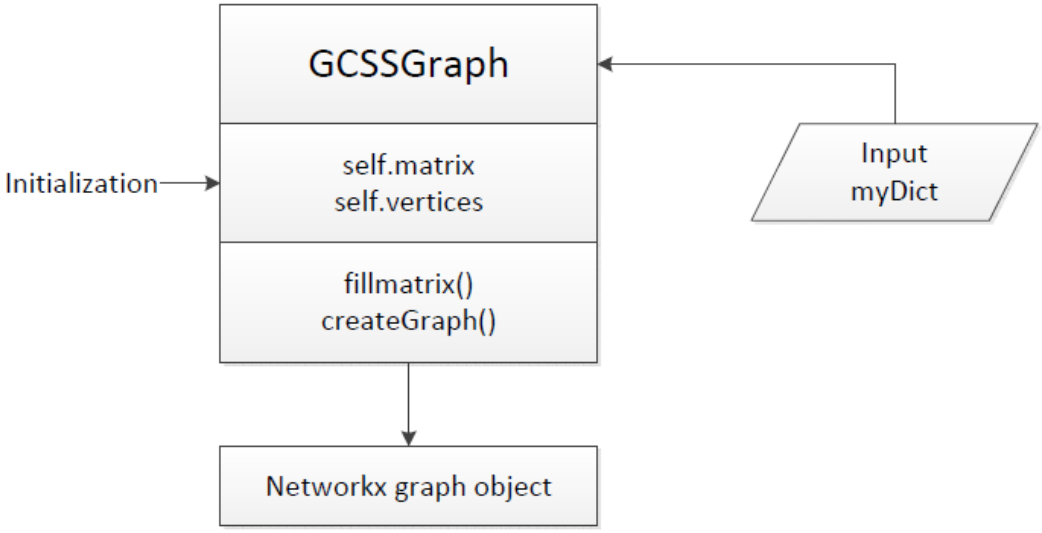


Figure 3.6. GCSSGraph Class Layout

We use a custom built class structure to construct the network using the length of the `myDict` data structure as a parameter. Figure 3.6 illustrates the class we define as `GCSSGraph`. A `GCSSGraph` object contains the main methods and data structures needed to build and store a complex network such as an adjacency matrix and a Networkx graph object. We initialize a `GCSSGraph` object by defining the amount of nodes the network has based on the length of `myDict`. Afterward, we create a nxn matrix that represents the adjacency matrix of the network where n is the number of nodes in the network. At this point, we have an object with an empty adjacency matrix. This adjacency matrix is the most basic representation of the network. If we were to visualize this network with an instantiated empty adjacency matrix, it would be a graph containing n nodes and no edges.

We apply the methods contained in `GCSSGraph` to continue to build the network. `GCSSGraph` contains the method `fillMatrix`. This method uses the `myDict` dictionary as a parameter to fill the adjacency matrix. First we apply the Python `list()` function to create a list of all of the TAMCNs from `myDict`. Then, for every TAMCN item, we build a local list called `comparisonList` containing the `myDict` value corresponding to

that TAMCN entry. For each item in `comparisonList`, which represents the attribute of the TAMCN, we compare it to every value of the key:value pairs in `myDict`. If a `myDict` key:value pair contains the item from `comparisonList`, then the key from `myDict`, which is a TAMCN, relates to the TAMCN whose value is the `comparisonList`. This means that the TAMCNs, as long as they are not the same TAMCN, contain an attribute that they share in common. This means that they share an edge. Since they share an edge, we increment the adjacency matrix entry a_{ij} where i and j are the two TAMCNs who share the attribute.

We then begin network construction after filling the adjacency matrix. We created a method called `createGraph` that constructs the graph from the adjacency matrix using a list of all TAMCNs from the `myDict` dictionary. First, `createGraph` instantiates a Networkx graph object we call `graph`. Then, for each TAMCN in the list of TAMCNs, we add the TAMCN to `graph` as a node. We instantiate two local integer variables within `createGraph` to represent the row and column of the adjacency matrix. We iterate through the adjacency matrix by row and column and if there is an adjacency matrix entry $a_{ij} \geq 1$, then an edge exists between the TAMCNs representing the row and column. Upon encountering 1 or greater, we use Networkx to add an edge between the TAMCNs. There exists the possibility of having an a_{ij} entry greater than 1. This represents two TAMCNs sharing multiple attributes. However, we only focus on undirected graphs in this thesis. Therefore, we only add one edge despite the possible existence of multiple attribute similarities. At this point, our `GCSSGraph` network object contains a completely filled adjacency matrix and exportable Networkx graph object that represents the matrix.

3.3.2 Analyzing the Network

In this section, we discuss the preparations needed for analysis. We implement the methods discussed in this section in Chapter 4. In Section 3.3.1, we construct the network according to TAMCNs and attributes exported to CSV as described in Section 3.2. We use tools in both Networkx and Gephi to analyze the graph. We also use Networkx to build synthetic models.

We use Networkx to find the eigenvector centrality values for each node in the network. We find the eigenvector values by inserting the graph object of the network as a parameter in the `networkx.eigenvector_centrality(graph object)` method. This method outputs

a dictionary where the key is the node and the value is the eigenvector measure associated with the node. Similarly, we find the assortativity of a network by inserting the graph object as a parameter to the `networkx.degree_assortativity_coefficient(graph object)` method. This method outputs the assortativity value r for the network as discussed in Section 2.3.9. Although Networkx provides the ability to find assortativity by different attributes, our methodology does not distinguish nodes by attribute; therefore, we use the degree assortativity.

As discussed in Section 3.1, Networkx provides the ability to export graphs to GEXF file formats. Once we finish constructing the network, we write the file to GEXF. We continue to analyze the graph in Gephi. We use Gephi to visualize the networks using the provided layouts and run statistics. We find the degree centrality of the network and plot the degree distribution through Gephi. We use Gephi to find other centrality measures including betweenness and closeness. Additionally, we use Gephi to find the modularity of the Network and the number of communities. For each of these measures, Gephi utilizes a Data Laboratory that allows the export of network properties and values to a CSV format. This allows us to export node labels and their corresponding centrality and network measurements to CSV. We use Gephi to identify the core of the graph as well.

In Section 2.3.8, we discussed synthetic models and their importance to Network Science. In this thesis, we use Networkx to construct the three main models discussed in Section 2.3.8. The method `networkx.erdos_renyi_graph(n, p)` creates a synthetic random graph with parameter n and p where n is the number of nodes in the synthetic network and p is the probability that edges form. The method `networkx.watts_strogatz_graph(n, k, p)` creates a synthetic small-world network with parameters n, k, and p where n is the number of nodes in the synthetic network, k is the number of nearest neighbors that each nodes connects to, and p is the probability that edges rewire to different endpoints. The method `networkx.barabasi_albert_graph(n, m)` creates a synthetic scale-free network with parameters n and m where n is the number of nodes in the synthetic network and m is the number of edges incoming nodes create as they join the network. For each synthetic network we use, we export the network to a GEXF file and analyze it using Gephi.

3.4 Summary

In this chapter, we provided a methodology to build a layer in a complex network according to attributes that we decide. We introduced the tools necessary to carry out the methodology and then demonstrated how to determine attributes and find and export tables from the GCSS-MC data. Afterward, we delved into constructing the network using a custom built class structure involving adjacency matrices and Networkx graph objects. We concluded the chapter with an explanation on analyzing the complex network. We apply the methodology discussed in this chapter and determine the most important TAMCN in Chapter 4.

CHAPTER 4:
Results and Analysis

In this chapter, we describe implementation of the methodology discussed in Chapter 3. We search through the GCSS-MC data for tables that contain information related to the three main attributes we chose to examine, specifically procurement, maintenance, and equipment loans and transfers. Once we find and select the tables, we convert the columns we desire to CSV format and run the `GCSSGraph` algorithm. We analyze the centrality measures and synthetic model comparisons for each graph produced by the algorithm in their own layer. Afterward, we discuss trends among the layers and determine the most important TAMCN.

4.1 Attribute Search Results

Table 4.1. Attribute Search Table

Criteria	**Search Hits**	**Tables Used**
TAMCN, REPAIR	140	
TAM, REP, NSN	187	CLC2S.REPAIR_DATA, GCSS_TEST.GCSS_IB_ASSET_MAINT, GCSS_TEST.GCSS_MARES_IB_D...
TAM, ORDER, NSN	192	LCMIDBA.DEADLINE_PARTS_ON_OR..
TAM, PARTS, NSN	119	LCMIDBA.CURRENT_OPEN_P..., LDRDBA.GCSS2_SR_HEADER_HST
TAM, MAINT, NSN	132	GCSS.R001_SRHEADERS
TAM, FIX, NSN	114	LDRDBA.GCSS_MARES_IB_DEADL...
TAM, SERVICE, NSN	118	MLS2FEEDS.X_GCSS_BR2_MER_SR_H..
TAM, LOAN, NSN	88	
LOAN	4	
LEND	0	
TRANSFER	3	
FROM	68	
RECEIVE	128	WSTIAC.WST_EROS

In this section, we methodically search for tables that contain information pertaining to procurement, maintenance, and equipment loans and transfers. Table 4.1 details the various search terms we entered into the `File_Extractor` program described in Section 3.2.1. For each search criteria, we provide the total number of tables that contain the strings used as search parameters. While searching through the tables we observe that many of the same tables appear as results for different criteria. We also provide names of tables we extracted and converted to CSV files. Notice that many of our search criteria contain the string `NSN`. Marine Corps Order 4400.150 [1] defines a national stock number (NSN) as a number that conveys information about a specific item of supply. NSNs belong to TAMCNs. For example, a truck can be a TAMCN. The truck is a system composed of many parts. Each part that comprises the truck needs to be identified. This identification occurs through the use of a NSN. We find that TAMCN and NSN pairs are the most useful pieces of information because we can easily establish relationships between them. Therefore, our searches favor tables that contain both TAMCNs and NSNs. We cautiously avoid pitfalls that lead to networks that do not convey any meaning. For example, we do not search for TAMCNs that share a cost. Although we can apply the methodology, the resulting network would be meaningless. All TAMCNs that cost the same amount of money would connect to one another and no other TAMCNs. This would produce a disjoint graph containing x completely connected communities where x equals the number of different prices. Using an NSN as a characteristic, on the other hand, provides depth. Since each TAMCN contains many NSNs, many different TAMCNs can relate through a common NSN.

Although we found 10 different tables containing information including TAMCNs and NSNs, we only use those tables which produced edges between TAMCNs. The following three tables are finally selected:

- LCMIDBA.CURRENT_OPEN_PARTS_ON_ORDER
- LCMIDBA.DEALINE_PARTS_ON_ORDER
- WSTIAC.WST_EROS.

The graphs with just a collection of nodes without any edges that correspond to unused layers have no NSNs in common. This signifies that for all of the tables we selected except for the three listed, no two TAMCNs shared an NSN in common. Additionally, we did not find any tables pertaining to loaning or transferring TAMCNs. At first we attempted to

search for specific words relating to lending an item. When that failed, we searched for tables containing the string `FROM` since it suggests some sort of exchange. Notwithstanding, we searched through all of the tables containing the string `FROM` and did not find any reference demonstrating one unit transferring or loaning a TAMCN to another unit.

4.2 Layer Analysis

In this section, we analyze each layer of the GCSS-MC data. For the Current Parts on Order layer, we visualize the network using Gephi and present a detailed analysis of the centrality measures and community structure. Additionally, we explore the structure of the core and determine the assortativity of the layer. We also compare the layer to each of the synthetic models outlined in Section 2.3.8. For each subsequent layer, we provide a summary of the results and any prominent observations.

4.2.1 Current Parts on Order

We retrieved the data for this layer from the `LCMIDBA.CURRENT_OPEN_PARTS_ON_ORDER` table. We used the search criteria `['TAM', 'PARTS', 'NSN']` to find this table. This table is important because it supports the procurement aspect of our analysis. The table provides information, as the table title states, regarding parts that are currently on order. We chose to extract two columns for this layer, the column containing the TAMCNs and the column containing the NSNs. Every row of the table contains one TAMCN and NSN pair. In this case, the NSN listed in a row with the TAMCN represents a part that has been purchased.

