A FRAMEWORK FOR THE FAST EVALUATION OF THE CAPABILITY-BASED CONNECTIVITY ROBUSTNESS OF A COLLABORATIVE INFORMATION NETWORK

A Thesis
Presented to
The Academic Faculty

by

Yuqian Dong

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
August 2016

Copyright © 2016 by Yuqian Dong

A FRAMEWORK FOR THE FAST EVALUATION OF THE CAPABILITY-BASED CONNECTIVITY ROBUSTNESS OF A COLLABORATIVE INFORMATION NETWORK

Approved by:

Professor Dimitri Mavris, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Professor Daniel Schrage
School of Aerospace Engineering
Georgia Institute of Technology

Professor Eric Feron
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Yen-Tai Wan
United Parcel Service

Dr. Kelly Griendling
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: June 29th 2016

To Mom, Dad, and Xiaofan.

ACKNOWLEDGEMENTS

First I want to thank my parents for their unconditionally love and support during my growth. They gave me the courage to be a dreamer. They also taught me to work hard to make dreams come true. Now, they are about to retire. I hope that, by that time, I will be able to support them and to help them realize their dreams in their new life journey.

Next, I want to express my highest gratitude to my advisor Dr. Mavris for having me as his student. Once I asked Dr. Mavris what his vision on Ph.D. education is. He told me that Ph.D. is not only about knowledge; it is also a process of self-discovering and learning to think independently. During my years in graduate school, I slowly learnt that, answers are important but what more important is to ask the right questions. Just like writing a Ph.D. thesis, to find the right topic has the most profound effects. This is not an easy process. For a long time, I had hoped that someone could tell me a good question so that I could finish my thesis earlier, which of course did not work. However, after all, I really appreciate the whole process. It gives me the methodology to face any new fields and challenges; it also enables me to see with my own eyes and think with my own brain.

In addition, I want to thank Kelly Griendling, who has been a great mentor during my time at ASDL. I cannot remember how many times that I walked into her office with confusion and frustration and walked out with confidence and new ideas. The discussions with her were always very insightful and productive. She could always provide me with helpful advices and cheering encouragements.

In the past few years, I have met many inspiring people, and some of them have become my lifetime friends. The most important one among them is my husband, Xiaofan Fei. He listens to me. He cares about me. He supports me. He is the anchor of strength for me. Because of him, I am becoming a more confident person. Because of him, I am becoming a better myself.

Ph.D. is only a period of life, but it shapes the rest of my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

CIN	Collaborative Information Network
UAV	Unmanned Air Vehicles
SUAV	Small Unmanned Air Vehicles
SoSE	System of Systems Engineering
SoS	System of Systems
G	Network
E	Set of All the Links of a Network
M	Number of Links within a Network, $M = \|E\|$
V	Set of All the Nodes of a Network
N	Number of Nodes within a Network, $N = \|V\|$
CP	Capability
CN	Connectivity
CP_0	Initial Capability (Built-in Capability)
RCN^{CP}	Capability-Based Connectivity Robustness
A	Adjacency Matrix
D	Degree Matrix

L	Laplacian
λ_i	The i th Eigenvalue of a Laplacian
ξ	Number of Spanning Trees in a Network.
ER	Effective Resistance
$ER_{i,j}$	Effective Resistance of a Node Pair
ER_G	Effective Resistance of a Network
L^+	Moore-Penrose Pseudoinverse of Laplacian
$nER_{i,j}$	Normalized Effective Resistance between Node Pair i, j
$m_{i,j}^X$	Number of Link Failures until Node Pair i, j Disconnected
$\bar{m}_{i,j}^X$	Average Number of Link Failures before Node Pair i, j Disconnected
$\tilde{\bar{m}}_{i,j}^X$	Estimated Average Number of Link Failures before Node Pair i, j Disconnected
θ	Smoothing Factor for $ER_{i,j}$
$\tilde{\theta}$	Estimated Smoothing Factor for $ER_{i,j}$
φ	Ratio of the Number of Structural Links to the Number of Redundant Links
ϕ	Augmented Effects of Redundant Links
C_k^v	Centrality of Network Node k
$C_{k,l}^e$	Centrality of Network Link k, l
$H_{i,j}$	Expected Number of Steps for a Random Walk Starting from Node i to Hit Node j for the First Time

$U_{i.j}$	Commute Time (Distance) for a Round Trip Random Walk Between Node i and Node j
λ	Information Packet Output Rate
Γ	Information Packet Processing Rate
Υ	Queue Capacity
$I_i(t)$	Internal Information Size of a Network Node at Time Stamp t
$DL_G(t)$	Total Information Packets Delivery Rate of the Entire Network at Stamp t
$DS_G(t)$	Total Information Packets Discard Rate of the Entire Network at Stamp t
$\tau_G(t)$	Time for a Packet Information to be Delivered within a Network Averaged from Time Stamp 0 to Time Stamp t
$\Omega_{k,l}^{i,j}$	Impact of Link k, l on the Capability-Based Connectivity Robustness Whose Target Node Pair is i, j
BC_k^V	Betweenness Centrality of Node k
aBC_k^V	Centrality of Node k
IC_k^V	Information Congestion Centrality of Node k
$p(m_{i,j})$	Probability of Node Pair i, j to Stay Connected with $m_{i,j}$ Number of Link Failures
$p_1(m_{i,j})$	Probability of Exact $m_{i,j}$ Number of Link Failures
$p_2(m_{i,j})$	Probability of Node Pair i, j to Stay Connected after $m_{i,j}$ Number of Link Failures

SUMMARY

Technology advancements have greatly extended the application scope of Collaborative Information Networks (CINs). Due to the unique application fields of CINs and the nature of this construction, the connectivity of the inter-connection structure under impairments is a profound but challenging requirement for a CIN. Most of the existing topological connectivity robustness measures were proposed from a pure structural perspective with little or no consideration of the capability of a network. They can describe the ability of a network to resist network fragmentation under impairments. However, the current evaluation practice provides no direct mapping between the measured connectivity robustness and the capability robustness of a network. By seeing this gap, the research objective of this thesis is to develop a method to measure the capability-based connectivity robustness of a CIN against link failures by using existing topological connectivity robustness measures.

A network model was chosen to represent the architecture of a CIN. The key to measure capability-based connectivity robustness is to link the capability of a CIN to its architecture structure. This can be done through network modeling. Network topological analysis is usually deployed to study the structure of a network. This thesis demonstrated the flexible use of network modeling. By modifying the network model of an infrastructure, network topological analysis can be used beyond pure structural analysis.

It was observed that, in order to output capability, one or more major information flows of a CIN should be maintained. The major information flows can be collapsed into the connection between several critical node pairs. To measure the capability-based connectivity robustness of a CIN is to measure the (structural) connectivity robustness of critical node pairs. The connectivity robustness of a node pair (i, j) can be directly quantified by the average number of link failures until its disconnection happens ($\overline{m}_{i,j}^{X}$), which can be estimated using the effective resistance between that node pair ($ER_{i,j}$). This

estimation method is fast and scalable. The estimation error stabilizes as network node number increases.

Centrality analyses for both existing and non-existing network entities were also performed in terms of their importance to the capability-based connectivity robustness of a network. The centrality of a network entity can be evaluated using the Moore-Penrose Pseudoinverse of a network Laplacian (L^+). Since L^+ is also used to calculate $ER_{i,j}$, the proposed centrality evaluation methods do not require any extra heavy computation other than several basic operations. As a result, the proposed methods can be used to help quickly allocate limited resources to protect network against impairments or to add additional links to strengthen connectivity.

In addition, a framework for the fast evaluation of the capability-based connectivity robustness of a CIN was constructed and was demonstrated on the example CIN followed by an alternative topology design generation process.

Assigning substitution nodes can also help strength connectivity. In this thesis, it was demonstrated how the proposed capability-based connectivity robustness measure can be used to evaluate the effects of having substitution nodes, which is a dynamic failure copying mechanism.

Finally, the effects of the capability-based connectivity robustness of a network on the required information processing capacity of each network node was also explored.

CHAPTER I

MOTIVATION AND INTRODUCTION

Humans are currently in the Information Age, which is also known as the Computer Age, Digital Age, or New Media Age. It is a period in human history characterized by the shift from traditional industry to an economy based on information computerization. The onset of the Information Age is associated with the Digital Revolution, just as the Industrial Revolution marked the onset of the Industrial Age. The entire human society is going towards the idea that, individuals will be able to transfer information freely, and to have instant access to knowledge that would have been difficult or impossible to be found previously. As we are marching towards that goal, our ways of doing things have been remarkably changed.

Information is useable data, inferences from data, or data descriptions [1]. The ability of gathering, translating and making sense of information has become one important factor that determines the success of an individual or an organization in the current knowledge-based society. Information exchange is critical to the performance of many networked systems, such as internet, air and ground transportation networks, business firms, military systems, and emergency respond systems [2], just to name a few. In order to increase the overall information level as to enhance performance or to complete tasks that are impossible to be achieved by individual participants alone, individual entities always work in collaboration[1] and form collaborative information networks. A collaborative information network (CIN) is a network within which component systems generate information and share it with others in the network via

[1] Collaboration means to work together in group(s) to achieve a common task or goal and irrespective of geographical separation. 3.Durugbo, C., et al., *Modelling collaboration using complex networks.* Information Sciences, 2011. **181**(15): p. 3143-3161..

information links to enhance the overall situation awareness and to increase performance and efficiency [4].

Rapid technological advances on electronic, sensor and communication technologies have greatly extended the scope of CIN operations with enhanced flexibility. One example is the use of networked small Unmanned Air Vehicles (SUAVs). SUAVs encompass the Micro, Mini and Close Range categories of Unmanned Air Vehicles (UAVs). According to [5], this classification means SUAVs have maximum takeoff weight less than or equal to 150 kg, maximum range of 30 km, and maximum altitude of 5 km mean sea level. Single-UAV systems have been in use for military missions since the beginning of UAV flight, due to their abilities to effectively operate in dirty, dull or dangerous missions [6]. Comparing to the use of Single-UAV systems, using a group of networked SUAVs has many advantages because of their better scalability [7], higher flexibility [8], greater accessibility [9], smaller radar cross-section [10] and relatively lower operation expenses[11]. The aforementioned technology advances have also enabled the ability to design and manufacture agile SUAVs at lower cost. As a result, the range of both military and civilian applications of networked SUAVs are getting wider, such as military target search and destroy operations [12], persistent surveillance [13-16], target tracking [17], wildfire control [18, 19], environment and weather monitoring [20-23], disaster management [24], and law enforcement [25].

For a CIN, especially when its operation scale is large, often than never, it is very hard to obtain well-documented performance data. The absence of performance data coupled with the increasing size and complexity of its interacting systems presents a large degree of uncertainty around a CIN operation [13]. This is especially pragmatic when the

CIN operation environment is at high stake. Hence, it is important to design a CIN with enough robustness to maintain its capability[2] under adverse changes during operation.

As mentioned earlier, effective communication is crucial for the cooperation and collaboration between entities within a CIN. Therefore, the capability robustness of a CIN highly depends on whether its architecture can provide robust networked communication. To achieve this, the most profound but challenging requirement is to maintain connectivity under network impairments [2, 30-32]. Network impairments refer to any kind of attack, multiple or cascading failures that can occur upon a network [33]. That is to say, connectivity loss under network impartments is a major cause for the capability loss of a CIN during operation.

Using networked SUAVs as an example. Unlike larger UAVs, SUAVs are in a unique regime where their capabilities to carry onboard connectivity loss mitigating technologies are limited yet their potential to be damaged is high [5] for the following reasons.

1. SUAV Platform Constraints
2. Adverse Environment Conditions
3. High Operation Mobility

The payload and space limitations of an SUAV are much higher than a traditional UAV. Those tighter constraints pose an important issue for the performance of SUAVs due to relative lower onboard power, sensing, communication and computation capabilities. Lower onboard power and communication capabilities can result in less reliable wireless communication channels and shorter communication ranges [34]. The operation environments of networked SUAVs are usually adverse, such as natural

[2] A capability is the ability to achieve a desired effect under specified standards and conditions through combinations of ways and means to perform a set of tasks. 26.Government. *Systems Engineering Guide for Systems of Systems*. 2008.

disaster scenes and impediment terrain structures. Under such circumstances, it can be hard or even impossible to maintain the communication links between SUAVs. Due to the high motilities of SUAVs, collision avoidance between SUAVs and SUAVs, SUAVs and obstacles becomes an important issue. Undoubtedly, lower sensing and communication capabilities increase the probability of collision [35]. In addition, as written in [36], latency is one of the most important design issues for all types of networks. Limited communication and onboard computation capabilities of SUAVs can not only increase the potential of vehicle loss [5, 37] but also diminish the overall delivered capability, especially when information timeliness is valued high [38].

Hence, to design a CIN that can maintain connectivity under network impairments during operation is essential for its capability robustness. In other words, that is to design a connectivity robust CIN to maintain desired overall capability.

1.1 *Capability-Based Connectivity Robustness Measure*

Connectivity robustness is not a new topic. It is defined as the ability of a network to remain connected when its component systems experience impairments [2]. As mentioned earlier, the connectivity robustness of a CIN should be directly linked to its capability robustness. In order to design and evaluate the connectivity robustness of a CIN, and understand how it supports the capability robustness, we need to be able to measure it first.

In recent years, a large amount of researches has been conducted on measuring the connectivity robustness of a network. According to [39], the most suitable connectivity robustness measure should be chosen based on the problem under investigation and the size of the network under analysis.

In general, connectivity robustness can be evaluated via direct simulation or topological measures. The problem of simulation results is that they lack transparency and the method itself is not scalable well to large networks. Whereas topological measures are more intuitive and have better scalability. In addition, they are more suitable when timely analysis result is critical.

An initial review of existing topological connectivity measures shows that almost all of the measures were proposed from a pure structural perspective with little or no consideration of the capability of a network. They can describe the ability of a network to resist network fragmentation under impairments. However, the current evaluation practice provides no direct mapping between the measured connectivity robustness and the capability robustness of a network.

Instead, a new type of connectivity robustness called capability-based connectivity robustness was proposed. It is defined as the ability of a network to maintain connectivity among component systems in a way that retains network capability under impairments. Mathematically, the capability-based connectivity robustness of a network can be expressed as following.

$$RCN^{CP} \propto Network\ Impairments\ Sustainable|CP \quad \textbf{1}$$

where,

RCN^{CP} represents capability-based connectivity robustness;

CP represents capability.

Equation 1 in essence says that, the capability-based connectivity robustness of a network should be proportional to the amount of network impairments a CIN can sustain while still outputs capability.

Many studies have shown that, a network can have different robustness behaviors depending on the type of impairments [2, 40-44]. The most basic way is to categorize

impairments by hit point type, which is either an individual entity or a communication link. Network impairments can also be grouped into either random or targeted. Random impairments are usually failures. In this thesis, focus will be given to link (random) failures for the following reasons.

1. Failures exist among all CIN operations, while attacks can only happen in certain operations.
2. An entity failure is equivalent to a set information transmission line failures.
3. For a CIN, communication link outrage happens more frequently than entity lose [5, 36].

1.2 *Network Model*

A model is a useful approximation of the object under modeling to aid the understanding and/or predicting of its behavior [45]. Since in this thesis topological measures are selected for evaluating the capability-based connectivity robustness of a network, it is nature to abstract or represent a CIN through a network model.

Network models have been widely used to represent and study the inter-connection structures of complex networks. They have simple constructions, elegant mathematical representation and unique capabilities to support various analyses that can yield fruitful results. Network models are based on graphs. The associated theory is network theory. Network theory origins in graph theory and is an area of applied mathematics. Network theory concerns itself with the study of graphs as a representation of either symmetric relations or, more generally, asymmetric relations between discrete objects. Sometimes, the term “network theory” is used interchangeable with “graph theory”. For the purpose of analysis, network models are grouped into two categories: real network models and synthetic network models. Real network models are abstractions

of real world networks. They are used to analyze or investigate existing networked architectures. Synthetic network models are usually used to generate networks or groups of networks with similar characteristics according to customized rules to study the general trends of certain network properties.

A network model is denoted as $G = (V, E)$. $V$is the set of nodes (vertices) and E is the set of links (edges). Network nodes (vertex) represent the entities in CIN network, and network links (edges) represent the communication links between network entities. Network topology is the graph that indicates the arrangement of the nodes and links of a network model.

If the starting and ending vertices of a link are the same, then that link forms a loop. A network without loops is a simple network. A network with no nodes and no links is an empty network. A network with only one node and no link is a trivial graph. In this thesis, are network models studied are simple and nontrivial. The most common way to categorize a network model is based on whether its links are directed (directed network) or not (undirected network). If the links of a directed network are all bidirectional, then such a directed network can be simply modeled as an undirected network. Figure 1 is a simple example of an undirected network.

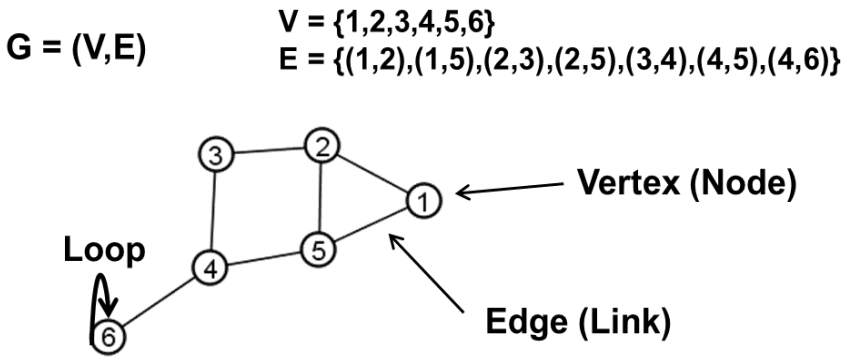


Figure 1. Network Topology of an Undirected Network $G = (V, E)$

A more elegant way to represent a network topology than graph is to use an adjacency matrix. Although an adjacency matrix is less intuitive and graphic than graph, it is more suitable to represent large-scale network topologies and enables mathematical operation on network topologies. Figure 2 provides two adjacency matrix examples. One is an undirected network topology, and the other one is a directed network topology.

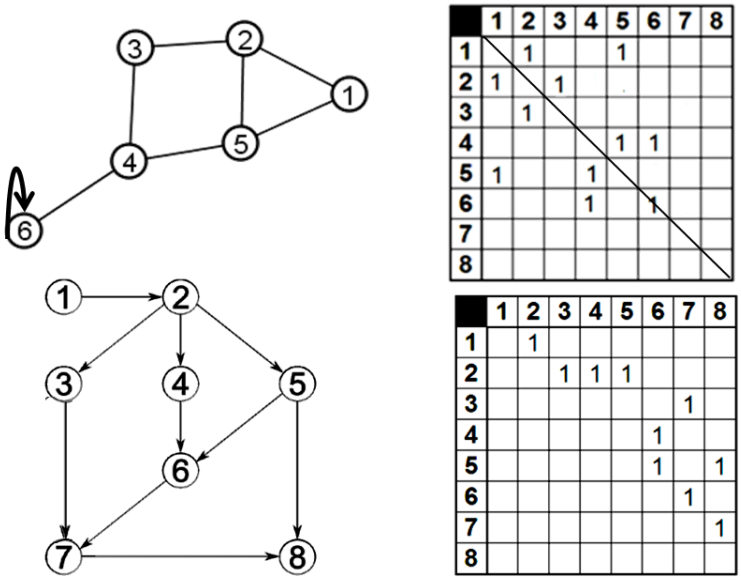


Figure 2. Adjacency Matrices for Directed and Undirected Networks

1.3 *Congestion Consideration*

As discussed earlier, connectivity loss results from network impairments (only link failures are considered in this thesis). By making that statement, there is a presumption, which is, all the component entities (network nodes) have enough information processing capabilities so that none of them will experience congestion during the CIN operation. Congestion is a result of information overload. When the total information input rate of a network node is higher than its information processing rate, information will accumulate at this node and eventually, this node will be overloaded and starts to experience congestion.

Network congestion can result in information lose, delay and impair network connectivity. Network connectivity loss will result in communication route loss or changes that may lead to information overload on one or more nodes. This means network congestion and network connectivity loss are inter-related. As mentioned above, researches on network connectivity always presume that no congestion will happen. While researches on network congestion always presume that no network topology change. By doing so, the two problems: connectivity and congestion are isolated, which can significantly simplify the analysis process. However, in reality those two problems should be considered simultaneously when design a CIN. Hence, in this thesis, both of the two problems are considered. In order to simplify the analysis process, congestion is viewed as a node (individual entity) level design requirement that can be derived from network topology (inter-connection structure) and its connectivity situation. First, assume all the network nodes have enough information processing capabilities to ensure no congestion will happen at any point of the CIN operation, even under network impairments. Next, re-examine that assumption by performing congestion analysis to derive node level design requirement, which in specific is the information processing capabilities of each network node.

Real world design practices are never conducted without a consideration on cost. For a CIN, by deploying more participant entities with high information processing capabilities, more communication channels (such as all entities can communicate to each other, a P2P structure[3]) with high reliability, the network can have very high capability-based connectivity robustness, but at a very high acquisition cost. On the other hand, even with the same number of participant entities, or communication channels, different

[3] For most operations, the current communication technologies and computation abilities are not able to support large scale, long distance P2P architecture. 36. Bekmezci, I., O.K. Sahingoz, and Ş. Temel, *Flying ad-hoc networks (FANETs): A survey.* Ad Hoc Networks, 2013. **11**(3): p. 1254-1270.

inter-connection structures can incur different costs due to different individual level design requirements, such as the information processing capabilities discussed earlier. The goal of this thesis is to provide a capability-based connectivity robustness measure for a CIN. The measure should be able to provide design insights on how the connectivity robustness of a CIN affects its capability robustness. In order to yield practical design insights or design guidelines, when the measure is used to analyze or design CINs, cost must be considered as well.

1.4 *Research Objective and Research Questions*

The rapid technological advances on electronic, sensor and communication technologies have greatly extended the scope of CIN operations with enhanced flexibility. However, there are many challenges to be addressed. Due to the unique application fields of CINs and the nature of this construction, its connectivity robustness against impairments is a profound but challenging requirement on a CIN design. Most of the existing topological connectivity robustness measures were proposed from a pure structural perspective with little or no consideration of the capability of a network. They can describe the ability of a network to resist network fragmentation under impairments. However, the current evaluation practice provides no direct mapping between the measured connectivity robustness and the capability robustness of a network. By seeing this gap, the research objective of this thesis is to develop a method to measure the capability-based connectivity robustness of a CIN against link failures by using existing topological connectivity robustness measures.

A network model is used to represent the inter-connection structure of a CIN. Since the objective is to use existing topological connectivity robustness measures, we need to transform the problem of measuring capability-based connectivity robustness into

the problem of measuring conventional (structural) connectivity robustness. With this, the following two research questions were developed.

Research Question 1: How to incorporate capability into the conventional network modeling process?

Research Question 2: Which existing topological connectivity robustness measure should be chosen?

1.5 *The Example Problem*

In this thesis, a disaster management application of networked SUAVs is used as the example problem. The example problem is illustrated in Figure 3. The SUAVs in this scenario forms a CIN. The main advantage of using networked SUAVs is to collect reliable data from a wide field of dangerous disaster scenes in an affordable way.

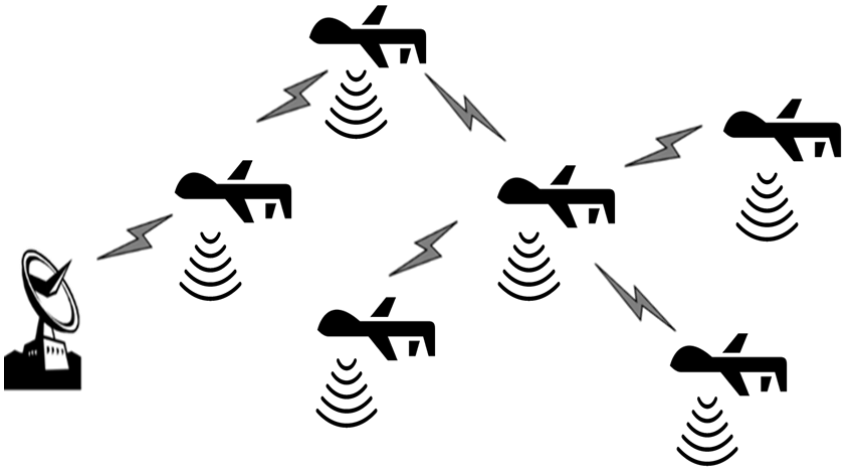


Figure 3. Disaster Management Application of Networked SUAVs

In this scenario, a city was struck by a severe earthquake. After the earthquake, a group of networked SUAVs equipped with sensors and cameras are to be dispatched for post-disaster inspection. Each SUAV is responsible for a field of the city and sends regular updates of its responsible field back to the command center. Using the collective information, the command center makes decision on where to send search and rescue

teams, and ends position update information back to the SUAVs. An SUAV also sends its position information to its nearby SUAVs to avoid collision and to keep each other within the communication rage. During the mission, an SUAV can send information back to the command center directly or relaying through other SUAVs. If no path exists between an SUAV and the command center, an SUAV cannot establish a new path itself either by relaying through other SUAVs or by directly connecting to the command center.

Therefore, an SUAV cannot send useful information back to the command center if there is no communication path established between the SUAV and the command center or if the SUAV moves too far away from its responsible field. Moreover, there are obstacles, hazardous weather conditions, such as strong wind, and heavy clouds, which can impair the wireless communication links of the CIN.

Since the major objective of this operation is to collect and stream-back sufficiently good quality data to the command center, those aforementioned connectivity loss issues can affect the outputted capability of this CIN so that it may not successfully complete this operation.

In order to make sure the CIN maintain connectivity to support its capability output during operation, it is asked to measure the capability-based connectivity robustness of the CIN. This CIN operation is a rapid response deployment to a natural disaster, so a timely result is critical.

CHAPTER II

CAPABILITY-BASED NETWORK MODELING

In Section 1.2, a brief introduction on the conventional network modeling process has been given. To construct a conventional network model for the CIN in the example problem, model the component SUAVs and their responsible fields as network nodes. Model the information transmission lines as network links. Use solid lines to represent the information transmission between the SUAVs and the command center. Use dashed lines to represent the information transmission between the SUAVs and their responsible fields. To simply the problem under examination, in this thesis, assume all the information transmission lines are bidirectional. This means a CIN can be simply modeled as an undirected network. For the information transmission lines between the SUAVs and the command center, it is not hard to conceive the bidirectional information transmission situation. For the information transmission between the SUAVs and their responsible fields, it is to assume the sensor equipped to the SUAVs are active sensors. Furthermore, ignore the characteristics of the information transmission lines, which is to assume all the network links are unweighted.

The conventional network model of the example CIN is shown in Figure 4. C represents the command center; u_1 to u_6 represent the SUAVs; and f_1 to f_6 represent the inspection fields of the SUAVs.

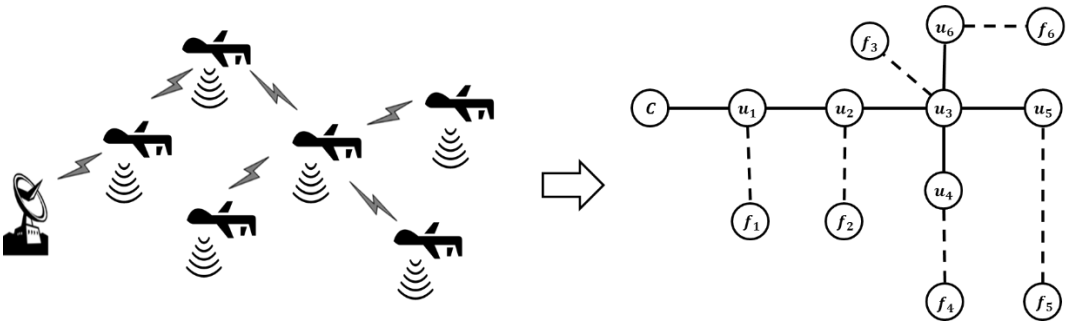


Figure 4. Conventional Network Model of the CIN in the Example Problem

By definition, the capability of a CIN is completely dependent on information sharing. In the representation of a CIN via a network model, the capability of a CIN manifests itself as a set of key information flows with logical relationships [45-47]. Hence, any CIN capability can then be represented as a series of logical operations on the key information flows. To incorporate the CIN capability into a conventional network model is to reflect the logical operations on the key information flows through network modeling.

There are two types of logical operations, one is OR and the other one is AND. OR operation is performed on the set of flows that have OR relationship, which means, as long as one of the key information flows is maintained, the capability of a CIN can be sustained. Use the example CIN as an example, the key information flows are the ones between the command center (node C) and the inspection fields (f_1 to f_6). Assume the relationship between those key information flows are OR. To reflect the OR relationship, collapse the key information flows into the connection between a node pair, which is denoted as the critical node pair. For the example CIN, this network modification is shown in Figure 5. In this example, to reflect the OR operation, it is to combine all the nodes that represent the inspection fields into one single node. The critical node pair resulted from this modification is node pair C, F. With this modification, to measure the capability-based connectivity robustness of a CIN is to measure the structural

connectivity robustness of the connection between the critical node pair, which in this example is the connection between the command center and the combined inspection field. Mathematically, it means $RCN^{CP} = RCN_{C,F}$, where $RCN_{C,F}$ represents the connectivity robustness between the critical node pair C, F.

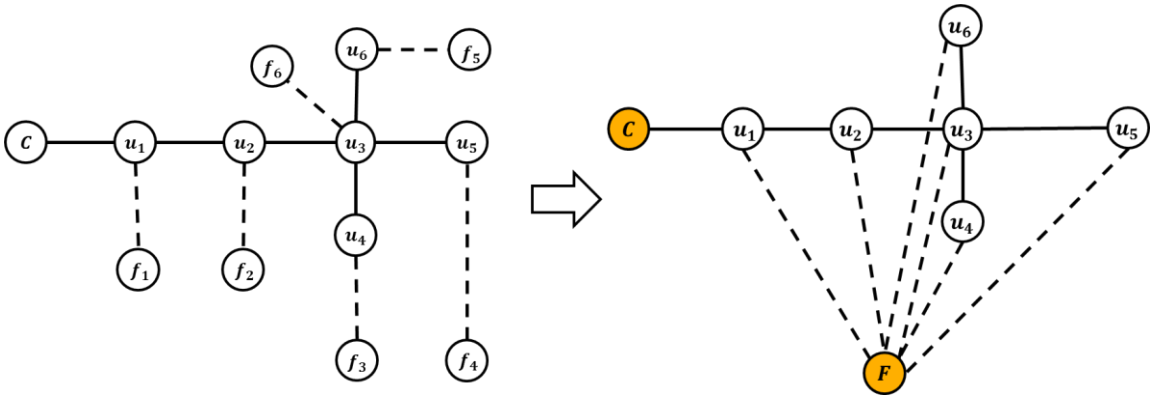


Figure 5. Network Modification to Reflect OR Operation on the Example CIN

AND operation is performed on a set of key information flows that have AND relationship, which means, all the key information flows have to be connected in order to sustain the capability of a CIN. In this case, all the key information flows are critical. To reflect the AND operation through network modeling, evaluate the structural connectivity robustness of each critical / AND flow separately. For the example CIN, the key information flows are still the ones between the command center (node C) and the inspection fields (f_1 to f_6). However, this time assume all the key information flows have AND relationship and hence all the key information flows are critical as shown in Figure 6. The capability-based connectivity robustness of a CIN in this case, is the minimum structural connectivity robustness among all the critical node pairs (each critical flow forms a critical node pair), which in this example is the minimum structural connectivity robustness among the connection between the command center and the inspection fields. Mathematically, it means $RCN^{CP} = \min(RCN_{C,f_i})$, where $i = 1, 2, \ldots, 6$.

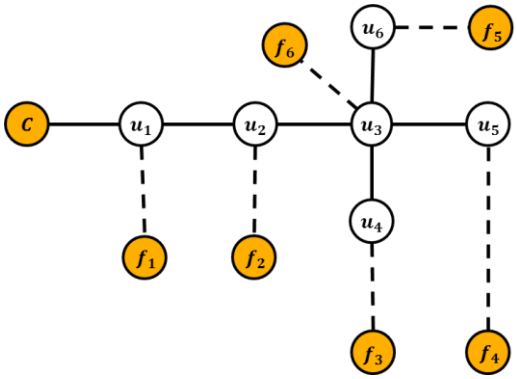


Figure 6. Network Modification to Reflect AND Operation on the Example CIN

What if both OR and AND relationships exist among the key information flows of a CIN? For example, modify the CIN in the example problem as shown in Figure 7 on the left and the conventional network model for this modified CIN is shown on the right.

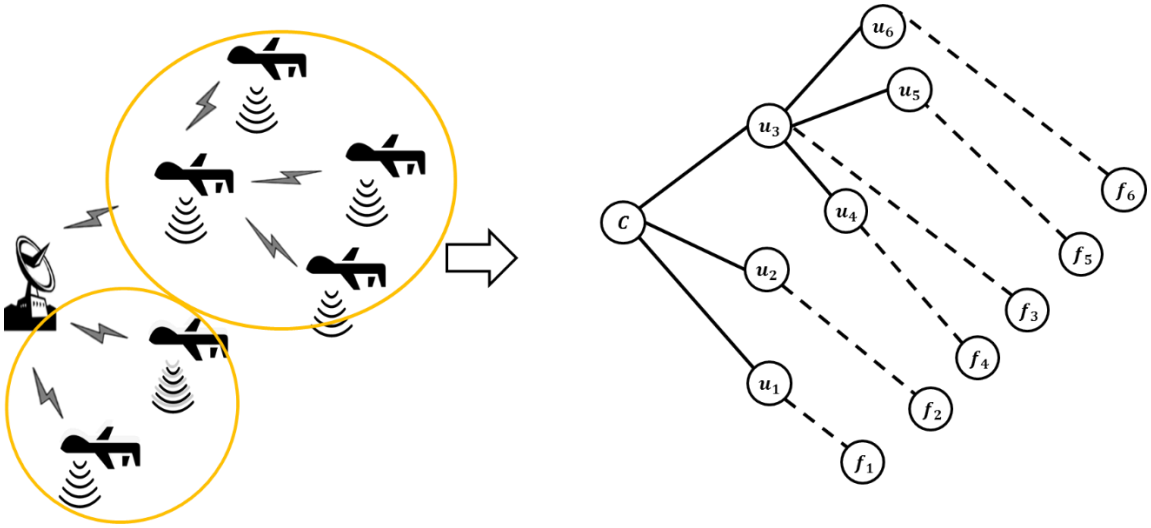


Figure 7. Variation of the Example CIN

The key information flows are still the ones between the command center and the inspection fields. However, this time, there are two OR flow groups. Within each OR group, the key information flows have OR relationship, and the two OR groups have AND relationship. In the situation when both OR and AND relationships exist, first perform OR operation within each OR group. Each OR group results in a critical node pair. Evaluate the structural connectivity robustness of each critical node pair. Next,

perform AND operation among the critical node pairs, which is to take the minimum structural connectivity robustness among all the critical node pairs. For the variation of the example CIN, first, collapse the key information flows within each OR group into the connection between a critical node pair as shown in Figure 8 and $RCN^{CP} = \min(RCN_{C,F_1}, RCN_{C,F_2})$.

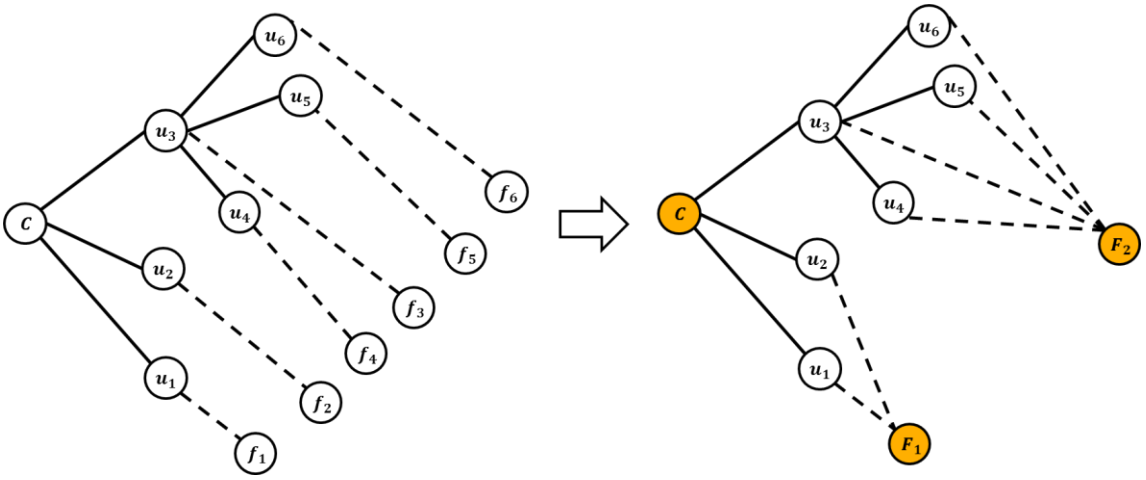


Figure 8. Network Modification to Reflect OR and AND Operations on the Example CIN Variation

The previous discussion can be summarized into a capability-based network modeling process.

1. Construct a conventional network model of a CIN.
2. Identify the key information flows and their logical relationships.
3. Apply logical operations on the key information flows and simplify the network model into the connection between critical node pairs.
4. Calculate the structural connectivity robustness of each critical node pair.
5. Take the minimum structural connectivity robustness among all the critical node pairs as the capability-based connectivity robustness of the CIN. The critical node pair with the smallest structural connectivity robustness is referred to as the capability critical node pair of the CIN.

$$RCN^{CP} = \min\left(RCN_{i,j}\right) = RCN_{i^*,j^*} \quad \textbf{2}$$

where, i, j denotes general critical node pairs and i^*, j^* denotes the capability critical node pair of a CIN.

Now with a capability-based network model, the problem of measuring the capability-based connectivity robustness of a CIN is successfully transformed into the problem of measuring the structural connectivity robustness between critical node pairs. With this, the first research question has been successfully answered. The next task is to find the answer to the second research question, which is to select a topological measure for the structural connectivity robustness between an arbitrary node pair against link failures.

CHAPTER III

MEASURING CAPABILITY-BASED CONNECTIVITY ROBUSTNESS

From now on, connectivity robustness will be used to refer to the structural connectivity robustness between a node pair against link failures. For the capability-based connectivity robustness of a CIN, it will always be specified.

In order to facilitate the selection of existing topological connectivity robustness measures, a set of requirements were developed. First a candidate measure should be quantitative to facilitate comparison. Next, since the problem has been transformed into measuring the connectivity robustness of critical node pairs, a candidate measure should be applicable to a node pair. In addition, with the pre-defined research scope, a candidate measure should be able to capture the connectivity change between a node pair under link failures. Finally, such a measure should also account for the effects of alternative (backup) paths between a node pair. It has been shown that the number of alternative paths or back-up paths and the extent to which they overlap are directly linked to the concept of connectivity robustness [48, 49].

3.1 *Review of Existing Connectivity Robustness Measures*

The following is a brief review of the current available topological connectivity robustness measures. There are two types of topological connectivity robustness measures according to [48]: classical and spectral.

3.1.1 Classical Robustness Measures

The classical measures refer to those directly related to the topology of a network. The following discussion covers four representative groups of classical connectivity robustness measures.

Connectivity Related Measures

The original connectivity measure is a binary measure, which is essentially just a graph connectivity indicator. It can only distinguish if a graph is a connected whole (value: 1) or has several disconnected components (value: 0). It cannot provide any detailed information on network structure other than being as an indicator. Apart from the classical connectivity measure, node/ link connectivity is defined as the minimal number of nodes/ links to be removed to disconnect a given network [48]. This measure was applied to study the connectivity robustness of a military architecture by Dekker in [50]. The major drawback of the node/link connectivity is that it cannot explicitly reflect any information on alternative or backup paths.

Distance Based Measures

The measures in this group are quite plenty, and the following is just a brief introduction of selected some.

Geodesic distance is the shortest path length from one node to another node in a network. There may be and often are more than one geodesic path between two nodes [51]. Average geodesic distance, which is usually considered as the characteristic path length of a network, is the averaged geodesic distance among all the node pairs of a network. It characterizes the average ability of two nodes in a graph to communicate with each other [40]. Diameter is the longest geodesic distance among all the nodes pairs of a network. This is used to measure the eccentricity of a given network topology and is

applied to detect abnormal change of a network [48]. Because the characteristic path length is more sensible to changes of network topology, it is used more often than diameter as a network connectivity robustness measure. When a network is disconnected, the values of those two measures will both be infinite. To deal with that issue, Latora and Marchiori proposed to use the reciprocal of the geodesic distance to calculate the characteristic reciprocal path length, which was introduced as network global efficiency in [42]. However, none of those measures considers alternative paths between node pairs [48].