Figure 4.1 visualizes the Current Parts on Order layer. The size of each node corresponds to the degree centrality of the node. The greater the degree centrality, the larger the node. The color of each node represents the modularity class Gephi calculates based on the Louvain community detection method. We apply the Force Atlas 2 layout to highlight the structure. An edge between two nodes signifies the relationship:

$$\exists \text{NSN such that NSN} \in \text{TAMCN X} \cap \text{TAMCN Y}. \tag{4.1}$$

Equation 4.1 states that an edge exists between two TAMCNs X and Y if there exists an NSN that both X and Y has on order. Figure 4.1 represents 512 of the 617 TAMCNs. We

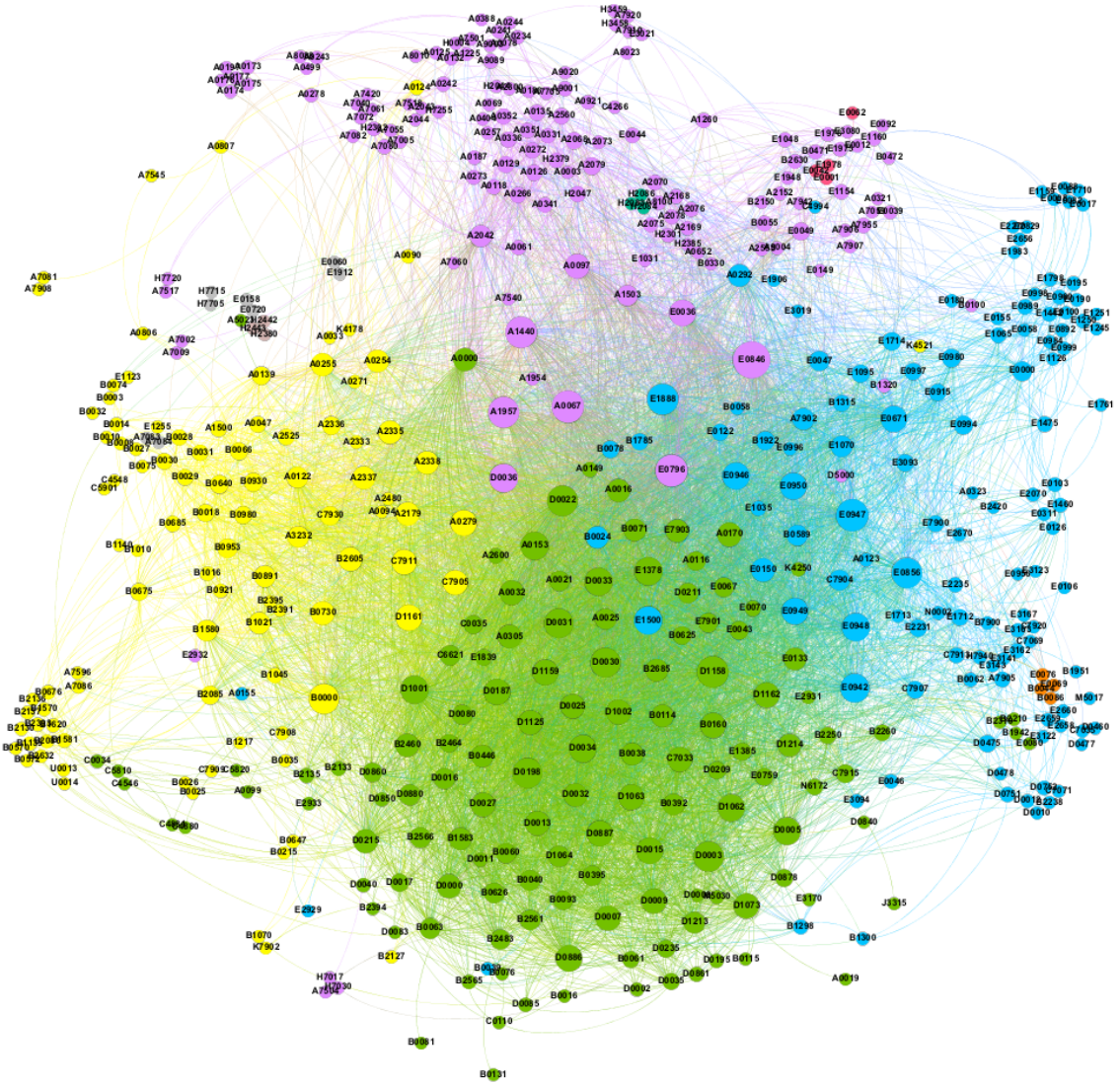
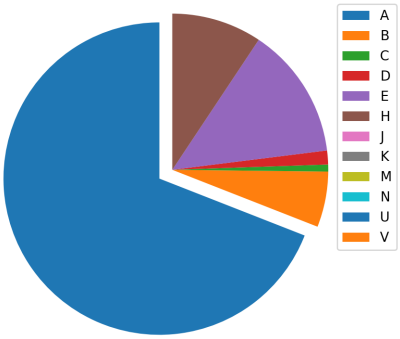
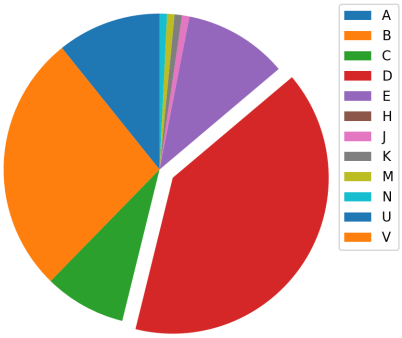


Figure 4.1. Current Parts on Order Layer

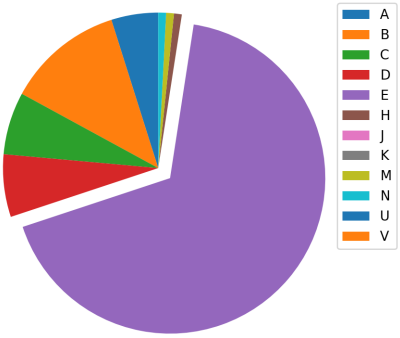
take every node into consideration for centrality calculations, modeling, and assortativity. However, we apply a $k = 1$-core filter for the visualization. This filters out any nodes that have a degree centrality equal to 0. This signifies that the nodes in the set of zero degree nodes do not possess an NSN on order that any other TAMCN possesses. Thus, the nodes that do not appear in Figure 4.1 possess unique NSNs while those that do appear have at least one NSN in common with a member node of the layer.



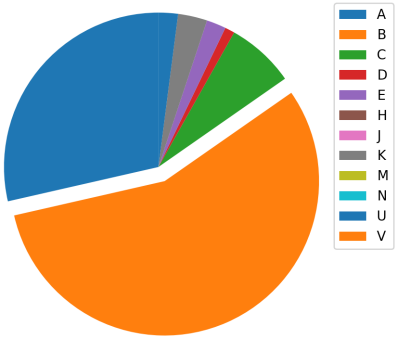
(a) Community 34 (22.53% of all nodes)



(b) Community 32 (21.07% of all nodes)



(c) Community 90 (19.94% of all nodes)



(d) Community 47 (15.88% of all nodes)

Figure 4.2. Community Distribution among the 4 Most Populous Communities

We discussed community detection in Section 2.3.9. We determine the modularity index and assign each node a modularity class before applying the community detection filter. The Louvain method assigned 117 different communities in this graph. However, many of those communities contain only one node. More than three quarters of all the TAMCNs in the graph pertain to four communities. The rest of the communities contain a maximum of 0.65% of nodes in the graph per community. Figure 4.2 represents the composition of the four most populous communities in this layer. We observe that although multiple TAMCN classes comprise each of the majority communities, a dominant TAMCN class does exist in each of the communities. In three of the four communities, one specific TAMCN seems to

dominate the community structure. In community 32, as represented in Figure 4.2 (b), even though *D* TAMCNs clearly represent the majority of the community, *B* TAMCNs comprise a larger portion of the community than other minority TAMCNs in the other communities.

Table 4.2. Top 15 Current Parts on Order Layer Centrality Measures

Degree	**Eigenvector**	**Betweenness**	**Closeness**
E0846 : 0.3685065	E0846 : 0.1269398	E0846 : 0.0558122	E0846 : 0.4967493
A0067 : 0.2987013	D1158 : 0.1223459	B0000 : 0.0419745	E0947 : 0.4697270
E0947 : 0.2987013	A0067 : 0.1220722	A1440 : 0.0312819	A0067 : 0.4622370
E0796 : 0.2922078	D0022 : 0.1202579	E0856 : 0.0264496	E0796 : 0.4590997
D1158 : 0.2873376	E0947 : 0.1202265	E0671 : 0.0235054	D1158 : 0.4585810
A1440 : 0.2824675	D0030 : 0.1185019	E0947 : 0.0196386	D0022 : 0.4554929
A1957 : 0.2775974	D0031 : 0.1174368	E0796 : 0.0183096	A1957 : 0.4544727
E1888 : 0.2775974	D0198 : 0.1167520	E1888 : 0.0165894	E1888 : 0.4539643
D0022 : 0.2743506	A1957 : 0.1166229	A0153 : 0.0150683	A1440 : 0.4529510
D0003 : 0.2678571	E1888 : 0.1162657	B2605 : 0.0147768	E0856 : 0.4519423
E0856 : 0.2678571	D0003 : 0.1160856	A0067 : 0.0146407	B0000 : 0.4489427
B0000 : 0.2662338	E0796 : 0.1157297	E0000 : 0.0145070	D0030 : 0.4479516
D0030 : 0.2581169	D1001 : 0.1123923	A2042 : 0.0129406	E0942 : 0.4445171
D0031 : 0.2516234	E0942 : 0.1110852	C7911 : 0.0129355	D0198 : 0.4435455

Table 4.2 lists the top 15 nodes for each centrality measure. For each centrality measure, TAMCN E0846 possesses the highest centrality value. Therefore, according to degree centrality, E0846 connects to more nodes than any other node in the graph. With respect to the layer, it has more parts on order in common with other TAMCNs than any other TAMCN. Since E0846 also possesses the highest eigenvector centrality measure, it is adjacent to important nodes as well. Figure 4.3 represents a plot of eigenvector centralities versus degree centralities. Figure 4.3 suggests a strong linear correlation between degree and eigenvector centrality for this layer. We determine that the higher the eigenvector centrality for any TAMCN, the importance of the nodes adjacent to that TAMCN correspond to their degree. In this example, E0846 is adjacent to more high degree nodes than any other TAMCN and correspondingly, the eigenvector centrality is higher. The strong linear correlation in Figure 4.3 also indicates possible ways that an algorithm assigning importance can be

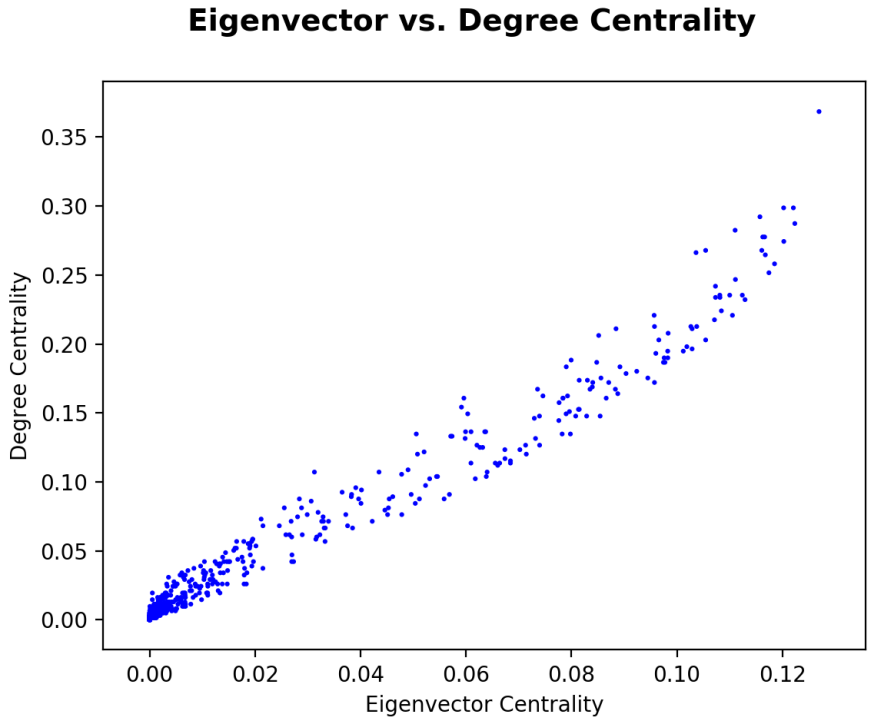


Figure 4.3. Eigenvector versus Degree Centrality for Current Parts on Order

written. Degree centrality is less computationally expensive than eigenvector centrality. Therefore, calculations using degree centrality may be used to estimate importance in lieu of eigenvector centrality.