Clustering Coefficient

A cluster in a graph refers to a group of nodes having relatively denser relations with each other than with the rest of the nodes in the graph. The clustering degree of a network is measured by clustering coefficient, which is a number ranging between 0 and 1. Although clustering coefficient was originally designed to study social networks, it is highly correlated with the notion of network robustness, since the number of alternative paths grows with the number of network triangles [48]. The problem of clustering coefficient is that, it cannot evaluate the connectivity situation between two specific nodes. It only considers the averaged connection density of the whole network, or the averaged neighborhood connection density of a single node.

Component Size Based Measures

A component is the maximal connected subgraph of a network. The largest component of a network is the one contains the largest number of nodes. Giant components refer to the ones whose component sizes (number of nodes) are larger than the giant component threshold. Examples of connectivity robustness measures related to this concept are the largest component size, the average component size, the fraction of

giant components. The measures in this group also surfer from the same problem as clustering coefficient.

3.1.2 Spectral Robustness Measures

Different from classical connectivity robustness measures, spectral connectivity robustness measures are not directly derived from network topologies. They are obtained based on spectrum graph theory, which more specifically is the Laplacian of a network. The Laplacian matrices of networks have great theoretical and practical importance [52]. For a network $G = (V, E)$, denote its adjacency matrix as A and its degree matrix as D. The degree matrix, D of a network is a diagonal matrix of network node out-degrees.

$$D_{i,j} = \begin{cases} \sum_{k=1}^{N} a_{i,k} & if\ i = j \\ 0 & if\ i \neq j \end{cases} \quad \textbf{3}$$

where,

$N = |V|$ is the number of nodes of a network.

Then the Laplacian L of network G can be obtained by taking difference between D and A.

$$L = D - A \quad \textbf{4}$$

Symmetric Laplacians associated to undirected graphs and their applications on analyzing network robustness have been deeply studied [53-57]. While asymmetric Laplacians that are associated with directed graphs are less explored. In order to symmetrize asymmetric Laplacians so that to apply those operations developed for symmetric Laplacians (e.g. Moore-Penrose pseudoinverse, Eigenvalue analysis), some normalization techniques on the asymmetric Laplacians are usually used. Depending on the research contents and analysis focuses, different normalization techniques have been

proposed. Although asymmetric Laplacians are now attracting more and more attentions [52, 57-61], it is still a working concept without conscience upon normalization techniques as well as the physical meanings behind them.

As discussed earlier, in this thesis, CINs are abstracted as undirected, unweighted networks. Therefore, in the following section, the focus will be given to spectral connectivity measures developed for undirected networks.

Singe Eigenvalue Based Connectivity Measure

A symmetric Laplacian is positive semidefinite and its rows sum up to 0. Therefore, its eigenvalues are real, non-negative and the smallest eigenvalue is 0. Denote the Eigenvalues of a symmetric Laplacian as λ_i for $i = 1, \dots, N$ and i is ordered in the following fashion $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$. Then λ_N and λ_2 can be used to indicate the connectivity robustness of an undirected network. In general, networks associated with larger eigenvalues have more node and link disjoint paths between node pairs. And the largest eigenvalue, λ_N can provide bounds on network connectivity robustness with respect to both link and node removals [27, 62]. The second smallest eigenvalue λ_2, which is also referred to as algebraic connectivity, is a more accurate measure to unfold the connectivity robustness of complex networks [63-65]. The larger the λ_2 value, the harder it is to break a network into islands or individual components. The problem of using λ_2 and λ_N as candidate connectivity robustness measures is that they can only be applied to a network as a whole but not a single node pair.

Average Eigenvalue Based Connectivity Measure

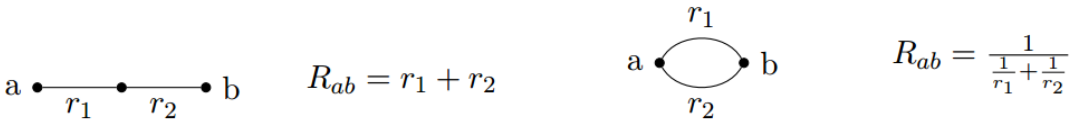
According to [48, 66], the number of spanning trees in a network (a spanning tree is a subgraph containing N-1 edges and no cycles) can be used as an indicator of network robustness. The number of spanning trees surfers from the same problem as λ_2 and λ_N.

$$\xi = \frac{1}{N}\prod_{i=2}^{N}\lambda_i \quad \textbf{5}$$

Moore-Penrose Inverse Based Connectivity Robustness Measure

The measure to be highlighted in this section is called effective resistance (ER). Different from the spectral measures discussed previously, which are only applicable for a network as a whole, ER can be used to measure connectivity robustness for both a single node pair and the entire network. In addition, by nature, effective resistance can capture the effects of alternative paths on network connectivity robustness against link failures. Use $ER_{i,j}$ to denote pairwise effective resistance and ER_G to denote the effective resistance of a network. For both $ER_{i,j}$ and ER_G, smaller values are desired.

The notion of pair wise effective resistance was originally developed to represent the resistance of the total system when a voltage source is connected between a node pair. That notion can be applied to calculate the connectivity robustness between a node pair within a network by seeing the network as an electrical circuit, where a link corresponds to a resistor of resistance r [67, 68]. r can be calculated as a function of link weights and the function form depends on the network type and the physical meaning of link weights. After defining the resistance of each link, the effective resistance between a node pair can be calculated using Kirchhoff's circuit Laws as illustrated in Figure 9.



(a) Resistors in series (b) Resistors in parallel

Figure 9. Effective Resistance between Two Points a and b in Simple Graphs [67]

For small and simple networks, it is viable to calculate $ER_{i,j}$ using Kirchhoff's circuit Laws. However, using Kirchhoff's circuit Laws to solve for pairwise effective

resistance has serious scalability issues and can be very cumbersome when the size of a network grows larger with more complex structures. A more elegant way to calculate $ER_{i,j}$ without scalability issues is to use the Moore-Penrose pseudoinverse. Use $L^{+}$to denote the Moore-Penrose pseudoinverse of a symmetric Laplacian. Then $ER_{i,j}$ can be calculated through the following equation:

$$ER_{i,j} = L^{+}_{i,i} - 2L^{+}_{i,j} + L^{+}_{j,j} \quad \textbf{6}$$

ER_G can be obtained by summing the pairwise effective resistances over all node pairs of a network G.

$$ER_G = \sum_{1 \le i < j \le N} ER_{i,j} \quad \textbf{7}$$

According to [68], the ER_G can also be calculated through aggregating Laplacian Eigenvalues as shown below.

$$ER_G = N \sum_{i=1}^{N} \frac{1}{\lambda_i} \quad \textbf{8}$$

Table 1 is a summary of the findings based on the above discussion.

Table 1. Summary of Network Connectivity Robustness Measures

Not Satisfied
With Exception
Satisfied

Types		Requirements			
		Quantitative	Node Pair	Link Failures	Alternative Paths
Connectivity Based Measures	Network Connectivity				
	Link/Node Connectivity				
Distance Based Measures	Characteristic Path Length				
	Efficiency				
Clustering Coefficient					
Component Size Based Measures					
Singe Eigenvalue Based Connectivity Measures	Largest Eigenvalue				
	Second Smallest Eigenvalue				
Average Eigenvalue Based Connectivity Measures	Number of Spanning Trees				
	Effective Resistance				

3.2 *Pairwise Effective Resistance*

It seems that only effective resistance (ER), in specific $ER_{i,j}$, satisfies all the requirements. In the literatures, $ER_{i,j}$ is always used to compare the connectivity robustness of node pairs connected via the same number of nodes; when used to compare two arbitrary node pairs, a normalization against the network node number N is suggested [54-56]. This leads to the first hypothesis of this thesis. Since there are two embodiment forms for the connectivity robustness under investigation (structural connectivity robustness against link failures between a node pair), $\bar{m}_{i,j}^{X}$ and $\frac{\bar{m}_{i,j}^{X}}{M}$, Hypothesis 1 has two forms. ($\bar{m}_{i,j}^{X}$ is the average number of link failures until the node pair of interest i,j disconnects; $\frac{\bar{m}_{i,j}^{X}}{M}$ is the average fraction of link failures until the node pair of interest i,j disconnects.) The reason for using the inverse form is that smaller values of effective resistance should correspond to larger numbers / fractions of link failures required until disconnection happens.

H^{1a}: $\frac{N}{ER_{i,j}}$ has higher correlation with $\bar{m}_{i.j}^{X}$ then $\frac{1}{ER_{i,j}}$.

H_0^{1a}: $\frac{N}{ER_{i,j}}$ does not have higher correlation with $\bar{m}_{i.j}^{X}$ then $\frac{1}{ER_{i,j}}$.

H^{1b}: $\frac{N}{ER_{i,j}}$ has high correlation with $\frac{\bar{m}_{i.j}^{X}}{M}$ then $\frac{1}{ER_{i,j}}$

H_0^{1b}: $\frac{N}{ER_{i,j}}$ does not have higher correlation with $\frac{\bar{m}_{i.j}^{X}}{M}$ then $\frac{1}{ER_{i,j}}$.

In order to test H^{1a}, H^{1b} the following experiment plan was developed. First an undirected synthetic network model, Step-Min network model was proposed to help systematically examine the relationship between connectivity robustness and $ER_{i,j}$.

As known, for a network with node number N, the maximum effective resistance value between a node pair is $N-1$. This corresponds to a line network with N nodes,

where the two end nodes form the node pair of interest. A line network with N nodes is the least robust connection structure between a node pair that N nodes can form. The minimum effective resistance value between a node pair is $\frac{2}{N}$. This corresponds to a fully connected network with N nodes and any node pair within the fully connected network can be the node pair of interest. A fully connected network is the most robust connection structure between a node pair with N nodes. Given a node number N, denote the node pair of interest as $1, N$. A Step-Min network family with N nodes is constructed in a way to thoroughly and systematically explore the range of possible $ER_{1,N}$ values between $[\frac{2}{N}, (N-1)]$ without having to resort to a full factorial.

To construct a Step-Min network family, first, decide the number of network nodes N. Then, connect the N nodes as a line. Index the nodes in the following fashion. Denote one end of the line as 1, and then increase the node index along the line until reaching the last node, whose index should be N. Now the network should have N nodes connected by $(N-1)$ links as a line. This line network will be referred to as the base network for network family N and the index of the base network of each network family will always be 1. In addition, node pair $1, N$ will always be the node pair of interest. Next, starting with the base network, each step add one link to the network that minimizes the decrease of $ER_{1,N}$ (According to [56], for a given node pair within a network, adding a link to the network will not increase the effective resistance between that node pair.). Repeat this process until a fully connected network is obtained. This process is illustrated in Figure 10.

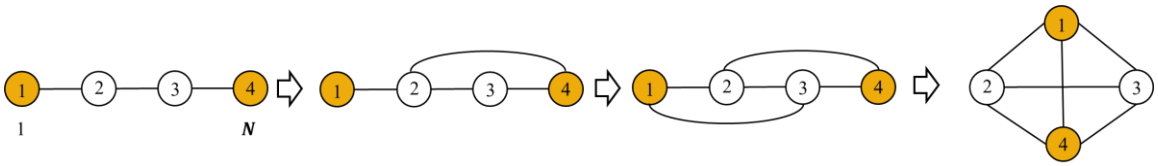


Figure 10. The Construction Process of Step-Min Network Family with 4 Nodes

Algorithm 1 is the pseudo code of this Step-Min network model. A Matlab program was written accordingly to generate Step-Min network families and calculate the $ER_{1,N}$ value of each network. Figure 11 is the $ER_{1,N}$ value history along link addition.

Algorithm 1. Step-Min Network Model

---

1. INPUT: $V \leftarrow N$ // Network node number (network family index)
2. $G_1 \leftarrow$ An undirected line network with N nodes. Index the two ends of this network as 1 and N separately.
3. $|E^*| = N - 1; G^* = G_1;\ \Delta ER_e^* \leftarrow$ INFINITY; $E^* \leftarrow E_{G^*};\ step \leftarrow 1$
4. WHILE $|E^*| \neq \frac{N(N-1)}{2}$ DO
5. FOR $a \leftarrow 1$ to $(N-1)$ DO
6. FOR $b \leftarrow a$ to N DO
7. IF $a, b \notin E^*$ THEN
8. IF $(\Delta ER_{a,b} < \Delta ER_e^*)$ THEN
9. $\Delta ER_e^* \leftarrow \Delta ER_{a,b};\ E^* \leftarrow a, b$
10. END IF
11. END IF
12. END FOR
13. END FOR
14. $|E^*| \leftarrow |E^*| + 1;\ G^* \leftarrow G^* \cup e^*;\ \Delta ER_e^* \leftarrow$ NFINITY; $E^* \leftarrow E_{G^*}; step \leftarrow step + 1$ // This is to connect Node Pair a, b through an undirected link.
15. RETURN $G_{step} \leftarrow G^*$
16. END WHILE

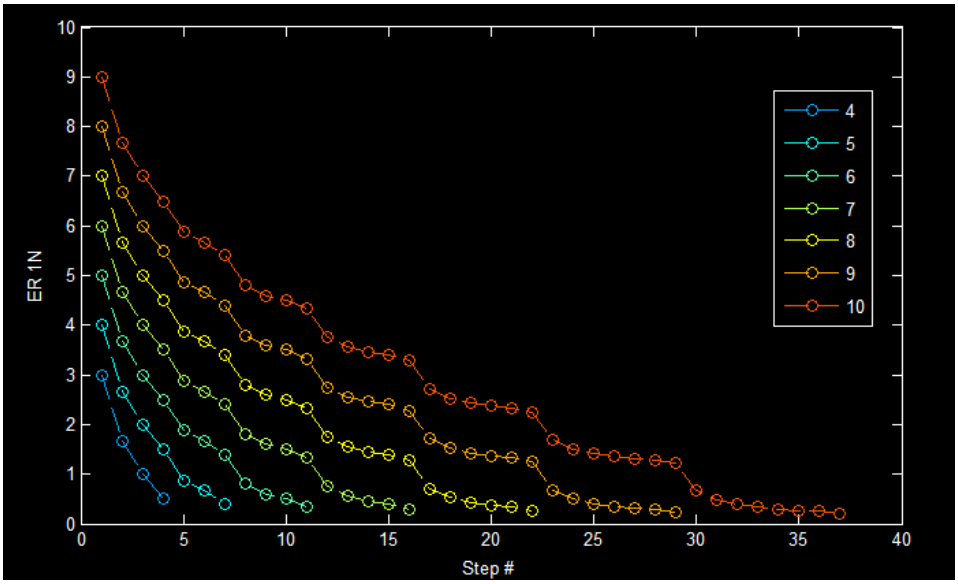


Figure 11. $ER_{1,N}$ History along Link Addition (Step-Min)

As can be seen in Figure 11, the networks with the same node number comprise a network family. Moreover, the $ER_{1,N}$ trend lines of different Step-Min network families have very similar behaviors.

Well established classical networks were also used to test H^{1a}, H^{1b}. To generate networks, two famous synthetic network models were used. One is the Barabasi-Albert (BA) scale free (SF) network model proposed in [69], and the other one is the Erdos-Renyi (ER) random (Rand) network model proposed in [70]. In the following discussions, for each of the two network models, two undirected networks were generated, one with 30 nodes and 60 links and the other one with 50 nodes and 100 links.

BA SF network model begins with an initially connected network of N_0 nodes and this network is called the base network. New nodes are added to the network one at a time. Each new node is connected to $0 \leq n \leq N_0$ existing nodes with a probability that is proportional to the number of links that the existing nodes already have. Denote the probability for each existing node to be chosen at each step as p_i. p_i can be calculated

through the following equation. Continuously adding nodes until the desired network node number N is achieved.

$$p_i = \frac{\sum_{k=1}^{N} a_{i,k}}{\sum_{i=1}^{N}\sum_{k=1}^{N} a_{i,k}} \quad \textbf{9}$$

ER Rand network model begins with N network nodes. The probability for a node pair to be connected (p) is the same and independent from each other. Using this probability, randomly select M unique node pairs to add links, where M is the designated network link number.

Next, a discrete-time link failure simulation model was constructed to obtain $\bar{m}_{i,j}^{X}$ and $\frac{\bar{m}_{i,j}^{X}}{M}$. Given a network, choose a node pair of interest, and denote this node pair as i, j.

First, apply a filter on a network to filter out all the redundant links for the connection between node pair i, j. A redundant link does not contribute to the connection between node pair i, j. In the example shown in Figure 12, the node pair of interest is 1,4 and link 3,5 is a redundant link for the connection between node pair 1,4. The reason to add this filter is that we want to study the relationship between $\bar{m}_{i,j}^{X}$ / $\frac{\bar{m}_{i,j}^{X}}{M}$ and $ER_{i,j}$. The existence of redundant links will not affect the value of $ER_{i,j}$, however, they will inflate the value of $\bar{m}_{i,j}^{X}$ obtained from simulation. If this filter is not applied, the simulation result will be inflated and will not correspond to $ER_{i,j}$.

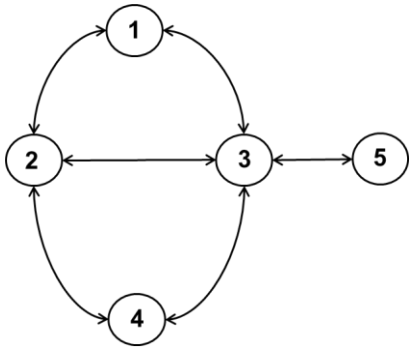


Figure 12. Example of a Pure Redundant Link (Link 3,5)

In the following discussion, the term "link failures" actually refers to structural link failures. "Redundant link failures" and "total link failures" are used to distinguish redundant and the overall link failures (both structural and redundant link failures) apart from structural link failures.

This filter can be turned off. When it is off, the simulation result obtained is the number of total link failures until node pair i, j disconnects. To implement the filter, temporarily disconnect network nodes one at a time to see if $ER_{i,j}$ increases. To temporarily disconnect a node n is to temporarily set all the entries in the n^{th} row and the n^{th} column of the network adjacency matrix to 0. Now the network nodes are separated into two groups: one whose removal results in $ER_{i,j}$ increase (Group 1), and one whose removal does not affect $ER_{i,j}$ (Group 2). The links with one or two end nodes within Group 2 are marked as redundant links and will be filtered out when the redundant link filter is on. Algorithm 2 is the pseudo code of this filter.

Algorithm 2. Network Redundant Link Filter

---

1. INPUT: $G = (V, E)$; Node Pair of Interest: i, j
2. $A^* \leftarrow A$; $ER^*_{i,j} \leftarrow ER_{i,j}$; $E_r \leftarrow \emptyset$; $E_b \leftarrow \emptyset$; $V_1 \leftarrow \emptyset$; $V_2 \leftarrow \emptyset$
3. Redundant_Flag$\leftarrow$1 // Decide if Filter is On (1) or Off (0)

4. //Classify network nodes
5. FOR $a \leftarrow 1$ to $N \backslash \{i, j\}$ DO
6. FOR $b \leftarrow 1$ to N DO
7. $A_{a,b} \leftarrow 0$; $A_{b,a} \leftarrow 0$
8. END FOR
9. IF $ER_{i,j} > ER_{i,j}^{*}$ THEN
10. $V_1 \leftarrow V_1 \cup a$
11. ELSE
12. $V_2 \leftarrow V_2 \cup a$
13. END IF
14. $A \leftarrow A^{*}$
15. END FOR
16. //Redundant link filter
17. IF Redundant_Flag == 1 THEN
18. E_r: A Collection of Network Links with One or Two End Nodes within V_2
19. END IF
20. RETURN $G \leftarrow (V_1, E \backslash E_r)$

After going through the filter (if the filter is turned off, then nothing will be done to the network), at each (time) step, a link is randomly chosen among the remaining links and is removed from the network. For an undirected network, if $a, b \in E$ is chosen, then both the values of $A_{a,b}$ and $A_{b,a}$ in the adjacency matrix will be set to zero. For a directed network, if $a, b \in E$ is chosen, then only the value of $A_{a,b}$ in the adjacency matrix will be set to zero. After each step, a new network topology can be obtained. Check the connectivity between node i and node j. One easiest way to check their connectivity is to

calculate the shortest distance between node i and node j. Since none of the network under investigation in this thesis has negative weights, Dijkstra's Algorithm was used to calculate the shortest distance for easy implementation and efficiency consideration.

Continuously removing links one at a time until node i and node j are disconnected. Document the number of link failures until i and j are disconnected ($m_{i,j}^X$). Repeat the entire process for several times (10000 is used in this thesis) and obtain the average number of link failures until disconnection $\bar{m}_{i,j}^X$ and the average fraction of link failures until disconnection $\frac{\bar{m}_{i,j}^X}{M}$. The following is the pseudo code for this simulation model and a C^{++} program was written accordingly to realize the model and carry out the simulations.

Algorithm 3. Link Failure Simulation Model

---

1. INPUT: $G = (V, E)$; Node Pair of Interest: i, j; Total Iteration: I
2. $e^* \leftarrow \emptyset$; $m_{i,j}^X \leftarrow \{0\}$; $count \leftarrow 0$; $\sigma_{i,j} \leftarrow 0$
3. Execute Algorithm 2 to obtain a filtered network
4. FOR $index$ = 1 to I DO
5. WHILE $\sigma_{i,j} < INFINITY$ DO
6. $e^* \leftarrow Random(E)$; $E \leftarrow E \backslash e^*$; $count \leftarrow count + 1$
7. END WHILE
8. $m_{i,j}^X[index] = count$
9. END FOR
10. RETURN $m_{i,j}^X$

Simulations were conducted on the previous generated Step-Min networks within each network family ($N = 4, \ldots 10$) with 10000 iterations per network. The results are summarized in Figure 13, Figure 14, Figure 15 and Figure 16.

In Figure 13, each data point is from the analysis of one network, where for each network, the two end nodes, $1, N$ form the node pair of interest. The blue data is $\bar{m}_{1,N}^{X}$ obtained from simulation, and it is being compared to calculated values of $\frac{N}{ER_{1,N}}$ (green) and $\frac{1}{ER_{1,N}}$ (red). In Figure 13, the data is sorted along the X-axis, first by number of nodes, and then by $\bar{m}_{1,N}^{X}$. Both $\frac{1}{ER_{1,N}}$ and $\frac{N}{ER_{1,N}}$ trend well with $\bar{m}_{1,N}^{X}$ within each Step-Min network family, suggesting that either could be used to measure connectivity robustness (in terms of $\bar{m}_{1,N}^{X}$) of node pairs connected via the same number of nodes.

However if the data is only sorted by $\bar{m}_{1,N}^{X}$ as shown in Figure 14, then the relationship is not nearly as clear.

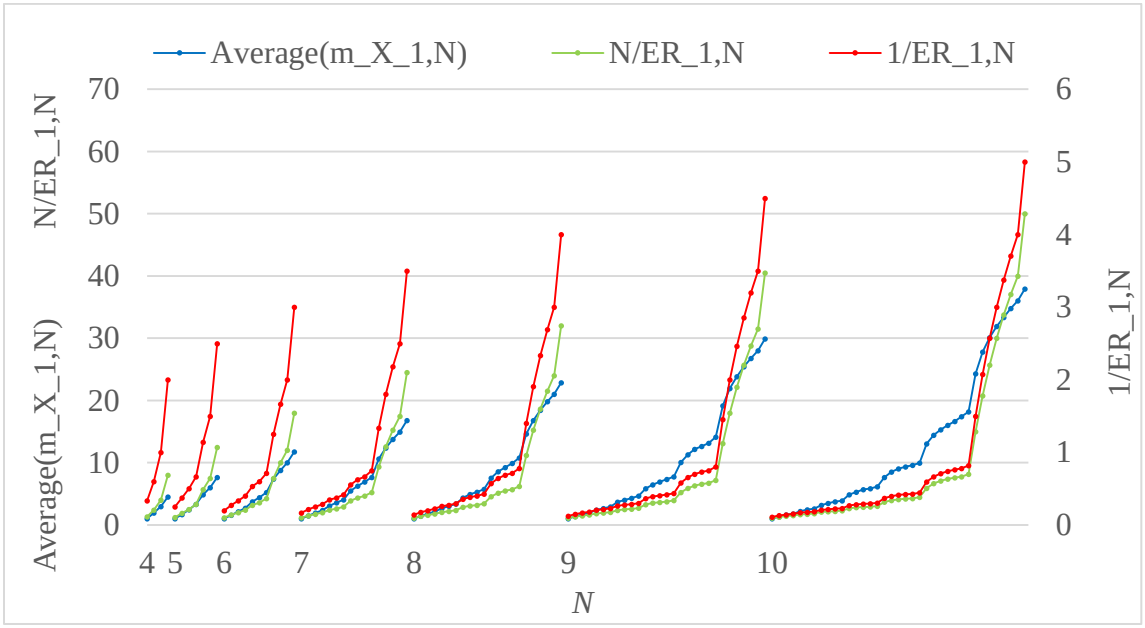


Figure 13. H^{1a} Test Results: Step-Min Network Families Sorted by N, then by $\bar{m}_{1,N}^{X}$

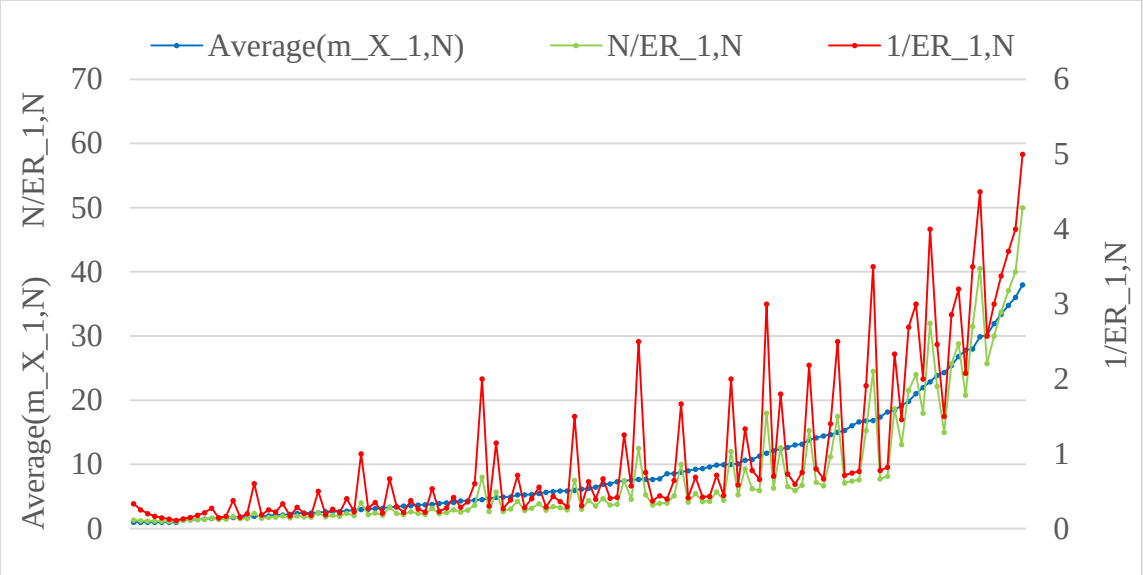


Figure 14. H^{1a} Test Results: Step-Min Network Families Sorted by $\bar{m}_{1,N}^{X}$

In Figure 15, again each data point is from the analysis of one network, where for each network, the two end nodes, $1, N$ for the node pair of interest. The blue data now is $\frac{\bar{m}_{1,N}^{X}}{M}$ obtained from simulation, and it is being compared to calculated values of $\frac{N}{ER_{1,N}}$ (green) and $\frac{1}{ER_{1,N}}$ (red). The data is sorted along the X-axis, first by the number of nodes, and then by $\frac{\bar{m}_{1,N}^{X}}{M}$. Both $\frac{1}{ER_{1,N}}$ and $\frac{N}{ER_{1,N}}$ trend well with $\frac{\bar{m}_{1,N}^{X}}{M}$, suggesting that either could be used to measure connectivity robustness (in terms of $\frac{\bar{m}_{1,N}^{X}}{M}$) of node pairs connected via the same number of nodes. However if the data is only sorted by $\frac{\bar{m}_{1,N}^{X}}{M}$ as shown in Figure 16, it seems only $\frac{1}{ER_{1,N}}$ can capture the trend of $\frac{\bar{m}_{1,N}^{X}}{M}$.

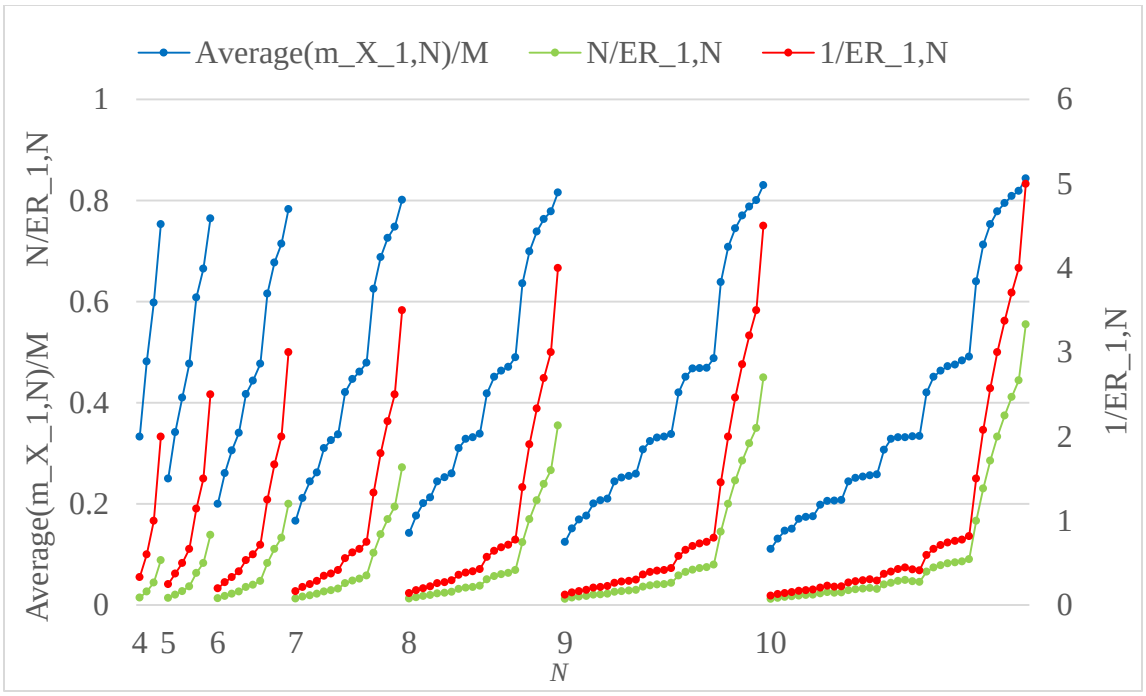


Figure 15. H^{1b} Test Results: Step-Min Network Families Sorted by N, then by $\frac{\bar{m}^{X}_{1,N}}{M}$

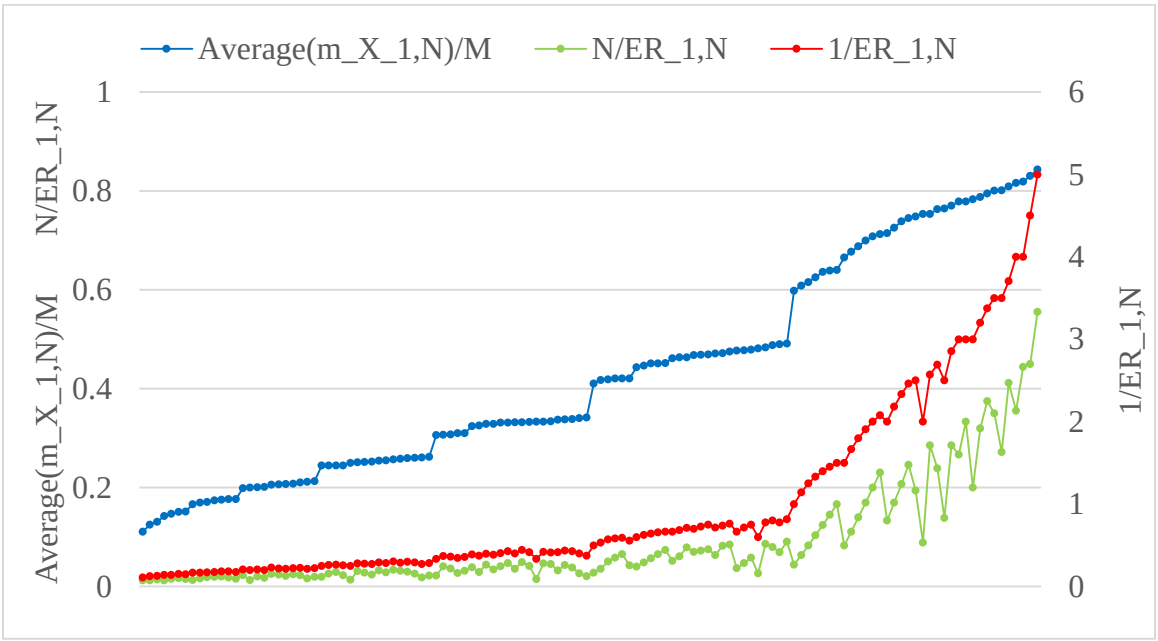


Figure 16. H^{1b} Test Results: Step-Min Network Families Sorted by $\frac{\bar{m}^{X}_{1,N}}{M}$

Comparing the above observation results, it seems only $1/ER_{i,j}$ (for Step-Min network families, $i, j = 1, N$) can be used to measure the connectivity robustness of two arbitrary node pairs in terms of $\frac{\bar{m}_{1,N}^{X}}{M}$.

Similar observations and conclusions can be made for the four classical networks as shown in Figure 17 and Figure 18. Since the classical networks do not come with pre-defined node pairs of interest, 20 node pairs were selected randomly for each network.

In Figure 17 and Figure 18, each data point is from the analysis of one node pair. In Figure 17, $\bar{m}_{i,j}^{X}$ is being compared to $\frac{N}{ER_{i,j}}$ and $\frac{1}{ER_{i,j}}$ and the data is only sorted by $\bar{m}_{i,j}^{X}$. In Figure 18, $\frac{\bar{m}_{i,j}^{X}}{M}$ is being compared to $\frac{N}{ER_{i,j}}$ and $\frac{1}{ER_{i,j}}$ and the data is only sorted by $\frac{\bar{m}_{i,j}^{X}}{M}$. Again, it seems that only $\frac{1}{ER_{i,j}}$ can be used to compare the connectivity robustness of two arbitrary node pairs in terms of $\frac{\bar{m}_{i,j}^{X}}{M}$.

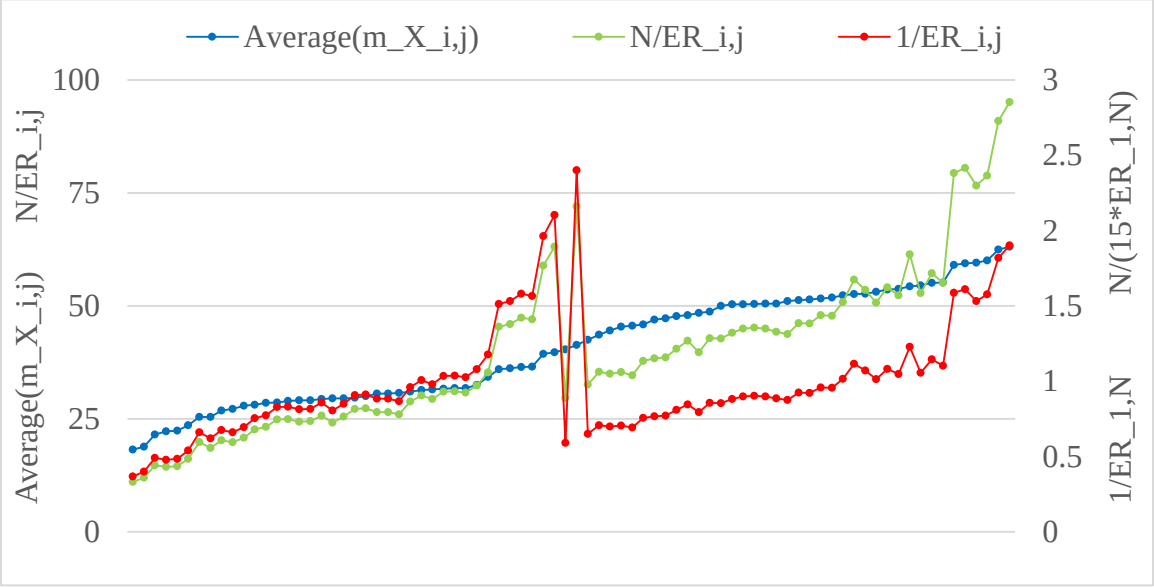


Figure 17. H^{1a} Test Results: Classical Networks Sorted by $\bar{m}_{i,j}^{X}$

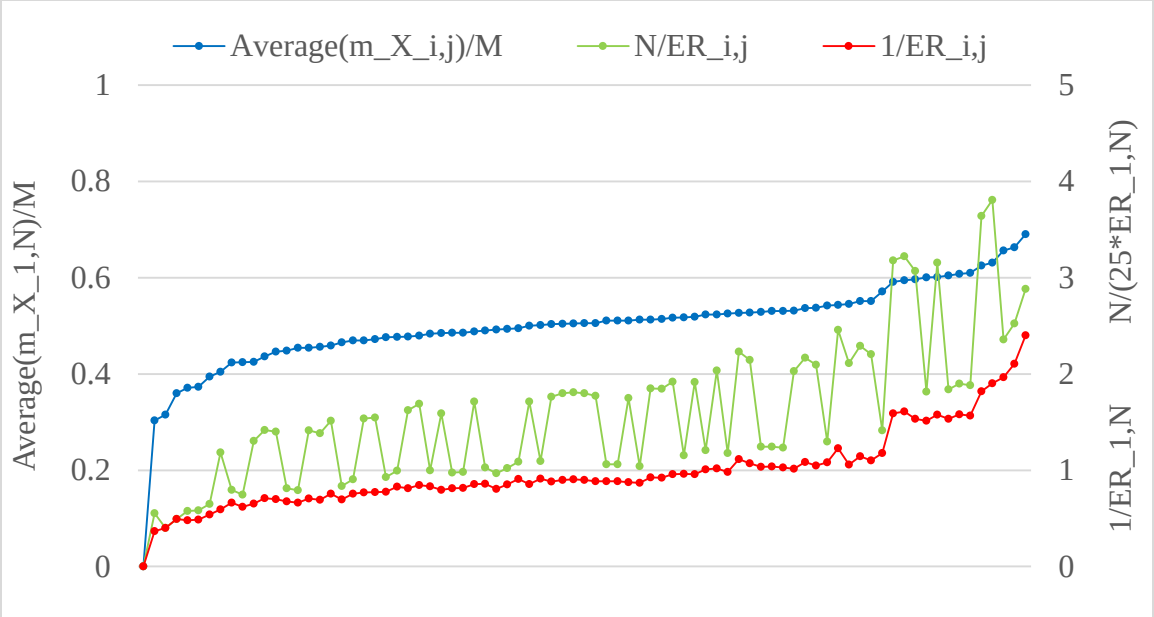


Figure 18. H^{1b} Test Results: Classical Networks Sorted by $\frac{\bar{m}_{i,j}^{X}}{M}$

The above observations suggest rejecting H_0^{1a} and failing to reject H_0^{1b}. Hence it can be concluded that when to compare the connectivity robustness of two arbitrary node pairs in terms of $\frac{\bar{m}_{i,j}^{X}}{M}$, $\frac{1}{ER_{i,j}}$ should be used.

3.3 *Estimating $\bar{m}_{i,j}^{X}$ from ER_{ij}*

As discussed earlier, connectivity robustness can be measured either in terms of the average fraction of link failures ($\frac{\bar{m}_{i,j}^{X}}{M}$) or in terms of the average number of link failures ($\bar{m}_{i,j}^{X}$). Comparing to $\frac{\bar{m}_{i,j}^{X}}{M}$, $\bar{m}_{i,j}^{X}$ is a more straight forward characterization of connectivity robustness. To be able to compare the connectivity robustness of two arbitrary node pairs in terms of $\bar{m}_{i,j}^{X}$, a way to estimated $\bar{m}_{i,j}^{X}$ from $ER_{i,j}$ is needed.

The first step in estimating $\bar{m}_{i,j}^{X}$ involved going back to the results of the Step-Min network families, to find the transformation that would best linearize the relationship

between $\frac{1}{ER_{1,N}}$ and $\frac{\bar{m}_{1,N}^{X}}{M}$ for a given number of nodes. The transformation is summarized below in Equation 10 and Equation 11.

$$\begin{cases} \left(\frac{1}{ER_{i,j}}\right)' = log\left(\frac{1}{ER_{i,j}}\right) & if\ ER_{ij} > 1 \\ \left(\frac{1}{ER_{i,j}}\right)' = log\left(\frac{1}{ER_{i,j}^{1/\theta}}\right) & if\ ER_{ij} \leq 1 \end{cases} \quad \textbf{10}$$

where,

θ is a function of network node number.

$$\left(\frac{\bar{m}_{i,j}^{X}}{M}\right)' = log\left(\frac{\bar{m}_{i,j}^{X}}{M}\right) \quad \textbf{11}$$

To find the equation for θ, the following optimization problem was formulated. The reason for choosing the following formulation will be discussed later.

$$Min: Z = \sum_{n} \left(\tilde{\bar{m}}_{i,j}^{X}\right)_{N_n} - \left(\bar{m}_{i,j}^{X}\right)_{N_n}$$

$$n \in \{\text{Networks within Step} - \text{Min Network Family } N\}$$

where,

$$\left(\tilde{\bar{m}}_{i,j}^{X}\right)_{x} = M_x * 10^{\left[\left(\left(\frac{1}{ER_{i,j}}\right)'_{x} - \left(\frac{1}{ER_{i,j}}\right)'_{LINE}\right)\frac{\left(\frac{\bar{m}_{i,j}^{X}}{M}\right)'_{FULL} - \left(\frac{\bar{m}_{i,j}^{X}}{M}\right)'_{LINE}}{\left(\frac{1}{ER_{i,j}}\right)'_{FULL} - \left(\frac{1}{ER_{i,j}}\right)'_{LINE}} + \left(\frac{\bar{m}_{i,j}^{X}}{M}\right)'_{LINE}\right]}$$

$$\begin{cases} \left(\frac{1}{ER_{i,j}}\right)' = log\left(\frac{1}{ER_{i,j}}\right) & if\ ER_{i,j} > 1 \\ \left(\frac{1}{ER_{i,j}}\right)' = log\left(\frac{1}{ER_{i,j}^{1/\theta}}\right) & if\ ER_{i,j} \leq 1 \end{cases}$$

$$\left(\frac{\bar{m}_{i,j}^{X}}{M}\right)' = log\left(\frac{\bar{m}_{i,j}^{X}}{M}\right)$$

This optimization problem was solved through Bisection Search method and the results were rounded to the second decimal place, which gives the following table from $N = 3$ to $N = 30$.

Table 2. θ Value of Different N

N	3	4	5	6	7	8	9	10	11	12	13	14	15	16
θ	1.00	1.72	2.34	2.86	3.27	3.68	3.73	3.84	3.95	3.99	4.10	4.13	4.24	4.25
N	17	18	19	20	21	22	23	24	25	26	27	28	29	30
θ	4.29	4.40	4.41	4.45	4.49	4.54	4.56	4.67	4.69	4.70	4.81	4.82	4.83	4.88

According to Table 2, it seems that as N getting bigger, the value of θ becomes more stabilized as shown in Figure 19.

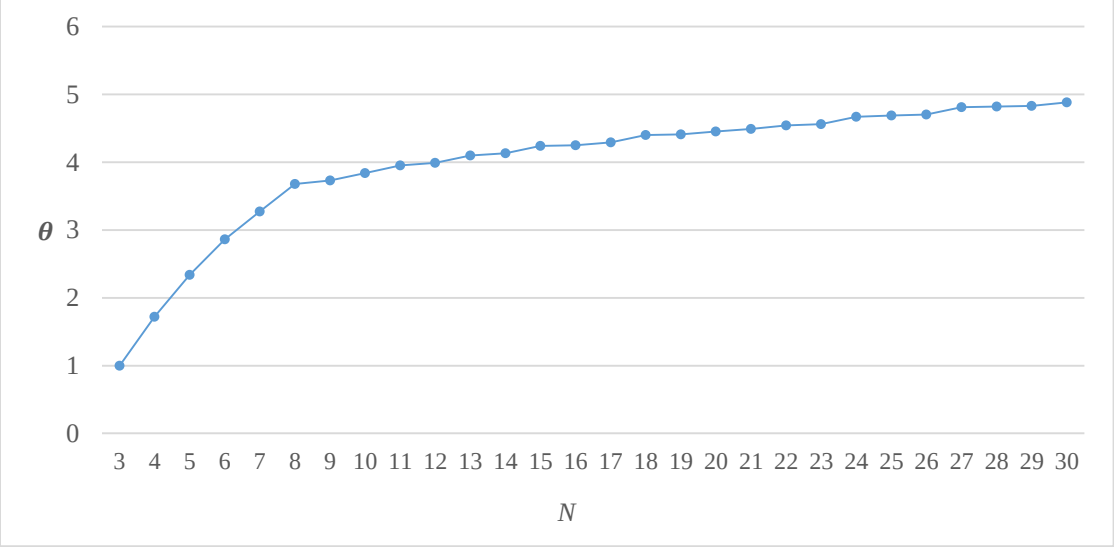


Figure 19. The Behavior of θ over Different N Values

The optimization to obtain θ relays on knowing all the networks within a Step-Min network family of node number N. As N grows bigger, it becomes more time consuming to construct the entire Step-Min network family. As already known, different Step-Min network families share similar characteristics since they are generated by the

same synthetic network mode. Observing the trend of θ as N increases, it seems that the value of e^{θ} is a linear function of N. Through the 28 data pairs shown in Table 2, the following fitting equation can be obtained.

$$e^{\tilde{\theta}} = 4.588 * N - 4.699 \quad \textbf{12}$$

```
lm(formula = exp_theta ~ N)

Residuals:
   Min      1Q  Median      3Q     Max
-8.0687 -2.6945 -0.2667  3.5992  7.6413

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -4.6986    1.9437     -2.417   0.023 *
N            4.5880    0.1058     43.363   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.522 on 26 degrees of freedom
Multiple R-squared:  0.9864,     Adjusted R-squared:  0.9858
F-statistic:  1880 on 1 and 26 DF,  p-value: < 2.2e-16
```

According to the fitting summary, it seems that Equation 12 fits the data very well. To further check the validity of this model (Equation 12) for extrapolation, the prediction (estimation) result of this model for $N = 50$ is compared with its actual optimization result as shown in Table 3. Since the error is only 1.81%, Equation 12 is used in this thesis for obtaining a close estimation of the θ value of a fully connected network with $N \geq 30$ nodes.

Table 3. Estimation Accuracy Summary of Equation 12 for $N = 50$

$\tilde{\theta}$	θ	Error in %
5.41	5.51	1.81%

Figure 10 is an example plot for the linearized data of the Step-Min network family with $N = 10$ using the proposed linearization method. In Figure 10, each data

point is the result of a network, where for each network, the two end nodes, $1, N$ form the node pair of interest.

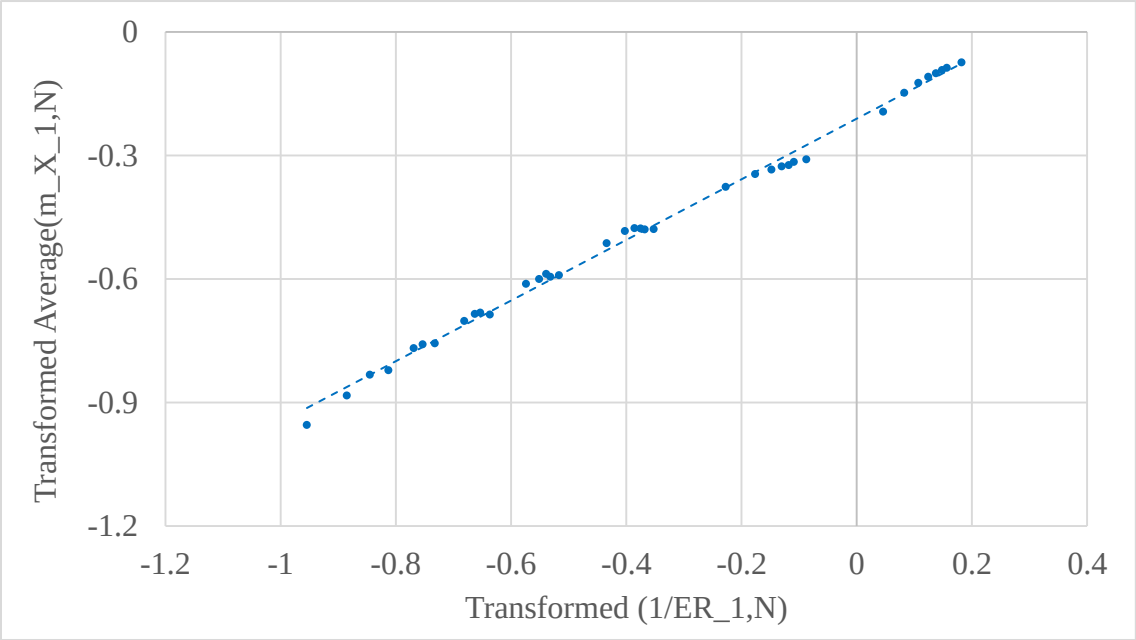


Figure 20. Example of the Linearized Data for the Step-Min Network Family with 10 Nodes

Next, the bounds on $\frac{1}{ER_{i,j}}$ and $\frac{\bar{m}_{i,j}^{X}}{M}$ can be directly calculated given a network with N nodes and M links through the following equation. It is important that these bounds can be calculated without any simulation, as the goal is to have a topological-based method that does not involve any costly simulation.

$$\frac{1}{ER_{i,j}} \in [\frac{1}{N-1}, \frac{N}{2}] \quad \textbf{13}$$

where,

$\frac{1}{N-1}$ corresponds to a line network with i, j as two ending nodes;

$\frac{N}{2}$ corresponds to a fully connected network.

$$\frac{\bar{m}_{i,j}^{X}}{M} \in \left[\frac{1}{M}, \frac{\left(\bar{m}_{i,j}^{X}\right)_{FULL}}{M}\right] \quad \textbf{14}$$

where,

$\frac{1}{M}$ corresponds to a line network with i, j as two ending nodes;

$\frac{\left(\bar{m}_{i,j}^{X}\right)_{FULL}}{M}$ corresponds to a fully connected network and $\left(\bar{m}_{i,j}^{X}\right)_{FULL}$ is a function of network node number.

The value of $\left(\bar{m}_{i,j}^{X}\right)_{FULL}$ can be obtained by feeding a fully connected network into the aforementioned link failure simulation mode. The simulation results obtained are summarized in Table 4 for $N = 3$ to N=30.

Table 4. $\left(\bar{m}_{i,j}^{X}\right)_{FULL}$ from $N = 3$ to $N = 30$

N	3	4	5	6	7	8	9
$\left(\bar{m}_{i,j}^{X}\right)_{FULL}$	2.34	4.52	7.65	11.75	16.83	22.85	29.90
N	10	11	12	13	14	15	16
$\left(\bar{m}_{i,j}^{X}\right)_{FULL}$	37.96	46.90	56.96	67.97	80.03	93.06	107.15
N	17	18	19	20	21	22	23
$\left(\bar{m}_{i,j}^{X}\right)_{FULL}$	122.07	138.12	155.03	173.15	192.01	212.20	233.16
N	24	25	26	27	28	29	30
$\left(\bar{m}_{i,j}^{X}\right)_{FULL}$	255.04	277.97	301.98	327.08	352.93	380.13	407.97

Although theoretically using simulation, it is possible to obtain the $\left(\bar{m}_{i,j}^{X}\right)_{FULL}$ value a fully connected network (undirected) with any node number N. the running time for such simulation exponentially increases as N becomes larger. It would

be very beneficial if a close estimation for $\left(\bar{m}_{i,j}^{X}\right)_{FULL}$ can be obtained from a simple equation.

The set of all fully connected networks can be viewed as a series of networks generated by a single synthetic network model. The model is, given the node number N, connecting each node with all the other nodes within the network except itself. As mentioned earlier, the networks generated by the same synthetic network model have similar characteristics. Therefore, all the fully connected networks should have similar characteristics. Plot the data summarized in Table 4 in Figure 21. The Y-axis of Figure 21 is for the $\left(\bar{m}_{i,j}^{X}\right)_{FULL}$ value and the X-axis of Figure 21 is for network node number. Figure 21 indicates that the values of $\left(\bar{m}_{i,j}^{X}\right)_{FULL}$ is a polynomial function of N.

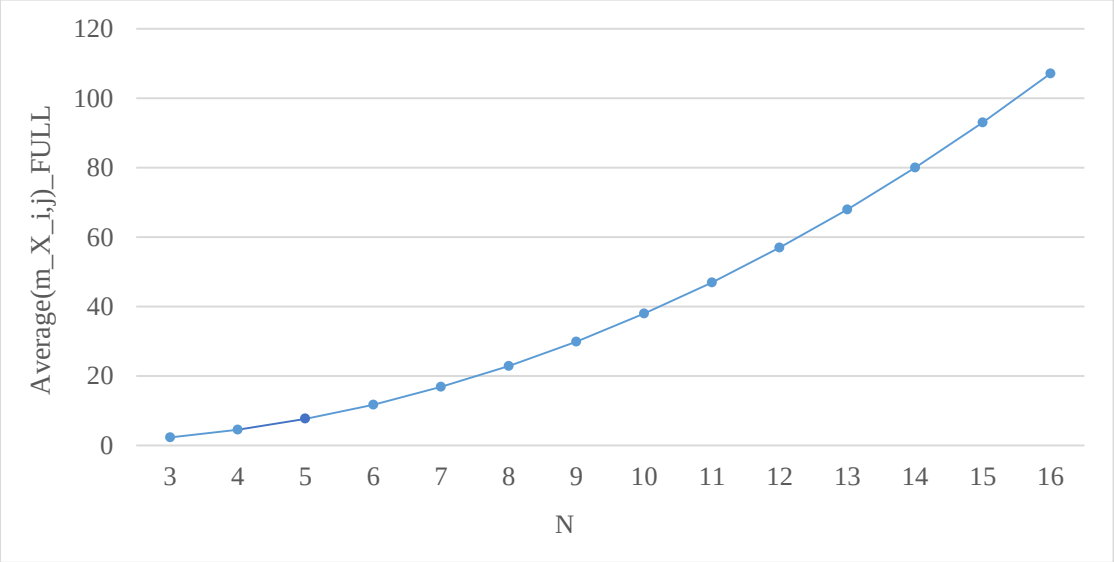


Figure 21. Behavior of $\left(\bar{m}_{i,j}^{X}\right)_{FULL}$ from $N = 3$ to $N = 16$

Fitting a second order linear regression model using those data, the following fitting equation was obtained using R.

$$\left(\bar{m}_{i,j}^{X}\right)_{FULL} = 0.4979395 * N^2 - 1.415395 * N + 2.2533249 \quad \textbf{15}$$

The fitting summary is shown below, based on which we can conclude the model fits the data very well. To further check the goodness-of-fit for N values that are greater than 30, $N = 50$ is fed into Equation 15 to obtain the value of $\left(\bar{m}_{i,j}^{X}\right)_{FULL}$ of a fully connected network with 50 nodes. The result is compared to the actual value of $\left(\bar{m}_{i,j}^{X}\right)_{FULL}$ obtained from simulation and summarized in Table 5. Since the error is only 0.16%, Equation 15 is used in this thesis for obtaining a close estimation of the $\left(\bar{m}_{i,j}^{X}\right)_{FULL}$ value of a fully connected network with $N > 30$ nodes.

```
Residuals:
lm(formula = mdis ~ N_2 + N)

Residuals:
     Min       1Q   Median       3Q      Max
-0.151395 -0.043599  0.006139  0.051579  0.152053

Coefficients:
             Estimate   Std. Error   t value   Pr(>|t|)
(Intercept)  2.2533249  0.0586286    38.43     <2e-16 ***
N_2          0.4979395  0.0002386    2087.29   <2e-16 ***
N           -1.4153950  0.0080582   -175.65    <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.07353 on 25 degrees of freedom
Multiple R-squared:      0.97,       Adjusted R-squared:      0.99
F-statistic: 4.028e+07 on 2 and 25 DF,  p-value: < 2.2e-16
F-statistic: 4.813e+06 on 2 and 27 DF,  p-value: < 2.2e-16
```

Table 5. Estimation Accuracy of Equation 15 for $N = 50$

$\left(\tilde{\bar{m}}_{1,N}^{X}\right)_{FULL}$	$\left(\bar{m}_{1,N}^{X}\right)_{FULL}$	Error %
1176.33	1178.25	0.16%

With the data linearized and two bonding points available, linear interpolation can be used to estimate $\frac{\bar{m}_{i,j}^{X}}{M}$ using the $ER_{i,j}$ value of a node pair. Multiplying the estimated

$\frac{\bar{m}_{i,j}^{X}}{M}$ value by the number of links (M), an estimation for $\bar{m}_{i,j}^{X}$ can be obtained. The estimation equation is shown below.

$$\left(\widetilde{\bar{m}}_{i,j}^{X}\right)_{x} = M_x * 10^{\left[\left(\left(\frac{1}{ER_{i,j}}\right)'_{x} - \left(\frac{1}{ER_{i,j}}\right)'_{LINE}\right)\frac{\left(\frac{\bar{m}_{i,j}^{X}}{M}\right)'_{FULL} - \left(\frac{\bar{m}_{i,j}^{X}}{M}\right)'_{LINE}}{\left(\frac{1}{ER_{i,j}}\right)'_{FULL} - \left(\frac{1}{ER_{i,j}}\right)'_{LINE}} + \left(\frac{\bar{m}_{i,j}^{X}}{M}\right)'_{LINE}\right]}$$

$$\begin{cases} \left(\frac{1}{ER_{i,j}}\right)' = log\left(\frac{1}{ER_{i,j}}\right) & if\ ER_{ij} > 1 \\ \left(\frac{1}{ER_{i,j}}\right)' = log\left(\frac{1}{ER_{i,j}^{1/\theta}}\right) & if\ ER_{ij} \leq 1 \end{cases} \qquad \textbf{16}$$

$$\left(\frac{\bar{m}_{i,j}^{X}}{M}\right)' = log\left(\frac{\bar{m}_{i,j}^{X}}{M}\right)$$

With Equation 16, the physical meaning of this optimization function is to find out the θ value that minimize the total squared error of $\widetilde{\bar{m}}_{i,j}^{X}$ summed over all networks within a Step-Min network family of N nodes.

The estimation results for $\bar{m}_{1,N}^{X}$ via Equation 16 for Step-Min network families from $N = 4$ to $N = 10$ are shown in Figure 22. The estimation errors for Step-Min network families from $N = 4$ to $N = 10$ and $N = 30, 50$ are summarized in Table 6.

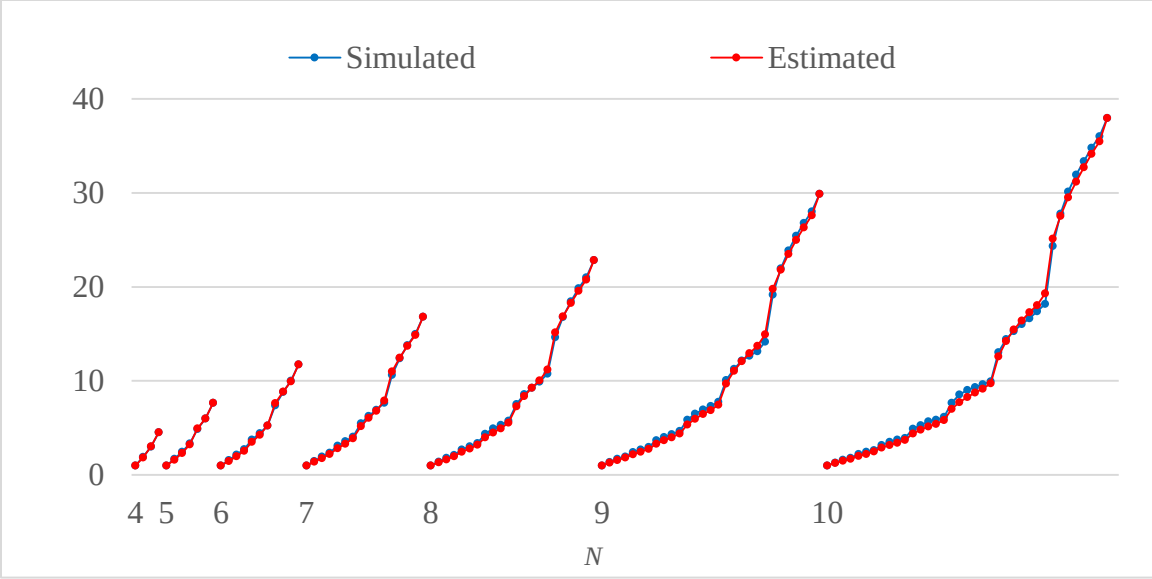


Figure 22. Estimation Results vs. Simulation Results for Step-Min Network Families

Table 6. Estimation Errors of Step-Min Network Families

N	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**30**	**50**
Average Error	1.5%	2.3%	3.0%	3.8%	4.1%	4.8%	5.2%	7.9%	9.2%

As can be observed from Figure 22 and Table 6, the proposed method can provide very good estimation for the $\bar{m}_{1,N}^{X}$ value of the node pair of interest within each Step-Min network.

The estimation results for node pairs within the classical networks are summarized in Figure 23 and Table 7. Again, the proposed method can provide close estimation for the $\bar{m}_{1,N}^{X}$ values of the node pairs within each classical network.

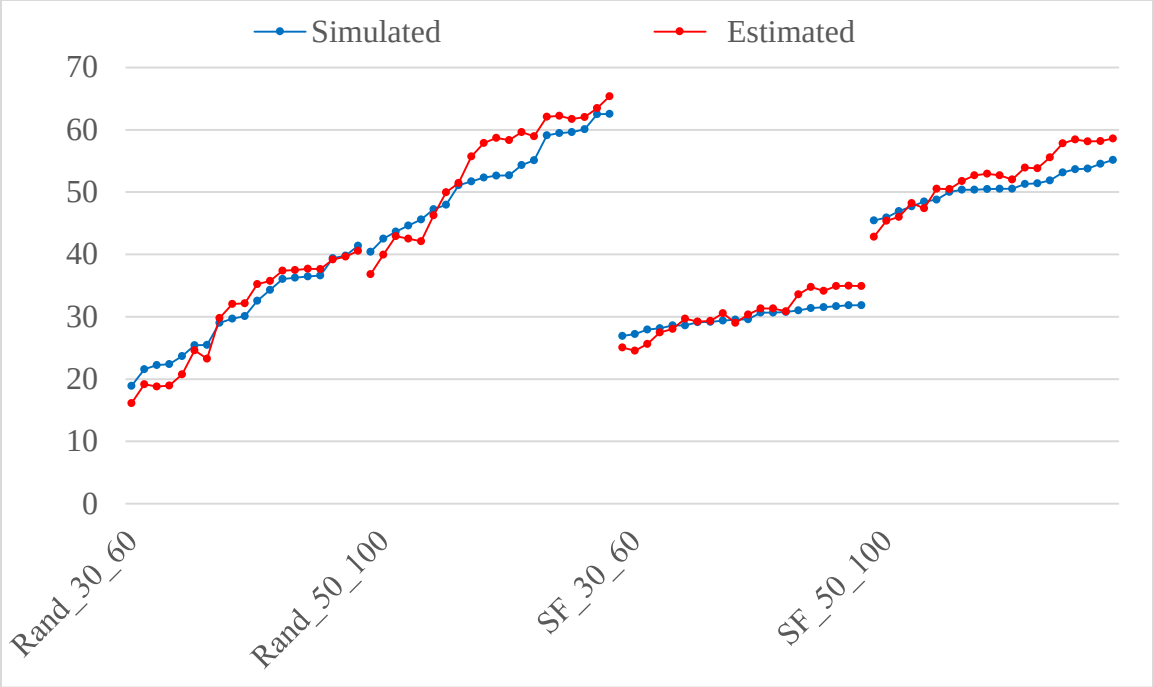


Figure 23. Estimation Results vs. Simulation Results for Node Pairs within Classical Networks

Table 7. Estimation Errors of Classical Networks

Network	Rand_30_60	Rand_50_100	SF_30_60	SF_50_100
Average Error	7.4%	4.4%	5.9%	3.4%

3.4 *Redundant Links*

Unlike structural links, the effects of redundant links on the number of link failures before node pair i, j disconnected against link failures are null or zero since redundant links do not contribute to the connection between node pair i, j and hence their existence or removal does not affect the value of $ER_{i,j}$. That is to say, redundant links do not affect the structural connectivity robustness between node pair i, j against link failures. However, this does not mean the existence of redundant links is useless. Under random link attacks, redundant links can server as "camouflage" and attract attacks away from structural links. This decreases the probability of structural links to be hit during

random attacks and as a result protects the network structure. Is it possible to quantify the effects of redundant links under random link attacks?

Group the links within a network G into two sets as shown below in Figure 24. The links in Set 1 are structural links (E_1), and the links in Set 2 are redundant links (E_2). Assume there are in total M links with M_1 in Set 1 and M_2 in Set 2.

$$G = (V, E)$$
$$E = E_1 \cup E_2$$
$$E_1 \cap E_2 = \emptyset$$

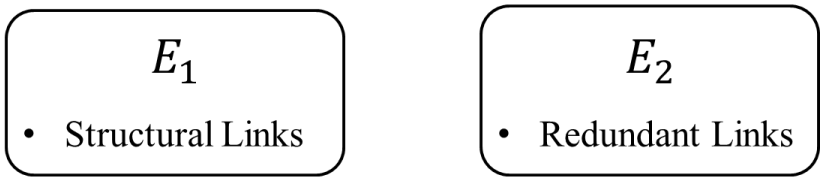


Figure 24. Partition of *E*

Next assuming remove $\left(m_{i,j}^X\right)_{E_1}$ links from Set 1 will result in the disconnection between node pair i, j. Apparently $\left(m_{i,j}^X\right)_{E_1}$ is a random variable based on the previous discussion. For a given network and a given node pair i, j, the sample space of $\left(m_{i,j}^X\right)_{E_1}$ is countable and limited. Assuming in total there are γ unique values within the sample space of $\left(m_{i,j}^X\right)_{E_1}$. Rearrange its elements in the following fashion: $\left(m_{i,j}^X\right)_{E_1}^1 < \left(m_{i,j}^X\right)_{E_1}^2 < \cdots \left(m_{i,j}^X\right)_{E_1}^\gamma$. Use $\left(m_{i,j}^X\right)_E$ to denote the total number of link failures from Set 1 and Set 2 that will result in the disconnection between node pair i, j. $\left(m_{i,j}^X\right)_E$ is also a random variable and its mean value, $\left(\bar{m}_{i,j}^X\right)_E$ or the ratio between $\left(\bar{m}_{i,j}^X\right)_E$ and $\left(\bar{m}_{i,j}^X\right)_{E_1}$ is what we want to estimate here. Mathematically, we have

$$\left(\bar{m}_{i,j}^{X}\right)_{E} = E\left[\left(m_{i,j}^{X}\right)_{E}\right] = E\left[E\left[\left(m_{i,j}^{X}\right)_{E}\middle|\left(m_{i,j}^{X}\right)_{E_1}^{l}\right]\right] \quad \textbf{17}$$

where $l = 1,2,\dots,\gamma$

Using indicator variables, we have

$$E\left[\left(m_{i,j}^{X}\right)_{E}\middle|\left(m_{i,j}^{X}\right)_{E_1}^{l}\right] = \left(m_{i,j}^{X}\right)_{E_1}^{l} + \frac{M_2 * \left(m_{i,j}^{X}\right)_{E_1}^{l}}{M_1 + 1} \quad \textbf{18}$$

where $l = 1,2,\dots,\gamma$

Then the following equation can be obtained for $\left(\bar{m}_{i,j}^{X}\right)_{E}$

$$\begin{aligned}\left(\bar{m}_{i,j}^{X}\right)_{E} &= E\left[\left(m_{i,j}^{X}\right)_{E}\right] \\ &= \sum_{l=1}^{\gamma}\left[\left(m_{i,j}^{X}\right)_{E_1}^{l} + \frac{M_2\left(m_{i,j}^{X}\right)_{E_1}^{l}}{M_1 + 1}\right] * p_{E_1}^{l} \\ &= \sum_{l=1}^{\gamma}\left[\left(m_{i,j}^{X}\right)_{E_1}^{l} * p_{E_1}^{l}\right] + \sum_{l=1}^{\gamma}\left[\frac{M_2\left(m_{i,j}^{X}\right)_{E_1}^{l}}{M_1 + 1} * p_{E_1}^{l}\right] \\ &= \left(\bar{m}_{i,j}^{X}\right)_{E_1} + \frac{M_2\left(\bar{m}_{i,j}^{X}\right)_{E_1}}{M_1 + 1} = \frac{M + 1}{M_1 + 1}\left(\bar{m}_{i,j}^{X}\right)_{E_1}\end{aligned} \quad \textbf{19}$$

Assuming the ratio between M_1 and M_2 is φ ($\varphi = \frac{M_2}{M_1}$), then based on Equation 19, the ratio between $\left(\bar{m}_{i,j}^{X}\right)_{E}$ and $\left(\bar{m}_{i,j}^{X}\right)_{E_1}$ is

$$\phi = \frac{M + 1}{M_1 + 1} = \frac{\varphi M_1}{M_1 + 1} + 1 \quad \textbf{20}$$

In order to check the validity of the expression (Equation 19 or Equation 20), the following was added to the link failure simulation model. At the beginning of a simulation, a redundancy ratio φ is decided. If a filtered network with M_1 structural links is fed into the simulation, then $M_2 = \lfloor \varphi M_1 \rfloor$ dummy links will be added to the original link pool, which forms an augmented link pool. During the simulation, a link is randomly

chosen at a step from the augmented link pool until the key node pair i, j disconnected. If the expression is right, then the ϕ value obtained from Equation 20 should correspond to the values obtained from simulation.

First, simulations were conducted on networks within the Step-Min network family with $N = 30$. For networks within this network family, the key node pair is always 1,30. The following φ values: 1, 0.8, 0.6, 0.4, 0.2, were used. The simulation results are presented Figure 25. The numbers at the lower right corner is the corresponding φ value of each plot. Y-axes are for plotting quantities, and X-axes are networks indices. The networks are ordered in increasing order of link number. Since network index does not matter, they are removed from the plots in Figure 25. The blue lines are the ϕ values obtained from simulation (ϕ_{simu}) and the red lines are the ϕ values calculated through Equation 20 (ϕ_{theo}). As can be seen from Figure 25, the lines of ϕ_{simu} follow the trend of ϕ_{theo} very well especially for networks whose link numbers are large. In order to quantitatively show how well ϕ_{theo} corresponds to ϕ_{simu}, the percentage difference between ϕ_{simu} and ϕ_{theo} were taken using the following equation. The distributions of ε are shown in Figure 26.

$$\varepsilon = \frac{\phi_{theo} - \phi_{simu}}{\phi_{simu}} \quad \textbf{21}$$

Figure 25. Co-plot of ϕ_{simu} and ϕ_{theo} for Step-Min Network Faimly of $N = 30$

Ordered by Link Number

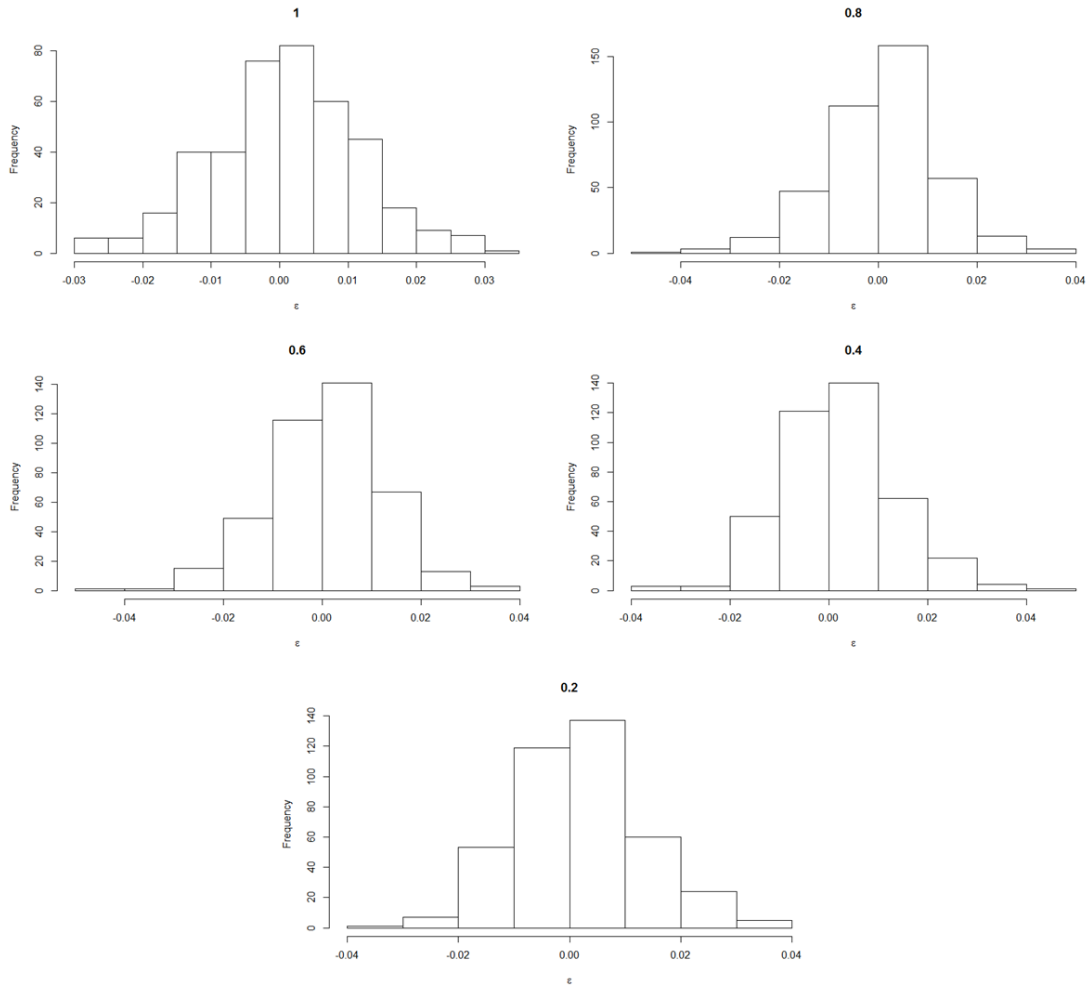


Figure 26. Distribution of ε for Step-Min Network Faimly of $N = 30$

As can be seen from Figure 26, given a φ value, the percentage error ε for networks within the Step-Min network family of $N = 30$ almost symmetrically distributed around 0 with the maximum absolute percentage error smaller or equal to 4%. Most of the data points are around 0 and as the absolute percentage error getting higher, the density becomes smaller. Those observations indicate that, Equation 20 and hence Equation 19 are valid. They can provide direct quantification of the effects of redundant links on the number of link breakdowns until the node pair of interest disconnects under random link attacks.

In order to confirm this conclusion, the above redundant link added link failure simulation was re-conducted on 80 node pairs that randomly selected from the two SF networks and the two Rand networks. For a given φ value, for each node pair, its percentage error ε can be obtained, with which, the average percentage error ε of each network can also be obtained (average over the 20 node pairs of each network). The results are summarized in Table 8. Since the average errors are low for all the four classical networks, it is confirmed that Equation 20 and hence Equation 19 are valid.

Table 8. Average Percentage Error ε of Classical Networks

Network Type	0.2	0.4	0.6	0.8	1.0
Rand_30_60	0.8%	0.9%	0.9%	0.9%	0.8%
Rand_50_100	0.7%	0.8%	0.8%	0.7%	0.8%
SF_30_60	1.0%	1.1%	1.1%	1.1%	1.0%
SF_50_100	3.3%	3.3%	3.4%	3.4%	3.4%

3.5 *Chapter Summary*

In Chapter 1, capability-based connectivity robustness (RCN^{CP}) was defined as the ability of a network to maintain inter-connection among individual entities to support network capability output under network impairments. The general mathematical expression of RCN^{CP} was given in Equation 1.

To reflect the relationship described in Equation 1, a capability-based network modeling process was developed as the answer to the first research question. With a capability-based network model, the problem of measuring the capability-based connectivity robustness of a network can be successfully transformed into the problem of measuring the connectivity robustness of critical node pairs.

In search for the answer to the second research question, a set of requirements on candidate connectivity robustness measures were proposed to help the measure selection process.

1. Quantitative
2. Be applicable to a node pair
3. Be able to capture the connectivity change between a node pair under link failures
4. Accounts for alternative paths between a node pair

Pairwise effective resistance $ER_{i,j}$ was identified as a candidate measure. By testing Hypothesis 1, it was concluded that, $ER_{i,j}$ can be used to compare the connectivity robusntess of two arbitrary node pairs in terms of the average fraction of link failures until disconnection happens ($\frac{\bar{m}_{i,j}^{X}}{M}$). In order to compare the connectivity robustness of two arbitrary node pairs in terms of the average number of link failures until disconnection happens ($\bar{m}_{i,j}^{X}$), Equation 16 was proposed to provide a close estimation for $\bar{m}_{i,j}^{X}$ given the $ER_{i,j}$ value of a node pair.

Finally, the effects of redundant links were discussed. The existence of redundant links does not affect the average number of link failures that a node pair can sustain before disconnection. This is because redundant links do not contribute to the connection between node pair i,j. However, under random link attacks, redundant links can serve as "camouflage" and attract attacks away from structural links. This decreases the probability of structural links to be hit during random attacks and as a result protects the network structure. The effect can be quantified using either Equation 19 or Equation 20. The validity of Equation 19 or Equation 20 was confirmed via simulation.

.

CHAPTER IV

CENTRALITY ANALYSES

In graph theory and network analysis, the centrality of a node or a link (network entity) is a quantitative value representing the importance of a network entity to a network property of interest [73]. The concept of centrality was first developed in social network analysis and now have many other applications, such as to help identify the super-spreaders of disease, the most critical infrastructures in the Internet, the bottlenecks in transportation network. In general, the centrality of a network entity is characterized by its position and/or the connectedness of network entities within networks, and depending on the research content, the centrality of the same network entity can be evaluated differently. In this chapter, the centrality of a network entity is measured by the extent to which it affects the capability-based connectivity robustness (RCN^{CP}) of a given network. In the following discussion, C_k^V will be used to denote the centrality of node $k \in V(G)$ and $C_{k,l}^E$ will be used to denote the centrality of link $k, l \in E(G)$. The contents of this chapter are arranged as following. First, the general equation for the centrality of a network entity in terms of RCN^{CP} will be given followed by a review of existing centrality measures. Next, the centrality measure for network nodes and network links proposed will be discussed followed by some analysis results.

One argument that can be derived from the definition of the centrality of a network entity is the that, the higher the centrality, the higher the impact of the removal of this network entity on the corresponding network quantity of interest [74, 75]. Based on this, the general mathematical expression of the centrality of a network entity in terms of RCN^{CP} can be written as below.

$$C_k^V = \Delta RCN^{CP}(k) \quad \textbf{22}$$

$$C_{k,l}^E = \Delta RCN^{CP}[(k,l)] \quad \textbf{23}$$

Based on the discussion in Chapter 2 and Chapter 3, Equation 22 (for network nodes) or Equation 23 (for network links) can be rewritten as following, where node pair i^*, j^* is the capability critical node pair of the network under study.

$$C_k^V = \Delta \bar{m}_{i^*,j^*}^X(k) \quad \textbf{24}$$

$$C_k^V = \Delta \bar{m}_{i^*,j^*}^X(k,l) \quad \textbf{25}$$

In general, there are two ways to calculate the centrality of a network entity. The first way is to measure it through Equation 24 (for network nodes) or Equation 25 (for network links) directly. This way of quantifying the centrality of a network entity is often referred to as sensitivity analysis or dynamic centrality in the literatures. It is sometimes normalized to the percentage form. The second way is to measure the centrality of a network entity through an indicator directly obtained through network topological analysis and often can reveal more information on the role of a network entity. In this thesis, the second method is used to calculate the centrality of a network entity.

In the literatures, several different centrality measures have been proposed to characterize the role of a network entity in different ways for different analysis purposes [72]. The simplest one is by degree. It is usually referred to as node degree centrality since this measure can only be applied to network nodes. Node degree is a local measure since it is only measured by the number the immediate neighbors of a node and not by, for example, the two-hop and three-hop neighbors of that node. Because of that, it is also referred to as first order/one-hop connectedness index. By increasing the number of hops, second order degree centrality, third order degree centrality and so on can be defined, which however, are used less often comparing to the first order one. Regardless of its order, degree centrality measure usually cannot help determine the overall position or the

connectedness of nodes within a network except for networks that display the so called rich-club connectivity [40, 72, 76, 77].

Another type of centrality is called geodesic closeness [78, 79]. Although the concept can be extended for network links, geodesic closeness is usually defined for network nodes. It is calculated through taking (the reciprocal of) the average geodesic distance from a selected node to all the other nodes in a network. Since information transmission between network nodes is not always through the geodesic path between them, other types of node distance are used to accommodate different information transmission rules and alternative information transmission paths. For example, information centrality proposed in [80], random-walk centrality proposed in [81] and effective resistance based centrality proposed in [72] are based on some all paths between node pairs within a network by using the random-walk path length instead of the geodesic distance between node pairs.