Interestingly, for each second highest centrality value node, the TAMCN class to which the nodes belong are different. For instance, the second highest degree, eigenvector, betweenness and closeness centrality TAMCNs are respectively A0067, D1158, B0000, and E0947. Notice that they each begin with a different letter. We consider E0947 the closest TAMCN to E0846 of the second highest value centrality TAMCNs because they both begin with the letter *E* and thus pertain to the same TAMCN class. The nomenclature for the TAMCNs more clearly reveals the similarity. Marine Corps Bulletin 3000 [47] lists that E0846 represents an Assault Amphibious Vehicle (AAV) Personnel variant and E0947 represents a Light Armored Vehicle (LAV) Light Assault variant. The E0947 TAMCNs is the most similar to E0846 in the sense that they both possess weapon systems that belong to the platforms and have the capacity to carry troops and traverse water. A0067, D1158, and B000 represent, respectively, a AN/MRC-148 High Frequency Vehicle System, a M1123

High Mobility Multipurpose Wheeled Vehicle (HMMWV), and an air conditioning unit. The top 5 TAMCNs with regard to centrality in this layer happen to be vehicles except for the air conditioning unit. Additionally, two of those TAMCNs are practically the same although they appear different since they belong to separate TAMCN classes. The AN/MRC-148 differs from the M1123 in that the purpose of that TAMCN is the radio it carries although the vehicle which carries it is a M1123 variant. This suggests that vehicles or TAMCNs that possess a vehicle are more important than other equipment.

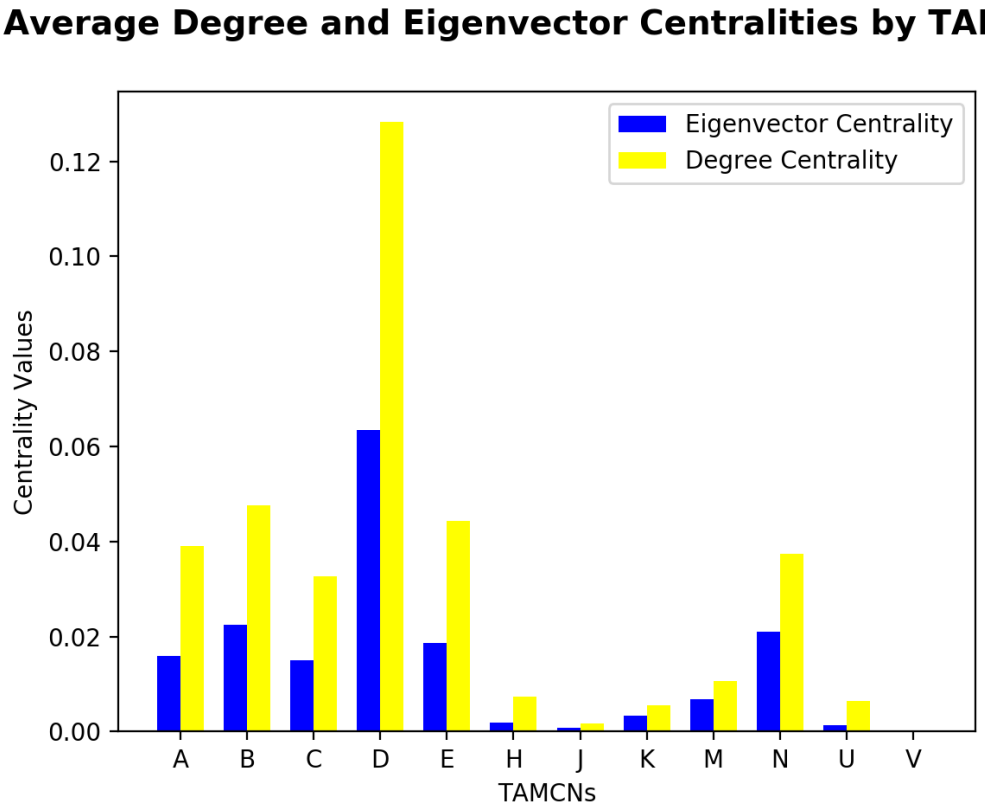
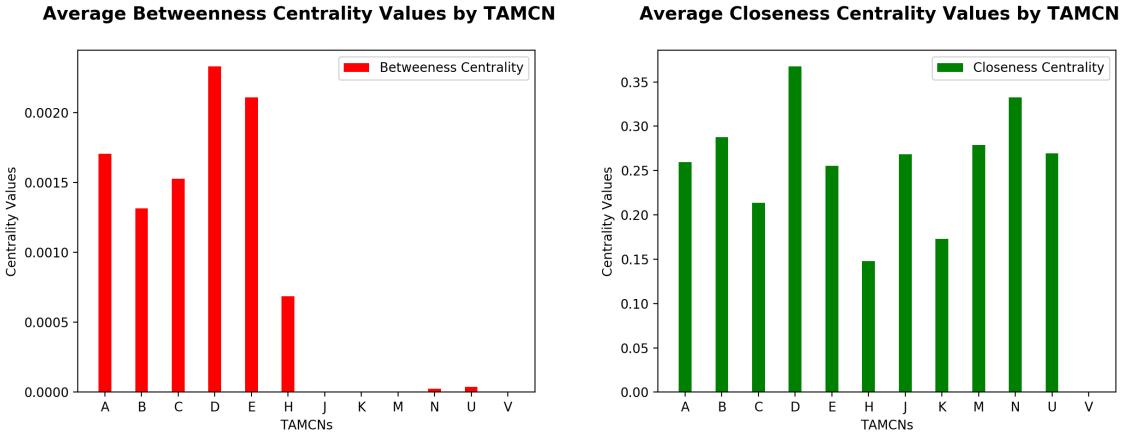


Figure 4.4. Average Degree and Eigenvector Centrality Comparison for Current Parts on Order

Figures 4.4 and 4.5 (a) and (b) represent the average centrality value for each TAMCN class. We plotted the average betweenness and closeness centrality in their own plot in order to more easily identify trends. Although the centrality values for E0846 are the highest for all of the centralities measured, the *E* TAMCN does not represent the highest average centrality for any centrality measure. Instead, the *D* TAMCN class possesses the highest average centrality for each centrality measured. Table 4.2 corroborates this observation for



(a) Average betweenness centrality

(b) Average closeness centrality

Figure 4.5. Average Betweenness and Closeness Centralities for Current Parts on Order

the top 15 TAMCNs. Generally, *E* and *D* TAMCNs populate the majority of the highest centrality values except for betweenness centrality. For betweenness centrality, not a single *D* TAMCN instance occurs in the list.

Both betweenness and closeness centrality rely on geodesics to establish importance. In this layer, a more important node with respect to betweenness centrality appears on a greater number of shortest paths than a less important node. As E0846 represents the node with the highest betweenness centrality, E0846 appears on the most amount of shortest paths. Since an edge represents an instance where two TAMCNs share a common part on order, a shortest path from one node to another represents a node or a series of nodes that connect the two endpoints. Consider Figures 4.6 and 4.7. Figure 4.6 highlights all of the nodes TAMCN C7909 connects to and Figure 4.7 similarly highlights TAMCN C7908 and its neighbors. Notice in Figure 4.6 that C7909's only neighbor is C7908. Figure 4.7 illustrates that C7908 is on the shortest path between nodes such as B0730, C7905, and B0891. This means that since C7909 does not connect to B0891 and B0730, C7909 relates to those TAMCNs through C7908.

Closeness centrality also relies on geodesics in its calculations except that unlike betweenness centrality, it measures the shortest distance from a node to all other nodes. In this layer, intuition suggests that since edges represent a common part on order between two TAMCNs, importance favors those TAMCNs that have many neighbors. A high degree

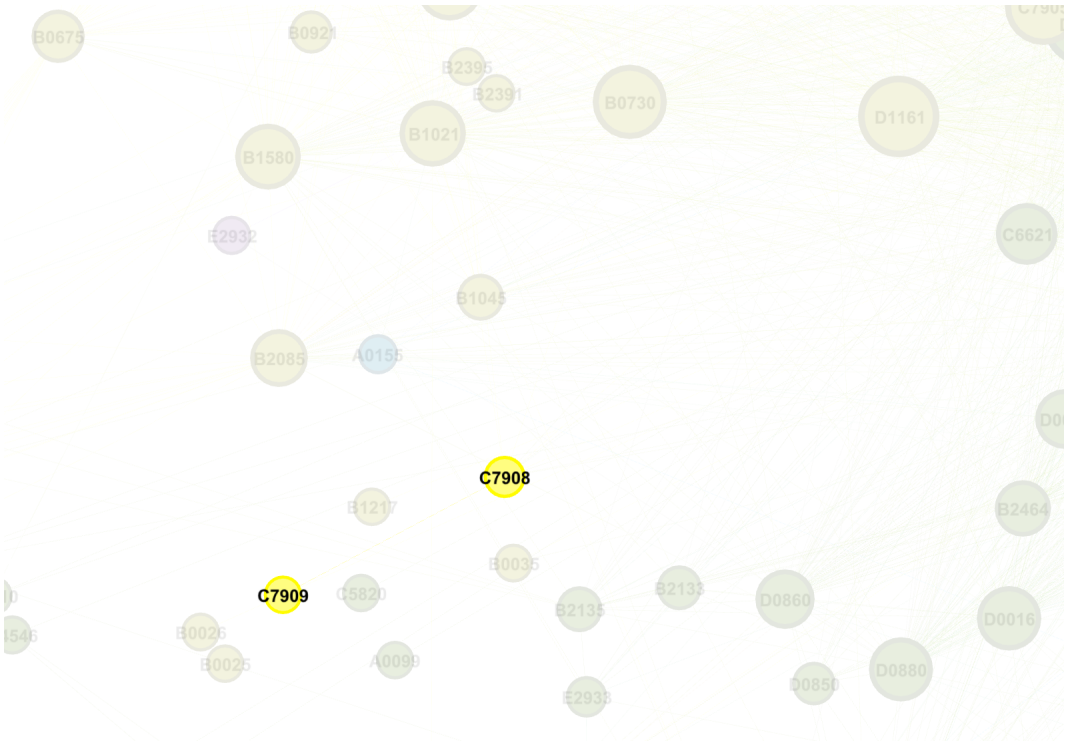


Figure 4.6. TAMCN C7909 Connections

centrality means that a TAMCN shares different parts with many other TAMCNs. However, Figure 4.8 plots betweenness centrality against degree centrality and illustrates that a linear correlation does not exist between the two. This lack of correlation suggests that closeness centrality favors TAMCNs that have high numbers of non-adjacent neighbors. Non-adjacent neighbors in this case are neighbors in the neighborhood of some TAMCN that do not share an edge.

Figure 4.9 illustrates non-adjacent neighbors. Figure 4.9 (a) highlights neighbors in the neighborhood of E3162 while Figure 4.9 (b) highlights neighbors in the neighborhood of E3141. Notice that although both E3141 and E3163 are in the neighborhood of E3162, E3141 and E3163 are not themselves neighbors. The concept of non-adjacent neighbors relates to closeness because the TAMCN with non-adjacent neighbors contains paths directly to each of those neighbors. We suspect the closeness centrality would increase if the non-adjacent neighbors are also pendants. A pendant is a node of degree equal to 1.