Another class of centrality is called betweenness and is defined based on how a pair of nodes are connected to each other. Betweenness can be defined for both network nodes and network links. In general, it is the number of node pair paths that pass through a node or a link and sometimes is normalized by the total number of node pair paths [82]. The path between a node pair is determined by routing rules, which can be either deterministic or stochastic. The two most commonly used betweenness centrality measures are geodesic path betweenness centrality (deterministic) and random walk betweenness centrality (stochastic). If link weights are considered, they can be modeled as network flows and use flow based centrality measure [82] or simply the weighted version of certain betweenness centrality measures. Betweenness centrality measures have been widely applied to different research fields due to their capability of reflecting the role played by a network entity in the communication between node pairs [72].

Some other types of centrality measures, which are used less often than the aforementioned ones, were proposed in association with certain network measures. In general, the choice of centrality measure should reflect the role of a network entity in affecting the network quantity under study and correspond to the way that quantity is measured. Therefore, in this thesis, the centrality measure used should correspond to the capability-based connectivity robustness measuring process developed in Chapter 3, which in specific is Equation 24 or Equation 25. In other words, the centrality measure should be able to capture the change of $\bar{m}_{i,j}^{X}$ (connectivity robustness) of a given node pair when a network entity is removed.

In [72], the author proposed a way to measure the centrality of a network entity based on effective resistance in affecting the connectivity robustness of the entire network. In [72], a quantity was proposed in the process of developing the centrality of a network node in affecting the connectivity robustness of a network as a whole. It was only used as an intermediate quantity in [72] and nothing was mentioned that it actually captures the centrality of a node in affecting the connectivity robustness between a node pair. In the following discussions, first, it will be shown that this quantity can reflect the node centrality in affecting $\bar{m}_{i,j}^{X}$. Then, from there, a new quantity was developed for measuring the centrality of a network link followed by some additional analyses and discussions.

4.1 *Node Centrality*

Before going into the equation for calculating the node centrality in affecting $\bar{m}_{i,j}^{X}$, first, the concept of random walk will be introduced. Random walk has been briefly mentioned in the previous discussion. As defined in [83], a random walk is a finite Markov Chain that is time-reversible and hence is a discrete stochastic process. The walk

starts at a given node i, which is usually called the source, and selects one of its neighbors to visit according to a designated probability distribution (usually node degree) as the time step increases by 1. The process repeats until reaching the destination node j. There are three concepts developed from the concept of a random walk that are closely related to the node centrality measure to be discussed here. One is the hitting time of a random walk from node i to node j, which will be denoted as $H_{i,j}$. $H_{i,j}$ is the expected number of steps for a random walk starting from node i to hit node j for the first time. The second one is the commute time (distance) of a round trip random walk between node i and node j ($U_{i,j}$). The relationship between $H_{i,j}$ and $U_{i,j}$ is shown in Equation 26.

$$U_{i,j} = H_{i,j} + H_{j,i} = U_{j,i} \quad \textbf{26}$$

The connection between the effective resistance and the random walk between a node pair lies in the following equation,

$$U_{i,j} = 2M\left(L^{+}_{i,i} - 2L^{+}_{i,j} + L^{+}_{j,j}\right) = 2MER_{i,j} \quad \textbf{27}$$

where M is the number of links of a network.

The last term is random detour, which is defined as the random walk from node i to node j that is forced to bypass a node k. $H_{i,k,j}$ will be used to denote the expected number of steps for a random detour from node i to node j via node k. This is the core concept that leads to the measure of node centrality that will be discussed here [72]. For a given node pair i, j, the difference between $H_{i,k,j}$ and $H_{i,j}$ can be calculated. Denote that difference as $\Delta H_{i,k,j}$. $\Delta H_{i,k,j}$ is the expected extra number of steps of a random walk from node i to node j if it is forced to bypass node k. Using the definition of $\Delta H_{i,k,j}$, the following equation can be obtained.

$$\Delta H_{i,k,j} = H_{i,k} + H_{k,j} - H_{i,j} \quad \textbf{28}$$

In general $H_{i,j} \neq H_{j,i}$. Hence, in general $\Delta H_{i,k,j} \neq \Delta H_{j,k,i}$, which means, the network has directed properties. However, as discussed earlier, the networks to be investigated in this thesis are all undirected networks, which require undirected network properties. To achieve this, instead of using $\Delta H_{i,k,j}$ or $\Delta H_{j,k,i}$, $\Delta U_{i,k,j}$ will be used. $\Delta U_{i,k,j}$ is defined as the expected extra number of steps of a round trip random walk between node i and node j if it is forced to bypass node k. $\Delta U_{i,k,j}$ can be calculated using Equation 29.

$$\Delta U_{i,k,j} = \Delta H_{i,k,j} + \Delta H_{j,k,i} = \Delta U_{j,k,i} \quad \textbf{29}$$

Inserting Equation 27 into Equation 29 yields

$$\Delta U_{i,k,j} = \Delta U_{j,k,i} = 2M\left(ER_{i,k} + ER_{k,j} - ER_{i,j}\right) \quad \textbf{30}$$

$$\Delta U_{i,k,j} = \Delta U_{j,k,i} = 4M(L^{+}_{k,k} + L^{+}_{i,j} - L^{+}_{i,k} - L^{+}_{j,k}) \quad \textbf{31}$$

From the definition of random walk and random detour, it can be seen that, $\Delta U_{i,k,j}$ is a nonnegative number. Intuitively, for the same network, the more peripheral node k is to the connection between node i and node j, the greater the value of $\Delta U_{i,k,j}$ will be. And hence, the less important node k is to the connectivity robustness between node i and node j. Therefore, it is reasonable to hypothesize that $-\Delta U_{i,k,j}$ can be used as the centrality measure of a network node corresponding to $\bar{m}^{X}_{i,j}$ as mathematically expressed below.

$$C^{V}_{k} = -\frac{\Delta U_{i,k,j}}{4M} \quad \textbf{32}$$

Since for a given network topology, the number of network links are constant when doing the centrality analysis of network entities, $\Delta U_{i,k,j}$ is divided by $4M$ to simplify the calculation process.

According to the definition of node centrality, the more important of a node is to the connectivity robustness between a given node pair i, j, the higher the impact its removal will have on $\bar{m}_{i,j}^{X}$, and hence the greater the value of $\Delta\bar{m}_{i,j}^{X}$ will be.

H^2: $-\Delta U_{i,k,j}$ is highly correlated with $\Delta\bar{m}_{i,j}^{X}(k)$.

H_0^2: $-\Delta U_{i,k,j}$ is not highly correlated with $\Delta\bar{m}_{i,j}^{X}(k)$.

In order to test Hypothesis 2, the following experiment was developed using the two SF networks and the two Rand networks discussed earlier in Chapter 3. For each network, randomly select 20 different node pairs. In total, there are 80 node pairs for four networks. First, for each node pair i, j, feed the original network G into the link failure simulation model stated in Algorithm 3 (with 10000 runs) to obtain the expected number of link failures before that node pair disconnects ($\bar{m}_{i,j}^{X}$). Next, at each step, remove a network node k from the original network and this will result in a new network $G' = G\backslash\{k\}$. Feed this newly obtained network G' into the link failure simulation mode (with 10000 runs) to obtain the expected number of link failures before node pair i, j disconnects ($\bar{m}_{i,j}^{X\,\prime}$). Repeat this process for each node pair until all the nodes except node i and node j within the original network have been removed once. So for a network with N network nodes, for each randomly selected node pair, the process should be repeated for $(N-2)$ times in total.

For each network, we can obtain a series of data pairs $[\Delta\bar{m}_{i,j}^{X}(k), -\Delta U_{i,k,j}]$, where $\Delta\bar{m}_{i,j}^{X}(k) = abs\left(\bar{m}_{i,j}^{X\,\prime} - \bar{m}_{i,j}^{X}\right) = \bar{m}_{i,j}^{X} - \bar{m}_{i,j}^{X\,\prime}$. To test H_0^2, calculate the nonlinear correlation between $\Delta\bar{m}_{i,j}^{X}(k)$ and $-\Delta U_{i,k,j}$ by fitting $\Delta\bar{m}_{i,j}^{X}(k)$ as a semi-parametrically estimated function, for example, a generalized additive model (GAM) of $-\Delta U_{i,k,j}$. This is to fit the following model as shown in Equation 33.

$$E\left(\Delta\bar{m}_{i,j}^{X}(k)\middle|-\Delta U_{i,k,j}\right) = \alpha + f\left(-\Delta U_{i,k,j}\right) + \epsilon_k \qquad \textbf{33}$$

This can be done by using the GAM() function in R. The fitting summary of a node pair within the Rand_30_60 network is shown below. A plot of the fitting model is also given to characterize the nature of the relationship between $\Delta\overline{m}_{i,j}^{X}(k)$ and $-\Delta U_{i,k,j}$ as shown in Figure 27.

```
Family: gaussian
Link function: identity

Formula:
Delta_m_dis_ij ~ s(Detour_k)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.77788    0.06419   43.28   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
              edf Ref.df     F p-value
s(Detour_k) 3.503  4.337 172.2  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.965   Deviance explained =   97%
GCV = 0.13746  Scale est. = 0.11535   n = 28
```

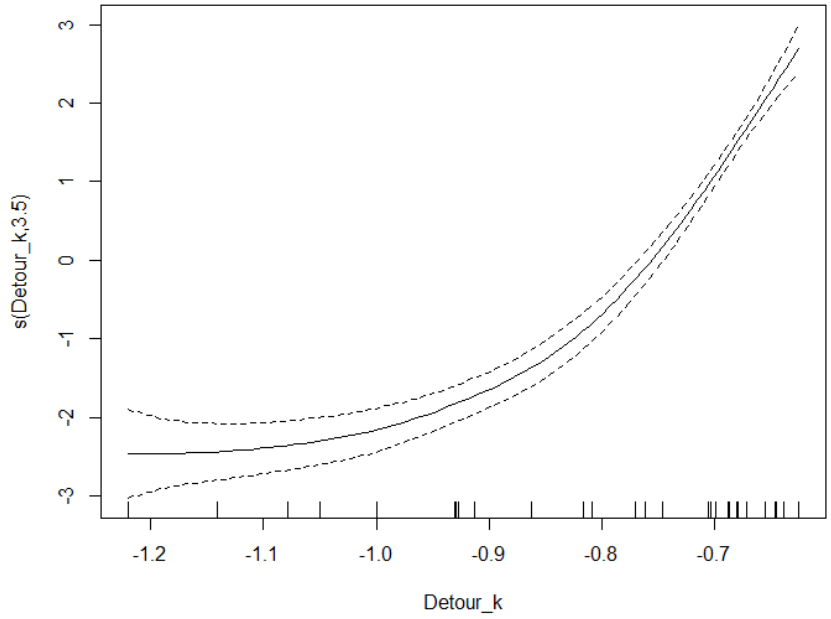


Figure 27. Plot of the Example GAM Model between $\Delta\bar{m}_{i,j}^{X}(k)$ and $-\frac{\Delta U_{i,k,j}}{4M}$

For each node pair, fit the GAM between $\Delta\bar{m}_{i,j}^{X}(k)$ and $-\Delta U_{i,k,j}$ and document the adjusted R^2 value. Calculated the averaged adjusted R^2 value of a network based on the 20 randomly node pairs. The results are summarized in Table 9.

Table 9. GAM Model Summary for Node Pairs within Classical Networks (Node Centrality)

Network	Rand_30_60	Rand_50_100	SF_30_60	SF_50_100
Average Adjusted R^2	0.94	0.83	0.93	0.95

Since the average adjusted R^2 values of all the classical networks are high, H_0^2 is rejected. Hence, $-\Delta U_{i,k,j}$ could be used as a measure for the centrality of a network node in terms of the connectivity robustness between a given node pair.

4.2 *Link Centrality*

In the previous section, it has been successfully shown that, the more peripheral a node is to the connection between a node pair, or in other words the greater the value of $\Delta U_{i,k,j}$, the less importance of node k to the connectivity robustness between node pair i,j. Intuitively, for a link k,l, the higher the values of $\Delta U_{i,k,j}$ and $\Delta U_{i,l,j}$, the more peripheral the position of link k,l is to the connection between node pair i,j. Therefore the less important link k,l is to the connectivity robustness between node pair i,j. This suggests the following relationship.

$$C_{k,l}^{E} \propto -(\Delta U_{i,k,j} + \Delta U_{i,l,j}) \quad \textbf{34}$$

Next, the following network property is used. For a network, the number of link failures that any link can sustain until all the node pairs within a network disconnect is 1. This means for the connection between all the node pairs of a network, the structural contributions of all the links are the same. For simplicity, a constant, W, is used to quantify the structural contribution of a link to the connection between all the node pairs of a network. According to the definition of link centrality, the following equation can be written.

$$\sum_{i=1}^{N}\sum_{j=1}^{N} f\left(C_{k,l}^{E}\right) = W \quad \textbf{35}$$

where,

W is a constant;

$f(\,)$ is a function with undefined properties.

In [72], the author proved the following relationship as written in Equation 36.

$$\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} \Delta U_{i,k,j} = \Delta U_k = 4Ml_{k,k}^{+} \quad \textbf{36}$$

So if $C_{k,l}^{E}$ is in the following form, Equation 35 is satisfied.

$$C_{k,l}^{E} = -\frac{\Delta U_{i,k,j} + \Delta U_{i,l,j}}{\Delta U_k + \Delta U_l} \quad \textbf{37}$$

This leads to the third hypothesis of this thesis.

H^3: $-\frac{\Delta U_{i,k,j}+\Delta U_{i,l,j}}{\Delta U_k+\Delta U_l}$ is highly correlated with $\Delta\bar{m}_{i,j}^{X}(k,l)$.

H_0^3: $-\frac{\Delta U_{i,k,j}+\Delta U_{i,l,j}}{\Delta U_k+\Delta U_l}$is not highly correlated with $\Delta\bar{m}_{i,j}^{X}(k,l)$.

The experiment to test Hypothesis 3 is similar to the one used for testing Hypothesis 2. The detailed process of the experiment is as following. For each of the four networks (two SF networks and two Rand networks), randomly select 20 node pairs. First, for each node pair i,j, feed the original network G into the link failure simulation model stated in Algorithm 3 (with 10000 runs) to obtain the expected number of link failures before that node pair disconnects ($\bar{m}_{i,j}^{X}$). Next, at each step, remove a network link k,l from the original network and this will result in a new network $G' = G\backslash\{(k,l)\}$. Feed this newly obtained network G' into the link failure simulation mode (with 10000 runs) to obtain the expected number of link failures before node pair $i.j$ disconnects ($\bar{m}_{i,j}^{X}{}'$). Repeat this process for each node pair until all the links within the original network have been removed once. So for a network with M network links, for each node pair, the process should be repeated for M times in total.

For each network, we can obtain a series of data pairs $[\Delta\bar{m}_{i,j}^{X}(k,l), -\frac{\Delta U_{i,k,j}+\Delta U_{i,l,j}}{\Delta U_k+\Delta U_l}]$. To test H_0^3, calculate the nonlinear correlation between $\Delta\bar{m}_{i,j}^{X}(k,l)$ and $-\frac{\Delta U_{i,k,j}+\Delta U_{i,l,j}}{\Delta U_k+\Delta U_l}$ by fitting $\Delta\bar{m}_{i,j}^{X}(k,l)$ as a semi-parametrically estimated function, for example, a generalized additive model (GAM) of $-\frac{\Delta U_{i,k,j}+\Delta U_{i,l,j}}{\Delta U_k+\Delta U_l}$. This is to fit the following model as shown in Equation 38.

$$E\left(\Delta\bar{m}_{i,j}^{X}(k,l)\middle|-\frac{\Delta U_{i,k,j}+\Delta U_{i,l,j}}{\Delta U_k+\Delta U_l}\right) \quad \textbf{38}$$

$$=\alpha+f\left(-\frac{\Delta U_{i,k,j}+\Delta U_{i,l,j}}{\Delta U_k+\Delta U_l}\right)+\epsilon_{k,l}$$

This can be done by using the GAM() function in R. The fitting summary of a node pair within the Rand_30_60 network is shown below. A plot of the fitting model is also given to characterize the nature of the relationship between $\Delta\bar{m}_{i,j}^{X}(k,l)$ and $-\frac{\Delta U_{i,k,j}+\Delta U_{i,l,j}}{\Delta U_k+\Delta U_l}$ as shown in Figure 28.

```
Family: gaussian
Link function: identity

Formula:
Delta_m_dis_ij ~ s(Detour_k_l)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -31.65853    0.03861    -820   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
               edf Ref.df    F p-value
s(Detour_k_l) 8.827   8.99 1416  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.995   Deviance explained = 99.6%
GCV = 0.10696  Scale est. = 0.089444  n = 60
```

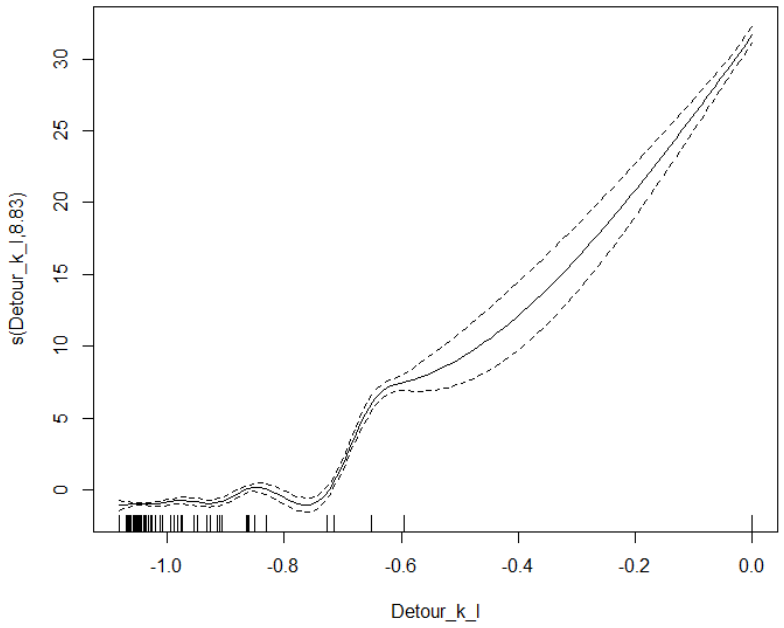


Figure 28. Plot of the Example GAM Model between $\Delta\bar{m}_{i,j}^{X}(k,l)$ and $-\frac{\Delta U_{i,k,j}+\Delta U_{i,l,j}}{\Delta U_k+\Delta U_l}$

For each node pair, fit the GAM between $\Delta\bar{m}_{i,j}^{X}(k,l)$ and $-\frac{\Delta U_{i,k,j}+\Delta U_{i,l,j}}{\Delta U_k+\Delta U_l}$ and document the adjusted R^2 value. Calculated the averaged adjusted R^2 value of a network based on the 20 randomly node pairs. The results are summarized in Table 10.

Table 10. GAM Model Summary for Node Pairs within Classical Networks (Link Centrality)

Network	Rand_30_60	Rand_50_100	SF_30_60	SF_50_100
Average Adjusted R^2	0.99	0.99	0.99	0.99

Since the average adjusted R^2 values of all the classical networks are high, H_0^3 is rejected. Hence, $-\frac{\Delta U_{i,k,j}+\Delta U_{i,l,j}}{\Delta U_k+\Delta U_l}$ could be used as a measure for the centrality of a network link in terms of the connectivity robustness between a given node pair.

4.3 *Chapter Summary*

In this Chapter, the centrality of network entities were discussed. The discussion started by giving general definition of the centrality. Depending on the research content, there are different types of centralities. The one considered here is the centrality of a network entity in terms of its impacts on the capability-based robustness of a network, the mathematical representations of which are provided in Equation 24 or Equation 25.

Depending on the research content, there are different types of centrality measures. In general, the choice of centrality measure should reflect the role of network entities in affecting the network quantity under study and correspond to the way that quantity is measured. Therefore, in this thesis, a candidate centrality measure should correspond to the capability-based connectivity robustness measuring process developed in Chapter 3. Based on the discussion in Chapter 3, a candidate centrality measure should be able to measure importance of a network entity to the connectivity robustness of a give node pair.

The proposed node centrality and link centrality measures are based on the concept of random detour as given in Equation 32 and Equation 37. By testing Hypothesis 2 and Hypothesis 3, the validity of the two proposed measures were confirmed.

CHAPTER V

A CAPABILITY-BASED CONNECTIVITY ROBUSTNESS EVALUATION FRAMEWORK

To summarize the discussions from Chapter 2 to Chapter 4 leads to a framework for the fast evaluation of the capability-based connectivity robustness of a CIN. The process is summarized in Figure 29. The application of this framework is demonstrated on the example described in Section 1.5.

The first step of the framework is to construct a capability-based network mode. This has already been done along the discussion in Chapter 2. Assume all the key information flows have OR relationship, then the capability-based network model of the example problem is the one illustrated in Figure 5 and the capability critical node pair is node pair C, F.

The second step is to measure the connectivity robustness of node pair C, F, which is the capability critical node pair of the example CIN. First, calculate the effective resistance between that node pair and that gives $ER_{C,F}$=1.16. Next, plug $N = 8, M = 12, ER_{C,F} = 1.61$ into Equation 16, and we can obtain $\left(\widetilde{\bar{m}}_{C,F}^{X}\right) = 5.16$. To check the accuracy of this estimation, the network topology of the CIN in this example problem was fed into the link failure simulation model. The simulation result ($\bar{m}_{C,F}^{X}$) and the accuracy of the above estimation are presented below in Table 11.

Table 11. Summary of the Accuracy of $\widetilde{\bar{m}}_{C,F}^{X}$

$\bar{m}_{C,F}^{X}$	5.14
$\widetilde{\bar{m}}_{C,F}^{X}$	5.16
% Diff	0.4%

As can be seen in Table 11, the difference between $\widetilde{\bar{m}}_{C,F}^{X}$ and $\bar{m}_{C,F}^{X}$ is only 0.4% and is negligible. Hence, $\widetilde{\bar{m}}_{C,F}^{X}$ is a close estimation for $\bar{m}_{C,F}^{X}$.

The third step is to evaluate the centrality of network entities in terms of their importance to the capability-based connectivity robustness of a CIN. The centrality of network nodes calculated via Equation 32 is summarized in Table 12. The centrality of network inter-connection links calculated via Equation 37 is summarized in Table 13.

Table 12. Summary of Node Centrality C_k^V and Impacts of Node Removal $\Delta\bar{m}_{C,F}^{X}(k)$

Node Index	C_k^V	$\Delta\bar{m}_{C,F}^{X}(k)$
u_1	0	5.14
u_2	-0.23	4.14
u_3	-0.26	3.22
u_4	-0.55	0.78
u_5	-0.55	0.78
u_6	-0.55	0.79

Table 13. Summary of Link Centrality $C_{k,l}^E$ and Impacts of Link Removal $\Delta\bar{m}_{C,F}^{X}[(k,l)]$

k	l	$C_{k,l}^E$	$\Delta\bar{m}_{C,F}^{X}[(k,l)]$
B	u_1	0	5.14
u_1	u_2	-0.34	4.14
u_2	u_3	-0.96	3.22
u_3	u_4	-1.19	0.80
u_3	u_5	-1.19	0.81
u_3	u_6	-1.19	0.82

As can be seen in Table 12 and Table 13, the proposed measures (Equation 32 and Equation 37) successfully captured the importance of network links/nodes to the connectivity robustness between node pair C, F.

According to Table 12, SUAV 1 is of the most importance. The failure SUAV 1 will result in immediate disconnection between node pair C, F, and hence the failure of the entire operation. The reason for SUAV 1 to be the most important node is its bottleneck role to the connection between node pair C, F. The second important node is SUAV 2, which is also due to its bottleneck position. However unlike SUAV 1, whose failure will completely disconnect the connection between node pair C, F, node pair C, F is still connected if only SUAV 2 fails. The next important node is SUAV 3, which relays SUAV 4, SUAV 5 and SUAV 6 to the command center. SUAV 4, SUAV 5 and SUAV 6 are of the same importance due to their structural similarity. They are also of the least importance. This is because the impact of the failure of any of them is isolated and will not impact the other pathways that connecting node pair C, F. According to Table 13, the inter-connection link between the command center and SUAV 1 is the most important link in terms the connectivity robustness between node pair C, F. This is due to its bottleneck role. The second important inter-connection link is the one between SUAV 1 and SUAV 2; while the third important link is the one between SUAV 2 and SUAV 3.

Based on the above centrality analysis results, to strengthen the capability-based connectivity robustness of the network in the example problem, additional failure protection mechanisms can be applied to network nodes, such as SUAV 1, SUAV 2 or to individual inter-connection links, such as the one between the command center and SUAV 1.

Figure 29. A Framework for the Fast Evaluation of the Capability-Based Connectivity Robustness of a CIN

5.1 *The Practicality of the Proposed Framework*

In the previous section, it has been demonstrated how the framework can help evaluate a given CIN. In this section, the discussion focuses on the practicality of the proposed connectivity robustness measure.

The key element of the proposed framework is to find the Moore-Penrose pseudoinverse of a network topology Laplacian, which is denoted as L^+. As written in [96], the Moore-Penrose pseudo inverse and the sub-matrix inverses of the Laplacian can reveal significant topological characteristic of a graph and have been applied to fields as diverse as probability and mathematical chemistry, collaborative recommendation systems and social networks, epidemiology and infrastructure planning. Alas, despite such versatility, the pseudo inverse and the sub-matrix inverses of the Laplacian suffer a practical handicap. Using the standard matrix factorization and inversion based methods (e.g. Cholesky factorization and inversion) on a serial processor to compute L^+ has an $O(N^3)$ computational time, where N is the number of network nodes. This means using the conventional methods, it is very expensive to compute L^+. This clearly impedes the practical utilities of the proposed connectivity robustness measure and the subsequent analyses as network size grows. This is especially problematic during the CIN design and optimization process that regular L^+ re-computations are required. In response to this, researchers have proposed several novel approaches to increase the efficiency of computing L^+. With a parallel architecture equipped with many processors, the time complexity of using the standard factorization and inversion methods to calculate L^+ could be reduced to $O(N)$ or even $O(logN)$ [97, 98]. If parallel computing is not available, using a divide-and-conquer based approach as proposed in [96] , the cost of computing L^+ of an undirected graph is at a cost of $O(N^2)$. With those, the time complexity of the proposed method is no longer a problem.

5.2 *An Alternative Design Generation Process*

In the following discussion, how the proposed measure can help design a CIN in terms of capability-based connectivity will be shown. The discussion starts by formulating the problem. The design problem will be decomposed into four sub-problems. Unconstrained situation will be considered first and then design constraints will be added gradually.

The base sub-problem is to design a CIN with enough capability-based connectivity robustness to complete an operation without specify the number of entities (nodes) and links. Of course, the more nodes and the more links used, the more connectivity robust the CIN is. However, real world design practices are never conducted without a consideration on cost. For a CIN, by deploying more participant entities with high information transmission capabilities, more communication channels (such as all entities can communicate to each other, a P2P structure) with high reliability, the network can have very high capability-based connectivity robustness, but also a very high acquisition cost. Hence, usually, a CIN design will specify the maximum number of entities that can be used for a specific operation. This formulates Sub-Problem 1. On the other hand, network links are also established at a cost. In order to be able to communicate within the network, a network node needs to be equipped with enough information transmission capabilities, such as bandwidth, information processing capacity, range, and power to enable the communication. The higher and the more comprehensive the information transmission capabilities, the higher the acquisition cost. Sometimes some of the required information transmission capabilities are not practical due to design constraints, such as the space, take-off weight constraints of SUAVs. Hence, link constraints are required. Unlike the node constraint, which is the number of network

nodes that can be deployed, link constraints are more complex. In this thesis, the following link constraint: the feasibility of establishing a link, will be considered. With this constraint, Sub-problem 2 is formulated.

This is to consider the feasibility of establishing a link considering the physical distance, interoperability between network entities. As defined in [99], interoperability is the *ability of two or more systems or components to exchange resources in the form of data, information, materiel, and services, and to use the resources that have been exchanged to enable them to operate effectively together*.

Finally, in Sub-problem 3, the network topology selection criteria are set and the best network topology design or designs are selected accordingly using the framework discussed earlier in this Chapter. Guided by the above problem decomposition, the following CIN design process was proposed.

The design activity corresponding to Sub-problem 0 or the base design problem is to specify the operation to be performed by a CIN. This is usually done by mission statement. For the example problem, the operation is to have a group of networked SUAVs operate over some fields and send regular updates to the command center regarding to the fields they monitor.

The design activity corresponding to Sub-problem 1 is to specify the number of network nodes and their general roles including the physical location of each network node. For the example problem, there are six monitoring fields. The original CIN design (Figure 3) uses six SUAV. Hence, there are in total eight network nodes within the corresponding network model with one node representing the command center (node C), six nodes representing those SUAVs and one node for the information source (node F).

The design activity corresponding to Sub-problem 2 starts by firstly constructing a fully connected network using all the network nodes. Then, decide the sets of infeasible network links and remove those links from the fully connected network.

Next, remove network links one at a step using the following rules. At each step, calculate the centrality of the existing network links in terms of $\overline{m}^{X}_{i^*,j^*}$ using the method proposed in Chapter 4 (Equation 37). Name the links whose removal will not result in any non-information source node disconnected from the command center as candidate links. Select the candidate link with the smallest link centrality. If there are more one candidate link with the smallest link centrality, then randomly select one. Remove the selected link from the network topology. Repeat the selection and removal process until no link left in the candidate link pool. The design process will stop here. Document the network topology obtained at each step, and calculate the average number of link failures until the capability critical node pair disconnects for each network topology. Need to note here, the capability critical node pair may vary along the design generation process. Name the network topologies obtained as the candidate network topologies.

For the example problem, the design activities corresponding to Sub-problem 2 starts with a fully connected undirected network with eight nodes as shown in Figure 30. Assume the physical impossible links are link B, u_4, link B, u_5, and link B, u_6. Next, using the above "link minus" approach, the link failure history for the example problem is shown in

Table 14. During the "link minus" process, all links selected are both feasible and viable.

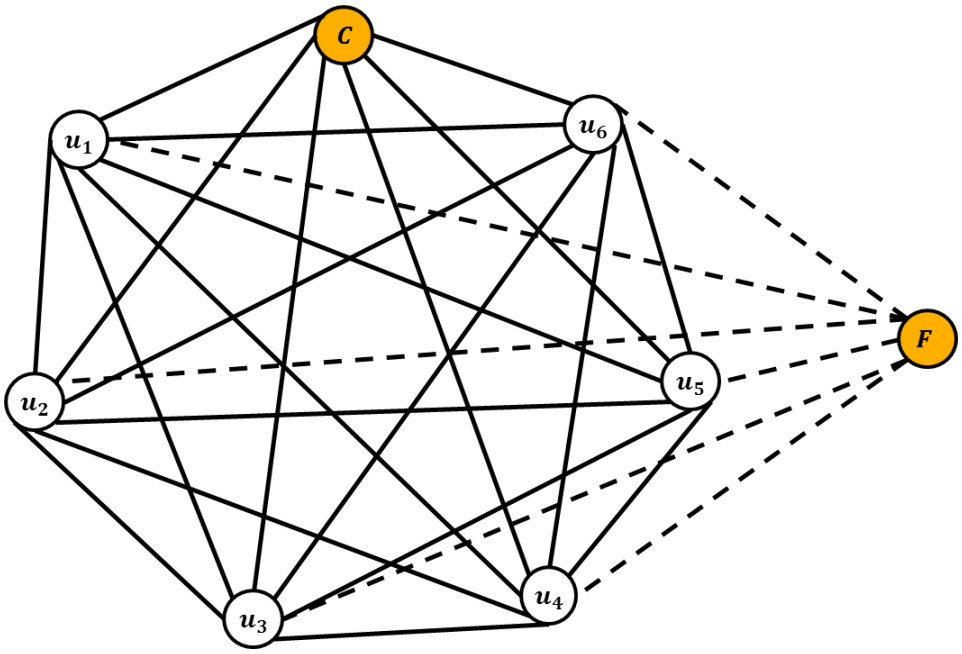


Figure 30. The Starting Topology of the Sub-Problem 2 for the Example CIN

Table 14. Results of the "Link Minus" Procedure

Step	M	k	l	$ER_{C,F}$	$\widetilde{\widetilde{m}}^{X}_{C,F}$	$\frac{\widetilde{\widetilde{m}}^{X}_{C,F}}{M}$
0	24	-----		0.52	16.84	70.2%
1	23	u_5	u_4	0.52	16.14	70.2%
2	22	u_6	u_4	0.52	15.44	70.2%
3	21	u_6	u_5	0.52	14.74	70.2%
4	20	u_4	u_2	0.53	14.02	70.1%
5	19	u_5	u_2	0.53	13.29	70.0%
6	18	u_6	u_2	0.54	12.56	69.8%
7	17	u_6	u_3	0.55	11.83	69.6%
8	16	u_4	u_3	0.56	11.10	69.4%
9	15	u_5	u_3	0.57	10.35	69.0%

10	14	u_3	u_2	0.57	9.66	69.0%
11	13	u_2	u_1	0.57	8.95	68.9%
12	12	u_3	u_1	0.58	8.24	68.7%
Original Design	12	------		1.61	5.16	43.0%

In Table 14, the first column is the step number. Step 0 corresponds to the base network topology obtained by removing all the physical impossible links. The second column is the number of network links within the network topology of each step. The third and the fourth columns contain the two ending nodes of the link removed at each step. The fifth column contains the effective resistance value between the critical node pair at each step. (Along the design generation process, node pair C, F is always the capability critical node pair of the CIN). The last second column contains the estimated average number of link failures that will result in the disconnection between the capability critical node pair of each step. Finally, the last column is the estimated average fraction of link failures that will result in the disconnection between the capability critical node pair of each step. The average fraction of link failures until the capability critical node pair disconnects can be viewed as the structural efficiency of a CIN.

From this table, the first observation can be made is, the capability-based connectivity robustness and the structure efficiency of all the candidate designs are higher than the original design. Next, plot the average number of disconnection link failures and the average fraction of disconnection link failures vs. the link number of each step as shown in Figure 31. All the candidate designs have similar structure efficiency.

Comparing the $ER_{C,F}$ column and the average fraction of disconnection link failures, the values in both columns are very stable. As the average fraction decreases, the $ER_{C,F}$value increases.

The design activity corresponds to the Sub-problem 3 is to specify the desired capability-based connectivity robustness (to link failures). Using the measure proposed in Chapter 3, it is to specific the minimum average number of link failures that can be tolerated during the CIN operation. Name that as the critical average number of link failures $\left(\bar{m}^{X}_{i^*,j^*}\right)^{C}$, where i^*, j^* is the capability critical node pair of the CIN. For the example problem, in order to maintain operation at the minimum level, node C and node F have to remain connected. The value of $\left(\bar{m}^{X}_{C,F}\right)^{C}$is decided to be five. In addition, from economic design perspective, the network topology with the smallest link number is selected (Step 12) and is shown in Figure 32. Comparing the optimized design and the original design, we can see the structural benefits are achieved by eliminating extra relaying hubs.

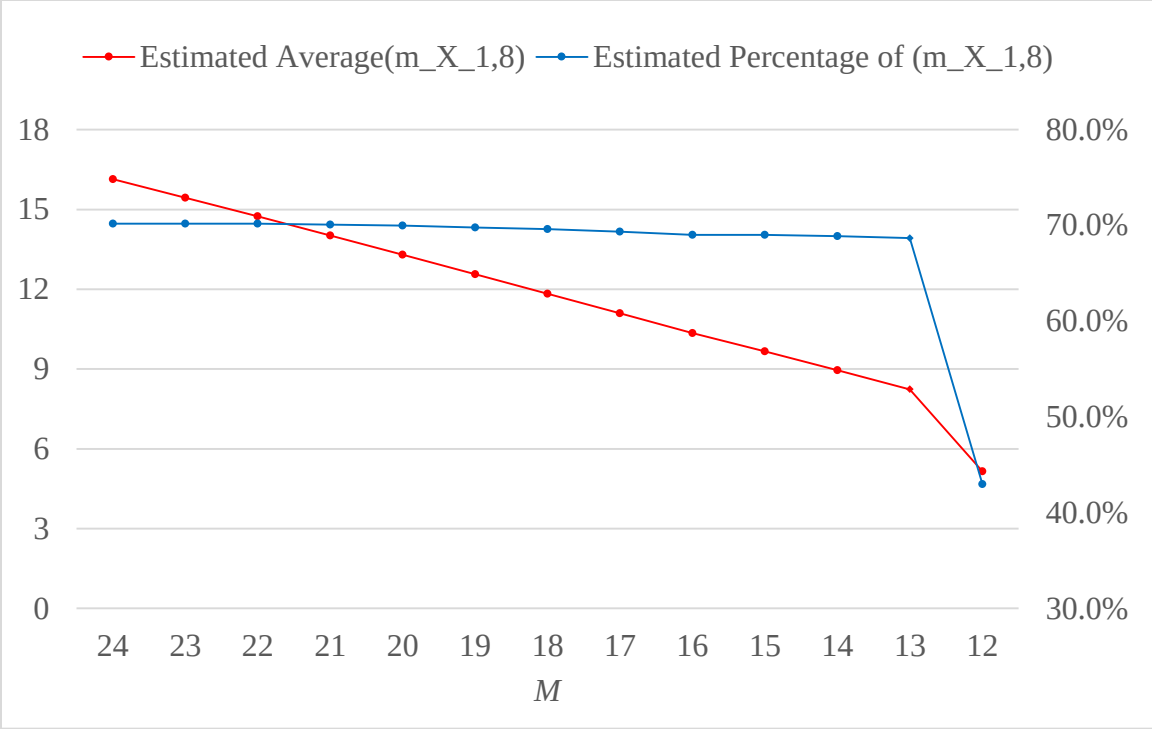


Figure 31. Co-plot of $\widetilde{\bar{m}}_{C,F}^{X}$ and $\frac{\widetilde{\bar{m}}_{C,F}^{X}}{M}$ vs. M

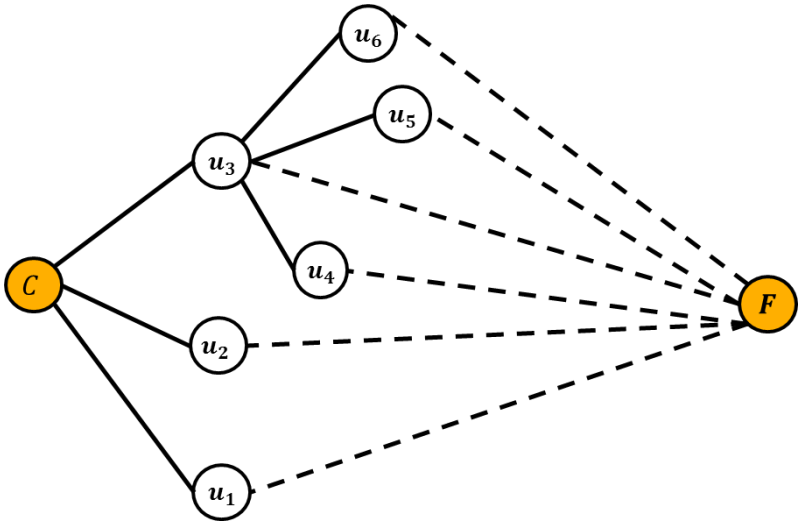


Figure 32. The Optimized Network Topologies of the Example CIN

5.3 *Chapter Summary*

The focus of this chapter is on the practical use of the proposed capability-based connectivity robustness measure and the subsequent analyses. First, a framework for the fast evaluation of the capability-based connectivity robustness of a CIN was proposed by summarizing the discussions provided in Chapter 2 to Chapter 4. The framework was demonstrated on the example discussed in Section 1.5. Next, the practicality of the proposed capability-based connectivity measure was discussed by stating its computational complexity. The key of the proposed measure and any subsequent analyses is to find the Moore-Penrose pseudoinverse of the Laplacian of a network topology, which is denoted as L^+. With a parallel architecture equipped with many processors, the time complexity of using the standard factorization and inversion methods to calculate L^+ could be reduced to $O(N)$ or even $O(logN)$. This greatly enhanced the practical use of the proposed measure and the subsequent analyses. Finally, an alternative design generation procedure was proposed. It is very easy for the proposed procedure to incorporate design constraints. The design process is repeatable and generates a pool of design candidates. It enables rapid trade-offs between capability-based connectivity robustness and other considerations, such as link number and information transmission range. Although the process demonstrated in Section 5.2 focuses on the capability-based connectivity robustness of a CIN, it is flexible to be used as a sub-design process of a more comprehensive, complex design process.

CHAPTER VI

HOW TO STRENGTHEN CAPABILITY-BASED CONNECTIVITY ROBUSTNESS

This chapter demonstrates how the measuring process for the capability-based connectivity robustness developed in Chapter 3 and the corresponding centrality measures discussed in Chapter 4 can be used to help develop some strategies to strengthen the capability-based connectivity robustness of a CIN. This is just to strengthen the connectivity robustness between the capability critical node pair of the CIN, or more specifically, to increase the value of $\bar{m}_{i^*,j^*}^X$. Two different strategies are considered. The first one is to add a new link into the existing network. This is to increase the static connectivity robustness between the capability critical node pair. The second strategy is to prepare a substitution for a network node, which is usually of great importance to the connectivity robustness between the capability critical node pair.

6.1 *How to Add a Link*

In this section, the strategy of adding a link will be discussed. The goal is to find a way to quickly determine the location to add a new link that increases the $\bar{m}_{i,j}^X$ value of a given node pair most. Such a position will be referred to as the optimal position and denoted as e^+.

The most straightforward way to find e^+ is to quantify the effects of adding a link on $\bar{m}_{i,j}^X$, and then to conduct an exhaustive search to identify the link whose addition results in the most $\bar{m}_{i,j}^X$ increase. For a given network topology, adding a link does not change the network node number. Use $ER_{i,j}^e$ to denote the new effective resistance value

between node pair i,j after link e is added to the original network. According to the discussion in Chapter 3, to compare the effects of adding a network link e on $\bar{m}_{i,j}^{X}$ is to compare the value of $\frac{1}{ER_{i,j}^{e}}$. In other words, to find the link whose addition results in the most $\bar{m}_{i,j}^{X}$ increase is to find the one with the lowest $ER_{i,j}^{e}$ value.

Hence, the complexity of comparing the effects of adding a link on the $\bar{m}_{i,j}^{X}$ is on the same order of calculating $ER_{i,j}$. As discussed earlier, using conventional calculation method on a serial processor, the complexity order of calculating $ER_{i,j}$ is $O(N^3)$ [55]. The network topologies of most CINs are sparse. This means the number of non-existing network links of a CIN network topology is on the same order of $O(N^2)$. As a result, the total complexity order of finding e^+ through an exhaust search could be $O(N^5)$. Clearly, there is a need for methods that determine e^+ in a computationally scalable fashion with high accuracy.

In [55], the authors proposed four different methods for finding e^+ in terms of the connectivity robustness of a whole network. The one based on L^+ has the best performance for all the networks tested. Motivated by that, in this section, a method based on L^+ for finding e^+ in terms of $\bar{m}_{i,j}^{X}$ was developed. Among all the possible link positions, the proposed method chooses the one with the highest Ω_e value. Assume the two ending nodes of a link e are k and l, Ω_e or $\Omega_{k,l}$ can be calculated through the following equation.

$$\Omega_{k,l}^{i,j} = -\frac{\Delta U_{i,k,j} + \Delta U_{i,l,j}}{\Delta U_k + \Delta U_l} - \frac{\Delta U_{k,i,l} + \Delta U_{k,j,l}}{\Delta U_i + \Delta U_j} \quad \textbf{39}$$

The superscript i,j is added to indicate that the node pair of interest is i,j. Equation 39 contains two parts. Referring back to the discussion in Chapter 3, the first part of $\Omega_{k,l}^{i,j}$ can be viewed as the centrality of the originally non-existing link k,l.

According to Hypothesis 3, the higher the centrality of an existing link as calculated by Equation 3738, the higher the impact of its removal on the value of $\bar{m}_{i,j}^{X}$. Hypothesis 3 has passed its test. However, this does not necessary mean failing to reject the following statement: The higher the centrality of a non-existing link as calculated by Equation 37 38is, the higher the impact of its addition to $\bar{m}_{i,j}^{X}$. In order to address this, the second part of Equation 39 was added. The second part measures the importance of the connectivity robustness between node i and node j to the connectivity robustness between node k and node l. The argument is that, if the non-existing link k, l is truly very important to the connectivity between node i and node j, either the value of $\frac{\Delta U_{i,k,j}+\Delta U_{i,l,j}}{\Delta U_k+\Delta U_l}$ or $\frac{\Delta U_{k,i,l}+\Delta U_{k,j,l}}{\Delta U_i+\Delta U_j}$ should be relatively small. This forms the fourth hypothesis of this thesis.

H^4: $\Omega_{k,l}^{i,j}$ can indicate the benefits of adding a non-existing link into a network on $\bar{m}_{i,j}^{X}$.

H_0^4: $\Omega_{k,l}^{i,j}$ cannot indicate the benefits of adding a non-existing link into a network on $\bar{m}_{i,j}^{X}$.

Hypothesis 4 was tested using the following experiment, which is a modified version of the one used in [55]. For a given key node pair i, j, the $ER_{i,j}^{e}$ values of all the possible link additions are calculated. Next, calculate the $\Omega_e^{i,j}$ values of all the possible link additions. Order the ER_{ij}^{e} value in ascending order, and denote this as $\left(ER_{i,j}^{e}\right)^*$. Now for a key node pair, the following data series can be obtained: $[\Omega_e^{i,j}, ER_{i,j}^{e}, \left(ER_{i,j}^{e}\right)^*]$. List the three columns together in the following fashion. For each non-existing network link, we can have a $\Omega_e^{i,j}$ value and a $ER_{i,j}^{e}$ value. Sort the data pairs in the two columns in descending order of $\Omega_e^{i,j}$. Finally, attach the $\left(ER_{i,j}^{e}\right)^*$ column to the sorted data table. Refer to this newly obtained data table the performance table of node pair i, j. After the

three columns are properly listed, the absolute relative difference between $ER_{i,j}^{e}$ and $\left(ER_{i,j}^{e}\right)^{*}$ is calculated for each link (each row). Denote the difference as $Err_{e}^{i,j}$.

$$Err_{e}^{i,j} = \frac{\left| ER_{i,j}^{e} - \left(ER_{i,j}^{e}\right)^{*} \right|}{\left(ER_{i,j}^{e}\right)^{*}} \quad \textbf{4}\textbf{0}$$

The experiments was carried out on the two SF networks and the two Rand networks developed in Chapter 3. For each network, 20 node pairs were randomly selected. The experiment results are summarized in Table 15 and Table 16. Table 15 contains the results of the two SF networks and Table 16 contains the results for the two Rand networks. In both tables, "first 1" is the $Err_{e}^{i,j}$ value of the first row (link) in the performance table for node pair i, j. It represents the accuracy of $\Omega_{k,l}^{i,j}$ identifying e^{+} for a given node pair. "First 2" is the average $Err_{e}^{i,j}$ value of the first two rows (links) in the performance table for key node pair i, j. It represents the accuracy of $\Omega_{k,l}^{i,j}$ identifying the optimal link e^{+}, and the second optimal link. "Overall" is the averaged $Err_{e}^{i,j}$ value over all the rows in the performance table for node pair i, j. It represents the overall accuracy of using $\Omega_{k,l}^{i,j}$ to compare the impact of the addition of non-existing network links.

From Table 15 and Table 16, it can be observed that, the "first 1" $Err_{e}^{i,j}$ values of almost all the node pairs are 0, which means $\Omega_{k,l}^{i,j}$ can successfully identify the optimal non-existing link. In addition, the "first 2" $Err_{e}^{i,j}$ values and the "overall" $Err_{e}^{i,j}$ values of all the node pairs are so small that H_0^4 is rejected. The fact that, for most of the node pairs, the "first 2" $Err_{e}^{i,j}$ values are bigger than the corresponding "overall" $Err_{e}^{i,j}$ values suggests the performance of $\Omega_{k,l}^{i,j}$ fluctuates and eventually stabilizes when identifying the benefits of adding a non-existing link as the link's $\overline{m}_{i,j}^{X}$ benefit decreases.

Table 15. Performance of $\Omega_{k,l}^{i,j}$ for Different Node Pairs (Rand)

NODE PAIR INDEX	Rand_30_60			Rand_50_100		
	First 1	First 2	Overall	First 1	First 2	Overall
1	0.00%	1.17%	0.66%	0.00%	1.10%	0.51%
2	0.00%	0.00%	0.58%	0.00%	3.84%	0.72%
3	0.00%	1.39%	0.64%	0.00%	2.40%	0.59%
4	0.00%	0.43%	0.73%	0.00%	1.73%	0.46%
5	0.00%	1.63%	0.53%	0.00%	5.16%	0.52%
6	0.00%	1.33%	0.65%	0.00%	1.66%	0.55%
7	0.00%	0.05%	0.52%	0.00%	4.66%	0.72%
8	0.00%	0.99%	0.49%	0.00%	3.76%	0.62%
9	0.00%	3.15%	0.74%	0.00%	1.25%	0.32%
10	0.00%	3.73%	0.86%	0.00%	0.82%	0.33%
11	0.00%	3.31%	0.68%	0.00%	0.55%	0.34%
12	0.00%	0.41%	0.53%	0.00%	1.03%	0.36%
13	1.94%	3.08%	0.41%	0.00%	2.63%	0.62%
14	0.00%	0.00%	0.81%	0.00%	1.03%	0.36%
15	0.00%	0.89%	0.35%	0.00%	1.44%	0.44%
16	0.00%	0.32%	0.62%	0.00%	3.55%	0.74%
17	0.00%	0.00%	0.72%	0.00%	3.40%	0.51%
18	0.00%	0.00%	0.97%	0.00%	4.85%	0.64%
19	0.00%	2.65%	0.57%	0.00%	4.53%	0.34%
20	0.00%	1.66%	0.58%	0.00%	0.08%	0.19%
AVERAGE	0.10%	1.31%	0.63%	0.00%	2.47%	0.49%
STD	0.42%	1.22%	0.15%	0.00%	1.57%	0.15%

Table 16. Performance of $\Omega_{k,l}^{i,j}$ for Different Node Pairs (SF)

NODE PAIR INDEX	SF_30_60			SF_50_100		
	First 1	First 2	Overall	First 1	First 2	Overall
1	0.00%	0.00%	0.44%	0.00%	0.85%	0.79%
2	0.00%	0.93%	0.55%	0.00%	0.24%	0.68%
3	0.00%	2.26%	0.67%	0.00%	4.62%	0.64%
4	0.00%	0.36%	0.37%	0.00%	3.03%	0.60%
5	0.00%	0.11%	0.53%	0.00%	0.00%	0.42%
6	0.00%	1.60%	0.48%	0.00%	0.58%	0.59%
7	0.00%	1.20%	0.38%	0.00%	0.78%	0.71%
8	0.00%	5.55%	0.61%	0.00%	2.81%	0.75%
9	0.00%	4.01%	0.53%	0.00%	0.00%	1.10%
10	0.00%	3.70%	0.68%	0.00%	2.63%	0.90%
11	0.00%	5.90%	0.55%	0.00%	0.34%	0.69%
12	0.00%	4.05%	0.62%	0.00%	0.00%	0.79%
13	0.00%	4.05%	0.62%	0.00%	0.00%	0.88%
14	0.00%	2.28%	0.48%	0.00%	0.00%	0.93%
15	0.00%	1.14%	0.38%	0.00%	0.00%	0.84%
16	0.00%	2.04%	0.60%	0.00%	1.05%	0.73%
17	0.00%	1.45%	0.44%	0.00%	0.80%	0.95%
18	0.00%	6.12%	0.68%	0.00%	3.44%	0.88%
19	0.00%	5.73%	0.33%	0.00%	2.50%	0.97%
20	0.00%	2.11%	0.45%	0.00%	0.09%	0.65%
AVERAGE	0.00%	2.73%	0.52%	0.00%	1.19%	0.77%
STD	0.00%	1.96%	0.11%	0.00%	1.39%	0.16%

6.2 *How to Prepare a Substitution*

Resilience has the same fundamental motivation and ultimate goal as robustness. They both originated as system level design concepts. Unlike robustness that has a concrete definition, resilience is a "work in progress" concept which, at present, could have a number of different meanings [84]. [85] provides a good review of existing definitions on resilience, and summaries that, robustness is the ability to resist or counteract adverse events, while resilience is the ability to adapt to or recover from those adverse events, while stability is acquired in a new state.

In Chapter 3, the capability-based connectivity robustness of a CIN is measured through analyzing the inter-connection structure of a CIN (network topology). In the following discussion, it will be shown that concept of capability-based connectivity robustness can be generalized and the proposed measuring process can be used to indicate the effectiveness of a prescribed link failure coping mechanism (resilience). The robustness obtained through the inter-connection structure of a CIN will be referred to as static robustness and the robustness achieved though some dynamic coping mechanism will be referred to as dynamic robustness.

Some researchers argue that robustness is a passive design character against adverse events, and resilience is an active design character against adverse events. Hence, resilience should be pursued instead of robustness. However, robustness and resilience are not two competing concepts. They both have their own merits. For an individual system, robustness in general is much easier to achieve comparing to resilience, which usually requires "self-healing" ability. Although it can be easier for a CIN to achieve resilience since the "self-healing" ability can be achieved by the interaction between different agents, there are many real issues to be addressed for practical resilience. Use networked SUAVs as an example. One proposed way to achieve resilience is through

network re-wiring or re-configuration. This may sound easy on paper. But in reality, to achieve that, an SUAV needs to be equipped with a very powerful sensor system for service discovery, a high computation capability for information processing and information transmission routing calculation, a strong information processing capacity to cope with information surge caused by information transmission routing change and SUAV-SUAV coordination, a high battery capacity and so on. It also requires enough space and take-off weight to carry all these supporting equipments [5, 36]. Even though instead of using distributed decision making, centralized decision making can lift the computation burden on each individual SUAV, strong if not stronger information processing capabilities are still required to send the control information to each individual SUAV in a timely manner. Moreover, most of the hardware technologies that can provide those aforementioned supports with high reliability are still open research questions [36]. As a result, built-in static robustness could be a more practical solution that delivers similar effects.

In summary, robustness and resilience are not two competing concepts. It is hard to say which one is better. To have how much robustness and how much resilience built in is design dependent. In general, robustness is easier to achieve and "act" immediately with no delay; while resilience can be harder and more costly to achieve and usually incurs a delay in action upon impairments, but it has the potential to be more effective and cost-efficient considering the entire acquisition life cycle of a CIN.

In this section, the strategy of preparing substitutions for some nodes will be discussed. The core concept is to build in substitution mechanism in a network for one or more important nodes. So that when such nodes malfunction or are unreachable, their substitution nodes can take on their responsibility and sustain the CIN operation. Different from the strategy of adding a link, this strategy does not change the topology of

a CIN. This strategy does not strength the connectivity robustness between the capability critical node pair of a CIN through increasing its static robustness. This strategy responds to network impairments dynamically through substitution nodes and strengthens the connectivity robustness between the capability critical node pair by building in dynamic robustness or in other words, resilience.

The key of using this concept lies in finding the right substitution for a given node. This requires a method to quantify the effects of this strategy. The following discussion starts with an example of this strategy: assigning a deputy leader and shows that the proposed capability-based connectivity robustness evaluation process discussed in Chapter 3 can also be used to quantify the effectiveness of this link failure coping mechanism,

Assuming in a CIN, collected information is merged up to a commander for decision-making and then decision information is transferred down to each entity within the network. The key to sustain the CIN operation is to keep the commander informed during the operation and make sure its decisions can be executed at the operation field. Because of the importance of the commander's role to the entire operation, a deputy commander role is assigned to another node within the network. In case something happens to the commander that it is disconnected from the network or malfunction, the deputy commander will take on the commander role and sustain the operation (if the deputy commander node functions well). This example is illustrated in Figure 33.

In Figure 33, C node represents the commander node and C' node is the node that assigned as the deputy commander. I represents information field. The solid lines represent the information transmission between network nodes. While the dashed lines represent information transmission between field and network nodes. The dashed lines can comprise more network nodes and the interconnection between them. However, for

the purpose of illustrating this strategy, they are not shown in details. With the existence of a deputy commander, the operation can be sustained as long as there is a connection between node I and node C or node I and node C'.

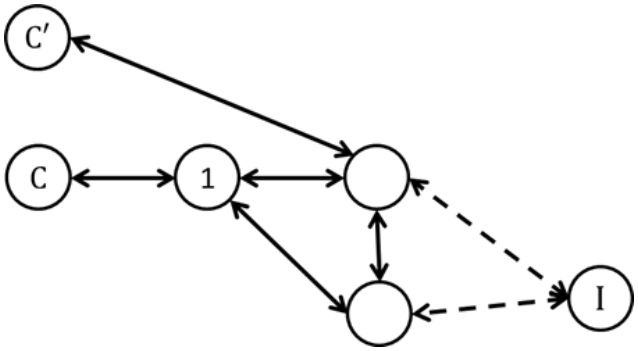


Figure 33. Example of Assigning a Deputy Commander in a CIN

(No Common Incident Nodes; Commander and Deputy Commander Not Connected)

Next, the capability-based connectivity robustness evaluation process proposed in Chapter 3 is used to evaluate the effectiveness of this strategy. Before the evaluation, some modification on a network topology is required to reflect this dynamic failure copying mechanism. As mentioned earlier, this strategy, in essence says, the CIN operation can be sustained as long as a connection exists between node I to either node C or node C', this is to shorting node C and node C'. In electrical engineering, shorting means to have the resistance between two nodes infinitely small. With this, the network can be modified by collapsing node C and node C' together as illustrated in Figure 34.

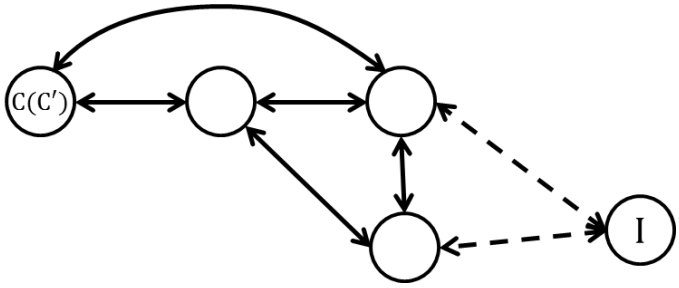


Figure 34. Modified Network Topology by Shorting Node C and Node C'

(No Common Incident Nodes; Commander and Deputy Commander Not Connected)

If there is no common node that is incident to the two collapsed nodes and the two collapsed nodes are not connected, then to quantify the effectiveness of this strategy is to measure the capability-based connectivity robustness of the modified network topology. For this example, it is to measure $\bar{m}^X_{C(C'),I}$ on the modified network, which will be denoted as $\left(\bar{m}^X_{C(C'),I}\right)_{G'}$, and to compare it with $\left(\bar{m}^X_{C,I}\right)_G$.

However, if there is any common node that is incident to the two collapsed nodes or the commander node and the deputy commander node are connected, then the effectiveness measured by the above process, which is to compare $\left(\bar{m}^X_{C(C'),I}\right)_{G'}$ to $\left(\bar{m}^X_{C,I}\right)_G$, will not yield the right result. An example of this scenario is shown in Figure 35.

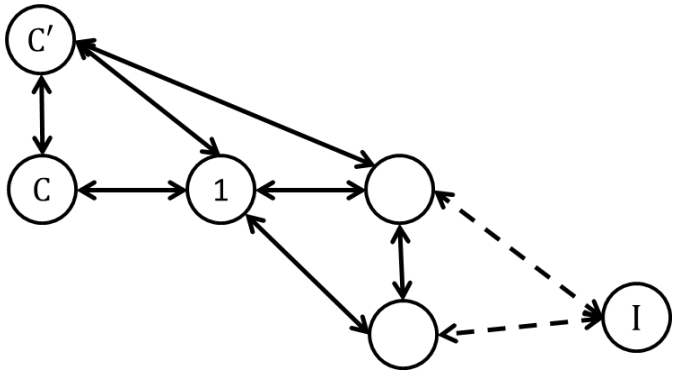


Figure 35. Example of Assigning a Deputy Commander in a CIN

(Common Incident Nodes; Commander and Deputy Commander Connected)

As shown in Figure 35, node 1 is connected to both node C and node C' and node C and node C' are connected to each other. When node C and node C' are collapsed together, there are actually two links connecting node $C(C')$ and node 1, which cannot be reflected by Figure 34. It seems that this issue can be solved by simply adding another link between node $C(C')$ and node 1 in Figure 34, which results in a non-simple graph (more than two links between a node pair) or a weighted network. However, the proposed evaluation process can only handle simple, unweighted networks. In order to use the

results obtained from Chapter 3, Figure 35 needs to be simplified into a simple, unweighted network.

This can be achieved by first calculating $\left(\bar{m}^{X}_{C(C'),I}\right)_{G'}$ without considering those extra links. Treat any extra link as a redundant link and inflate $\left(\bar{m}^{X}_{C(C'),I}\right)_{G'}$ using the method discussed in Section 3.4. This method is depicted in Figure 36. If there is M links in the original network topology, and the network resulted from this modification has M' non-redundant links, then $\left(\bar{m}^{X}_{C(C'),I}\right)_{G'}$ should be inflated by $\frac{M+1}{M'+1}$.

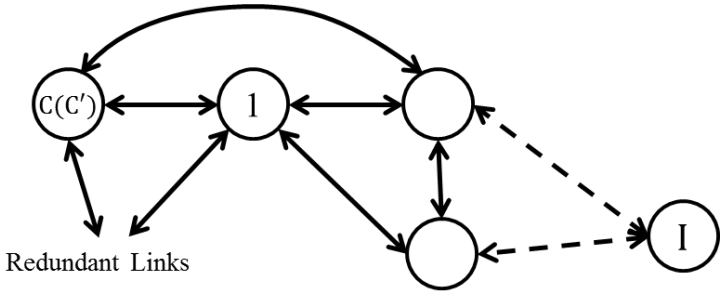


Figure 36. Modified Network Topology by Shorting Node C and Node C'

(Common Incident Nodes; Commander and Deputy Commander Connected)

In order to test the performance of the proposed evaluation method, the following experiment was carried out on the two SF and the two Rand networks. For each network, 20 node pairs were randomly selected.

Denote the two end nodes of a node pair as i and j. For each node pair, randomly select a node within the network that is different from i and j. Name this node k. Node k is used as the substitution of node j. Keep a copy of the original network topology and denote it as G. Then modify the original network topology using the network topology modification method illustrated in Figure 36. This is to collapse node j and node k, and consider all the extra links resulted from this modification as pure redundant links. The network topology obtained from this modification will be denoted as G'. Feed the two

network topologies, G and G'into the link failure simulation model developed in Chapter 3 with 10000 runs (the pure redundant links need to be removed first before feeding into the simulation model). For network G, stop each simulation run when both node pair i, j and node pair i, k are disconnected. Take the average number of link failures of the 10000 runs and denote it as $\left(\bar{m}^X_{i,j(k)}\right)_G$. For network G', stop the simulation when node pair $i, j(k)$ is disconnected. Denote the average number of link failures as $\left(\bar{m}^X_{i,j(k)}\right)_{G'}$. As mentioned earlier, $\left(\bar{m}^X_{i,j(k)}\right)_{G'}$needs to be adjusted to account for the effects of extra links resulted from node collapsing. The adjusted average number of link failures will be denoted as $\left(\bar{m}^X_{i,j(k)}\right)'_{G'}$. Compare the value $\left(\bar{m}^X_{i,j(k)}\right)'_{G'}$ to the value of $\left(\bar{m}^X_{i,j(k)}\right)_G$ and calculate their percentage differences (errors) using to the following equation, which will be denoted as $Err^{i,j(k)}_{G'}$.

$$Err^{i,j(k)}_{G'} = \frac{\left|\left(\bar{m}^X_{i,j(k)}\right)'_{G'} - \left(\bar{m}^X_{i,j(k)}\right)_G\right|}{\left(\bar{m}^X_{i,j(k)}\right)_G} \qquad 41$$

For each node pair, the above process was repeated until all the network nodes within a network that are different from node i and node j have been selected once. For each network node pair, calculate the average percentage error over all the network nodes that are different from node i and node j. The experiment results are summarized in Table 17. The estimation errors of the proposed evaluation method of all the node pairs are negligible regardless of network types. Hence, it can be concluded that, the proposed evaluation method can provide a close estimation for the effects of designating a substitution node within a network. In addition, based on the previous discussion, the effects can be directly estimated (measured) without using simulation.

Table 17. Performance of the Proposed Resilience Evaluation Method on Classical Networks

NODE PAIR INDEX	Rand_30_60	Rand_50_100	SF_30_60	SF_50_100
1	1.08%	2.32%	1.54%	1.32%
2	0.84%	2.75%	1.56%	2.19%
3	0.86%	2.16%	2.29%	1.76%
4	1.17%	0.98%	2.71%	2.10%
5	2.12%	2.06%	2.22%	2.72%
6	0.86%	1.71%	1.60%	1.19%
7	1.45%	1.21%	1.32%	1.62%
8	1.49%	1.28%	2.87%	2.24%
9	1.16%	1.21%	1.47%	2.83%
10	1.73%	1.10%	2.70%	1.43%
11	1.53%	0.83%	2.90%	2.15%
12	0.76%	1.19%	3.29%	2.10%
13	2.39%	1.08%	3.21%	1.97%
14	1.13%	1.03%	3.79%	1.89%
15	1.62%	0.78%	2.13%	2.44%
16	1.43%	1.20%	1.39%	1.11%
17	1.20%	0.89%	1.27%	2.31%
18	1.48%	1.21%	2.28%	2.33%
19	1.19%	0.89%	2.16%	2.25%
20	1.56%	1.19%	3.94%	2.41%
AVERAGE	1.35%	1.35%	2.33%	2.02%
STD	0.41%	0.54%	0.80%	0.47%

6.3 *Chapter Summary*

In this chapter, two different strategies were proposed to strengthen the capability-based connectivity robustness of a CIN. Based on the discussion in Chapter 3, the goal can be translated to strengthen the connectivity robustness between a given node pair.

The first strategy is to add a new link to an original network, which is to increase the static connectivity robustness. The second strategy is to designate substitution nodes for one or more important nodes within a network, which is to increase the dynamic connectivity robustness.

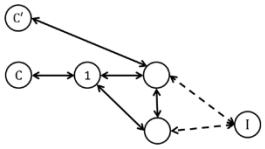
The first strategy was discussed in Section 6.1. The key of the first strategy is have a method to quickly determine the optimal position to add an additional link within a network in terms of increasing $\bar{m}_{i,j}^{X}$. $\Omega_{k,l}^{i,j}$ was proposed as an indicator for the impact of an originally non-existing link on $\bar{m}_{i,j}^{X}$. $\Omega_{k,l}^{i,j}$ can be calculated through Equation 39.

The bigger the $\Omega_{k,l}^{i,j}$ value is, the more impact the originally non-existing link has on $\bar{m}_{i,j}^{X}$. Equation 39 was validated by rejecting H_0^4.

The second strategy was discussed in Section 6.2. The key of using this strategy lies in finding the right substitution for a given node. This requires a method to quantify the effectiveness of this strategy. It was demonstrated that the capability-based connectivity robustness evaluation process proposed in Chapter 3 together with a simple network topology modification procedure could be used to quantify the effectiveness of a dynamic link failure coping mechanism as illustrated in Figure 37.

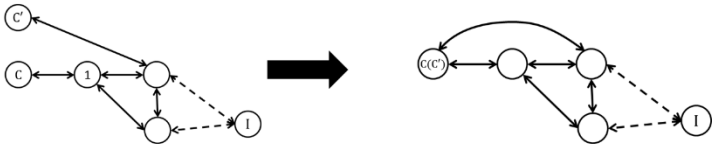
Situation 1:

- There is no common nodes that are linked to both the protected node and its substitution node and
- The protected node and its substitution node are not connected.



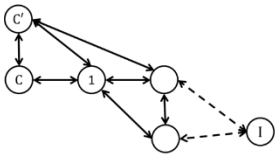
Evaluation Method:

- Modify the original network by collapsing the protected node and its substation node.
- Measure the connectivity robustness between the original target node pair using the new network topology.



Situation 2:

- There is one or more common nodes that are linked to both the protected node and its substitution node or
- The protected node and its substitution node are connected.



Evaluation Method:

- Modify the original network by collapsing the protected node and its substitution node.
- Treat any extra link resulted from the collapsing as pure redundant links.
- Measure the connectivity robustness between the original target node pair using the new network topology without any redundant links.
- Adjust the measure result by adding the effects of those pure redundant links. Assume the new network topology has M' non-redundant links, and the original network topology has M links, then the measure result should be inflated by multiplying $\frac{M+1}{M'+1}$. Need to note here, before doing the adjustment, the measure result need to be transferred into the form of "average number of link removal".

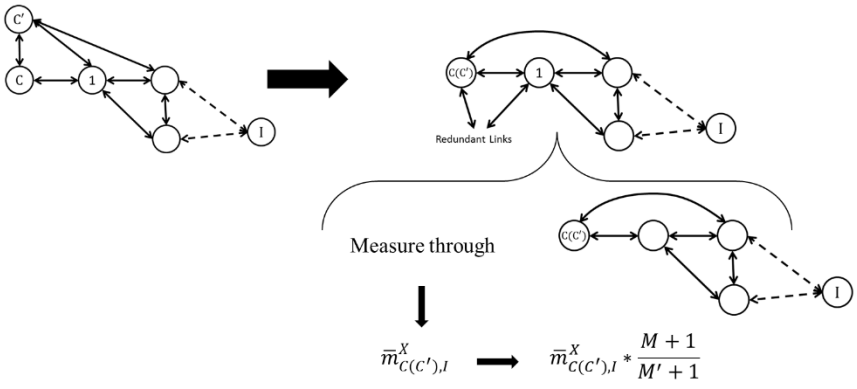


Figure 37. Summary of the Proposed Evaluation Methods

CHAPTER VII

CONGESTION CONSIDERATION

As discussed in Section 1.3, information congestion can also result in connectivity loss. The discussion in Chapter 3 is based on the assumption that congestion will not happen. In other words, the assumption says that each network node is always congestion robust. In the literatures, congestion robustness of a network node is defined as the ability of a network node to sustain information overload. It is also defined as the tendency of a network node to experience congestion. In order to avoid congestion, each network node should be equipped with enough information processing capacity.

For a CIN comprised by a given number of links and nodes, there is an upper boundary on the capability-based connectivity robustness of the CIN against link failures assuming that all the entities have enough information processing capacity. To further increase the capability-based connectivity robustness, one can add links or nodes to the architecture. It may seem that to add a network link between existing network nodes is much cheaper than adding another network node. However, only adding links between existing network nodes may be neither economically viable nor technically feasible due to the extra information processing capacity required on relevant nodes.

Network node congestion robustness depends on the information exchange dynamics within a network, which includes [2, 86, 87]:

1. Information processing behaviors
    a. Information output rate (constrained by bandwidth)
    b. Queue type, queue capacity, queue discipline, and service rate (information processing capacity)

2. Information distribution behaviors
    a. The probability of information exchange to exist between a node pair
    b. The probability of the information exchange path between a given node pair to include a particular node

In [86, 88, 89], Z. P. Hu etc. investigated the effects of network structures, packet information generation rate, routing plans, queue types and disciplines on the information exchange dynamics and congestion robustness of a given network through simulation. In Section 7.1, focus will be given to individual nodes to understand how information exchange dynamics affect the congestion behaviors of individual nodes.

Changes to any element within the above list may affect the congestion robustness of a network node. A thorough study of the relationship between information transmission dynamics and congestion robustness itself can be the content of a thesis. For the analysis purpose of this thesis, only the following scenario was considered.

Table 18. Information Transmission Scenario

Packet Output Rate	Uniform λ
Routing Strategy	Shortest Path
Queue Type	Single Server, Limited Capacity (Discard)
Queue Service Rate (Information Processing Capacity)	Uniform Γ
Queue Capacity	Uniform Υ
Queue Principle	FIFO
Exchange Matrix	Modified Uniform

For simplicity, assume all the network nodes have the same packet output rate, which is denoted as λ. Shortest path routing strategy is used. The shortest path routing strategy is a global routing strategy, which means when a packet information is generated, its transmission route will be associated with it. Next, assume each node is a single server,

First-In-First-Out (FIFO) queue with uniform information processing capacity (service rate) Γ. Moreover, assume each node has limited queue capacity Υ and all the network nodes have the same queue capacity. If a packet of information transmitted to an intermediate node that does not have enough capacity to store that packet, that packet of information will be discarded. Last, a node chooses its destination node according to a probability matrix, which is the information exchange matrix in Table 18. Details of this matrix will be given in the following discussion.

7.1 *Understanding Congestion Behaviors*

A discrete-time model is usually used to study the traffic dynamics within a complex network [86, 88-90] assuming time is slotted (discretized). Hence, a discrete-time simulation model on information transmission and processing within a network was developed based on the scenario described in Table 18. During each time slot (stamp), each network node generates a packet at rate λ and picks the destination node for this newly generated information packet according to the exchange matrix specified in Table 18. During each time slot (stamp), each node also processes and transfers information out to one of its neighbor according to the shortest path routing plan. When a packet reaches an intermediate node, it can be processed immediately (no queue and enough remaining information processing capacity), stored (enough queue space but not enough remaining information processing capacity), or discarded (not enough queue space and not enough remaining information processing capacity). When a packet reaches its destination, it is either absorbed by the destination node (the destination node can be viewed as the information sink for that information packet) or discarded if the queue of the destination node is full. Besides the settings given in Table 18, the model is also based on the following assumptions.

1. Undividable Information

   An information packet can only be sent or accepted as a whole at once during each time stamp.

2. No information addition or loss during transmission unless being absorbed or discarded.

   After an information packet is generated, unless it is absorbed or discarded, its packet size will keep constant.

3. Ignore the time required for information processing.
4. The information transmission time between any adjacent nodes is the same.

   Ignore the actual distance between two connected nodes within a network.

5. All information needs to be proceed before sent out by a network node.

   This is to simplify the information transmission and processing model by not distinguishing information by its generation source. In addition, all the information outputted by a node needs to be proceed and takes the information processing capacity of that node.

6. A node cannot choose itself as information destination.

A discrete-time information transmission and processing simulation model captures the reality well. In reliability, one of the most popular wireless information transmission and processing method is packet switching and processing. Packet switching is a digital network communication method that groups all transmitted data into suitably sized blocks, called packets, which are transferred via a medium that might be shared by multiple simultaneous communication sessions. Packet processing refers to the wide variety of algorithms that are applied to a packet data or information as it moves through the various network elements of a communication network. The reason that packet switching and processing receives widely acceptance is that it can increase network

efficiency, robustness and enables technological convergence of many applications operating on the same network. [91]

The detailed simulation process was carried out on two different network topologies: SF_10_20 and Rand_10_20. The two network topologies were generated separately using the BA SF model and the ER Rand model. Sample from a uniform distribution matrix to obtain an information exchange matrix. Each cell of this uniform distribution matrix is a uniform distribution between 0 and 1. The size of the matrix is the same as the adjacency matrix of the network. A raw information exchange matrix generated by this will have cell entries that are too small to be meaningful, which will hinder the investigation of node congestion behaviors (too distributed information traffic results in no prominent congestion behaviors). In order to correct this, set the values of those cells with probability less than or equal to 0.5 to 0. Re-normalize the matrix to make sure each row adds up to 1 and the newly obtained matrix will be used as the information exchange matrix. This process was carried out twice and two different information exchange matrices were obtained. The values of the two matrices are provided in Appendix I.

For each information exchange probability matrix, choose three different valued for the information processing capacity of each node Γ: 1, 1.5, and 2. For each Γ value, change the packet out rate λ from 0.05 to 0.95 with 0.05 increment (19 different λ values in total). Further, there are two different queue capacity settings ϒ: 5, 10. For each input combination (network topology G, information exchange matrix $P(j|i)$, information processing capacity Γ, packet output rate λ, and queue capacity Γ), the simulation was carried out with 300 time stamps. The simulation results will be discussed below. To facilitate the discussion, the following quantities will be used based on [89]:

1. Internal Information Size: $I_i(t)$

Internal information size of a network node at time stamp t.

2. Total Delivery Rate: $DL_G(t)$

   The total information packets delivery rate of the entire network at stamp t.

3. Total Packet Discard Rate: $DS_G(t)$

   The total information packets discard rate of the entire network at stamp t.

4. Total Deliver Time $\tau_G(t)$:

   The time for a packet information to be delivered within a network averaged from time stamp 0 to time stamp t.

Figure 38 and Figure 39 are two figures for the time averaged internal information size of each network node ($\bar{I}_i = \frac{\sum_{t=1}^{300} I_i(t)}{300}$) versus λ. Figure 38 is for SF_10_40 network and Figure 39 is for Rand_10_40 network. The network nodes that are not plotted in Figure 38 and Figure 39 have $\bar{I}_i$ values constantly 0. Such nodes do not serve as inter-transmission nodes for information exchange between any node pair.

Therefore, the first conclusion can be drawn is that if a network node does not serve as an inter-transmission node for information exchange between any node pair as prescribed in the information exchange probability matrix, it will never experience congestion as long as its packet output rate does not exceed its information processing capacity.

As λ increases, eventually a network nodes will experience congestion. This can be seen in both Figure 38 and Figure 39. In all the plots, initially the $\bar{I}_i$ value of each network node is close to zero and then gradually increases as λ increases. Some nodes will experience a surge of its $\bar{I}_i$ value as λ continuously increases and passes a certain value. Then their $\bar{I}_i$ values will peak to its queue capacity level or a level close to its queue capacity with little fluctuations. As λ continuously increases, eventually the $\bar{I}_i$ values of all the network nodes (except those whose $\bar{I}_i$ value is constantly 0) reach the

queue capacity level or a level close to the queue capacity little fluctuations. Based on this observation, a quantitative congestion definition for nodes with limited queue capacity can be given: node congestion occurs when its total internal information size increases to the level close to its queue capacity with little fluctuations. Name the λ value when a node starts to experience congestion as the critical λ value of this node, which will be denoted as λ_i^C.

In [89], three traffic stages were proposed to characterize the congestion behaviors of a network based on the total internal information size of the entire network. The concepts of the three states are adapted in the following discussion. Instead of using the average total internal information size of the entire network, the average internal information size of each network node ($\bar{I}_i$) will be used.

1. Light Traffic State (LTS)

   In this state, $\bar{I}_i$ remains almost unchanged or gradually increases as λ increases.

2. Moderate Congestion State (MCS)

   As λ increases, after $\lambda \geq \lambda_G^{C_L}$, the $\bar{I}_i$ value of one or more nodes starts to increase dramatically and reaches the level close to queue capacity Υ .

   $\lambda_G^{C_L}$ is the lower critical λ value of network G. It is the λ value when the first node congestion happens, and $\lambda_G^{C_L} = \min(\lambda_i^C), i \in V$.

3. Heavy Congestion State (HCS)

   If λ continuously increases, after $\lambda \geq \lambda_G^{C_U}$, the $\bar{I}_i$ values of all network nodes are close to queue capacity Υ.

   $\lambda_G^{C_U}$ is the upper critical λ value of network G. It is the λ value when the last node congestion happens, and $\lambda_G^{C_U} = \max(\lambda_i^C), i \in V$.

Those three congestion stages can be observed in both Figure 38 and Figure 39. Regardless of the network topology type and the information exchange matrix used, the higher the information processing capacity is, the higher the values of λ_{C_i} are for all network nodes. Therefore, the higher the values of $\lambda_G^{C_L}$ and $\lambda_G^{C_U}$. With everything else the same, different network topologies affect the node congestion behaviors differently. This can be seen by the difference between the set of nodes whose $\bar{I}_i$ values are greater than 0 and their λ_i^C values. Similar observation can be made for the effects of different information exchange probability matrices. By increasing the information processing capacity of a network node, its congestion can be delayed to occur at a greater λ value, which in other words is to increase its λ_i^C value. If the values of $\min(\lambda_i^C)$ and $\max(\lambda_i^C)$ are affected and increased, the onsite of MCS and HCS stages of the entire network can also be delayed.

It can be seen in both Figure 38 and Figure 39 that, it is possible that more than one nodes whose λ_i^C values are equal to $\min(\lambda_i^C)$. However, those nodes can have different $\bar{I}_i(\lambda_i^C)$ values. $\bar{I}_i(\lambda_i^C)$ is the $\bar{I}_i$ value of a network node when λ reaches λ_i^C. Under the simulation scenario prescribed by Table 18, the node that has the highest $\bar{I}_i$ value is the most prone to congestion. The reason that several nodes can have the same λ_i^C values that equal to $\min(\lambda_i^C)$ while different $\bar{I}_i(\lambda_i^C)$ values is the nature of the simulation model. As discussed earlier, the model is for packet switching information transmission and processing method and is a discrete-time simulation model. In addition, the increment of λ is 0.05. Those discreteness results in less discretion of λ_i^C values. Because of this, instead studying the exact λ_i^C value of each network node, group the λ_i^C values of network nodes into different tier. The nodes that experience congestion first as λ increases have their λ_i^C values equal or close to $\lambda_G^{C_L}$. They start to experience congestion with a sudden

jump in their $\bar{I}_i$ values. Group such nodes into the first tier. The onsite of the congestion of the first tire nodes marks the change of network congestion state from LTS to MCS. For the nodes that enter congestion state latest as λ increases, their λ_i^C values are equal or close to $\lambda_G^{C_U}$. Those nodes also enter congestion states with a surge in their $\bar{I}_i$ values. Group such nodes into the third tier. The onsite of the congestion of the third tier nodes marks the change of network congestion state from MCS to HCS. In between the onsites of MCS and HCS, there is a congestion development period as λ increaes. During the development period, the nodes that do not belong to either the first tier or the third tier start to experience $\bar{I}_i$ increase (e.g. node 8 and node 10 in Figure 39) as λ increases before experiencing congestion. Those nodes will be grouped into the second tier. Different from the nodes from the first tier and the third tier, the congestion of the second tier nodes is gradually developed.