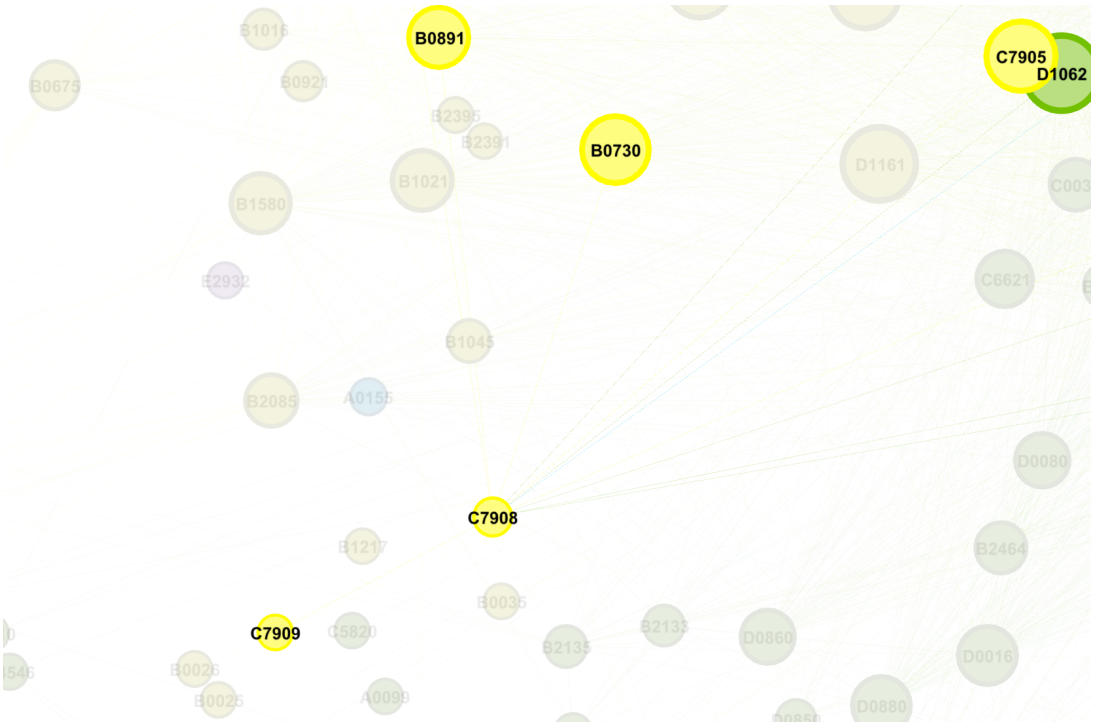


Figure 4.7. TAMCN C7908 Connections

Figures 4.4 and 4.5 (a) and (b) provide a graphical depiction of the average centrality measure for each centrality and for each TAMCN. Although useful in generalizing the data, it may obscure trends. Figures 4.10 and 4.11 plot the cumulative distribution function (CDF) for each of the centrality measures and for each TAMCN. The CDF provides a more granular display of the centrality measures for each TAMCN. For each centrality measure, we divide the TAMCN classes in half. We group the *A*, *B*, *C*, *D*, *E*, and *N* TAMCNs together according to their similar average centrality measures. We group the rest of the TAMCNs together although they also share lower average centrality measures in general. Figures 4.10 (a)-(d) all illustrate that *A*, *C*, and *D* TAMCNs possess the highest centrality measures. *D* TAMCNs differ from other TAMCNs in their degree and eigenvector centrality distributions. While the other TAMCNs enclose approximately 60% of their nodes in degree centrality less than .05 and eigenvector centrality less than .03, D TAMCNs almost linearly distribute in degree and eigenvector centrality. Figure 4.10 (d) differs from the others in that the majority of TAMCNs possess high closeness centrality nodes. This suggests that in general, TAMCNs are close to each other, meaning they have short distances to other nodes. Using Gephi we find that the average path length for this layer is 2.575 albeit the diameter of this layer

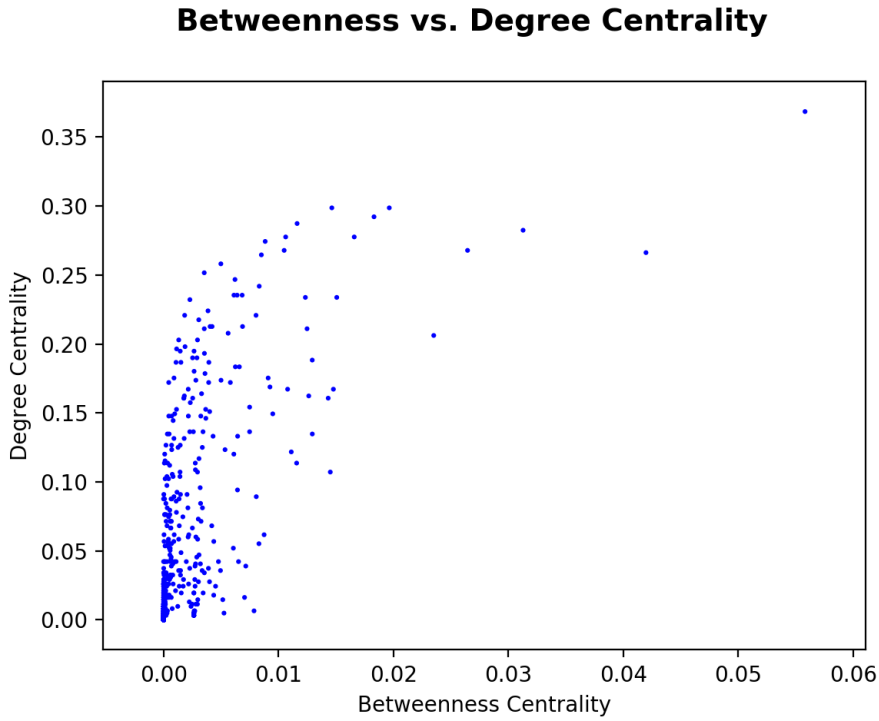
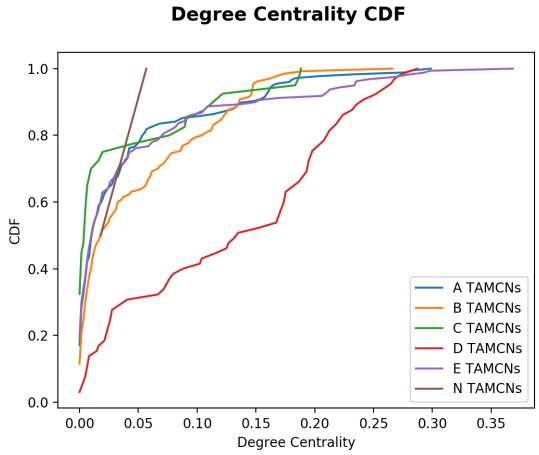


Figure 4.8. Current Parts on Order Betweenness versus Degree Centrality

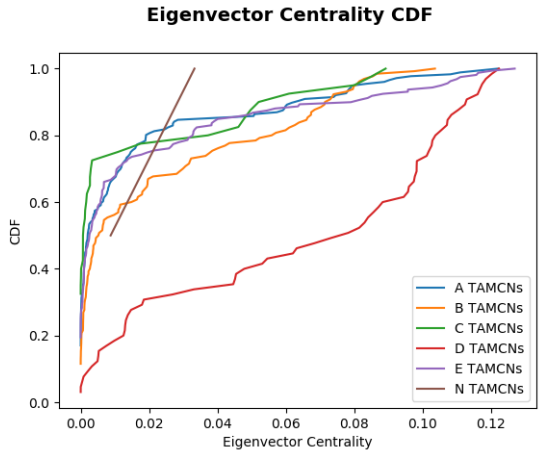
(a) E3162 TAMCN neighbors

(b) E3141 TAMCN neighbors

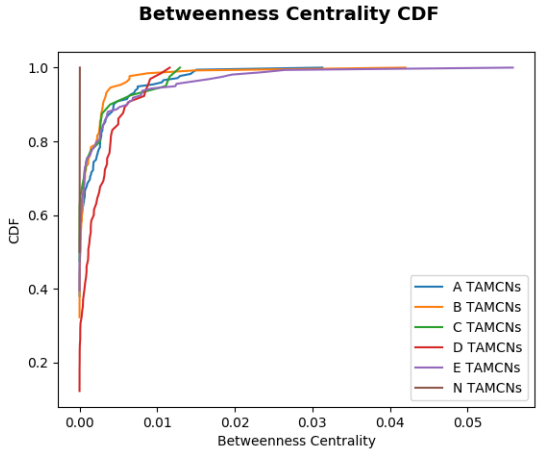
Figure 4.9. Sample Connections from Current Parts on Order Graph Illustrating Non-Adjacent Neighbors



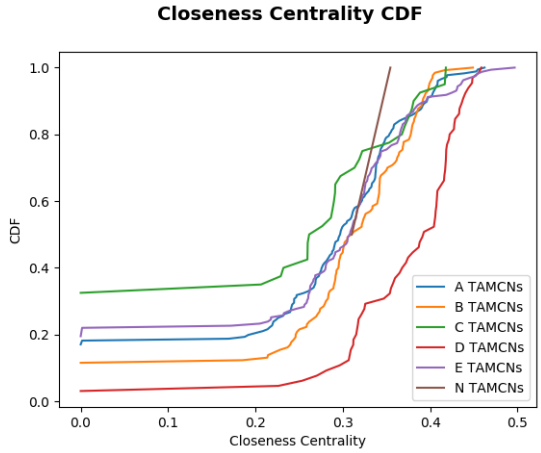
(a) Degree centrality CDF



(b) Eigenvector centrality CDF



(c) Betweenness centrality CDF

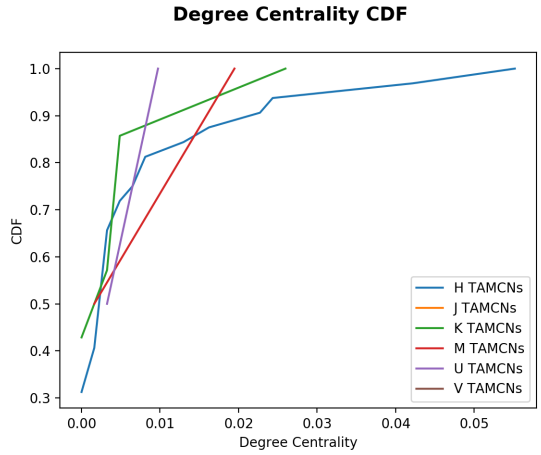


(d) Closeness centrality CDF

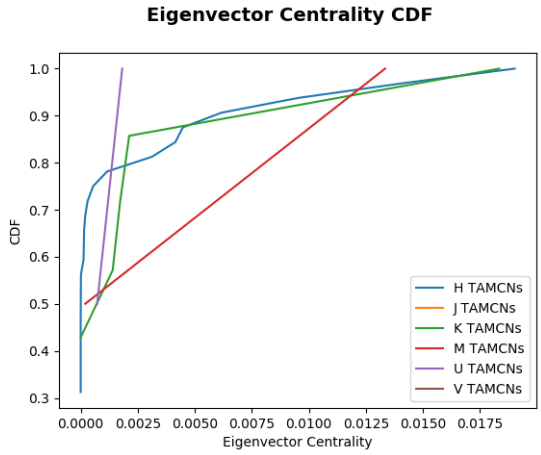
Figure 4.10. A, B, C, D, E, and N TAMCN Centrality Measure CDF Graphs

is 7. Figures 4.11(a)-(d) illustrate that the majority of J, K, M, U, and V TAMCNs have centrality measures close to 0. H TAMCNs demonstrate a less exponential centrality CDF but are still generally less than the TAMCNs found in Figure 4.10. In general, we conclude that J, K, M, U, and V TAMCNs are less important.

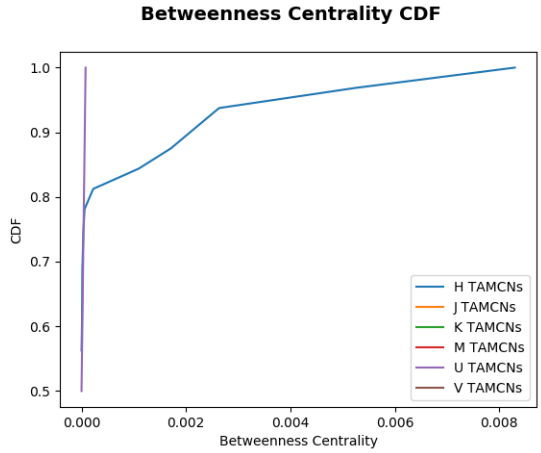
Figure 4.12 visualizes the core for this layer. The core contains 64 nodes which represent 10.37% of the total nodes in this layer and 2016 edges which represent 21.72% of the total edges in the layer. We find that the k-core value for this layer equals $k = 63$. Each node represented in the core connects to at least 63 other nodes. We also find that the core is a clique, which signifies that every node in the core is connected to at least every other



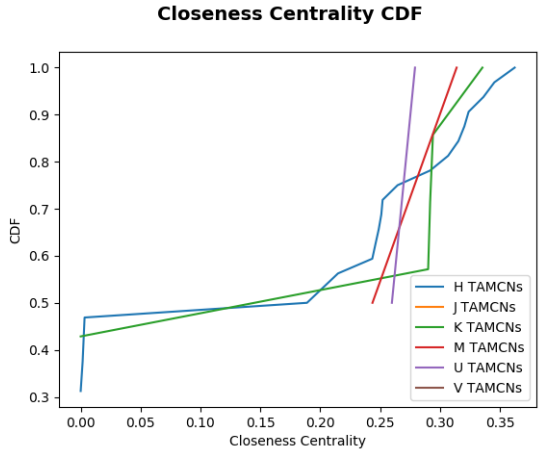
(a) Degree centrality CDF



(b) Eigenvector centrality CDF



(c) Betweenness centrality CDF



(d) Closeness centrality CDF

Figure 4.11. H, J, K, M, U, and V TAMCN Centrality Measure CDF Graphs

node in the core. This implies that changes in one of the core nodes can potentially affect every other node in the core if it pertains to a part they all have in common. Since an edge represents a common part on order, we infer that if a node in the core orders a new part, other nodes in the core are likely to need the part as well. We filtered out nodes from the graph found in Figure 4.1 until only the core nodes remained. Therefore, the size of the nodes and their coloring still correspond to the original degree centrality and community assignment initially calculated.

We observe that community 32 colored in green possesses the largest amount of core nodes. Figure 4.2 depicts the composition of community 32 for the entire graph. For $k = 0$, where

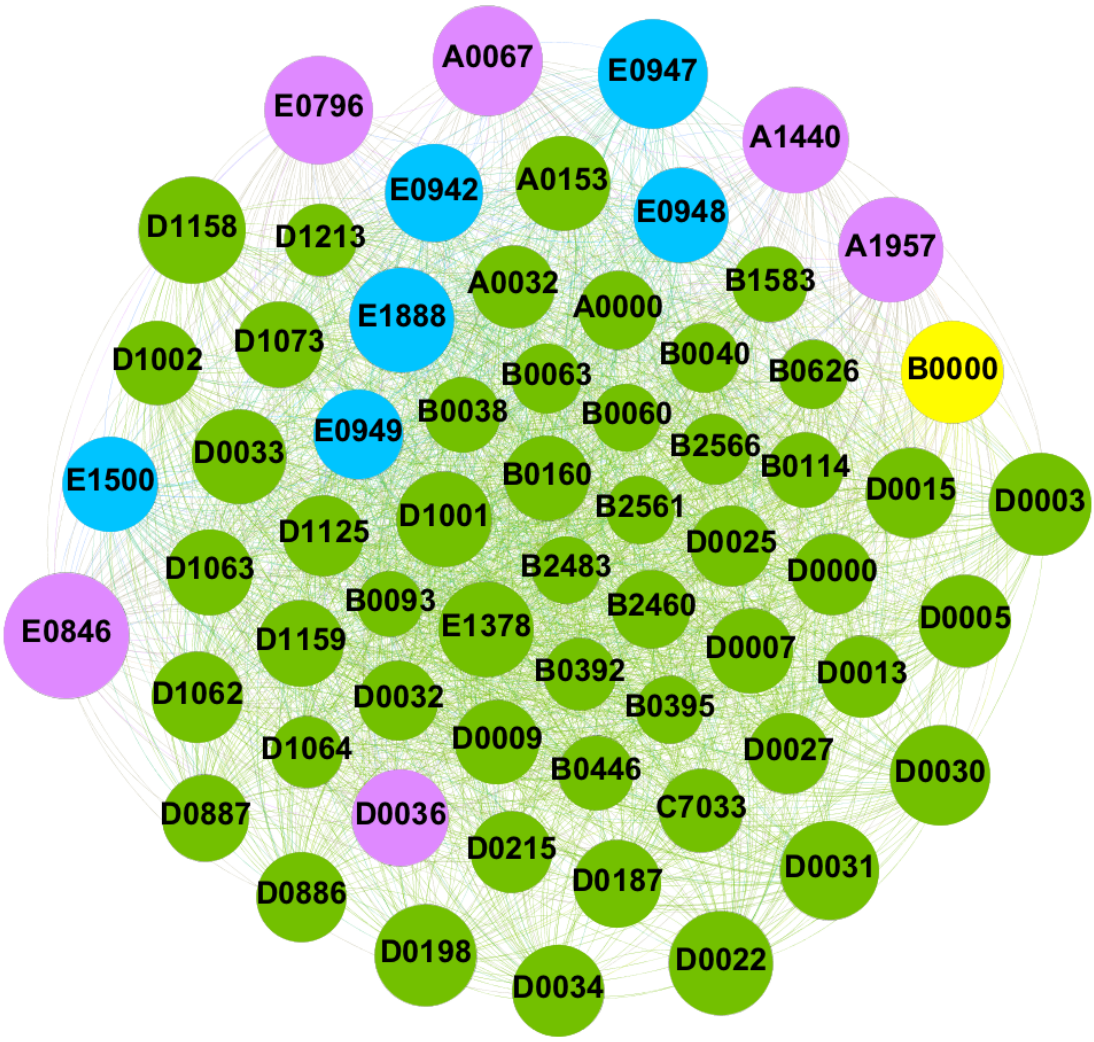


Figure 4.12. Current Parts on Order Core

every node and edge in the graph is present, D TAMCNs comprise the majority of the community. Figure 4.13 (b) depicts the composition of community 32 within the core. Although D TAMCNs comprise the majority of community 32 when $k = 0$ and when $k = 63$, D TAMCNs comprise a larger portion of the community in the core. This implies that the higher the degree or eigenvector centrality for D TAMCNs, the more important they become in the community.

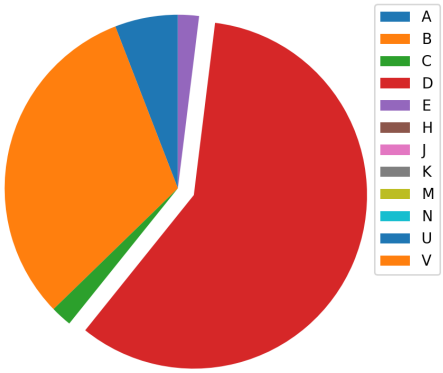
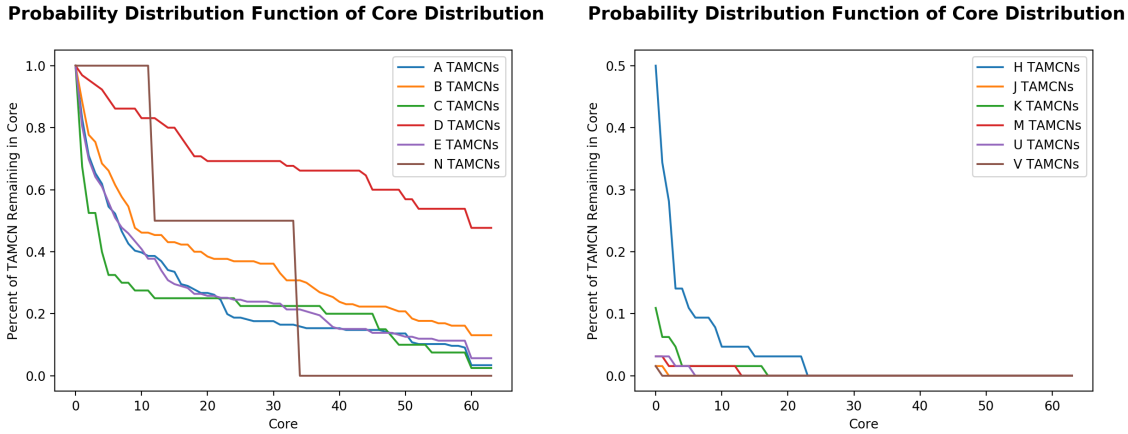


Figure 4.13. Community 32 Core Composition



(a) PDF of core distribution

(b) PDF of core distribution

Figure 4.14. PDF of the Core Distribution for All TAMCNs

Although Figure 4.12 provides an accurate visualization of the core, it lacks the granularity demonstrating the evolution of the graph according to k in the k-core. Figure 4.14 illustrates the probability distribution function (PDF) for each TAMCN class as k increases. We organize TAMCN classes according to their average degree. Higher degree TAMCNs including A, B, C, D, E, and N are represented in Figure 4.14 (a) while the rest are represented in Figure 4.14 (b). Each curve represents the percentage of the TAMCN class present for values of k where $0 \leq k \leq 63$ and 0 and 63 represent the entire graph and the core respectively. We observe from Figure 4.14 that generally the percentage of TAMCNs

remaining in the k-core fall precipitously as k increases for each TAMCN except for D TAMCNs. This seems to explain why the portion of D TAMCNs in community 32, which is the community representing the most nodes in the core, is greater in the core than for the overall graph. Since approximately 50% of D TAMCNs remain in the core for $k = 63$, we know that at least half of all D TAMCNs contain degree equal to 63 or greater.

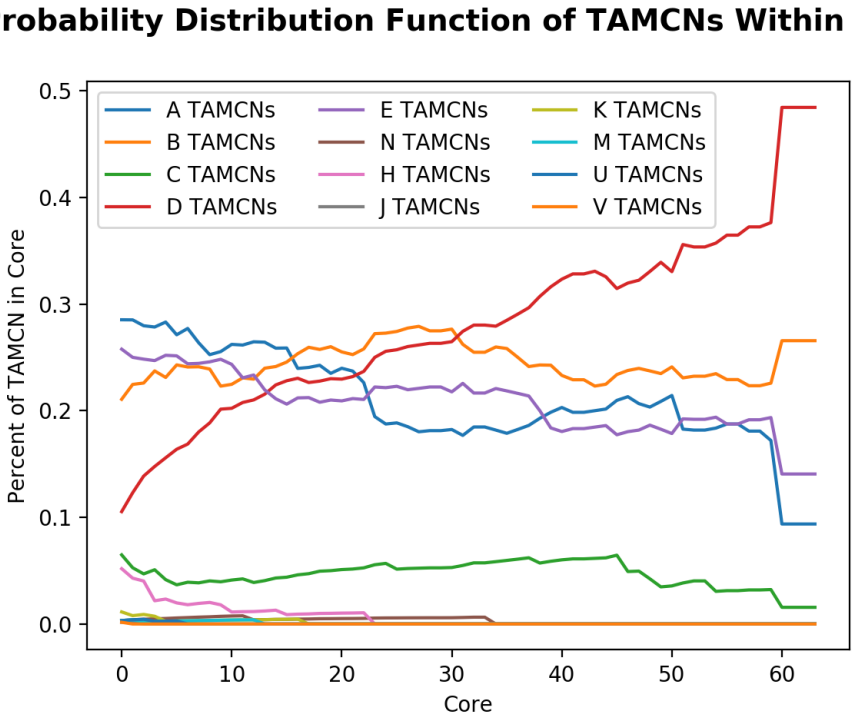
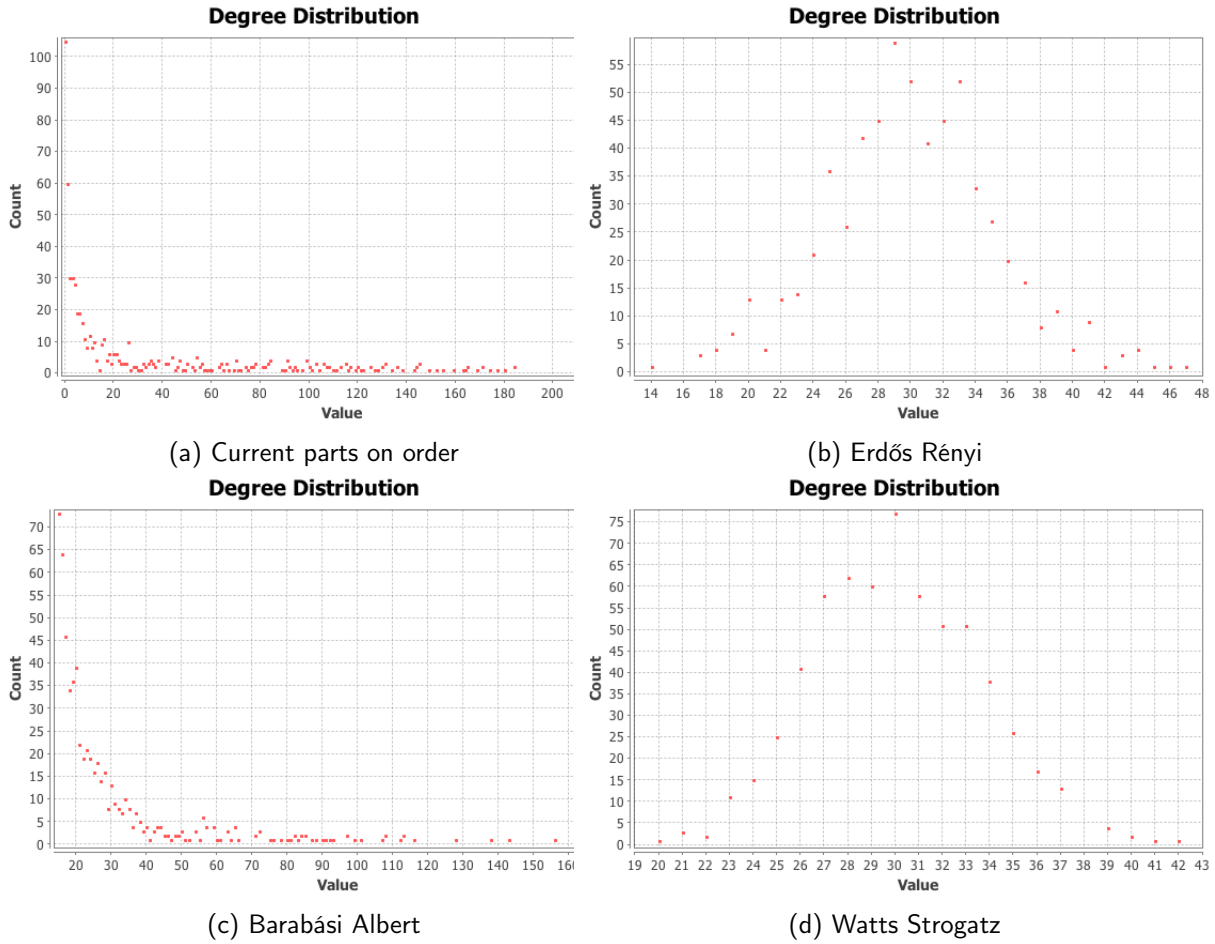


Figure 4.15. PDF of TAMCNs within Core

While Figure 4.14 illustrates the PDF of each TAMCN class as k increases, Figure 4.15 depicts the PDF for the k-core as k increases. For each value of k, Figure 4.15 represents the percentage of each TAMCN present. We observe that initially when $k = 0$, D TAMCNs comprise approximately only 10% of all nodes. The D TAMCN class is the only TAMCN class that consistently increases in percentage as k increases. As k approaches 63, D TAMCNs comprise nearly 50% of all nodes in the core. We also observe that when k reaches $k = 60$, the core no longer changes significantly. This signifies that all the nodes in the core behave the same when $60 \leq k \leq 63$.