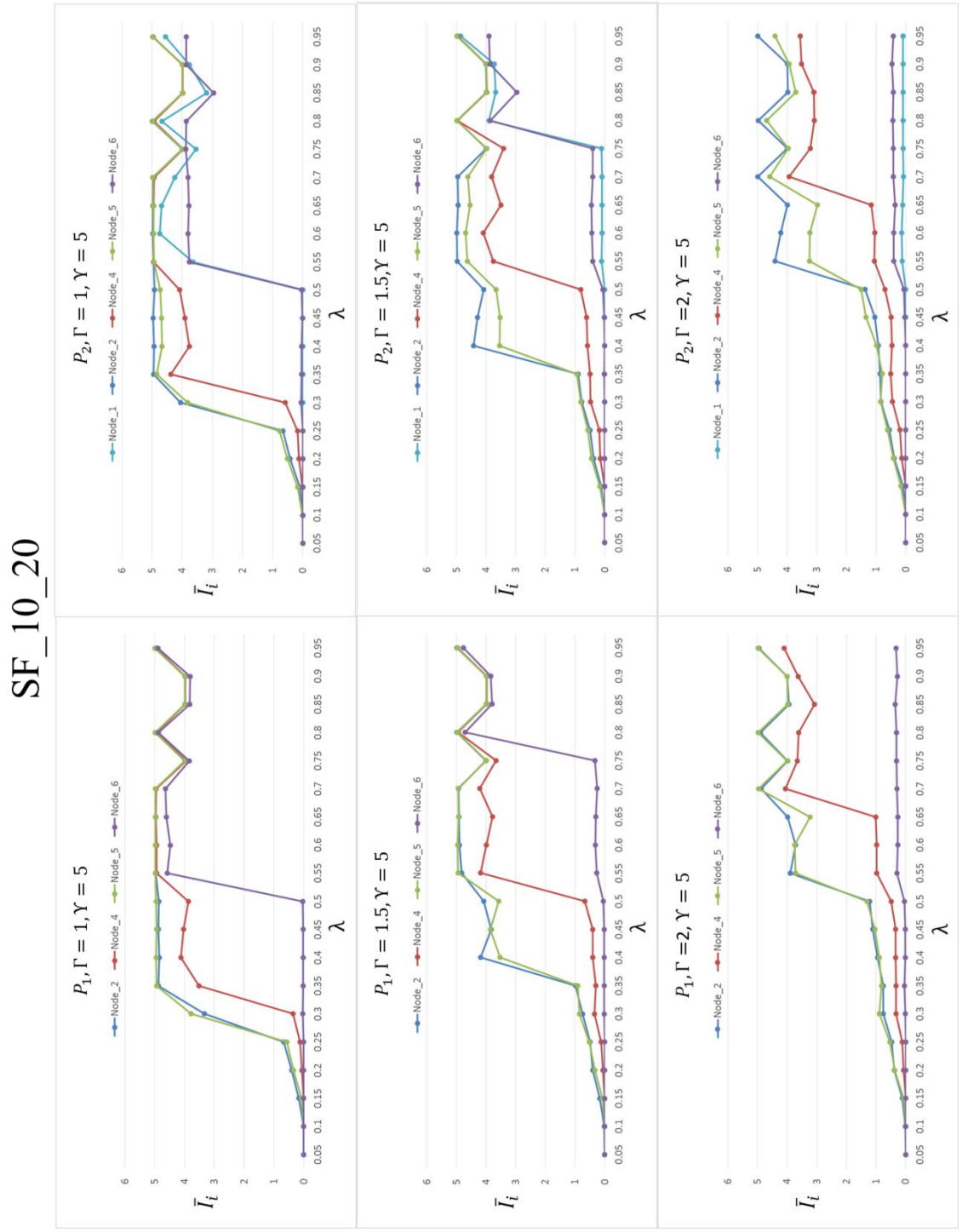


Figure 38. Plots of $\bar{I}_i$ versus λ for SF_10_20

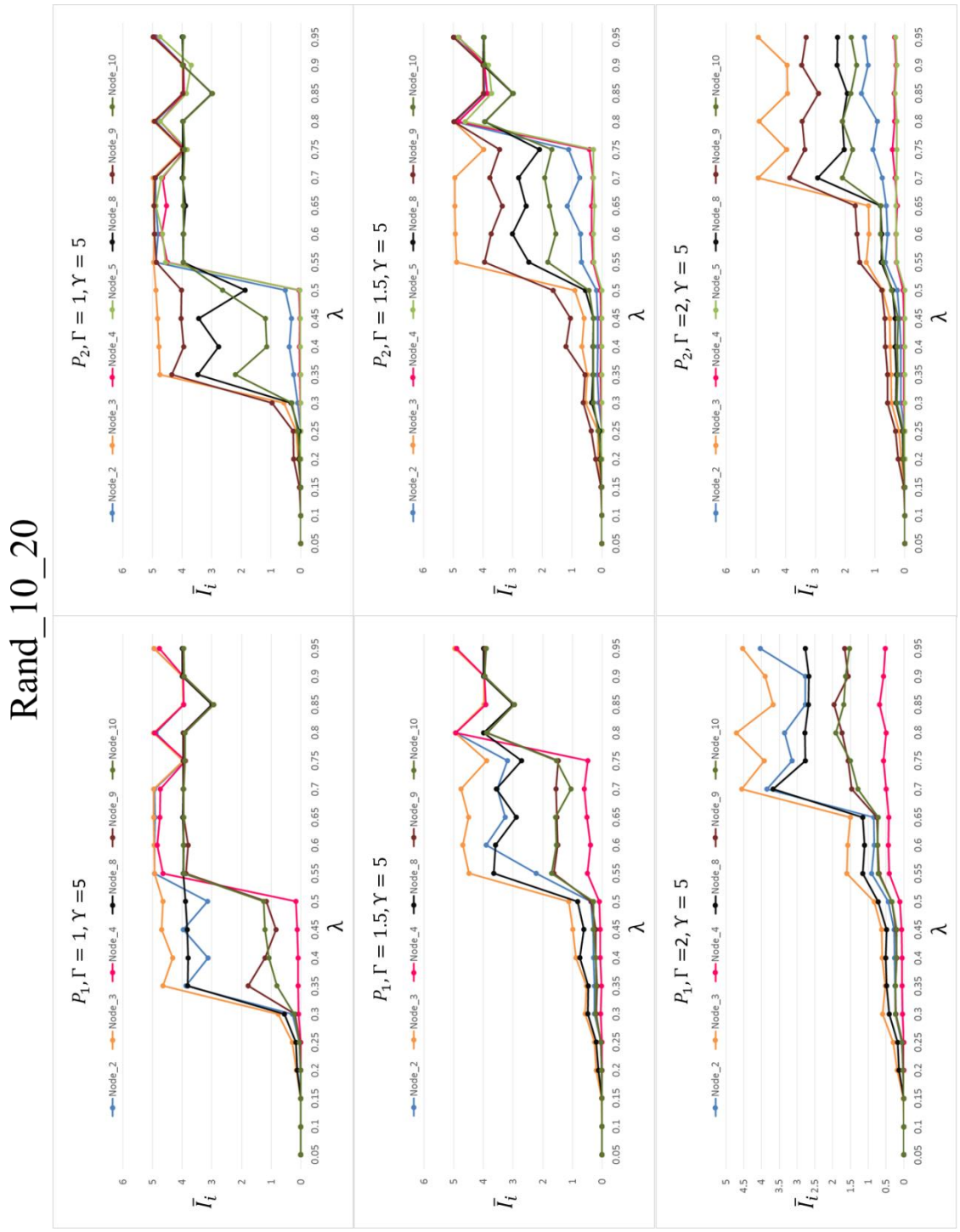


Figure 39. Plots of $\bar{I}_i$ versus λ for Rand_10_20

Next, the time averaged total information delivery rate ($\overline{DL}_G$) and the time averaged information discard rate ($\overline{DS}_G$) versus λ were plotted as shown in Figure 40 and Figure 41. The gray lines in those plots represent the summation of the $\overline{DL}^G$ and $\overline{DS}^G$, which represents the average amount of information packets generated per time stamp within the network. Since there are 10 nodes within each network, and at each time stamp each network node generates 1 packet information, theoretically the average amount of information packets generated per time stamp should be 10. As can be seen in those plots, the gray lines always stay on the level of 10, which proves the validation of this simulation. Next, look at the red lines, which are for $\overline{DL}_G$. In all the plots, for all the simulation scenarios, the trend lines for $\overline{DL}_G$ have several stages, which correspond to the LTS, MCS and HCS stages (when $\Gamma = 2$, there is no HCS stage) discussed previously. At LTS stage, $\overline{DL}_G$ stays at 10, which is the total amount of information packets generated per time stamp. This means there is no information loss at LTS stage. At MCS stage, the value of $\overline{DL}_G$ decreases to an intermediate level between 0 and 10 as congestion starts to develop within network nodes and further decreases to the lowest level at HCS stage. At each network congestion stage, the value of $\overline{DL}_G$ stays relatively constant. Since the addition of $\overline{DL}_G$ and $\overline{DS}_G$ stays constant, same observations can be made for the trend lines of $\overline{DS}_G$ (blue lines) except the value change direction as λ increases.

The above discussion reveals one of the adverse impacts of congestion, which is information loss. As can be seen from Figure 40 and Figure 41, information loss starts ($\overline{DS}_G$ greater than 0) as soon as the network enters MCS congestion stage.

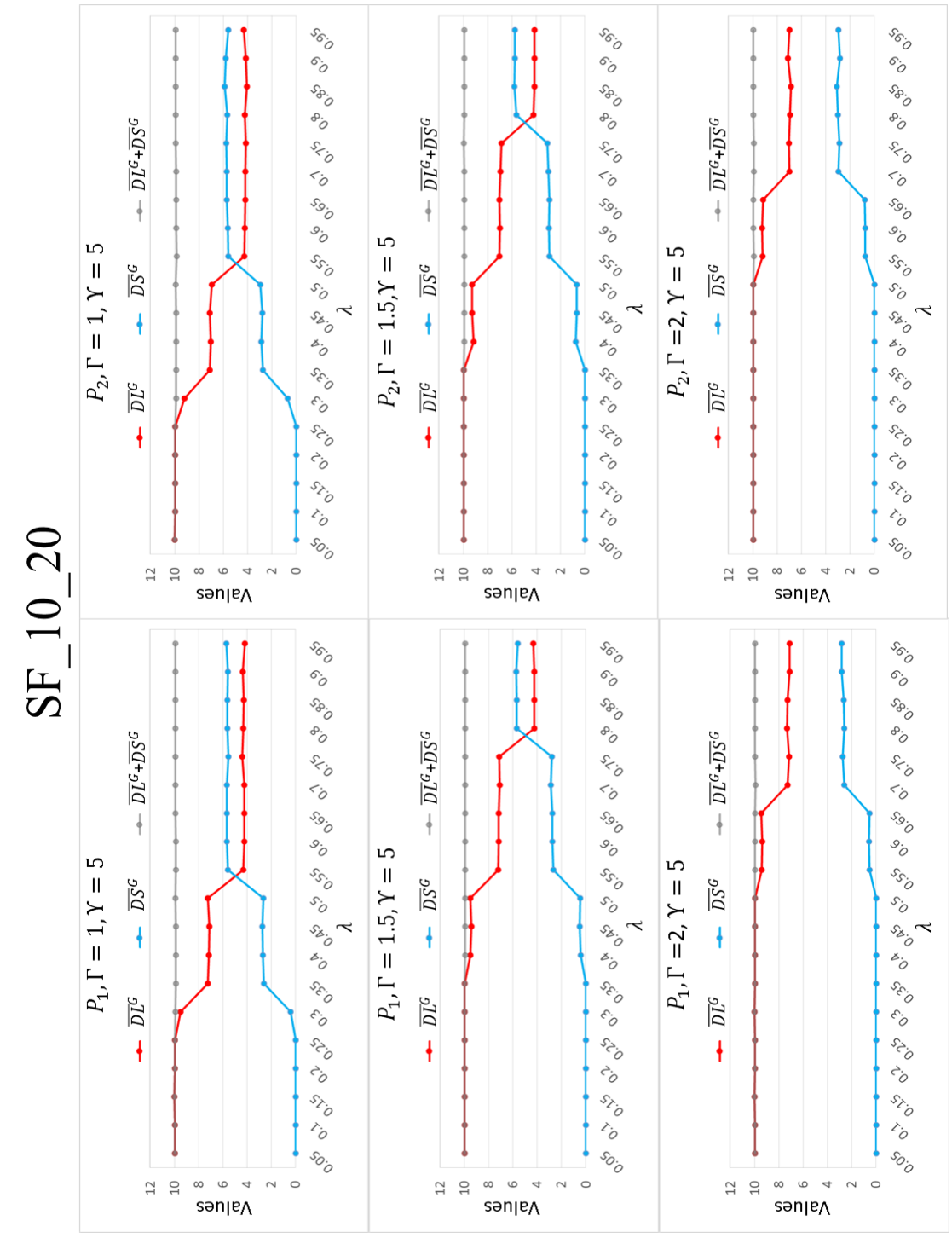


Figure 40. Co-plot of $\overline{DL^G}$, $\overline{DS^G}$versus λ for SF_10_20

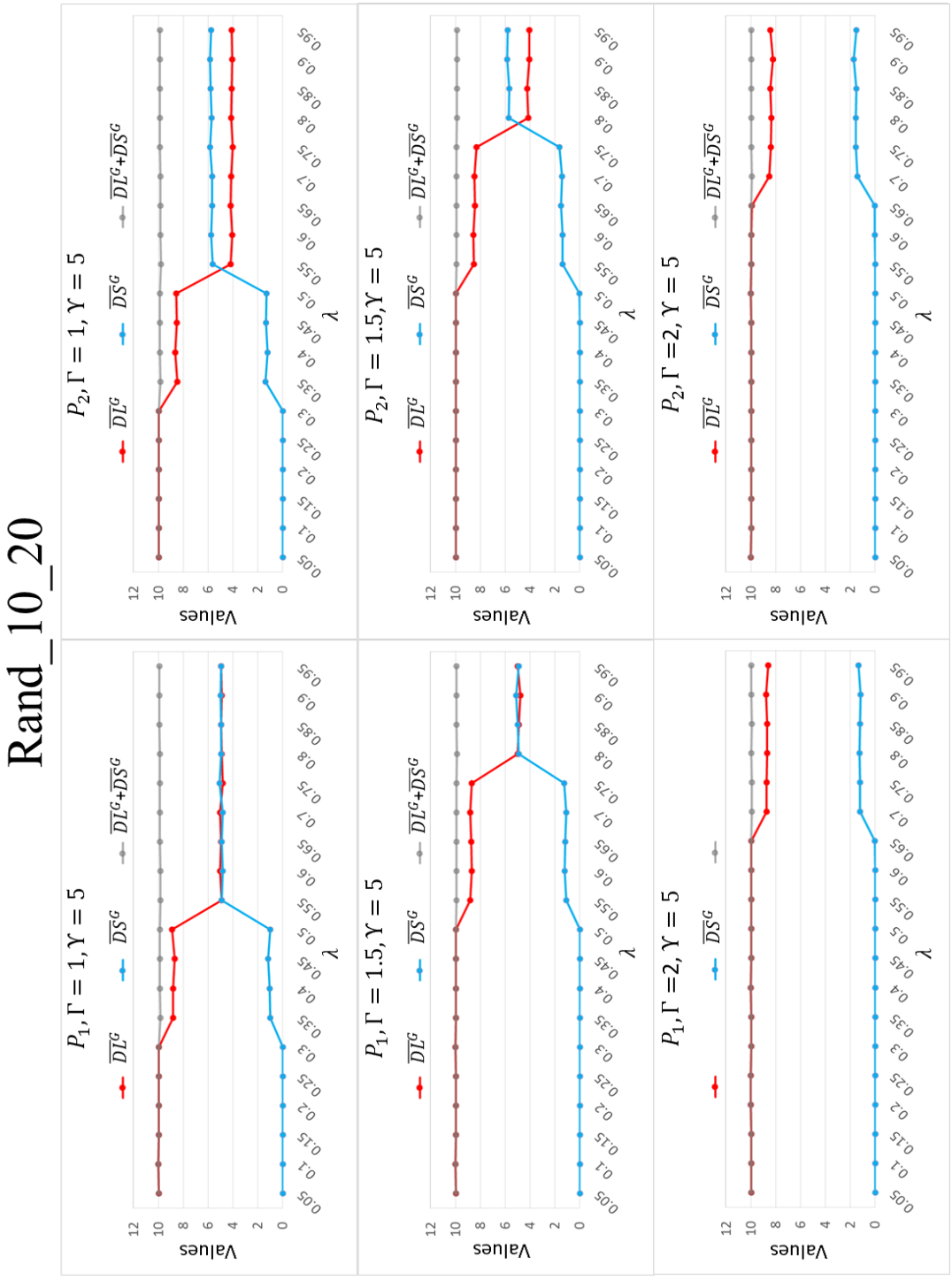


Figure 41. Co-plot of $\overline{DL^G}$, $\overline{DS^G}$ versus λ for Rand_10_20

Figure 42 is a comparison between the $\bar{I}_i$ value of each network node under the first information exchange matrix, with $\Gamma = 1$ and $\Upsilon = 5$ and 10 separately. From Figure 42, one immediate observation can be made is that, increasing the queue capacity from 5 to 10 does not affect the λ_i^C value of any network node. Moreover, the plots within Figure 42 contain the same set of network nodes and the shapes and trends of the lines within each plot are very similar to each other. This suggests very similar information exchange dynamics and congestion behaviors between $\Upsilon = 5$ scenario and $\Upsilon = 10$ scenario. From those observations, we can conclude that, when the node queue capacity is higher than a certain value, further increasing its value will not yield any additional congestion benefits. According to the plots, apparently node congestion can happen at LTS when its queue capacity is low enough. Name such kind of node congestion as early congestion. Early congestion prevents the information processing capacity of a node to be fully used and results in non-economic designs. Hence, each network node should be equipped with enough information storage space (queue capacity) to prevent early congestion.

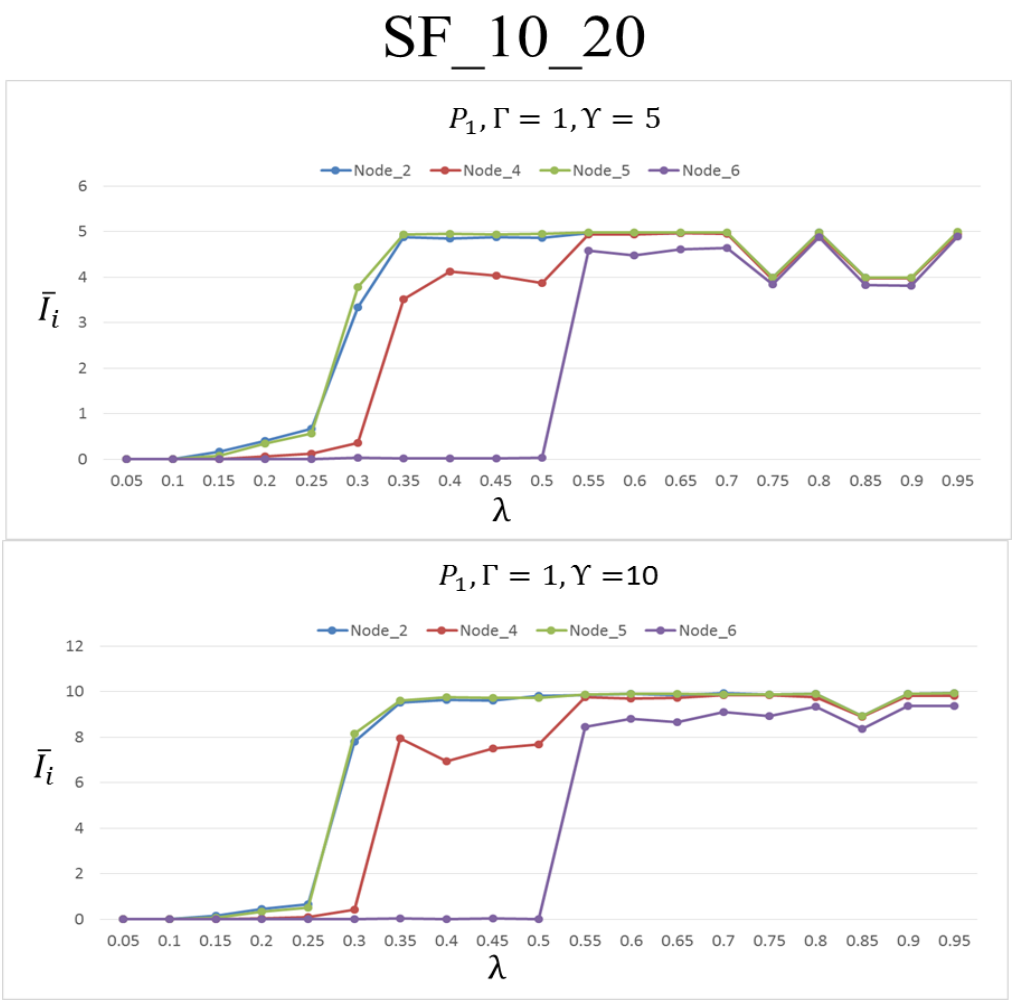


Figure 42. Plots of $\bar{I}_i$ versus λ for SF_10_20 under Different Υ Values

7.2 *Critical Information Processing Capacity*

In Section 7.1, it has been shown that, the information processing capacity Γ of network nodes can affect their congestion behaviors. In addition, the effects vary under different CINs (network topologies) and different information exchange matrices. In this section, focus will be given to deriving the required information processing capacity of each network node.

There are two guidelines based on the conclusions made in the previous section.

1. Each network node should be equipped with enough information processing capacity Γ_i so that no node will experience congestion during operation. Denote the critical Γ_i value as Γ_i^C. Γ_i^C is the minimum information processing capacity node i required to avoid congestion.
2. Γ_i^C of all the network nodes should be set in they way that minimum the network polarization value of Γ_i^C about $\min(\Gamma_i^C)$, which will be denoted as π_{Γ^C}.

For a network quantity, its polarization can be defined as following:

$$\pi_* = \frac{\left|*_{max/min} - <*>\right|}{<*>} \qquad \textbf{42}$$

where,

$<*>$ means the average of quantity *.

The first guideline is to ensure that no network node will experience congestion due to inadequate information processing capacity. The second guideline aims for economic CIN designs. The above two guidelines are rather qualitative. Based on these two design requirements, in the following discussion, a quantitative design requirement on Γ_i^C will be given. It is desired to know the required information processing capacity of each network node that can be derived from the inter-connection structure and the connectivity situation of a CIN.

7.2.1 Static Critical Information Processing Capacity

In order to achieve that, a link between Γ_i^C, network topology and node congestion robustness is needed. The relationship between a network topology and node congestion robustness is usually quantified by betweenness (centrality). The betweenness of a node is the number of paths between all the other node pairs that pass through that node [73]. The paths are decided by the information routing strategy and the network

topology deployed. Since the shortest distance routing strategy is used in this thesis (Table 18), the shortest path (σ) betweenness will be used. Node betweenness centrality can be viewed as normalized node betweenness. Equation 43 is the equation for calculating node betweenness centrality. Betweenness (centrality) can also be defined for network links. In this thesis, only node betweenness centrality will be discussed. For simplicity, in the following discussions, when referring to betweenness centrality, it means node betweenness centrality.

$$BC_k^V = \sum_{\substack{i=1 \\ i \neq k}}^{N} \sum_{\substack{j=1 \\ j \neq i \\ j \neq k}}^{N} \frac{\sigma_{ij}(k)}{\sigma_{i,j}} \quad \textbf{43}$$

From the discussion in Section 7.1, information exchange matrix can also affect the congestion behavior of a node. Betweenness (centrality) only considers the routing strategy and the topology of a network. It does not incorporate any information on the information exchange matrix used. Instead, the following augmented betweenness centrality will be used to incorporate the effects different information exchange matrices.

$$aBC_k^V = \sum_{\substack{i=1 \\ i \neq k}}^{N} \sum_{\substack{j=1 \\ j \neq i \\ j \neq k}}^{N} p(j|i)p(k|i,j) \quad \textbf{44}$$

where,

$p(j|i)$ is the probability of node i choosing node j as its information destination;

$p(k|i,j)$ is the probability for node k to be on the information transmission path from node i to node j.

Equation 44 is motivated by the probabilistic interpretation of betweenness centrality and the (traffic-aware) node utilization proposed in [92]. There are two

differences between the augmented betweenness centrality proposed here and the betweenness utilization proposed in [92]. First, the augmented betweenness centrality utilizes conditional probability to quantify the information distribution behaviors within a network. Second, the augmented betweenness centrality separates the effect of the information exchange matrix ($p(j|i)$) from the effect of the network topology and the routing strategy ($p(k|i,j)$) of a CIN. The information exchange matrix of a CIN reflects the collaboration relationship (structure) between individual entities. To separate the effects allows the effects of different information exchange matrices on network congestion behaviors to be explored.

According to Equation 44, aBC_k^V represents the probability for node k to relay information within a network. Incorporating aBC_k^V with the information transmission rate between node pairs and dividing it by the information processing capacity of node k yields a ratio that represents the average information accumulation rate within node k. Name this ratio as information congestion centrality denoted as IC_k^V. The following is the mathematical representation of IC_k^V.

$$IC_k^V = \frac{\sum_{\substack{i=1 \\ i\neq k}}^{N} \sum_{\substack{j=1 \\ j\neq i \\ j\neq k}}^{N} p(j|i)p(k|i,j)\,\lambda_{i,j} + \lambda_k}{\Gamma_k} \quad \textbf{45}$$

where,

$p(j|i)$ is the probability of node i choosing node j as its information destination;

$p(k|i,j)$ is the probability for node k to be on the information transmission path from node i to node j;

$\lambda_{i,j}$ is the information output rate from node i to node j;

Γ_k is the information processing capacity of node k.

Under uniform information output rate and processing rate scenario, Equation 45 becomes:

$$IC_k^V = \frac{\lambda}{\Gamma}(aBC_k^V + 1) \quad \textbf{46}$$

If it can be shown that IC_k^V is an indicator for node congestion robustness, it can be used to derive Γ_i^C. For each simulation case plotted in Figure 38 and Figure 39, the values of BC_k^V, aBC_k^V, IC_k^V and λ_k^C, $\bar{I}_k(\lambda_k^C)$ are summarized together in Table 19 sorted by the value of IC_k^V.

Table 19. Summary of $BC_k, aBC_k, IC_k^V, \lambda_{C_k}, \bar{I}_k(\lambda_{C_k})$ Values

(For Simulation Scenarios Plotted in Figure 38 and Figure 39)

a. SF_10_20

k	BC_k^V	aBC_k^V	$P_1, \Gamma = 1, \Upsilon = 5$			$P_1, \Gamma = 1.5, \Upsilon = 5$			$P_1, \Gamma = 2, \Upsilon = 5$		
			IC_k^V	λ_k^C	$\bar{I}_k(\lambda_k^C)$	IC_k^V	λ_k^C	$\bar{I}_k(\lambda_k^C)$	IC_k^V	λ_k^C	$\bar{I}_k(\lambda_k^C)$
2	0.22	2.28	0.82	0.25	0.65	0.77	0.35	0.98	0.82	0.50	1.21
5	0.18	2.28	0.82	0.25	0.51	0.77	0.35	0.67	0.82	0.50	1.30
4	0.12	1.28	0.68	0.30	0.44	0.76	0.50	0.91	0.74	0.65	1.01
6	0.02	0.30	0.65	0.50	0.01	0.65	0.75	0.34	--	--	--
1	0.02	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0

k	BC_k^V	aBC_k^V	$P_2, \Gamma = 1, \Upsilon = 5$			$P_2, \Gamma = 1.5, \Upsilon = 5$			$P_2, \Gamma = 2, \Upsilon = 5$		
			IC_k^V	λ_k^C	$\bar{I}_k(\lambda_k^C)$	IC_k^V	λ_k^C	$\bar{I}_k(\lambda_k^C)$	IC_k^V	λ_k^C	$\bar{I}_k(\lambda_k^C)$
5	0.18	2.43	0.86	0.25	0.77	0.80	0.35	0.94	0.86	0.50	1.51
2	0.22	2.33	0.83	0.25	0.66	0.78	0.35	0.89	0.83	0.50	1.37
4	0.12	1.46	0.74	0.30	0.58	0.82	0.50	0.81	0.80	0.65	1.16
6	0.02	0.43	0.72	0.50	0.04	0.72	0.75	0.41	--	--	--
1	0.02	0.10	0.55	0.50	0	0.55	0.75	0.11	--	--	--
3	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0

b. Rand_10_20

k	BC_k^V	aBC_k^V	$P_1, \Gamma = 1, \Upsilon = 5$			$P_1, \Gamma = 1.5, \Upsilon = 5$			$P_1, \Gamma = 2, \Upsilon = 5$		
			IC_k^V	λ_k^C	$\bar{I}_k(\lambda_i^C)$	IC_k^v	λ_k^C	$\bar{I}_k(\lambda_i^C)$	IC_k^V	λ_k^C	$\bar{I}_k(\lambda_k^C)$
5	0.16	1.71	0.81	0.30	0.76	0.90	0.50	1.13	0.88	0.65	1.50
2	0.12	1.49	0.75	0.30	0.56	0.83	0.50	0.83	0.81	0.65	1.16
4	0.04	1.08	0.62	0.30	0.29	0.69	0.50	0.38	0.68	0.65	0.84
6	0.12	0.93	0.97	0.50	1.16	0.97	0.75	1.49	--	--	--
1	0.06	0.91	0.96	0.50	1.26	0.96	0.75	1.53	--	--	--
3	0.04	0.47	0.74	0.50	0.17	0.74	0.75	0.49	--	--	--
7	0	0	0	0	0	0	0	0	0	0	0
8	0.06	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0

k	BC_k^V	aBC_k^V	$P_2, \Gamma = 1, \Upsilon = 5$			$P_2, \Gamma = 1.5, \Upsilon = 5$			$P_2, \Gamma = 2, \Upsilon = 5$		
			IC_k^V	λ_k^C	$\bar{I}_k(\lambda_i^C)$	IC_k^v	λ_k^C	$\bar{I}_k(\lambda_i^C)$	IC_k^V	λ_k^C	$\bar{I}_k(\lambda_i^C)$
5	0.12	1.75	0.83	0.30	0.97	0.92	0.50	1.64	0.89	0.65	1.66
2	0.16	1.58	0.77	0.30	0.56	0.86	0.50	0.91	0.84	0.65	1.22
4	0.12	1.11	0.63	0.30	0.33	0.70	0.50	0.55	--	--	--
6	0.06	0.99	1.00	0.50	0.3	1.00	0.75	1.69	--	--	--
1	0.04	0.74	0.87	0.50	0.53	0.87	0.75	1.12	--	--	--
3	0.04	0.32	0.66	0.50	0.06	0.66	0.75	0.42	--	--	--
7	0.06	0.27	0.64	0.50	0.03	0.64	0.75	0.29	--	--	--
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0

The following observations can be made from Table 19. First, BC_k^V cannot capture the information transmission criticalness of node k relaying information within a network. However, under uniform packet output and processing rates, aBC_k^V can.

This conclusion is made by the following observations. Firs, compare the values of BC_k^V and aBC_k^V to the values of λ_k^C and $\bar{I}_k(\lambda_k^C)$. As can be seen in the four sub-tables, under uniform packet output and processing rates, it is always the case that the network nodes with bigger aBC_k^V values have smaller λ_k^C values regardless of the simulation setting. In addition, for the network nodes with the same λ_k^C value, the ones with higher aBC_k^V values also have the higher $\bar{I}_k(\lambda_k^C)$ values. While the same observation cannot be made for aBC_k^V. Second, if a node does not serve as intermediate node for any information transmission, its information processing capacity will only be used for processing the packet generated by itself. As long as the packet output rate of such a node does not exceed its information processing rate, it will not experience congestion and its $\bar{I}_k(\lambda_k^C)$ value should always be zero. Since in all the simulation settings, the packet output rate is smaller than the information processing rete of a node, the $\bar{I}_k(\lambda_k^C)$ values of the non-relaying nodes should be zero. As can be observed from the four sub-tables, the aBC_k^V values of the nodes whose $\bar{I}_k(\lambda_k^C)$ values are zero are also zero. While the same observations cannot be made for BC_k^V. Therefore, comparing to BC_k^V, aBC_k^V is a better quatity that captures the criticalness of a node relaying information within a network.

If keep everything else the same but assume information can be sent and processed in a more continuous manner, then a bigger λ_k^C value can be observed for each network node under the same simulation setting. It is because for packet switching and processing, if a packet information exceeds the size of the remaining information

processing capacity of a network node, the packet cannot be processed and have to wait in queue until there is enough available information processing capacity.

According to the physical meaning of Equation 46 and the definition of node congestion, under continuous situation information transmission and processing situation, a network will enter into MCS from LTS when the IC_k^V value of **any network node** exceeds one. Hence in order to avoid the onsite of MCS within a network, IC_k^V should not exceed one during the entire CIN operation period. With this, the following relationship can be developed to avoid the onsite of MCS within a network.

$$IC_k^V = \frac{\max(\lambda)}{\Gamma}(aBC_k^V + 1) \leq \frac{\lambda_k^C}{\Gamma}(aBC_k^V + 1) = 1 \quad \forall k \in V \qquad \textbf{47}$$

where,

$\max(\lambda)$ is the maximum information output rate during a CIN operation.

Rewrite Equation 47 as shown in Equation 48. Clearly, $\Gamma_k^C = \max(\lambda)\,(aBC_k^V + 1)$.

$$\Gamma \geq \Gamma_k^C = \max(\lambda)\,(aBC_k^v + 1) \qquad \textbf{48}$$

Equation 48 seems fine except that it does not consider the discreteness of packet switching and processing method. For packet switching and processing method, with everything else the same, using packet switching and processing method would result in lower λ_k^C values for network nodes comparing to using continuous packet switching and processing method. That means for packet switching and processing method, a network would enter into MCS from LTS when the IC_k^v value of **any network node** exceed a value smaller than one. This can be confirmed by observing the first row of each sub-table within Table 19. Therefore, for packet switching and processing method, Equation 47 and Equation 48 should be modified as following.

$$IC_k^V = \frac{\max(\lambda)}{\Gamma}(aBC_k^V + 1) \leq \frac{\lambda_i^C}{\Gamma}(aBC_k^V + 1) < 1 \quad \forall k \in V \qquad \textbf{49}$$

$$\Gamma_k^C > \max(\lambda)\,(aBC_k^v + 1) \quad \textbf{50}$$

or

$$\Gamma_k^C = [\max(\lambda)\,(aBC_k^V + 1) + \varepsilon_k] \quad \textbf{51}$$

For uniform Γ situation, the critical Γ value can be decided as

$$\Gamma_C^G = \max(\Gamma_k^C) \quad \textbf{52}$$

If information can be sent and processed in a continuous manner, then setting Γ_k to Γ_k^C will meet the two design guidelines proposed at the beginning of this section simultaneously. For packet switching and processing, the task now becomes to select the ε_k value for each network node so that to meet the two design guidelines at the same time. The exact value ε_k depends on the packet output rate and is a more complex issue that will not be addressed in the content of this thesis.

$\max(\lambda)\,(aBC_k^V + 1)$ can be viewed as the minimum required information processing capacity of each network node for avoiding congestion regardless of the information transmission and processing method used. For packet switching and processing, it is easy to see from Equation 50 and Equation 51. However, why it is also the case if continuous information switching and processing?

That is because regardless of the method used, the above discussion assumes no network impairments. In other words, the network topology used to calculate aBC_k^V does not change. When a network is impaired, the information traffic within that network will experience redistribution. If network nodes are impaired or link impairments that result in network disconnectness, then the network will also experience decease in the total information generation rate. Information redistribution can induce congestion in an originally not congested node if more information has to be relayed through that node. It

can also alleviate the information transmission burden placed on network nodes due to decrease in total information generation rate.

Those are two competing effects on node congestion situations within a network. Normally under light network impairments situation, the effect of information redistribution dominates. As impairment scale increases, the effect of the decrease in total information generation rate starts to be more dominant. To show this, the SF_10_20 network was fed into the link failure simulation model with 1000 runs. Within each run, randomly select and remove a network link one at a time until node 3 and node 6 disconnected. After each link failure, calculate and document the aBC_k^V value of each network node. The first information exchange matrix as shown in Table 21 will be used.

For run number r, denote the aBC_k^V value of node k corresponding to the number of link failures that does not result in discussion between node 3 and node 6 as $(aBC_k^V)_r^{m_{3,6}}$. Taking the average of $(aBC_k^V)_r^{m_{3,6}}$ over the 1000 runs yields $(\overline{aBC}_k^V)^{m_{3,6}}$ ($m_{3,6}$ represents the number of impaired links that does not result in discussion between node 3 and node 6.). In Figure 43, the $(\overline{aBC}_k^V)^{m_{3,6}}$ values of all the 10 network nodes are plotted together against the values of $m_{3,6}$.

According to Equation 46, under uniform packet output and processing rate, aBC_k^V can be used to compare the congestion robustness of network nodes. If λ and Γ stay constant along network impairment process, then aBC_k^V can also be used to compare the congestion robustness of network nodes along network impairment process.

As can be seen in Figure 43, except node 4 and node 5, the $(\overline{aBC}_k^V)^{m_{3,6}}$ values of all the other nodes firstly increase and then decrease as $m_{3,6}$ increases. $(\overline{aBC}_4^V)^{m_{3,6}}$ slightly decreases when $m_{3,6}$ is very small and then increases back to its original level before decreases to zero. $(\overline{aBC}_5^V)^{m_{3,6}}$ first decreases at a very small rate and then decreases to zero at a much bigger rate when the $(\overline{aBC}_k^V)^{m_{3,6}}$ values of the other nodes

start to decrease. This observation supports the previous discussion on the two competing effects of network impairments on node congestion situations.

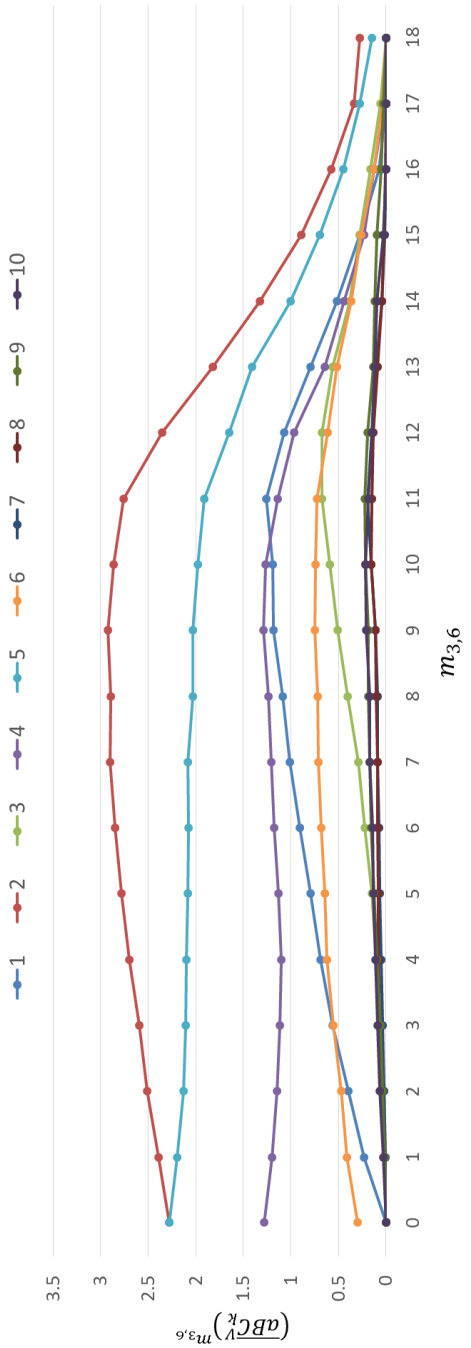


Figure 43. Plot of $\left(\overline{\boldsymbol{aBC}}_{\boldsymbol{k}}^{\boldsymbol{V}}\right)^{\boldsymbol{m}_{3,6}}$ vs. $\boldsymbol{m}_{3,6}$ for SF_10_20

So even for continuous information transmission and processing, if each network node within a network is equipped with $\Gamma_C^G = \max(\lambda)\,(aBC_k^V + 1)$ information processing capacity, no network node will experience information congestion if the network is not impaired. However, under network impairments, it is very possible that one or more network nodes will experience congestion due to the effect of information traffic redistribution. Therefore, $\max(\lambda)\,(aBC_k^V + 1)$ is the minimum required information processing capacity of each network node for avoiding congestion regardless of the information transmission and processing method used.