Although an E TAMCN represents the largest centrality measure from each centrality measured, D TAMCNs are the most important TAMCN in this layer. D TAMCNs are the

most important because although they do not have the largest centrality measures, Figure 4.10 suggests a larger majority of *D* TAMCNs possess high centrality measures compared to the other TAMCN classes. Additionally, Figures 4.2 and 4.13 both show *D* TAMCNs as majority members of a significant community with regards to size of the community. Also, since *D* TAMCNs comprise the largest portion of the core, we infer that changes to those TAMCNs are more likely to affect other TAMCNs throughout the network.



(a) Current parts on order

(b) Erdős Rényi

(c) Barabási Albert

(d) Watts Strogatz

Figure 4.16. Current Parts on Order and Synthetic Model Degree Distributions

Figure 4.16 depicts the degree distribution for this layer and for each of the synthetic models discussed in Section 2.3.8. This illustrates the amount of nodes for each degree in the network. We discussed the methodology needed to create the synthetic models in Section 3.3.2. For each synthetic model, we set the amount of nodes to 617, which

represents the total nodes in the graph. Figure 4.16 (b) depicts the degree distribution for an Erdős Rényi random graph. We determine the parameter for the random graph by inputting the size and order of the actual network into Equation 2.15. We find that the probability p needed to create a random network with approximately 9282 edges is:

$$\begin{aligned} p &= \frac{e(g)}{\binom{n(g)}{2}} \\ &= \frac{9282}{\binom{617}{2}} = \frac{9282}{190036} \\ &\approx 0.048843377. \end{aligned} \tag{4.2}$$

Figure 4.16 (c) depicts the degree distribution for a Barabási Albert scale-free network. We know that the parameter for scale-free networks determines how many nodes an incoming node will attempt to connect to. We find that in order to create a graph with approximately the same number of edges as the actual network, a node will attempt to connect to 15 other nodes since $617 * 15 = 9255$ where 617 is the number of nodes in the graph and 15 is the parameter. Figure 4.16 (d) depicts the degree distribution for a Watts Strogatz small-world network. We use the average degree of the actual network as an integer value, 30 specifically, as a parameter in the small-world network which determines how many of its nearest neighbors it connects to. As the probability parameter for small-world networks approaches 0, the graph becomes more modular. We know the modularity index for the actual network and adjusted the probability parameter until finding a value $p = 0.65$ that produces a graph with a similar modularity index.

Table 4.3 lists the metrics we use to compare the synthetic models to the Current Parts on Order layer. Together with Figure 4.16, we conclude this layer models a Barabási Albert scale-free network. We find that the Erdős Rényi random graph is the least similar model to the actual network. Not only does the degree distribution follow a Poisson distribution instead of a power law, the clustering coefficient is low. Although the Watts Strogatz small-world model accurately captured the modularity of the actual network, it severely underestimated the clustering coefficient and also differs in its degree distribution. Even though the Barabási Albert model underestimates both the modularity and the clustering, it closely models the degree distribution. We conclude the degree distribution is the most important metric to compare because it characterizes the network and affects the rest of

Table 4.3. Current Parts on Order and Synthetic Model Comparison

Name	**Actual Network**	**Erdős Rényi**	**Barabási Albert**	**Watts Strogatz**
Average Degree	30.088	30.016	29.271	30
Modularity	0.249	0.154	0.149	0.251
Clustering Coefficient	0.649	0.048	0.111	0.07
Average Path Length	2.575	2.171	2.184	2.19
k-core	63	22	15	23
% of nodes in Core	10.37	91.09	100	98.54

the metrics. All of the models failed to accurately model the core of the network. The core of the network helped identify important communities and subsequently highlighted TAMCNs that appear important due to their presence as k increases.

Analyzing the synthetic networks does not reveal importance for any particular nodes since the distribution is randomly assigned. However, the synthetic networks model the structure of the actual network. Despite the variance in the metrics found, we obtain bounds on the behavior of this layer. With the random graph as a lower bound, we find that the Current Parts on Order layer is not random. This implies that the information stored in the GCSS-MC data contains structure and the relationships between the data are not random. We infer that this layer is a variation of a scale-free small-world network due to the similar modularity to a small-world network and the degree distribution of a scale-free network.

We introduced assortativity in Section 2.3.9. Newman [38] describes degree assortativity as the mixing of high degree nodes with low degree nodes. He explains that networks with high degree assortativity, in which a value of $r = 1$ represents a perfectly assortative network while $r = -1$ conversely represents a perfectly disassortative network, mix high degree nodes with other high degree nodes and similarly for low degree nodes. This relationship manifests itself in a densely populated core with loosely connected peripheries for highly assortative networks and clustered networks with a small core for disassortative networks. He provides a table listing assortativity coefficients for social, technological, and biological

networks. Social networks tend to have assortativity coefficients greater than 0, signifying higher assortativity while technological networks such as power grids, the internet, and software dependencies have negative values and are less assortative. Biological networks such as protein interactions, metabolic networks, and neural networks have the lowest negative values in the table and represent disassortative networks.

The degree assortativity coefficient for the Current Parts on Order layer is $r = 0.02750$. We find that this layer is assortative not only due to its assortativity coefficient, but also its core structure. As previously stated, the core is a clique with every node in the core possessing a degree of at least 63. We observe that this layer counter-intuitively resembles a social network instead of technological and biological networks. Considering the GCSS-MC database contains information regarding inanimate objects, we assumed the data would show a stronger correlation to technological networks. Additionally, due to the assortativity coefficient, we infer that high degree nodes mix with other high degree nodes. With regards to the database, we imply that TAMCNs with many parts on order mix together.

4.2.2 Deadline Parts on Order

We retrieved the data for this layer from the `LCMIDBA.DEALINE_PARTS_ON_ORDER` table. We located this table with the `['TAM', 'ORDER', 'NSN']` search criteria. We consider this table unique not only its application to the procurement aspect of our analysis, but also in its relationship to maintenance. A TAMCN is considered deadlined when it contains a NSN that needs repair which leaves the entire TAMCN inoperable unless repaired or replaced. Therefore, if a TAMCN appears as a node in this layer, we also know that it is in the maintenance cycle. Similar to the Current Parts on Order Layer found in Section 4.2.1, we extracted the columns containing TAMCNs and NSNs which produced a CSV file with rows containing TAMCN, NSN pairs. We assume that entries in this layer are distinct from entries in the Current Parts on Order layer.

An edge in this layer follows the same format found in Equation 4.1. Although the edge represents the same relationship between nodes, the layer represents a different attribute we analyze, namely TAMCNs that are deadlined which have parts on order. The order and size of this layer are 265 and 2358, respectively, which is smaller than the size and order of the Current Parts on Order layer. Table 4.4 displays the metrics associated with

Table 4.4. Deadline Parts on Order and Synthetic Network Metrics

Name	Actual Network	Erdős Rényi	Barabási Albert	Watts Strogatz
Average Degree	18.48	19.193	17.411	18
Modularity	0.195	0.197	0.187	0.197
Clustering Coefficient	0.705	0.072	0.141	0.065
Average Path Length	2.45	2.172	2.218	2.219
k-core	50	13	9	13
% of nodes in Core	18.55	92.73	99.64	96.36
Assortativity	0.04324	-	-	-
Most Important TAMCN	E0846	-	-	-

this layer. Similar to the Current Parts on Order layer, TAMCN E0846 represents the most important TAMCN in the layer. Unlike the Current Parts on Order layer, E0846 does not occupy the highest centrality node for every centrality measured. E0846 represents the highest centrality measure for each centrality except for eigenvector centrality where TAMCN D003 represents the greatest centrality instead.

The core contains a higher percentage of nodes than the core of the Current Parts on Order Layer. This implies there exists a greater number of high degree nodes. The larger assortativity coefficient asserts that high degree nodes tend to connect with other high degree nodes. The larger clustering coefficient and lower modularity imply that although the nodes cluster, many edges exist in between those clusters. This layer also follows a power law degree distribution. Similar to the Current Parts on Order layer, we also categorize this layer as a Barabási Albert small-world scale-free network.

4.2.3 Open Service Request

We retrieved the data for this layer from the `WSTIAC.WST_EROS` table. This table contains information regarding open service requests. We introduced the concept of a service request as it pertains to GCSS-MC in Chapter 1. This layer supports the procurement and

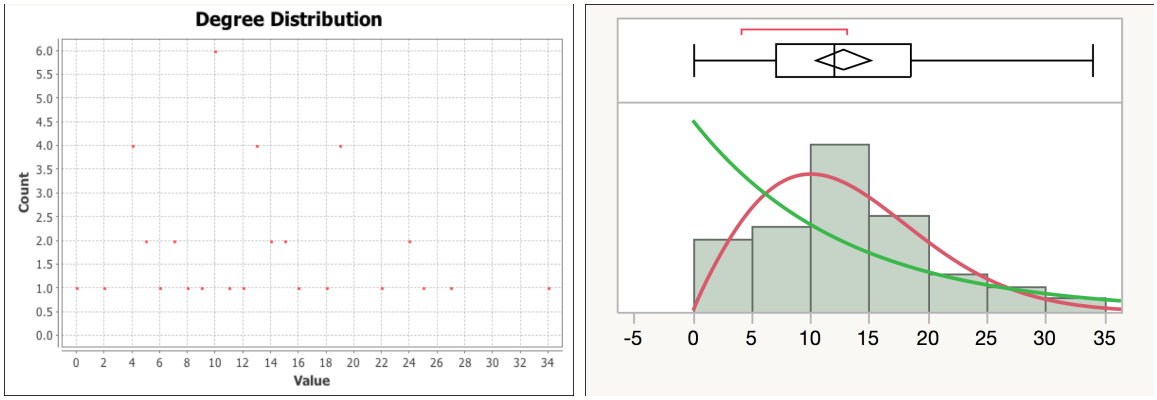
maintenance aspect of our analysis. An open service request attached to a TAMCN represents an instance where a service is required for that TAMCN. A service includes actions such as purchasing parts, requesting maintenance, and recording preventative maintenance. We exported columns containing TAMCNs and NSNs for this layer as well. The rows in the resulting CSV file represent TAMCN, NSN pairs that signify an open service request regarding a NSN for its corresponding TAMCN.

Table 4.5. Open Service Requests and Synthetic Network Metrics

Name	**Actual Network**	**Erdős Rényi**	**Barabási Albert**	**Watts Strogatz**
Average Degree	12.78	12.976	12.878	12
Modularity	0.197	0.167	0.145	.178
Clustering Coefficient	0.679	0.31	0.415	0.293
Average Path Length	1.735	1.679	1.684	1.71
k-core	10	10	8	9
% of nodes in Core	46.34	80.49	100	90.24
Assortativity	-0.1887	-	-	-
Most Important TAMCN	D1158	-	-	-

Edges in this layer also follow the format found in Equation 4.1. The attribute presented in this layer, namely open service requests, is the difference between this layer and previous layers discussed. The order and size of this layer are 41 and 262, respectively. Beginning with the order and size of the layer, we observe that this layer differs from the previous layers. This anomaly possibly arises from the data sampled. The table whose data we analyzed may be a subset of another table or may be incomplete. Nonetheless, for this thesis we assume that the table is independent of any other tables.

The most important node according to the centrality measures for this layer differs from the other layers. In this layer, TAMCN D1158 possesses the highest centrality measure for each of the different centralities measured. While E0846 dominated the highest centrality values for the other layers, it only appears in the top 15 TAMCNs for betweenness centrality.



(a) Scatter plot degree distribution (b) Bin degree distribution with lines of fit

Figure 4.17. Open Service Request Layer Scatter-Plot and Binned Degree Distributions

Another key difference lies in the degree distribution. Figures 4.17 (a) and (b) depict the degree distribution for this layer. Figure 4.17 (a) displays the degree distribution in a scatter plot. While the degree distributions from the other layers obviously depicted a power law abiding exponential distribution, the degree distribution from this layer does not follow a neat distribution. Therefore, we binned the degrees and re-plotted the graph as depicted in Figure 4.17 (b). The x-axis represents the degree while the y-axis represents the amount of nodes in each bin. Afterward, we fitted the graph to both an exponential distribution in green and a Weibull variate distribution in red. The Weibull variate distribution clearly fits the data more closely than an exponential distribution since a greater portion of the data is contained within its distribution. Besides the distribution, this network follows a Watts Strogatz small-world model. Since it does not follow a power law, we infer that this layer is not scale-free. Additionally, the negative assortativity coefficient implies that high degree nodes are unlikely to connect with other high degree nodes and likewise for low degree nodes. It also suggests that this layer compares more closely with biological networks than social networks. Curiously, this layer also defies intuition in that as with the previous layers analyzed, we expected this layer to behave similarly to technological networks.

4.3 Overall Importance

While in Section 4.2 we analyzed each layer independently, in this section, we present a ranking system that determines the overall most important TAMCN in the GCSS-MC

database. First of all, we identify trends among the three layers. We observe that the denser the layer, the more it follows a scale-free power law degree distribution. The denser layers also possess higher positive assortativity coefficients. However, this does not imply that the assortativity increases as the amount of nodes and edges increase since the Deadline Parts on Order assortativity coefficient is higher than the Current Parts on Oder layer. All the layers possess high clustering coefficients. This signifies a large number of connections within communities. According to our community analysis, although the community breakdown does not separate communities entirely by TAMCN class, each community is made up of a majority of a distinct TAMCN class. The high clustering coefficients for each of the layers implies TAMCNs have many connections among their own TAMCN class. Additionally, we notice that D TAMCNs are important in each of the layers.

We create a ranking system that ranks TAMCNs based on their importance in each of the layers. A more important TAMCN will have a higher centrality measure. In order to find the most important overall TAMCN, first we create a Python dictionary data structure in which the key is the TAMCN and the value is a list of each centrality measure associated with the TAMCN. Consider a TAMCN that appears in each layer. The length of the list associated with the TAMCN is: $4 \cdot \phi(\text{TAMCN})$, where $\phi(\text{TAMCN})$ is the number of layers the TAMCN appears in and 4 represents the 4 total centrality measures. We find the greatest value of ϕ which is $max(\phi)$ in the dictionary and set a starting value of $\gamma = 4 \cdot max(\phi)$. We find a ranking value by calculating the summation of each centrality value associated with a TAMCN and subtracting it by γ:

$$rank\_value(\text{TAMCN}) = \gamma - \left(\sum_{i=1}^{m} C_D(i) + \sum_{i=1}^{m} C_E(i) + \sum_{i=1}^{m} C_B(i) + \sum_{i=1}^{m} C_C(i) \right), \qquad (4.3)$$

where C_D, C_E, C_B, C_C are degree, eigenvector, betweenness, and closeness centralities, respectively, $m = \phi(\text{TAMCN})$, and i is the layer containing the centrality measure. Not all TAMCNs are present in each one of the layers. We incorporate a penalization for TAMCNs that do not appear in a layer through the use of γ. We know that the centrality value is less than 1 for each centrality since the values are normalized. Therefore, Equation 4.3 demonstrates that if a TAMCN has a value $m < max(\phi)$, then the TAMCN's ranking value is penalized by a value of $\gamma - (4 \cdot (max(\phi) - m))$. In other words, consider a TAMCN that appears in each layer. For each layer the TAMCN appears in, it has 4 opportunities to

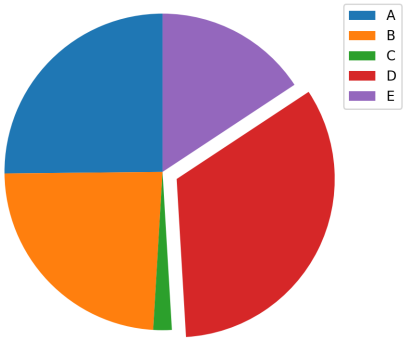
subtract from the starting value γ according to its 4 centrality values. Thus, for every layer a TAMCN does not appear in, it loses 4 opportunities to subtract from γ. We finalize the ranking by sorting the TAMCN in ascending order according to the $rank\_value$ associated with each TAMCN. The TAMCN with the lowest $rank\_value$ is the most important TAMCN in the GCSS-MC database.

Table 4.6. Top 25 TAMCNs and Top 25 Maintenance Drivers

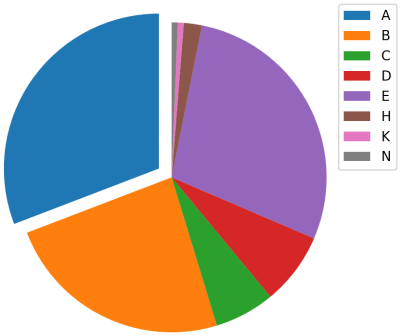
Maint Driver	**Top TAMCNs**	**Maint Driver**	**Top TAMCNs**
1 : E08467K	1 : D1158	14 : A20427G	14 : A0067
2 : E18887M	2 : D0198	15 : D00037K	15 : E0947
3 : E09477M	3 : E0846	16 : A01297G	16 : D0003
4 : A00977G	4 : A1440	17 : A03367G	17 : D1059
5 : D00367K	5 : E0796	18 : E09487B	18 : D0022
6 : E06717M	6 : D1159	19 : D00227K	19 : A1957
7 : A20687G	7 : D1002	20 : E07967K	20 : D0030
8 : B05897B	8 : D1125	21 : A01267G	21 : D0031
9 : D00307K	9 : D1001	22 : D00157K	22 : E0942
10 : D11587K	10 : D1062	23 : A21797G	23 : E1888
11 : E13787K	11 : D0209	24 : D01987K	24 : D0036
12 : B01607B	12 : D1213	25 : E08567K	25 : D0033
13 : A15037G	13 : D1073		

Table 4.6 lists by comparison the top 25 maintenance drivers and the top 25 TAMCNs from our methodology. We obtained the top 25 maintenance drivers from [48]. The maintenance drivers are TAMCNs that are of elevated interest. Although our methodology did not rank the TAMCNs in the exact order as the maintenance drivers, the methodology did capture several of the TAMCNs found in the maintenance drivers. Of the top 25 TAMCNs identified by our methodology, 10 are also contained in the top 25 maintenance drivers. We successfully matched nearly half of the TAMCNs in the top 25 current maintenance drivers using only three layers and a static snapshot of data nearly three years old.

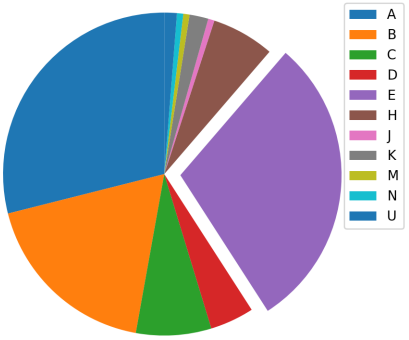
Figure 4.18 depicts the TAMCN quartile breakdown for the overall ranking. We split the TAMCNs into four groups according to their ranking. We observe that the first quartile has the smallest sampling of TAMCN classes. This signifies that the majority of important



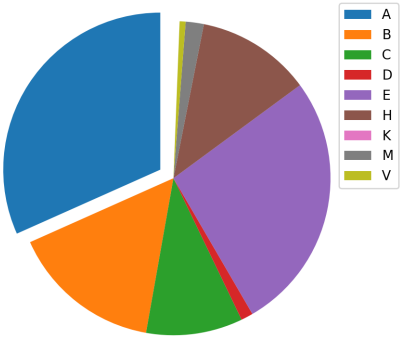
(a) First quartile



(b) Second Quartile



(c) Third Quartile



(d) Fourth quartile

Figure 4.18. TAMCN Quartile Breakdown

TAMCNs fall under five TAMCN classes. Figure 4.18 (a) illustrates that *D* TAMCNs comprise the majority of the first quartile. The proportion of *D* TAMCNs decrease as the quartile increases. This suggests that *D* TAMCNs are more important since they concentrate in the first quartile while the other TAMCNs distribute more evenly through the quartiles. This observation supports our layer analyses which show that *D* TAMCNs are the most important TAMCN class. *E* TAMCNs comprise the majority of the third quartile but appear in large numbers throughout all quartiles. Although *E* TAMCNs do not comprise a majority of the first or second quartiles, individual *E* TAMCNs convey importance as seen in Table 4.6. Low degree nodes tend to appear more often in the third and fourth quartiles.

4.4 Summary

In this chapter, we applied the methodology discussed in Chapter 3. We systematically searched for tables that contain information prevalent to creating a complex network based on the GCSS-MC database. Once we located the tables, we extracted the TAMCN and NSN columns via SQL Developer and exported them to a CSV file. We then constructed a complex network consisting of three layers from the information obtained from the CSV files. We provided an in depth analysis of the Current Parts on Order layer and then summarized the Deadline Parts on Order layer and the Open Service Request layer. We identified trends and the most influential TAMCN per layer and concluded by identifying the most important TAMCN of all three layers. We compared our ranking to current maintenance drivers and found that we identified nodes considered important within the maintenance cycle. We conclude our thesis with a summary and analysis of our findings in Chapter 5.

CHAPTER 5:
Findings

This thesis investigated the application of network science in determining importance within the GCSS-MC database. A TAMCN is considered important based on its impact to other TAMCNs in the network. We used eigenvector, degree, betweenness, and closeness centrality to determine importance in the GCSS-MC database network consisting of three layers. Additionally, other network attributes such as community structure, core structure, and assortativity aid in the understanding of relationships between nodes and edges and provide insight into the relationships between the nodes. Network models generalize behavioral properties which allow the comparison to other phenomena and provide foresight into potential changes and growth in the network. We developed a process to find columns within tables belonging to the GCSS-MC database based on search criteria related to Marine Corps procurement, maintenance, and equipment loan and transfer functions. Then we export the columns to a CSV file through Oracle SQL Developer. Each table whose columns are exported represents a relationship between the columns selected. This relationship serves as the attribute used to construct a layer in the complex network. We developed an algorithm to create a complex network consisting of multiple layers. Finally, we constructed a ranking algorithm that ranks TAMCNs in order from most to least important.

This thesis concludes the following:

1. We demonstrated a methodology that ranks TAMCNs from most to least important based on centrality measures. This methodology also identifies the most important TAMCN in each layer.
2. This methodology also identified the most influential regions within the database. Influential regions consist of TAMCN classes with high centrality measures through the analysis of the community and core structure of each layer.
3. According to the data analyzed, the GCSS-MC network behaves similarly to social networks. All of the layers model small-world networks – and two of the three are also scale-free – in their degree distribution, low average path length, and high clustering coefficients and whose values suggests a strong similarity to social networks [11],

[34], [10], [49].

4. Two of the three layers contain positive degree assortativity coefficients signifying that high degree nodes are more likely to connect with other high degree nodes and likewise for low degree nodes. The assortativity coefficients of these layers resemble assortativity values found in social networks [38].
5. We note that a non-random network structure does exist in the GCSS-MC database. As such, the database should be optimized based on the network properties discovered in this thesis with regard to the GCSS-MC database. Specifically, the GCSS-MC database should be optimized for D TAMCNs according to the data analyzed.