7.2.2 Dynamic Critical Information Processing Capacity

Continue the discussion under the previous link failure experiment. To decide how much more information processing capacity node k needs to avoid congestion under network impairments, the value of $\max(aBC_k^V)$ is needed. And Equation 51 becomes

$$\Gamma_k^C = \max(\lambda)\,[\max(aBC_k^V) + 1] + \varepsilon_k \quad \textbf{53}$$

However, in reality, it may not necessary to obtain the $\max(aBC_k^V)$ value over the entire $m_{i,j}$ value field. What needed is the $\max(aBC_k^V)$ value over a practical $m_{i,j}$ value field in terms of CIN operation. "Practical" means the probability of node pair i,j to stay connected with $m_{i,j}$ number of link failures during the operation period of a CIN is significant. Denote such a probability as $p(m_{i,j})$ and its significant level as $p^0(m_{i,j})$. Hence the practical field of $m_{i,j}$ is from 0 to some value determined by p^0, and denote the $m_{i,j}$ value corresponds to p^0 as $\left(m_{i,j}\right)_{p^0}$. $p(m_{i,j})$ can be obtained by taking the product of the probability of exact $m_{i,j}$ number of link failures during operation, $p_1(m_{i,j})$, and the probability of node pair i,j to stay connected after $m_{i,j}$ number of link failures, $p_2(m_{i,j})$.

$$p(m_{i,j}) = p_1(m_{i,j})p_2(m_{i,j}) \quad \textbf{54}$$

To decide the practical $m_{i,j}$ value field is to find out all the $m_{i,j}$ values that satisfy:

$$p(m_{i,j}) \geq p^0 \quad \textbf{55}$$

The probability of a link to fail during operation is determined by its reliability. Assume the reliability distributions of all the network links are iid and follow an exponential distribution with scale parameter Θ equals MTTF. Figure 44 is the PDF and CDF of an exponential distribution with scale parameter $\Theta = 8$ (hours). The PDF is the probability of the "life time length" of a link under the previous assumption. The CDF represents the probability of a link failure after operating length t.

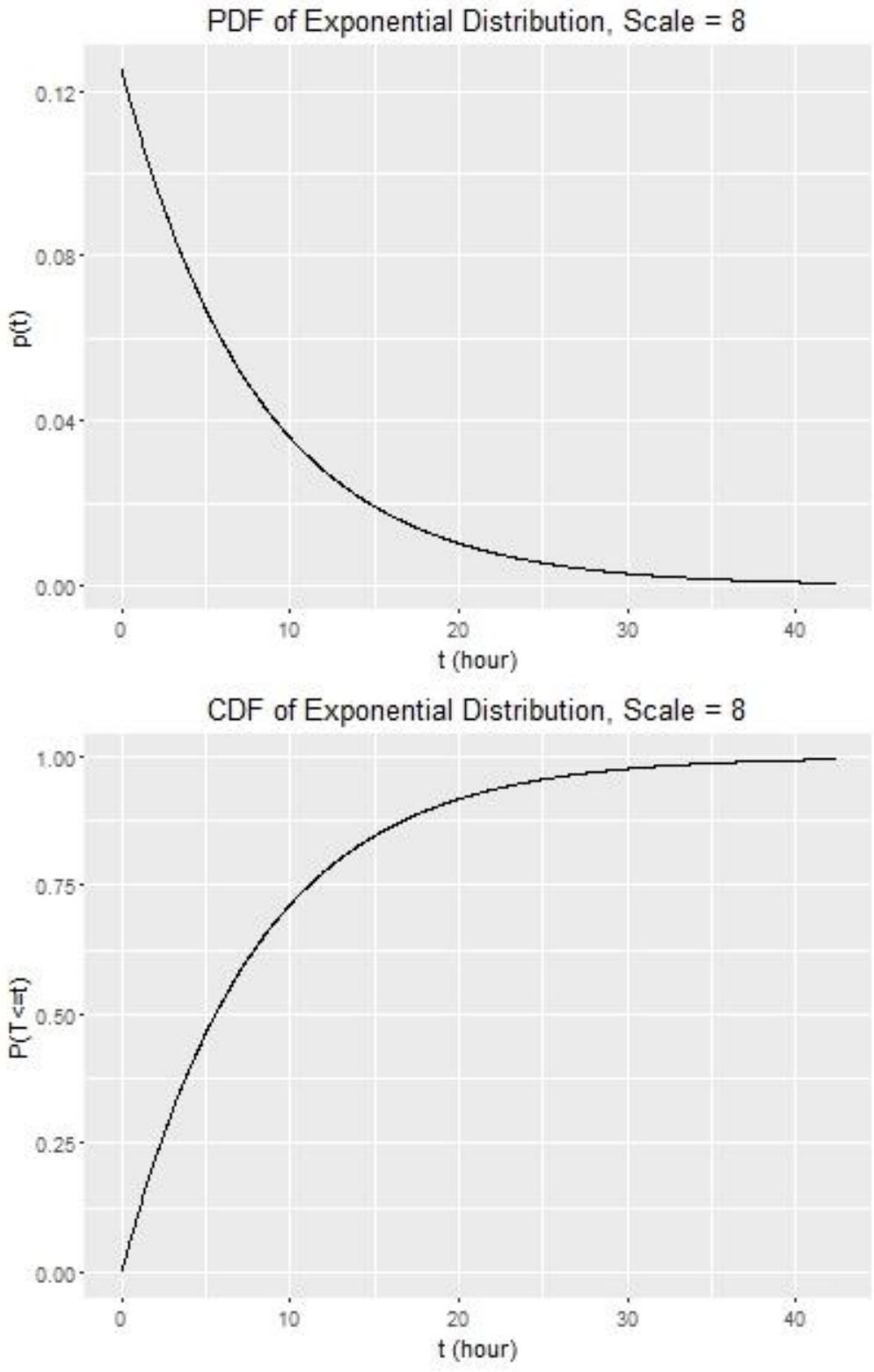


Figure 44. PDF and CDF of an Exponential Distribution (Scale = 8)

Assume an operation has length T, then the number of exact $m_{i,j}$ link failures during operation is a passion distribution with rate $\frac{T}{\Theta}$.

$$p_1(m_{i,j}) = p_{Possion(\frac{T}{\Theta})}(m_{i,j}) \quad \textbf{56}$$

Review the definition of $m_{i,j}$, which is the number of link fails that will not result in node pair i,j disconnection. Hence, the probability node pair i,j to stay connected after $m_{i,j}$ number of link failures can be expressed as following.

$$p_2(m_{i,j}) = 1 - P_{i,j}^X(m_{i,j}^X \leq m_{i,j}) \quad \textbf{57}$$

In Equation 57, $P_{i,j}^X(m_{i,j}^X \leq m_{i,j})$ represents the probability of node pair i,j to disconnect after $m_{i,j}$ link failures. It is the CDF of the probability of node pair i,j to disconnect at exact $m_{i,j}$ link failures. Equation 57 in essence says, the probability of node pair i,j to stay connected after $m_{i,j}$ link failures is equal to 1 minus the probability of node pair i,j to disconnect after $m_{i,j}$ link failures.

In order to obtain p_2, the key is to find $p_{i,j}^X$ (PMF) or $P_{i,j}^X$ (CDF). Unlike p_1, it is usually hard to obtain the exact distribution for either $p_{i,j}^X$ or $P_{i,j}^X$ given an arbitrary network topology. Here, a method based on the principle of maximum entropy to estimate the distribution of $p_{i,j}^X$ is proposed. This method also uses the $ER_{i,j}$ based $\bar{m}_{i,j}^X$ estimation method proposed in Chapter 3.

Given a network G with N nodes, using the method proposed in Chapter 3, we can easily obtain close estimations for $(\bar{m}_{i,j}^X)_{FULL}$ and $\bar{m}_{i,j}^X$. Now there are three pieces of information handy for estimating $p_{i,j}^X$: its central or typical value ($\tilde{\bar{m}}_{i,j}^X$), its sample space ($[0, (\tilde{\bar{m}}_{i,j}^X)_{FULL}]$) and it is a discrete probability distribution (rounding of $(\tilde{\bar{m}}_{i,j}^X)_{FULL}$]) is necessary if its value is not integer).

Next, the principle of maximum entropy will be used to construct the PMF of $p_{i,j}^{X}$. The principle of maximum entropy states that when one searches for a probability distribution that satisfies some constrains (evidence or information) known, the correct one to choose is the one that maximizes the uncertainty or entropy subject to these constrains [93-95]. The maximum entropy distribution that satisfies the three pieces of information known is Poisson distribution. Hence, the PMF equation of $p_{i,j}^{X}$ is shown below.

$$p_{i,j}^{X} = p_{pisson(S)}\left(m_{i,j}^{X}\right) = \frac{S^{m_{i,j}^{X}} e^{-S}}{m_{i,j}^{X}!} \quad \textbf{58}$$

where,

$$S = \tilde{\tilde{m}}_{i,j}^{X}$$

$$m_{i,j}^{X} \in \left[0,1,2,\dots,\left(\tilde{\tilde{m}}_{i,j}^{X}\right)_{FULL}\right]$$

For the SF_10_20 network and node pair 3,6, the values for $\tilde{\tilde{m}}_{i,j}^{X}$ and $\left(\tilde{\tilde{m}}_{i,j}^{X}\right)_{FULL}$ is shown in Table 20. The corresponding maximum entropy distribution is shown in Figure 45.

Table 20. $\tilde{\tilde{m}}_{i,j}^{X}$ and $\left(\tilde{\tilde{m}}_{i,j}^{X}\right)_{FULL}$ of SF_10_20

$\tilde{\tilde{m}}_{3,6}^{X}$	$\left(\tilde{\tilde{m}}_{3,6}^{X}\right)_{FULL}$
12.3	38

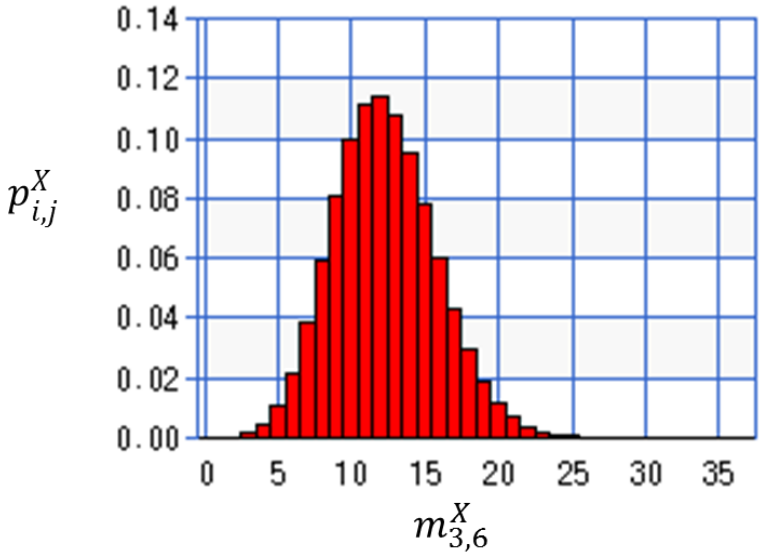


Figure 45. The Maximum Entropy Distribution of $p_{3,6}^X$

On the other hand, as can be seen in Figure 45, the probability for both tails of the distribution are very small ($< 2\%$). Those $m_{i,j}^X$ numbers are very unlikely to happen in real networks. Therefore, the following modifications were proposed for the PMF shown in Equation 58. That is to set the probability to zero if it is smaller than 2% and re-normalize the values to sum up to one.

$$p_{i,j}^X = \frac{p_{pisson(S)}^{0.02}(m_{i,j}^X)}{\sum_0^{(\widetilde{\widetilde{m}}_{i,j}^X)_{FULL}} p_{pisson(S)}^{0.02}(m_{i,j}^X)} \quad \textbf{59}$$

where,

$$S = \widetilde{\widetilde{m}}_{i,j}^X$$

$$m_{i,j}^X \in \left[0,1,2,\dots,(\widetilde{\widetilde{m}}_{i,j}^X)_{FULL}\right]$$

$$p_{pisson(S)}^{0.02}(m_{i,j}^X) = \begin{cases} 0 & \text{if } p_{pisson(S)}(m_{i,j}^X) < 0.02 \\ p_{pisson(S)}(m_{i,j}^X) & \text{otherwise} \end{cases}$$

In Figure 46, the blue lines are the PMF and CDF of $p_{3,6}^X$ constructed using Equation 59 and the red lines are the PMF and CDF of $p_{3,6}^X$ constructed using the

simulation results. As can be seen in Figure 46, the blue lines resembles the shape of the red lines very well, which means that Equation 59 can provide a good estimation for $p_{i,j}^{X}$.

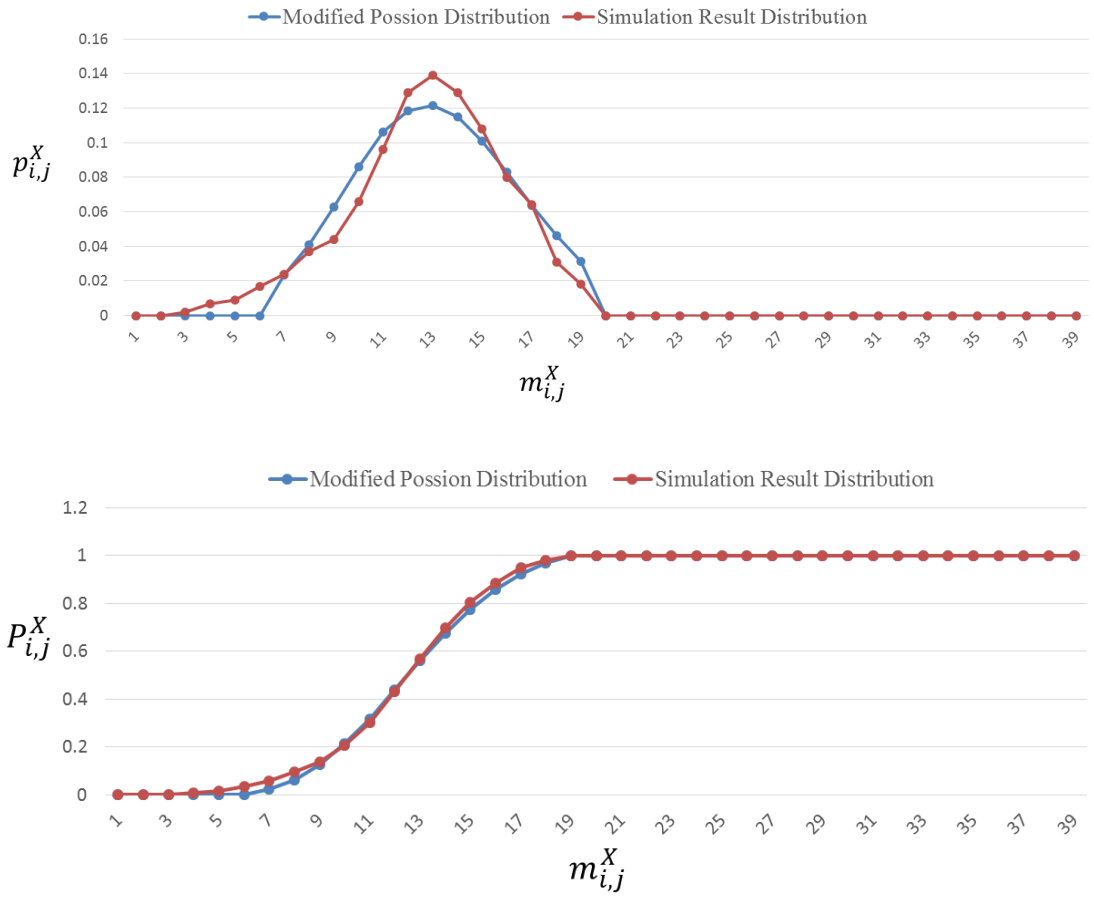


Figure 46. PDF and CDF of $p_{3,6}^{X}$ Constructed by Equation 59 and Simulation Results

With Equation 56, Equation 57 and Equation 59, Equation 54 can be rewritten as below. Figure 47 is an example plot for $p(m_{3,6})$ with $T = 10$ (hours) and $\Theta = 8$ (hours).

$$p(m_{i,j}) = p_{Possion\left(\frac{T}{\Theta}\right)}(m_{i,j})\left[1 - P_{i,j}^{X}(m_{i,j}^{X} \le m_{i,j})\right] \quad \textbf{60}$$

where,

$$p_{i,j}^{X} = \frac{p_{pisson(S)}^{0.02}(m_{i,j}^{X})}{\sum_{0}^{(\tilde{\tilde{m}}_{i,j}^{X})_{FULL}} p_{pisson(S)}^{0.02}(m_{i,j}^{X})}$$

$$S = \widetilde{\widetilde{m}}_{i,j}^{X}$$

$$m_{i,j}^{X} \in \left[0,1,2,\dots,\left(\widetilde{\widetilde{m}}_{i,j}^{X}\right)_{FULL}\right]$$

$$p_{pisson(S)}^{0.02}\left(m_{i,j}^{X}\right) = \begin{cases} 0 & \text{if } p_{pisson(S)}\left(m_{i,j}^{X}\right) < 0.02 \\ p_{pisson(S)}\left(m_{i,j}^{X}\right) & \text{otherwise} \end{cases}$$

Θ is the MTTF of a link

T is the length of a CIN operation

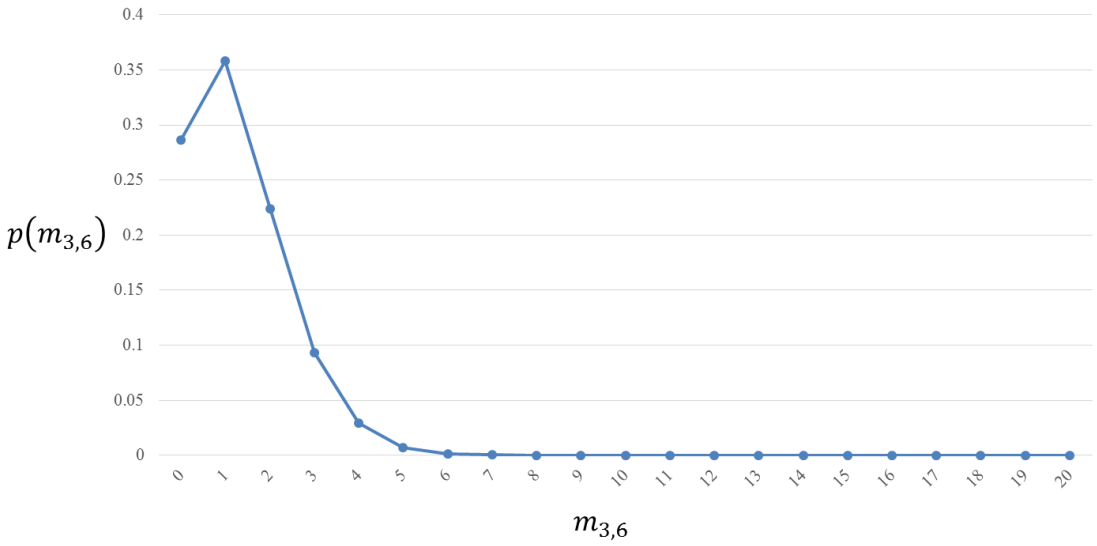


Figure 47. Example Plot of $p(m_{3,6})$ with $T = 10$ (Hours) and $\Theta = 8$ (Hours)

Go back to the discussion at the beginning of this section. The reason that the distribution of $p(m_{i,j})$ is needed is to obtain a practical $m_{i,j}$ value field, $\left[0, \left(m_{i,j}\right)_{p^0}\right]$, for deciding the value of$\max(aBC_k^V)$. According to Equation 53, the higher the value of $\max(aBC_k^V)$ the more information processing capacity is needed for a network node. Hence, it is desired to have a small $\max(aBC_k^V)$ value. To restrict the value of $\max(aBC_k^V)$ can be achieved by controlling the value of $\left(m_{i,j}\right)_{p^0}$. In order to show the

effects of $(m_{i,j})_{p^0}$ on $\max(aBC_k^V)$, overlay Figure 43 and Figure 47 together as shown below.

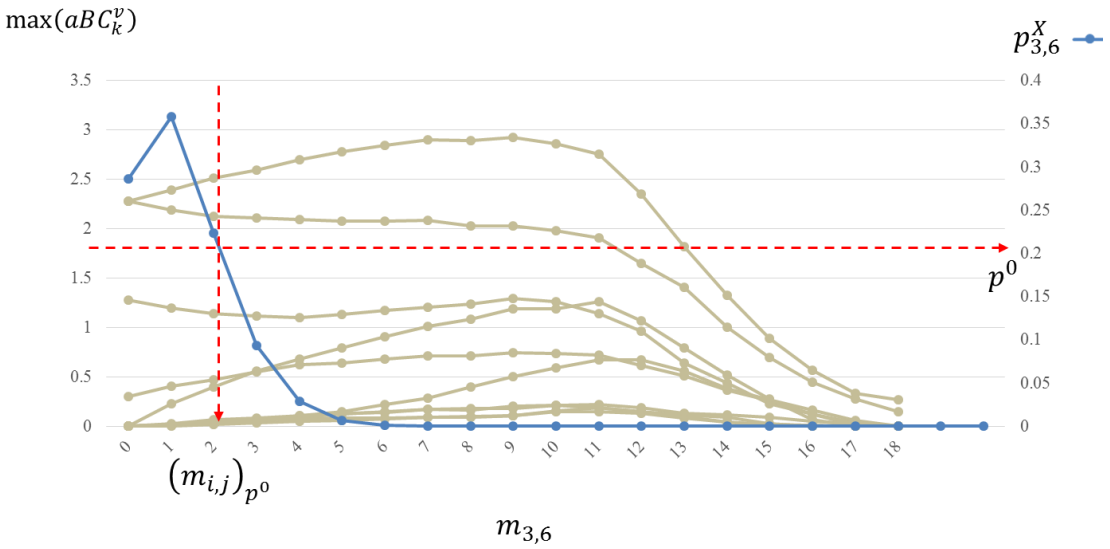


Figure 48. Overlay of Figure 43 and Figure 47

As can be seen in Figure 48, the value of $\max(aBC_k^V)$ will decrease (most likely) or at least stay at the same level as the value of $(m_{3,6})_{p^0}$ decreases. It is not hard to see that this observation is true for any node pair within any network topology.

Based on Equation 60, under the information transmission scenario prescribed in Table 18, for a given $p^0(m_{i,j})$ value, CIN operation length T, CIN topology, to decrease the value of $(m_{i,j})_{p^0}$ can be achieved through increasing the value of Θ, which is to increase the MTTF (reliability) of a link.

According to Equation 60, to increase the value of Θ will affect $p(m_{i,j})$ by increasing the probability of $p_1(m_{i,j})$ for smaller $m_{i,j}$ values. Therefore, to increase the MTTF of network links through increasing the reliability of network links (system design) will decrease the value of $(m_{i,j})_{p^0}$, and hence decrease the critical information processing capacity of each network node (Γ_k^C).

From now on, the congestion behaviors without any network impairment will be called as static congestion behaviors and those under network impairments will be named as dynamic congestion behaviors. Moreover, refer to the Γ_k^C value calculated without considering network impairments (Equation 51) as static Γ_k^C, or $(\Gamma_k^C)_s$ and refer to the Γ_k^C value calculated considering network impairments (Equation 53) as dynamic Γ_k^C, of $(\Gamma_k^C)_d$. From the previous discussion, it can be seen that, the value of $(\Gamma_k^C)_d$ depends on the value of $(\Gamma_k^C)_s$, and the $(\Gamma_k^C)_s$ value can be viewed as the lower bond of the $(\Gamma_k^C)_d$ value.

As mentioned earlier in Chapter 1, a CIN network is usually constructed by SUAVs. The payload and space limitations are much higher than traditional UAVs. This poses a much tighter constraint on the communication and computation capabilities an SUAV can be equipped. It is possible that the required information processing capacity, $(\Gamma_k^C)_d$, cannot not be met by the current available technologies or can only be met at a very high acquisition cost on supporting technologies. Under such circumstance, based on the discussion in Section 7.2, the required information processing capacity, $(\Gamma_k^C)_d$, can be decreased.

First, for a CIN, to decrease the value of $(\Gamma_k^C)_d$ can be achieved by decreasing the corresponding $(\Gamma_k^C)_s$ value. The discussion in Section 7.2.1 says that to decrease the value of $(\Gamma_k^C)_s$ can be done by carefully selecting the network topology, the routing strategy ($p(k|i,j)$), and the collaboration structure ($p(j|i)$) of the CIN. Based on the discussion in Section 7.2.2, with the network topology selected, for a given CIN operation length, the value of $(\Gamma_k^C)_d$ can be further decreased by increasing the reliability of network links.

Most existing studies related to network congestion are mainly on static congestion behaviors. Till now, this thesis has provided some experiment results and

discussions on dynamic congestion behaviors and a quantitative design requirement on the critical information processing capacity of network nodes (Equation 51 and Equation 53). Since the research objective of this thesis is on measuring the capability-based connectivity robustness of a CIN, the discussion on network dynamic congestion behaviors will not be extended further. This topic will be deemed as future work that will be discussed in more details in Chapter 8.

7.3 *A Final Note*

During a CIN operation, it is very possible that the collaboration structure between network nodes changes during the operation period. This will result in more than one information exchange matrix among network entities. Along the previous discussion, it is assumed that the information exchange matrix is fixed. In order to include the effects of different information exchange matrix on Γ_k^C, the following approach can be used. Assume the set of information exchange matrices and the duration of each matrix to be used during a CIN operation is known. Then the following two equations can be obtained.

$$\Gamma_k^C = \max[(\Gamma_k^C)_t] \quad \textbf{61}$$

where,

$(\Gamma_k^C)_t$ is the critical information processing capacity of node k calculated using the t^{th} information exchange matrix.

$$\Gamma_k^C = \sum_{t=1} T_t (\Gamma_k^C)_t \quad \textbf{62}$$

where,

$(\Gamma_k^C)_t$ is the critical information processing capacity of node k

calculated using the t^{th} information exchange matrix;

T_t is the duration of a CIN operation segment when the t^{th} information exchange matrix.is used.

Equation 61 considers the effects of different information exchange matrices by taking the maximum value of the critical information processing capacity of node k during the entire CIN operation period. While Equation 62 considers the effects of different information exchange matrices by taking the time averaged critical information processing capacity of node k during the entire CIN operation period.

Equation 61 and Equation 62 are two examples meant to show how to incorporate the effect of having more than one information exchange matrix during a CIN operation. The method should be chosen based on the problem at hand and the design emphases.

On the other hand, both Equation 51 and Equation 53 were derived based on Equation 45 assuming uniform information packet output rate. If remove this assumption, similar equations can be derived by using the original form of Equation 45, which is Equation 46.

$$\Gamma_k^C = \max\left(\sum_{\substack{i=1\\ i\neq k}}^{N}\sum_{\substack{j=1\\ j\neq i\\ j\neq k}}^{N} p(j|i)p(k|ij)\,\lambda_{ij}\right) + \lambda_k + \varepsilon_k \qquad \textbf{63}$$

where,

$p(j|i)$ is the probability of node i choosing node j as its information destination;

$p(k|i,j)$ is the probability for node k to be on the information transmission path from node i to node j;

λ_{ij} is the information output rate from node i to node j;

ε_k is the extra information transmission capability required for using packet switching and processing method.

Again, if Equation 63 is evaluated using the original network topology without considering network impairments, then the result is the minimum information processing capacity required to be equipped for a network node in order to avoid congestion. In order to decide how much more information processing capacity a node needs to avoid congestion under network impairments, Equation 63 should be evaluated over the set of practical network topologies under network impairments.

7.4 *Chapter Summary*

Information congestion will result in both connectivity and information loss during CIN operation. It may also result in hardware impairments due to information surge. Hence, congestion should be avoided. The discussion in the previous chapters are based on the assumption that no congestion will happen within a network. In other words, the assumption says that, each network node is congestion robust. Congestion robustness of a network node is defined as the ability of a network node to sustain information overload or its tendency to experience congestion. In the literatures, congestion robustness and connectivity robustness are usually studied in different contexts because they are two coupling issues. In this chapter they were studied within the same context. In order to simplify the problem, congestion robustness was treated as a system level design requirements on each network node. The discussions from Chapter 3 to Chapter 6 focused on connectivity robustness assuming that no network node will experience information congestion under any circumstance. In this chapter, focus was given to that assumption.

Many things that can affect the congestion robustness of a network node and hence there are many system level design requirements for network nodes to be congestion robust during a CIN operation. Since the focus of this thesis is to study the capability-based connectivity robustness of a network, only the required information processing capacity of each network node was investigated. In addition, the discussion was constrained to the information transmission scenario specified in Table 18.

The congestion behaviors without any network impairment are referred to as static congestion behaviors and those under network impairments are referred to as dynamic congestion behaviors.

First, a discrete time simulation model for information transmission was constructed under the scenario specified in Table 18 to study the static congestion behaviors of network nodes. The following conclusions were obtained.

If a network node does not serve as an intermediate information transmission node for any node pair, it will never experience congestion as long as its packet output rate does not exceed its information processing capacity. Node congestion occurs when its total internal information size increases to the level close to its queue capacity with little fluctuations. The λ value when a node starts to experience congestion is the critical λ value of this node, which will be denoted as λ_i^C.

A network can have three congestion stages, namely, LTS, MCS, and HCS based on the average total internal information size within the entire network [89]. It has been shown that the three congestion stages of a network can also be characterized by the average information size within a network node $\bar{I}_i$, or in other words, the congestion behaviors of network nodes. Denote the lower critical λ value of a network as $\lambda_G^{C_L}$, and $\lambda_G^{C_L} = \min(\lambda_i^C)$. Denote the upper critical λ value of a network as $\lambda_G^{C_U}$, and $\lambda_G^{C_U} = \max(\lambda_i^C)$. The first tier nodes have λ_i^C values equal or close to $\lambda_G^{C_L}$. The third tier nodes

have λ_i^C values equal or close to λ_G^{CU}. The rest nodes whose $\bar{I}_i$ values are not constantly 0 belong to the second tier. The onsite of the congestion of the first dire nodes marks the change of network congestion state from LTS to MCS. The onsite of the congestion of the third tier nodes marks the change of network congestion state from MCS to HCS.

Different network topologies and different information exchange matrices have different effects on the congestion behaviors of network nodes. However, regardless the network topology type and the information exchange matrix used, it is always the case that, the higher the information processing capacity a node has, the higher its λ_i^C value will be.

Congestion results in information loss. In order to prevent congestion, each network node should be equipped with enough information processing capacity. On the other hand, early congestion will happen if any network node does not have enough information storage space (queue capacity) to prevent early congestion. Early congestion prevents the information processing capacity of a node to be fully used and results in non-economic designs and hence should be prevented. However, when the queue capacity of a network node is higher than a certain value, further increasing its value will not yield any additional congestion benefits.

Next, based on those conclusions, two design guidelines were proposed for the information processing capacity of each network node. One is each network node should be equipped with at least the minimum information processing capacity, which is also the critical information processing capacity (Γ_k^C) to avoid information congestion. The other ones is to minimum the network polarization value of Γ_i^C about $\min(\Gamma_i^C)$ to ensure economic architecture design.

If there is no network impairments, the static critical information processing capacity of each network node can be obtained by Equation 51.

If network impairments are considered, then the dynamic critical information processing capacity of each network node can be obtained by Equation 53.

"Practical" means the probability of node pair i, j to stay connected with $m_{i,j}$ number of link failures during the operation period of a CIN is significant. This probability is denoted as $p(m_{i,j})$, and its significant value is denotd as $p^0(m_{i,j})$. Under the information transmission scenario prescribed in Table 18, for a given $p^0(m_{i,j})$ value, $p(m_{i,j})$ can be estimated through Equation 60.

Comparing Equation 51 and Equation 53, to decrease the value of $(\Gamma_k^C)_d$ can be achieved by decreasing the corresponding $(\Gamma_k^C)_s$ value.

Equation 51 establishes the relationship between the required information processing capacity of a network node and the network topology, the routing strategy $(p(k|i,j))$ as well as the collaboration structure $(p(j|i))$ of a CIN. Equation 53 and Equation 60 further establishes the relationship between the required information processing capacity of a network node and the capability-based connectivity robustness of the CIN, the reliability of network links.

Finally, the effect of variable collaboration structure were discussed. The method used to incorporate the effects of having more than one information exchange matrix during a CIN operation on the required information processing capacity of a network node should be chosen based on the problem at hand and the design emphases of the CIN.

CHAPTER VIII

CONCLUSIONS AND CONTRIBUTIONS

8.1 *Resolution of Research Questions and Hypotheses*

Technology advancements have greatly extended the application scope of Collaborative Information Networks (CINs). Due to the unique application fields of CINs and the nature of this construction, the connectivity of the inter-connection structure under impairments is a profound but challenging requirement for a CIN. Most of the existing topological connectivity robustness measures were proposed from a pure structural perspective with little or no consideration of the capability of a network. They can describe the ability of a network to resist network fragmentation under impairments. However, the current evaluation practice provides no direct mapping between the measured connectivity robustness and the capability robustness of a network. By seeing this gap, the research objective of this thesis is to develop a method to measure the capability-based connectivity robustness of a CIN against link failures by using existing topological connectivity robustness measures.

The research objective immediately leads to two research questions.

Research Question 1: How to incorporate capability into the conventional network modeling process?

Research Question 2: Which existing topological connectivity robustness measure should be chosen?

In search for the answer to the first research question, a capability-based network modeling process was developed. The process was motivated by the following observation. In order to output capability, one or more major information flows of a CIN

should be maintained. The major information flows can be collapsed into the connection between several critical node pairs. To measure the capability-based connectivity robustness of a CIN is to measure the (structural) connectivity robustness of critical node pairs.

Now with a capability-based network model, the problem of measuring the capability-based connectivity robustness of a CIN is successfully transformed into the problem of measuring the structural connectivity robustness between critical node pairs. The next task is to find the answer to the second research question, which is to select a topological measure for the structural connectivity robustness against link failures between an arbitrary node pair.

Pairwise effective resistance $ER_{i,j}$ was identified as a candidate measure. By testing Hypothesis 1, it was concluded that, $ER_{i,j}$ can be used to compare the connectivity robustness of two arbitrary node pairs in terms of the average fraction of link failures until disconnection happens ($\frac{\bar{m}_{i,j}^{X}}{M}$). In order to compare the connectivity robustness of two arbitrary node pairs in terms of the average number of link failures until disconnection happens ($\bar{m}_{i,j}^{X}$), Equation 16 was proposed to provide a close estimation for $\bar{m}_{i,j}^{X}$ given the $ER_{i,j}$ value of a node pair. This estimation method is fast and scalable. The estimation error stabilizes as network node number increases. With this, the second research question was also answered.

The existence of redundant links does not affect the average number of link failures that a node pair can sustain before disconnection. This is because redundant links do not contribute to the connection between node pair i, j. However, under random link attacks, redundant links can server as "camouflage" and attract attacks away from structural links. This decreases the probability of structural links to be hit during random

attacks and as a result protects the network structure. The effect can be quantified using either Equation 19 or Equation 20. The validity of Equation 19 or Equation 20 was confirmed via simulation.

Centrality analyses for network entities existing were also performed in terms of their importance to the capability-based connectivity robustness of a network. Network node centrality can be calculated via Equation 32 and network link centrality can be calculated via Equation 37. By testing Hypothesis 2 and Hypothesis 3, the validity of the two proposed measures is confirmed. Both measures are based on the Moore-Penrose Pseudoinverse of a network Laplacian (L^+). Since L^+ is also used to calculate $ER_{i,j}$, the proposed centrality evaluation methods do not require any extra heavy computation other than several basic operations. As a result, the proposed measures can be used to help quickly allocate limited resources to protect network against impairments.

A framework for the fast evaluation of the capability-based connectivity robustness of a CIN was constructed and was demonstrated on the example CIN followed by an alternative topology design generation process.

In addition, two capability-based connectivity robustness strengthen strategies were proposed and discussed. The first strategy is to increase the static robustness via adding network links. Equation 39 was proposed to help decide the optimal link addition process that results in the most robustness increase benefit. The second strategy is to prepare substitution nodes for some important network nodes. It was demonstrated that the capability-based connectivity robustness evaluation process proposed in Chapter 3 together with a simple network topology modification procedure could be used to quantify the effectiveness of a dynamic link failure coping mechanism. By testing Hypothesis 4, the validity of Equation 39 is confirmed.

Finally, the effects of the capability-based connectivity robustness of a network on the required information processing capacity of each network node was also explored. In this thesis, information congestion was treated as a system level design requirement on each network node. To avoid information congestion, a network node needs to be incorporated with enough information transmission capabilities. This thesis focuses on studying the required information processing capacity under a given information transmission scenario.

The analyses were conducted using a discrete-time simulation model on information transmission and processing within a network. Equation 51 establishes the relationship between the required information processing capacity of a network node to the network topology, the routing strategy ($p(k|i,j)$), and the collaboration structure ($p(j|i)$) of a CIN. Equation 53 and Equation 60 further establishes the relationship between the required information processing capacity of a network node to the capability-based connectivity of the CIN and the reliability of network links.

The hypotheses proposed along the discussion process are summarized below. The test results suggest rejecting Hypothesis 1 and accepting Hypothesis 2, 3, 4.

H^{1a}: $\frac{N}{ER_{i,j}}$ has higher correlation with $\bar{m}_{i.j}^{X}$ then $\frac{1}{ER_{i,j}}$.

H_0^{1a}: $\frac{N}{ER_{i,j}}$ does not have higher correlation with $\bar{m}_{i.j}^{X}$ then $\frac{1}{ER_{i,j}}$.

H^{1b}: $\frac{N}{ER_{i,j}}$ has high correlation with $\frac{\bar{m}_{i.j}^{X}}{M}$ then $\frac{1}{ER_{i,j}}$

H_0^{1b}: $\frac{N}{ER_{i,j}}$ does not have higher correlation with $\frac{\bar{m}_{i.j}^{X}}{M}$ then $\frac{1}{ER_{i,j}}$.

H^{2}: $-\Delta U_{i,k,j}$ is highly correlated with $\Delta\bar{m}_{i,j}^{X}(k)$.

H_0^{2}: $-\Delta U_{i,k,j}$ is not highly correlated with $\Delta\bar{m}_{i,j}^{X}(k)$.

H^{3}: $-\frac{\Delta U_{i,k,j}+\Delta U_{i,l,j}}{\Delta U_k+\Delta U_l}$ is highly correlated with $\Delta\bar{m}_{i,j}^{X}(k,l)$.

H_0^3: $-\frac{\Delta U_{i,k,j}+\Delta U_{i,l,j}}{\Delta U_k+\Delta U_l}$ is not highly correlated with $\Delta\overline{m}_{i,j}^X(k,l)$.

H^4: $\Omega_{k,l}^{i,j}$ can indicate the benefits of adding a non-existing link into a network on $\overline{m}_{i,j}^X$.

H_0^4: $\Omega_{k,l}^{i,j}$ cannot indicate the benefits of adding a non-existing link into a network on $\overline{m}_{i,j}^X$.

8.2 *Contributions*

Contribution 1

This thesis demonstrated the flexible use of network modeling. Network topological analyses are usually deployed to study the structure of a network. By modifying the network model of an infrastructure, network topological analysis can be used to analyze the effects besides network structure such as the capability-based connectivity robustness and the resilience strategy of a CIN.

Contribution 2

For the first time, it was pointed out that $ER_{i,j}$ or $\frac{ER_{i,j}}{N}$ can only be used to compare the connectivity robustness of different node pairs from the same network or the same node pair of networks within the same network family. The connection between $ER_{i,j}$ and the connectivity robustness under link impairments is actually established though the average faction of link failures until a node pair disconnected ($\frac{\overline{m}_{i,j}^X}{M}$).

Contribution 3

In this thesis, $\overline{m}_{i,j}^X$ is used as a direct measure of the capability-based connectivity robustness of a CIN. A quick and scalable method was proposed that can provide close

estimation for the average number of link failure until a node pair disconnected ($\bar{m}_{i,j}^{X}$). The error of this estimation stabilizes as network node number increases.

Contribution 4

The fourth contribution of this thesis is that it provides quick and scalable ways to quantify the centrality of existing network nodes and links as well as the centrality of non-existing links in terms of $\bar{m}_{i,j}^{X}$, which can help effectively allocate limited resources to protect network against impairments or to add additional links to strengthen robustness

Contribution 5

The fifth contribution of this thesis is to consider congestion robustness and connectivity robustness under the same content. This thesis demonstrated that congestion robustness could be treated as a system level design requirement on each network node that could be derived from the inter-connection structure and the connectivity situation of a CIN.

Contribution 6

The final contribution of this thesis is a network topology design and selection process based on the proposed capability-based connectivity robustness measure, which can also be used as a sub-design process of a more comprehensive, complex design process.