Alderson [50] (see also: [51], [52], [53], [54]) describes characteristics of complex networks and provides common pitfalls researchers tend to make in their analyses of networks. We examine these pitfalls in the context of the resultant GCSS-MC network produced through our methodology:

1. **Taking data at face value.** We obtained the data from the same source as [4] and [5]. Also, the methodology described in this thesis identifies and retrieves the data used to create the network.
2. **Misconceptions about power laws.** Two of the three layers depict a power law degree distribution. However, the data obtained represents a portion of a snapshot in time. Thus, we do not have insight into the growth of this network. We do not assume the GCSS-MC network follows preferential attachment growth. Since the relationships between TAMCNs are based on properties related to NSNs, hubs in the network should be affected by NSN changes.
3. **Insufficient validation.** The network constructed in this thesis does not take into consideration all possible layers or provide a history of changes in the database. Nonetheless, the methodology presented in this thesis demonstrates the ability to create complex networks from the GCSS-MC database.
4. **Reducing a complex system to a simple graph.** We recognize the complexity involved with the relationships in the GCSS-MC database and address it through the use of a multilayer network. Each layer of the network presents a different attribute, as described in Chapter 3.
5. **Confusing "Disorganized Complexity" with "Organized Complexity."** Individual TAMCNs represent systems with organized complexity. For example, a tank

exhibits technologically organized complexity in that a tank contains many components engineered to fit together in a deliberate fashion. A tank cannot be constructed by randomly connecting its components. The GCSS-MC data is a collection of technologically complex items grouped by type. However, the organization of the data within the database does not appear to be deliberate. The network created with our methodology demonstrates that structure exists within the database.

5.1 Future Work

This thesis presented a methodology to identify the most important TAMCN. The future work associated with this thesis focuses on automating the process, incorporating big data analytic techniques, and expanding the size of the network by adding layers. In this section, we discuss these ideas in further detail by providing a high level overview designed to continue the research started by this thesis.

First of all, we were unable to locate tables from the data set which reference equipment loans and transfers. Further research would benefit from obtaining data containing transfers of equipment from one unit to another. This thesis concentrated on undirected graphs. Utilizing data containing loans and transfers would enable the use of a digraph. While this thesis used four centrality measures, including a digraph would allow the use of the page rank algorithm, which may offer further insight into the GCSS-MC database. While all of the layers in the network created in this thesis consisted of nodes in $V(G)$ which represent TAMCNs, a digraph can have nodes represent units. Then, an edge connecting one node to another represents one unit loaning or transferring a TAMCN to another. Our methodology can be easily adjusted to create a digraph instead of an undirected graph. However, our ranking formula does not take into account nodes belonging to digraphs.

Automating the methodology can be divided into two parts. Figure 3.1 illustrates the methodology overview. We use different tools in order to accomplish each step depicted in Figure 3.1. Instead, one program can combine all of the steps needed to create the complex network. We iterate through the tables in order to find columns using the `File Extractor` program written in Python. Future work can entail using the `cx_Oracle` Python library [55] in order to connect to the database and pull columns without having to export it to CSV first. The same program can then use the `GCSSGraph` class to create a graph directly

from the data. Searching through tables was a time-consuming aspect of the methodology. Machine learning can be incorporated into the methodology to train an algorithm to search for columns that contain information desired to create a network. Also, since the GCSS-MC database is dynamic, the data contained in the database changes. Our current methodology ranks a static snapshot of the GCSS-MC database. Machine learning can be used to learn how the GCSS-MC data changes over time and predict important TAMCNs.

This thesis explored three well-known models, and our methodology compared the resultant networks to the models. We observed that although the degree distribution estimated the actual network, each synthetic model underestimated the composition of the core. The synthetic models also varied in their estimation of modularity and clustering coefficients. Future work can entail designing a synthetic model that closely models the behavior found in the GCSS-MC network. This model can assist in studying growth as the number of nodes and edges in the graph increase. This research can help determine whether the graph builds and grows through preferential attachment.

Lastly, big data analytic techniques can be applied to this project. This thesis does not attempt to parallelize or distribute the computation involved in creating the network. The largest CSV file we applied our methodology to is only 22.3 MB. As the data grows, the capabilities of the laptop may not sufficiently handle the processing power necessary to create the graph. For example, the `fillMatrix` method in the `GCSSGraph` class is a suitable candidate for parallelization. For large files, a HDFS cluster can be used to fill in specific a_{ij} entries pertaining to the adjacency matrix. Applying big data analytic techniques to network science in order to study the GCSS-MC database can provide decision makers with fast, accurate information regarding relationships within the database among the TAMCNs.

List of References

[1] *Consumer-Level Supply Policy*, MCO 4400.150, United States Marine Corps, Washington, DC, 2014.

[2] U. S. M. C. concepts and programs. Global Combat Support System - Marine Corps (GCSS-MC). [Online]. Available: https://marinecorpsconceptsandprograms.com/programs/command-and-controlsituational-awareness-c2sa/global-combat-support-system-marine-corps. Accessed May 25, 2017.

[3] "GCSS-MC welcome to GCSS-MC basics," MarineNet Course GCSS11BC0. Available: https://www.marinenet.usmc.mil/MarineNet/Courses/CourseDetails.aspx

[4] N. Bitto, "Adding big data analytics to GCSS-MC," M.S. thesis, Naval Postgraduate School, Monterey, Ca, 2014.

[5] A. Das, "Marine Corps Logistics Command Master Data Repository (MDR) study," Internal NPS Report, Oct. 2016.

[6] D. B. West *et al.*, *Introduction to Graph Theory*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2001.

[7] K. H. Rosen, *Discrete Mathematics and Its Applications*, 7th ed. New York, NY: McGraw-Hill, 2012.

[8] D. Ortiz-Arroyo, "Discovering sets of key players in social networks," in *Computational Social Network Analysis*. Springer, 2010, pp. 27–47.

[9] H. Burch and B. Cheswick, "Mapping the internet," *Computer*, vol. 32, no. 4, pp. 97–98, Apr. 1999.

[10] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, "Complex networks: Structure and dynamics," *Physics Reports*, vol. 424, no. 4, pp. 175–308, Feb. 2006.

[11] S. H. Strogatz, "Exploring complex networks," *Nature*, vol. 410, no. 6825, pp. 268–276, Mar. 2001.

[12] K. W. Kohn, "Molecular interaction map of the mammalian cell cycle control and dna repair systems," *Molecular Biology of the Cell*, vol. 10, no. 8, pp. 2703–2734, May 1999.

[13] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1978.

[14] T. Opsahl, F. Agneessens, and J. Skvoretz, "Node centrality in weighted networks: Generalizing degree and shortest paths," *Social Networks*, vol. 32, no. 3, pp. 245–251, 2010.

[15] J. Scott and P. J. Carrington, *The SAGE Handbook of Social Network Analysis*. Los Angeles: SAGE Publications, 2011.

[16] F. G. Laxe, M. J. F. Seoane, and C. P. Montes, "Maritime degree, centrality and vulnerability: Port hierarchies and emerging areas in containerized transport (2008–2010)," *Journal of Transport Geography*, vol. 24, pp. 33–44, Sept. 2012.

[17] F. C. I. Commission, *The Financial Crisis Inquiry Report: Final Report of the National Commission on the Causes of the Financial and Economic Crisis in the United States*. New York: PublicAffairs, 2011.

[18] M. Pióro, Á. Szentesi, J. Harmatos, A. Jüttner, P. Gajowniczek, and S. Kozdrowski, "On open shortest path first related network optimisation problems," *Performance Evaluation*, vol. 48, no. 1, pp. 201–223, May 2002.

[19] S. Zhou and R. J. Mondragón, "Accurately modeling the Internet topology," *Physical Review E*, vol. 70, no. 6, p. 066108, Dec. 2004.

[20] S. P. Borgatti, "Centrality and network flow," *Social Networks*, vol. 27, no. 1, pp. 55–71, Jan. 2005.

[21] E. Costenbader and T. W. Valente, "The stability of centrality measures when networks are sampled," *Social Networks*, vol. 25, no. 4, pp. 283–307, Oct. 2003.

[22] H.-W. Ma and A.-P. Zeng, "The connectivity structure, giant strong component and centrality of metabolic networks," *Bioinformatics*, vol. 19, no. 11, pp. 1423–1430, Jul. 2003.

[23] P. Bonacich, "Some unique properties of eigenvector centrality," *Social Networks*, vol. 29, no. 4, pp. 555–564, Oct. 2007.

[24] A. J. Seary and W. D. Richards, "Spectral methods for analyzing and visualizing networks: an introduction," in *Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers*. Washington, D.C.: National Academies Press, 2003, pp. 209–228.

[25] M. Newman, *Networks: An Introduction*. New York, NY: Oxford University Press, Inc., 2010.

[26] G. Lohmann, D. S. Margulies, A. Horstmann, B. Pleger, J. Lepsien, D. Goldhahn, H. Schloegl, M. Stumvoll, A. Villringer, and R. Turner, "Eigenvector centrality mapping for analyzing connectivity patterns in fmri data of the human brain," *PloS One*, vol. 5, no. 4, p. e10232, Apr. 2010.

[27] D. Yuhas. (2012, Jun. 21). What's a voxel and what can it tell us? A primer on fMRI. *Scientific American*. [Online]. Available: https://blogs.scientificamerican.com/observations/whats-a-voxel-and-what-can-it-tell-us-a-primer-on-fmri/

[28] A. Beveridge and J. Shan, "Network of thrones," *Math Horizons*, vol. 23, no. 4, pp. 18–22, Apr. 2016.

[29] E. M. Daly and M. Haahr, "Social network analysis for information flow in disconnected delay-tolerant MANETs," *IEEE Transactions on Mobile Computing*, vol. 8, no. 5, pp. 606–621, Nov. 2009.

[30] P. Hui, J. Crowcroft, and E. Yoneki, "BUBBLE rap: Social-based forwarding in delay-tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 11, pp. 1576–1589, Dec. 2011.

[31] P. Erdős and A. Rényi, "On random graphs, I," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

[32] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.

[33] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Apr. 1998.

[34] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, Oct. 1999.

[35] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, p. 026113, Feb. 2004.

[36] 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, Oct. 2008.

[37] M. E. Newman, "Assortative mixing in networks," *Physical Review Letters*, vol. 89, no. 20, p. 208701, May 2002.

[38] M. E. Newman, "Mixing patterns in networks," *Physical Review E*, vol. 67, no. 2, p. 026126, Feb. 2003.

[39] G. D. Bader and C. W. Hogue, "Analyzing yeast protein–protein interaction data obtained from different sources," *Nature Biotechnology*, vol. 20, no. 10, pp. 991–997, 2002.

[40] S. B. Seidman, "Network structure and minimum degree," *Social Networks*, vol. 5, no. 3, pp. 269–287, Sept. 1983.

[41] Madebyoliver, www.flaticon.com author page, 2017. Available: http://www.flaticon.com/authors/madebyoliver

[42] Freepik, www.flaticon.com author page, 2017. Available: http://www.flaticon.com/authors/freepik

[43] Networkx. Networkx overview. [Online]. Available: https://networkx.readthedocs.io/en/stable/overview.html. Accessed April 27, 2017.

[44] Gephi. Gephi overview. [Online]. Available: https://launchpad.net/gephi. Accessed April 28, 2017.

[45] Oracle. (2014, May). What is SQL Developer. [Online]. Available: http://www.oracle.com/technetwork/developer-tools/sql-developer/what-is-sqldev-093866.html

[46] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian, "Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software," *PloS One*, vol. 9, no. 6, p. e98679, Jun. 2014.

[47] *Marine Corps Readiness Reportable Ground Equipment*, MCBul 3000, United States Marine Corps, Washington, DC, 2016.

[48] Headquarters I&L, "Top 25 maintenance drivers," presented at Enterprise Ground Equipment Management Corporate and Executive Boards, Quantico, Va, 2017.

[49] M. E. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.

[50] D. Alderson, "Catching the network science bug," presented at NPS MA4404 Guest Lecture, Monterey, Ca, 2017.

[51] W. Willinger, D. Alderson, and J. C. Doyle, "Mathematics and the Internet: A source of enormous confusion and great potential," *Notices of the AMS*, vol. 56, no. 5, pp. 586–599, 2009.

[52] D. L. Alderson, "OR forum—catching the 'network science' bug: Insight and opportunity for the operations researcher," *Operations Research*, vol. 56, no. 5, pp. 1047–1065, Oct. 2008.

[53] D. L. Alderson and J. C. Doyle, "Contrasting views of complexity and their implications for network-centric infrastructures," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 40, no. 4, pp. 839–852, Jun. 2010.

[54] J. C. Doyle, D. L. Alderson, L. Li, S. Low, M. Roughan, S. Shalunov, R. Tanaka, and W. Willinger, "The 'robust yet fragile' nature of the Internet," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 41, pp. 14 497–14 502, Oct. 2005.

[55] Oracle. Using Python with Oracle Database 11g. [Online]. Available: http://www.oracle.com/technetwork/articles/dsl/python-091105.html. Accessed May 24, 2017.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California