8.3 *Recommendations for Future Studies*

The Moore-Penrose pseudoinverse of a symmetric Laplacian (L) is the key for most of the analyses in this thesis. Symmetric L and their applications have been deeply studied [53-57]. While asymmetric L arise in connection with directed graphs are less

explored. To obtain the Moore-Penrose pseudoinverse, an asymmetric L needs to be normalized. Depending on the research contents and analysis focuses, different normalization techniques have been proposed. Although asymmetric L are now attracting more and more attentions [52, 57-61], it is still a working concept without conscience upon normalization techniques as well as the physical meanings behind them. Most importantly, unlike a symmtric L that is strictly related to the connectivity properties of the corresponding undirected network topology, a normalized asymmetric Lusually does not reflect the connectivity of the corresponding directed network topology well. To extend the results obtained from this thesis to directed networks, a normalization technique is needed so that the $ER_{i,j}$ calculated based on the normalized L is closely related to the average percentage of link failures until a node pair disconnected within a directed network.

In addition, symmetric L can be used for weighted network. However, the proposed connectivity robustness measure and the subsequent analyses can only handle unweighted networks since all network link failures are treated the same. Network link weights can be used to model some connection properties between network entities, such as interoperability. Future researches can focus on extend the results of this thesis to weighted network to account for the effects of network connection properties.

Another area that can be explored further is how to design a congestion robust routing strategy under network impairments. As discussed earlier, the congestion robustness of network nodes can be affected by the information transmission capabilities of network nodes, the routing strategy and the network topology. A dynamic routing strategy that responds to network impairments can reduce the required information capabilities of network nodes.

The next area that can be studied further is to understand the difference between the network topologies obtained from step-wise "optimization" (greedy algorithm) as shown in this thesis and the ones obtained from solving optimization directly (global, or true optimization). In addition, the results obtained from step-wise network topology "optimizations" are initial point dependent. How sensitive the results are to different initial points and how to pick a good initial point that can lead to network topology designs that is close to the global optimal is an area that worths further exploration.

APPENDIX I

ADDITIONAL GRAPHS AND TABLES

Table 21. Information Exchange Matrix 1 Used in Chapter 7

	1	2	3	4	5	6	7	8	9	10
1	0	0	0.21	0.25	0	0	0.25	0.29	0	0
2	0.4	0	0	0	0	0.26	0	0	0.34	0
3	0	0.22	0	0	0.15	0.13	0.11	0.17	0.11	0.11
4	0.22	0	0.2	0	0.16	0	0.14	0	0.12	0.16
5	0.11	0.14	0.11	0	0	0	0.14	0.18	0.16	0.16
6	0	0	0.29	0.33	0	0	0.38	0	0	0
7	0	0	0.2	0.17	0	0.28	0	0	0.2	0.15
8	0.19	0.34	0	0	0.22	0	0	0	0.25	0
9	0.21	0.14	0.12	0.17	0.14	0.1	0	0.12	0	0
10	0.31	0.29	0	0	0.23	0	0	0	0.17	0

Table 22. Information Exchange Matrix 2 Used in Chapter 7

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0.27	0	0.24	0	0.18	0.31
2	0.18	0	0.25	0.23	0.16	0	0	0	0	0.18
3	0	0	0	0	0	0.38	0	0.33	0.29	0
4	0.15	0.25	0.2	0	0.15	0.25	0	0	0	0
5	0	0	0.57	0	0	0	0	0	0.43	0
6	0.28	0.36	0.36	0	0	0	0	0	0	0
7	0	0.21	0	0.33	0	0	0	0.25	0	0.21
8	0.12	0.22	0	0.1	0	0.17	0.17	0	0.12	0.1
9	0.2	0	0	0.17	0	0	0.17	0.2	0	0.26
10	0.47	0	0.53	0	0	0	0	0	0	0

APPENDIX II

MODELING AND SIMULATION

A. Step-Min Network Family Generation (Matlab)

This program is used to generate networks within a Step-Min Network Family as well as calculating the $ER_{1,N}$ value of each network. The only input for this code is the Step-Min ntwork family index (the number of nodes). For the detailed logic of this code, please refer to Algorithm 1.

```
clear;
clc;
result = cell([1,1]);
% mkdir('ER');
% colorSpace = jet(40);
% figure
for nodeSize = 11:29 % CHANGE HERE: NODE NUMBER
   adjMatrix = zeros(nodeSize, nodeSize);
   for i = 1: nodeSize-1
      adjMatrix(i,i+1) = 1;
      adjMatrix(i+1,i) = 1;
   end
   linkSpace = (nodeSize-1)*nodeSize / 2 - (nodeSize-1);
   stepMin = zeros(linkSpace + 1,1);
   stepMin(1,1) = nodeSize - 1;
   if linkSpace > 0
      stepMinPair = zeros(2,1);
      ER_Temp = 0;
      mkdir(num2str(nodeSize));
      for i = 1:linkSpace
         for u = 1:nodeSize-1
            for v = u+1:nodeSize
               if adjMatrix(u,v) == 0
                  adjMatrix(u,v) = 1;
```

```
                    adjMatrix(v,u) = 1;
                    ER_Temp = ER_Cal(nodeSize, adjMatrix, 1, nodeSize);
                    if stepMin(1+i, 1) == 0
                        stepMin(1+i, 1) = ER_Temp;
                        stepMinPair = [u;v];
                    else
                        if ER_Temp > stepMin(1+i, 1)
                            stepMin(1+i, 1) = ER_Temp;
                            stepMinPair = [u;v];
                        end
                    end
                    adjMatrix(u,v) = 0;
                    adjMatrix(v,u) = 0;
                end
            end
        end
        adjMatrix(stepMinPair(1,1),stepMinPair(2,1)) = 1;
        adjMatrix(stepMinPair(2,1),stepMinPair(1,1)) = 1;
        dlmwrite(strcat(num2str(nodeSize), '\', num2str(nodeSize), '_', num2str(i+1),
'.txt'), adjMatrix);
    end
end

%     dlmwrite(strcat('ER\', num2str(nodeSize), '.txt'), stepMin);
%
%     result{nodeSize-1,1} = stepMin;
% %     semilogy(result{nodeSize-1,1}.^(-1), ['--','o'], 'color',colorSpace(nodeSize-1,:))
%     hold on
end
% axis([0 40 0.01 1])
% hold off
```

B. Rand and SF Network Generation Code (C^{++})

This program is used to generate the topologies for Rand and SF networks used in the experimetns in this thesis. This code repqures input the number of network nodes and the total number of network links. The inputs are the network node number N, the

network link number *E* for the network to be gerenated, and the network number of the fully connected network used if the network to be generated is a SF network.

```
#include <iostream>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <fstream>
#include <random>
#include <iostream>
#include <string>
#include <sstream>
using namespace std;

namespace patch
{
   template < typename T > std::string to_string( const T& n )
   {
      std::ostringstream stm ;
      stm << n ;
      return stm.str() ;
   }
}


static const int numberOfV = 10;

static const int numberOfE = 40;

//Generate the seed for random number generation functions
mt19937 gen(time(NULL));

//Uniform integer random number generator
int uniIntRand(int n) {
   uniform_int_distribution<int> distribution(1, n);
   return distribution(gen);
}
```

```
//Uniform real number generator
double uniRealRand() {
   uniform_real_distribution<double> distribution(0.0,1.0);
   return distribution(gen);
}

//Output the generated adjacent matrix to screen to check results validity
void outputToText( string networkName, int AMatrix[][numberOfV], int
Node_num){
   ofstream myfile;
   const string fileName = "Network Topology/" + networkName + ".txt";
   myfile.open (fileName);
   for (int i=0; i < Node_num; i++){
      for (int j=0; j < Node_num; j++){
         if (j==49) {
           myfile<< AMatrix[i][j];
         }
         else{
            myfile<< AMatrix[i][j]<< ",";
         }
      }
   myfile<<"\n";
   }
   myfile.close();
}


//Random Network
void generate_Node(int * Node,int range){
   Node[0]=uniIntRand(range)-1;
   Node[1]=uniIntRand(range)-1;
}

void Rand_Topology(int Node_num,int Link_num){
   //generate the desired matrix
  int array[numberOfV][numberOfV]={};
  int Node[2]={};
```

```
  //generate the random topology of the matrix
  for (int i=0;i<Link_num;i++){

     while (1){
        generate_Node(Node,Node_num);
        int N1 = Node[0];
        int N2 = Node[1];
     if (array[N1][N2]==0&&array[N2][N1]==0&& N1!=N2){
          array[N1][N2]=1;
          array[N2][N1]=1;
          break;
     }
     else
        {continue;}
     }
  }

  string fileName = "Rand_" + patch::to_string(numberOfV) + "_" +
patch::to_string(numberOfE);
  outputToText(fileName, array, Node_num);
}


//Scale Free Network
int PickNode(int D[numberOfV], int N, int DTotal){
  int i;
  double Prob_Pick = uniRealRand();
  double Degree_Pick= Prob_Pick * double(DTotal);
  double Add_Degree = 0.0;

  for (i = 0; i < N; ++i) {
     Add_Degree = Add_Degree + double(D[i]);
     if (Degree_Pick <= Add_Degree) {
        return i;
        break;
     }
  }
}
```

```
void Scalefree_Topology(int Node_num, int existing_node_num){
   int D_Total = 0;  //sum of the node degrees over the entire network
   int D_Matrix[numberOfV] = {};
   int array[numberOfV][numberOfV]={};

//Generate a fully connected network with desired number of nodes
   for (int i = 0; i < existing_node_num; ++i) {
      D_Matrix[i] = existing_node_num - 1;
      for (int j = 0; j < existing_node_num; ++j) {
         if (i!=j){
         array[i][j]=1;
         D_Total += 1;
         }
      }
   }

//Preferential node selection
   for (int i = existing_node_num ; i < Node_num; ++i) {
      int Picked_Node1;
      int Picked_Node2;

      while(1){
         Picked_Node1 = PickNode(D_Matrix, i, D_Total);
         if (Picked_Node1!=i)break;
         else continue;
      }

      while(1){
         Picked_Node2=PickNode(D_Matrix,i,D_Total);
         if (Picked_Node2==Picked_Node1||Picked_Node2==i)
            continue;
         else break;
      }

      D_Matrix[Picked_Node1]++;
      D_Matrix[Picked_Node2]++;
      D_Matrix[i] = 2;
```

```
        D_Total += 4;
        array[i][Picked_Node1]=1;
        array[i][Picked_Node2]=1;
        array[Picked_Node1][i]=1;
        array[Picked_Node2][i]=1;
    }
    string fileName = "SF_" + patch::to_string(numberOfV) + "_" +
patch::to_string(numberOfE);
    outputToText(fileName, array, Node_num);

}


int main (){
    Rand_Topology(numberOfV,numberOfE/2);
    Scalefree_Topology(numberOfV,5);
    return 0;
}
```

C. The Link Failure Simulation Model Code (C++)

This program is used to simulate the link failure process. The redundant link filter is not included. That part is a small process conducted in Matlab. If a filtered network is fed into this simulation model, then the output is the number of structural link failures until the target node pair disconnected. If an unfiltered network is fed into this simulation model, then the output is the number of total link failures until the target node pair disconnected. The inputs for this code are the network topology, the number of network nodes, and the target node pair index.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <iomanip>
#include <stdlib.h>
#include <algorithm>
```

```
#include <cmath>

#define INFINITY 999999

using namespace std;

static const int number = 10;

namespace patch
{
   template < typename T > std::string to_string( const T& n )
   {
      std::ostringstream stm ;
      stm << n ;
      return stm.str() ;
   }
}

class Dij{
   public:
      static const int numOfV = number;
      int predecessor[numOfV], distance[numOfV];
      int adjMatrix[numOfV][numOfV];
      void readTopology(string);
      int tree[numOfV][numOfV];
      bool mark[numOfV];
      int source;
      int dest;
      void initialize();
      void calculateDistance(int,int,int);
      int getClosestUnmarkedNode();
      void printPath(int, ofstream&);
};


//Read network topology
void Dij::readTopology(string  fileName){

   ifstream file(fileName);

   int col_read = number;
   int row_read = number;

   for(int row = 0; row < row_read; ++row)
```

```
    {
        string line;
        getline(file, line);

        stringstream iss(line);
        for (int col = 0; col <col_read; ++col)
        {
            string val;
            getline(iss, val, ',');
            int connectivity;
            connectivity = atoi(val.c_str());
            adjMatrix[row][col] =  connectivity;
        }
    }
};


void Dij::initialize(){
    for(int i = 0; i < numOfV; i++){
        mark[i] = false;
        predecessor[i] = -1;
        distance[i] = INFINITY;
    }
    distance[source] = 0;
};


int Dij::getClosestUnmarkedNode(){
    int minDistance = INFINITY;
    int closestUnmarkedNode;
    for(int i = 0; i < numOfV; i++){
        if((!mark[i]) && (minDistance >= distance[i])){
            minDistance = distance[i];
            closestUnmarkedNode = i;
        }
    }
    return closestUnmarkedNode;
};


void Dij::calculateDistance(int exclude_1, int exclude_2, int endNode){
    initialize();
    int closestUnmarkedNode;
    int count = 0;
```

```
   mark[exclude_1] = true;
   mark[exclude_2] = true;

   while(count < numOfV){
      closestUnmarkedNode = getClosestUnmarkedNode();
      mark[closestUnmarkedNode] = true;
      for(int i = 0; i < numOfV; i++){
         if((!mark[i]) && (adjMatrix[closestUnmarkedNode][i] > 0)){

            if(distance[i] > distance[closestUnmarkedNode] +
adjMatrix[closestUnmarkedNode][i]){
               distance[i] = distance[closestUnmarkedNode] +
adjMatrix[closestUnmarkedNode][i];
               predecessor[i] = closestUnmarkedNode;
            }

         }
      }
      count++;
   }
};

void Dij::printPath(int node, ofstream & myfile){
   if(node == source)
      myfile<<node<<",";
   else if(predecessor[node] == -1)
      myfile<<"No path from “<<source<<”to "<<node;
   else {
      printPath(predecessor[node], myfile);
      myfile<<node<<",";
   }
}


mt19937 gen(time(NULL));
//Uniform integer random number generator
int uniIntRand(int n) {
   uniform_int_distribution<int> distribution(1, n);
   return distribution(gen);
}


void generate_Node(int * Node,int range){
```

```
    Node[0]=uniIntRand(range)-1;
    if(range == 1){
        Node[1] = Node[0];
    }else{
        while(1){
            Node[1]=uniIntRand(range)-1;
            if(Node[1] != Node[0]){
                break;
            }
        }
    }

}

int mainSimulation_1_pair(string topoFileName, int node_i_1, int node_j_1, int
numberOfE){
    int impairment = 0;


    Dij G;
    G.readTopology(topoFileName);

    vector<string> nodePairPool_Rem;

    for(int i = 0; i < number; i++){
        for(int j = 0; j < number; j++){
            if(G.adjMatrix[i][j]> 0){
                nodePairPool_Rem.push_back(patch::to_string(i) + "_" +
patch::to_string(j) );
            }
        }
    }

    int link_imp_s = 0;
    int link_imp_t = 0;

    int distij_1 = 0;

    G.source = node_i_1;

    while(impairment < numberOfE){
        int Node_Rem[2] = {};
        generate_Node(Node_Rem, (numberOfE - impairment));
```

```
        link_imp_s = atoi(nodePairPool_Rem[Node_Rem[0]].substr(0,
nodePairPool_Rem[Node_Rem[0]].find("_",0)).c_str());
        link_imp_t =
atoi(nodePairPool_Rem[Node_Rem[0]].substr(nodePairPool_Rem[Node_Rem[0]].fin
d("_",0) + 1).c_str());
        nodePairPool_Rem.erase(remove(nodePairPool_Rem.begin(),
nodePairPool_Rem.end(), patch::to_string(link_imp_s) + "_" +
patch::to_string(link_imp_t)), nodePairPool_Rem.end());
        nodePairPool_Rem.erase(remove(nodePairPool_Rem.begin(),
nodePairPool_Rem.end(), patch::to_string(link_imp_t) + "_" +
patch::to_string(link_imp_s)), nodePairPool_Rem.end());
        G.adjMatrix[link_imp_s][link_imp_t] = 0;
        G.adjMatrix[link_imp_t][link_imp_s] = 0;
        impairment = impairment + 2;


        G.calculateDistance(-1, -1, -1);
        distij_1 = G.distance[node_j_1];
        if(distij_1 >= INFINITY){
           break;
        }
    }

    nodePairPool_Rem.clear();
    return impairment;
 }

int mainSimulation_1_pair_Pure_Redundancy(float pure_redundancy_ratio, string
topoFileName, int node_i_1, int node_j_1, int numberOfE){
    int pure_redundancy = floor(numberOfE * pure_redundancy_ratio);
    pure_redundancy = 0;
    int impairment = 0;
    int real_impairment = 0;

    Dij G;
    G.readTopology(topoFileName);

    vector<string> nodePairPool_Rem;

    for(int i = 0; i < number; i++){
       for(int j = 0; j < number; j++){
          if(G.adjMatrix[i][j]> 0){
             nodePairPool_Rem.push_back(patch::to_string(i) + "_" +
patch::to_string(j) );
```

```
        }
      }
    }


    int link_imp_s = 0;
    int link_imp_t = 0;

    int distij_1 = 0;

    G.source = node_i_1;

    while(impairment < (numberOfE + pure_redundancy)){
      int Node_Rem[2] = {};
      generate_Node(Node_Rem, (numberOfE + pure_redundancy - impairment));

      if(Node_Rem[0] < (numberOfE - real_impairment)){
        link_imp_s = atoi(nodePairPool_Rem[Node_Rem[0]].substr(0,
nodePairPool_Rem[Node_Rem[0]].find("_",0)).c_str());
        link_imp_t =
atoi(nodePairPool_Rem[Node_Rem[0]].substr(nodePairPool_Rem[Node_Rem[0]].fin
d("_",0) + 1).c_str());
        nodePairPool_Rem.erase(remove(nodePairPool_Rem.begin(),
nodePairPool_Rem.end(), patch::to_string(link_imp_s) + "_" +
patch::to_string(link_imp_t)), nodePairPool_Rem.end());
        nodePairPool_Rem.erase(remove(nodePairPool_Rem.begin(),
nodePairPool_Rem.end(), patch::to_string(link_imp_t) + "_" +
patch::to_string(link_imp_s)), nodePairPool_Rem.end());
        G.adjMatrix[link_imp_s][link_imp_t] = 0;
        G.adjMatrix[link_imp_t][link_imp_s] = 0;
        impairment = impairment + 2;
        real_impairment =real_impairment + 2;
      }else{
        impairment = impairment + 2;
      }


      G.calculateDistance(-1, -1, -1);
      distij_1 = G.distance[node_j_1];

      if(distij_1 >= INFINITY){
        break;
      }
    }
```

```
    nodePairPool_Rem.clear();
    return impairment;
}

int main(){
    float pure_redundancy_ratio = 0;
    int simu_num = 10000;
    int node_i_1 = 0;
    int node_j_1 = 3;

    int bridgeNum = 0;

    for(int linkSpace = 1; linkSpace <= 1; linkSpace++){
        Dij G;

        string topoFileName = "try.txt";
        G.readTopology(topoFileName);
        int numberOfE = 0;


        for(int i = 0; i < number; i++){
            for(int j = 0; j < number; j++){

                if(G.adjMatrix[i][j]> 0){

                    numberOfE = numberOfE + 1;
                }
            }
        }

        G = {};


        for(int i = 1; i <= 1; i++){


            ofstream myfile1;

            cout<<i<<"_"<<node_i_1<<"_"<<node_j_1<<"\n";

            string resultFileName1 = "Result_6_modi.csv";
            myfile1.open(resultFileName1, ios_base::app);
```

```
        for(int rep = 0; rep < simu_num; rep++){


myfile1<<mainSimulation_1_pair_Pure_Redundancy(pure_redundancy_ratio,
topoFileName, node_i_1, node_j_1, numberOfE)<<"\n";

        }

        myfile1.close();

    }

  }

return 0;
}
```

D. The Code Used to Obtain the Optimized θ Value

This program is used to solve the optimization problem speicified in Secion 2.2.3.

```
clc;
clear;
mdisij_full = [2.3372
4.5195
7.6476
11.7464
16.8265
22.8509
29.8991
37.9593
46.9035
56.9614
67.9692
80.0329
93.0555
107.146
122.0693
138.1219
155.0319
173.1482
192.0141
```

```
212.2045
233.1596
255.0404
277.9741
301.9826
327.0849
352.9298
380.126
407.9661
];

x = 0.9;
xs = zeros(28,1);
for nodeNum = 3:30
   ER1N = zeros(nodeNum*(nodeNum-1)/2 - nodeNum + 2,1);
   ER1N(1,1) =  nodeNum-1;
   for linkSpace = 2: (nodeNum*(nodeNum-1)/2 - (nodeNum-1) + 1)
      adjMatrix = zeros(nodeNum, nodeNum);
      adjMatrix =
dlmread(strcat(num2str(nodeNum),'\',num2str(nodeNum),'_',num2str(linkSpace),'.txt'
));
      ER1N(linkSpace,1) =  ER_Cal(nodeNum, adjMatrix, 1, nodeNum);
   end


   ERR = 10^10;
   e_mdis = zeros(nodeNum*(nodeNum-1)/2 - nodeNum + 2,1);
   mdis = zeros(nodeNum*(nodeNum-1)/2 - nodeNum + 2,1);
   mdis(1,1) = 1;
   mdis(2:nodeNum*(nodeNum-1)/2 - nodeNum + 2,1) =
dlmread(strcat('Results\Result_',num2str(nodeNum),'_Mean.txt'));
   e_mdis(1,1) = 1;
   t = 0.1;
   while 1
      ERR_old = ERR;
%       for linkSpace = 2: (nodeNum*(nodeNum-1)/2 - (nodeNum-1) + 1)
%          e_mdis(linkSpace,1) = e_mdisij(nodeNum, linkSpace + nodeNum - 2,
ER1N(linkSpace,1), x, mdisij_full(nodeNum-2,1));
%       end
%
      ERR = sum(abs((mdis-log10((linkSpace + nodeNum - 2)./ER1N.^x)).^2));
```

```
        if abs(ERR-ERR_old) < 10^(-4)
            x
            break;
        end

        if ERR >= ERR_old
            x = x - t;
            t = t/10;
        end
        x = x + t;
    end
    xs(nodeNum-2,1) = x;

end
```

E. The Code Used to Calculated $ER_{i,j}$ (Matlab)

This customized Matlab function calculates the effective resistance between a givne node pair ($ER_{i,j}$). The inputs of this function are network node number, network topology, and the target node pair.

```
function ERij = ER_Cal(nodeSize, adjMatrix, node_i, node_j)
L = zeros(nodeSize, nodeSize);

for j = 1 : nodeSize
    for k = 1:nodeSize
        if j == k
            L(j,k) = sum(adjMatrix(j,:));
        else
            if adjMatrix(j,k) > 0
                L(j,k) = -adjMatrix(j,k);
            end
        end
    end
end

QQ = pinv(L);
```

```
ERij = QQ(node_i, node_i) - 2*QQ(node_i, node_j) + QQ(node_j, node_j);
```

F. The Code Used to Calculate the Centrality of Existing Network Nodes (Matlab)

This customerized matlab function calculates the centrality of all the nodes within a network in terms of the capability-based connectivity robustness between the target node pair. The inputs of this function are network node number, network topology, and the target node pair.

```
function C = NodeImp(nodeSize, adjMatrix, node_i, node_j)

C_temp = zeros(nodeSize, 1);
L = zeros(nodeSize, nodeSize);

for j = 1 : nodeSize
   for k = 1:nodeSize
      if j == k
         L(j,k) = sum(adjMatrix(j,:));
      else
         if adjMatrix(j,k) > 0
            L(j,k) = -adjMatrix(j,k);
         end
      end
   end
end

QQ = pinv(L);

for i = 1:nodeSize
   C_temp(i,1) = (2*QQ(i,i)-QQ(node_i,i)-QQ(i,node_i)-QQ(node_j,i)-QQ(i,
node_j)+QQ(node_i, node_j)+QQ(node_j, node_i))/2;
end
C = C_temp;
End
```

G. The Code Used to Calculate the Centrality of Existing Network Links (Matlab)

This customerized matlab function calculates the centrality of all the existing links within a network in terms of the capability-based connectivity robustness between the target node pair. The inputs of this function are network node number, network topology, and the target node pair.

```
function C = linkIMP_2(nodeSize, adjMatrix, node_i, node_j)
   C_temp_1 = zeros(nodeSize, nodeSize);
   C_temp_2 = zeros(nodeSize, nodeSize);
   C_temp_3 = zeros(nodeSize, nodeSize);
   C_temp_4 = zeros(nodeSize, nodeSize);
   C_temp = zeros(nodeSize, nodeSize);
   L = zeros(nodeSize, nodeSize);

   for j = 1 : nodeSize
      for k = 1:nodeSize
         if j == k
            L(j,k) = sum(adjMatrix(j,:));
         else
            if adjMatrix(j,k) > 0
               L(j,k) = -adjMatrix(j,k);
            end
         end
      end
   end

   QQ = pinv(L);


   for i = 1:nodeSize
      for j = 1:nodeSize
         C_temp_3(i,j) = (2*QQ(i,i)-QQ(node_i,i)-QQ(i,node_i)-QQ(node_j,i)-QQ(i,
node_j)+QQ(node_i, node_j)+QQ(node_j, node_i))/2;
         C_temp_4(i,j) = (2*QQ(j,j)-QQ(node_i,j)-QQ(j,node_i)-QQ(node_j,j)-QQ(j,
node_j)+QQ(node_i, node_j)+QQ(node_j, node_i))/2;
         C_temp_1(i,j) = (2*QQ(node_i,node_i)-QQ(node_i,i)-QQ(i,node_i)-
QQ(node_i,j)-QQ(j, node_i)+QQ(i,j)+QQ(j,i))/2;
         C_temp_2(i,j) = (2*QQ(node_j,node_j)-QQ(node_j,i)-QQ(i,node_j)-
QQ(node_j,j)-QQ(j, node_j)+QQ(i,j)+QQ(j,i))/2;
```

```
            C_temp(i,j) = (C_temp_3(i,j) + C_temp_4(i,j))/(QQ(i,i) + QQ(j,j)); %This is
the right one

        end
    end

    C = C_temp;
End
```

H. The Code Used to Calculate the Centrality of Non-existing Network Links (Matlab)

This program is used to calucate the centrality of all the non-existing links within a network in terms of the capability-based connectivity robustness between the target node pair. The inputs of this function are network node number, network topology, and the target node pair.

```
clc;
clear;
nodeNum = 50;
adjMatrix = importdata('Network Topology\SF_50_100.txt', ',' , 0);
C_Node = zeros(50,40);
nodePairs = importdata('50_SF_Node Pair.txt', ',' , 0);

newER = zeros(nodeNum*(nodeNum-1)/2-100,5*40);

for k = 1:40
    C_Link = linkIMP_2(nodeNum, adjMatrix, nodePairs(k,1)+1, nodePairs(k,2)+1);
    count = 1;
    for i = 1:nodeNum-1
        for j = (i+1):nodeNum
            if adjMatrix(i,j) == 0
                C_Link_2 = linkIMP_2(nodeNum, adjMatrix, i, j);
                adjMatrix_new = adjMatrix;
                adjMatrix_new(i,j) = 1;
                adjMatrix_new(j,i) = 1;
                newER(count,1+(k-1)*5) = i;
                newER(count,2+(k-1)*5) = j;
```

```
            newER(count,3+(k-1)*5) = ER_Cal(nodeNum, adjMatrix_new,
nodePairs(k,1)+1, nodePairs(k,2)+1);
            newER(count,4+(k-1)*5) = C_Link(i,j) + C_Link_2(nodePairs(k,1)+1,
nodePairs(k,2)+1);
            newER(count,5+(k-1)*5) = abs(C_Node(i,1)-C_Node(j,1));
            count = count + 1;
        end
    end
  end
end
```

I. The Code Used to Generate a Resource Exchange Matrix (Matlab)

This program is used to generate a modified random resource exchange matrix. The only input of this program is number of network nodes.

```
clc;
clear;
nodeNum = 8;
resMatrixIndex = 3;

ResourceExMatrix = randi(10,nodeNum, nodeNum)*10;
for i = 1:nodeNum
  for j = 1:nodeNum
    if ResourceExMatrix(i,j)<=50 || i==j
      ResourceExMatrix(i,j) = 0;
    end
  end
end

for i = 1:nodeNum
  rowSum(i) = sum(ResourceExMatrix(i,:));
end

for i = 1:nodeNum
  ResourceExMatrix_NormalizedSimu(i,:) = floor(ResourceExMatrix(i,:) /
rowSum(i)*100)/100;
  addOnLocation = find(ResourceExMatrix(i,:)==max(ResourceExMatrix(i,:)));
  addOnLocation = addOnLocation(1,1);
```

```
    ResourceExMatrix_NormalizedSimu(i,  addOnLocation) =
ResourceExMatrix_NormalizedSimu(i, addOnLocation) + 1 -
sum(ResourceExMatrix_NormalizedSimu(i, :));

end

ResourceExMatrix_Normalized = ResourceExMatrix_NormalizedSimu;

for i = 1:nodeNum
   currentValue = 0;
   for j = 1:nodeNum
      if ResourceExMatrix_NormalizedSimu(i,j) > 0
         currentValue = currentValue + ResourceExMatrix_NormalizedSimu(i,j);
         ResourceExMatrix_NormalizedSimu(i,j) = currentValue;
      end
   end
end

dlmwrite(strcat('ResExMatrix_No Fluc\10_40_0.05_0.95\ResourceMatrix_',
num2str(nodeNum),'_', num2str(resMatrixIndex),'.txt'),
ResourceExMatrix_Normalized);
dlmwrite(strcat('ResExMatrix_No Fluc\10_40_0.05_0.95\ResourceMatrixSimu_',
num2str(nodeNum),'_', num2str(resMatrixIndex),'.txt'),
ResourceExMatrix_NormalizedSimu);
```

J. The Code Used to Calculate the Augmented Betweenness Centrality of Network Nodes (Matlab)

This program is used to calculate the augmented betweenness centrality of all the network nodes within a network. The inputs for this program is the network node number, the network link number, the network topology and the resource exchange matrix used for information transmission.

```
clc;
clear;
nodeNum = 7;
linkNum = 6;
netType = 'Rand';
```

```
matrixIndex = 1;

fileName_dij = strcat('Shortest_Distance\Dij_ExampleProblemAdj.txt');
fileName_res = strcat('ResExMatrix_No
Fluc\10_40_0.05_0.95\ExampleProblemRE.txt');
dijDist = csvread(fileName_dij);
resExMatrix = csvread(fileName_res);

maxLength = size(dijDist);
maxLength = maxLength(1,2);

coutPass = zeros(nodeNum,1);
coutPass_weighted = zeros(nodeNum,1);
numTotalPath = nodeNum*(nodeNum);
resExWeight = zeros(numTotalPath, 1);

k = 0;
for i = 1: nodeNum
   for j = 1: nodeNum
      k = k + 1;
      resExWeight(k,1) = resExMatrix(i,j);
   end
end

for i = 1: numTotalPath
   j = 2;
   while(j < maxLength)
      if dijDist(i,j+1)~= 0
         coutPass(dijDist(i,j),1) = coutPass(dijDist(i,j),1) + 1;
         coutPass_weighted(dijDist(i,j),1) = coutPass_weighted(dijDist(i,j),1) +
resExWeight(i,1);
      end
   j = j+1;
   end
end

strBet = coutPass;
strBet_Nor = coutPass / numTotalPath;

funcBet = coutPass_weighted;
```

K. The Code Used to Generated the Shortest Path between Two Nodes within a Network Using Dij Algorithm (C++)

This program is based on Dij algorithm to generate the shortest path between any node pair within a network. The inputs for this program are the network topology and the number of network nodes.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <iomanip>
#include <stdlib.h>

#define INFINITY 999

using namespace std;


static const int number = 10;

class Dij{
   public:
      static const int numOfV = number;
      int predecessor[numOfV], distance[numOfV];
      int adjMatrix[number][number];
      void trys(string);
      int tree[numOfV][numOfV];
      bool mark[numOfV];
      int source;
      int dest;
      void initialize();
      void calculateDistance();
      void output();
      void printPath(int, ofstream &);
      int getClosestUnmarkedNode();
};

void Dij::trys(string  fileName){
   ifstream file(fileName);

   int col_read = number;
```

```
    int row_read = number;

    for(int row = 0; row < row_read; ++row)
    {
        string line;
        getline(file, line);

        stringstream iss(line);
        for (int col = 0; col <col_read; ++col)
        {
            string val;
          getline(iss, val, ',');

            int number;
            number=atoi(val.c_str());
            adjMatrix[row][col] = number;
        }
    }
}

void Dij::initialize(){
    for(int i = 0; i < numOfV; i++){
        mark[i] = false;
        predecessor[i] = -1;
        distance[i] = INFINITY;
    }
    distance[source] = 0;
}


int Dij::getClosestUnmarkedNode(){
    int minDistance = INFINITY;
    int closestUnmarkedNode;
    for(int i = 0; i < numOfV; i++){
        if((!mark[i]) && (minDistance >= distance[i])){
            minDistance = distance[i];
            closestUnmarkedNode = i;
        }
    }
    return closestUnmarkedNode;
}

void Dij::calculateDistance(){
    initialize();
```

```
   int minDistance = INFINITY;
   int closestUnmarkedNode;
   int count = 0;
   while(count < numOfV){
      closestUnmarkedNode = getClosestUnmarkedNode();
      mark[closestUnmarkedNode] = true;
      for(int i = 0; i < numOfV; i++){
         if((!mark[i]) && (adjMatrix[closestUnmarkedNode][i] > 0)){
            if(distance[i] > distance[closestUnmarkedNode] +
adjMatrix[closestUnmarkedNode][i]){
               distance[i] = distance[closestUnmarkedNode] +
adjMatrix[closestUnmarkedNode][i];
               predecessor[i] = closestUnmarkedNode;
            }
         }
      }
      count++;
   }
}

void Dij::printPath(int node, ofstream &myfile){
   if(node == source){
      if(node == dest){
         myfile<<node+1;
      }else{
         myfile<<node+1<<",";
      }
   }
   else if(predecessor[node] == -1)
      myfile<<"No path from <<source<<to "<<node<<endl;
   else {
      printPath(predecessor[node], myfile);
      if(node == dest){
         myfile<<node+1;
      }else{
         myfile<<node+1<<",";
      }
   }
}

void Dij::output(){
   if(dest == source)
      cout<<source<<".."<<source;
   else
```

```
        //printPath(dest);

    cout<<"->"<<distance[dest]<<endl;
}


int main(){

    for (int networkIndex = 1; networkIndex <= 1; networkIndex++){
        string networkNames[4] = {"StarLike_50_97", "SF_10_40", "Rand_10_40",
"Ring_50_100"};
        string networkName = networkNames[networkIndex];
        ofstream myfile;
        myfile.open("ShortestDistance/Dij_" + networkName + ".txt");
        int totalDistance = 0;

        Dij G;
        G.trys("Network Topology/" + networkName + ".txt");

        for(int i = 0; i < number; i++){
                G.source = i;
                G.calculateDistance();
                for(int j = 0; j < number; j++){
                    G.dest = j;
                    totalDistance = totalDistance + G.distance[G.dest];
                    //myfile<<i<<","<<j<<","<<G.distance[G.dest]<<"\n";
                    G.printPath(G.dest, myfile);
                    myfile<<"\n";
                }
        }
        float avgDistance = float(totalDistance) * 2.0 / float(number* (number-1));
        myfile<<avgDistance<<"\n";
        myfile.close();
    }
    return 0;
}
```

L. The Information Transmission Simulation Model Code[4] (C++)

This is a C++ project used to simulate the information transmission process within a network based on the information transmission scenario specified in Table 18. The inputs for this program are the network topology, the resource exchange used, the ouput packet rate and the information storage capacity of each node.

a. Main1.CPP

```
#include "nodeevent.hpp"

#include <iostream>
#include <vector>
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>

float** resourceEx(string  fileName){
   ifstream file(fileName); //Change resource exchange file name YD

   int col_read = NUMBER;
   int row_read = NUMBER;

   float** resExMatrix = new float *[NUMBER];

   for(int row = 0; row < row_read; ++row)
   {
      string line;
      getline(file, line);
      resExMatrix[row] = new float [NUMBER];
      stringstream iss(line);
      for (int col = 0; col <col_read; ++col)
      {
         string val;
         getline(iss, val, ',');
         float probValue;
         probValue=atof(val.c_str());
         resExMatrix[row][col] = probValue;
```

[4] Part of this mode was constructed under the help of Chengwe Li.

```
        }
    }
    return resExMatrix;
}


int main()
{

    float** resExMatrix = resourceEx("ResExMatrix_No
Fluc/10_40_0.05_0.95/ExampleProblemRE_Simu.txt");

    Dij G[4];
    Init_Graph(&G[0]);
    //Create the Node_queue
    for(int i = 1; i<20; i++){
        Node_queue node_queue;
     //run the simulation
        simulation(node_queue,G[networkIndex], 0.05*i, resExMatrix, brandWidth);
   /*
   queue<int> a;
   for (int i=0;i<50;i++){
    for (int j=0;j<50;j++){
      std::cout<<"Path for "<<i<<"and"<<j<<std::endl;
    a=Shortest_Path(i,j,G[2]);
    Print_Path(a);
    Clear_Path(&G[2].shortestPath);
   }

  }*/

        node_queue = {};

    }


    return 0;
}
```

b. nodeevent.cpp

```
#include "nodeevent.hpp"
```

```
#define Queue_Capacity 5

using namespace std;



//!!!!!!!!!!!!!dimension should be changed into a variable
//generate the random NUMBER between 0~1
int rand(int a,int Range){
   srand(a);
   int A=rand()%Range;
   return A;
};

void send_to_Next(Node_Info_Passing&info, Node_queue&node_queue,int i,int
simu){
   info.position = info.Path.front();

   if(info.position != info.Destination){
      if ((node_queue.Queue_Size[info.position] + info.info_size) <=
Queue_Capacity){
         info.Path.pop();
         info.Simul_step = simu + 1;
         node_queue.addToQueue(info);
         node_queue.Queue_Size[info.position] += info.info_size;
         node_queue.success_pass_counter += 1;
      }else{
         node_queue.aborted_counter += 1;
      }
   }else{
      Printtofile_node(info, simu + 1);
      node_queue.success_deliver_counter += 1;
   }
   //decrease the total info-size of the current Queue_size
   node_queue.Queue_Size[i] -= info.info_size;
};


void prepare_node_to_send(Node_queue&node_queue, float Bandwidth, int
Num_node, int simu){
   float Band_width = Bandwidth;
```

```
   while(Band_width>0){
      if
(node_queue.Info_Queue[Num_node].size()&&node_queue.Info_Queue[Num_n
ode].front().Simul_step <= simu){
         if(node_queue.Info_Queue[Num_node].front().info_size <=
Band_width){
            send_to_Next(node_queue.Info_Queue[Num_node].front(),
node_queue, Num_node, simu);
            Band_width -= node_queue.Info_Queue[Num_node].front().info_size;
            node_queue.Info_Queue[Num_node].pop();
         }else{
            break;
         }

      }else{
         break;
      }

   }
};


//wrapper function for the Send_Node function
void Send_Node(Node_queue & node_queue,float Bandwidth,int simu){
   for(int i=0; i<NUMBER; i++){
   //message sending protocol, limited to the bandwidth of the node
      prepare_node_to_send(node_queue, Bandwidth, i, simu);
   }
};


mt19937 gen(time(NULL));
int uniIntRand(int n) {
   uniform_int_distribution<int> distribution(1, n);
   return distribution(gen)-1;
};

void Generate_node(Node_queue& node_queue, bool a, int n, double
info_size_max, int simu, Dij &G, float infoSize, float ** resExMatrix){
   for (int i=0;i<n;i++){
```

```
        Node_Info_Passing info;
        info.Origin = i;
        info.Simul_step = simu;
        info.Initial_Sim_Step = simu;
        info.position = i;
        info.info_size = infoSize; //Normal Distribution YD

        int dest;

        int diMethod = 2; //Change distribution method YD


//        ofstream SDPair;
//        SDPair.open ("Output/SDSPair.txt",ios::app);


        if(diMethod == 1){
           while(1){
              dest = uniIntRand(NUMBER);
              if(dest != info.Origin){
                 break;
              }
           }
        }else if(diMethod == 2){
           float destProb = uniIntRand(100);
           destProb = destProb / 100;
           for(int dest_i = 0; dest_i < NUMBER; dest_i++){
              if(destProb < resExMatrix[info.Origin][dest_i]){
                 dest = dest_i;
                 //SDPair<<destProb<<","<<resExMatrix[info.Origin][dest_i]<<
","<<info.Origin<<","<<dest<<"\n";
                 break;
              }
           }
        }

        info.Destination = dest;

        Compute_path(info, G);

        node_queue.addToQueue(info);
```

```
        node_queue.Queue_Size[info.position] += info.info_size;
    }
};


void Compute_path( Node_Info_Passing&info, Dij &G){
    info.Path = Shortest_Path(info.Origin, info.Destination, G);
    Clear_Path(&G.shortestPath);
    Print_Path(info.Path); //Origin is not included.
};


//wrapper function to get the simulation running
void simulation(Node_queue&node_queue,Dij &G, float infoSize, float **
resExMatrix, float brandWidth){
    float Bandwidth = brandWidth;
    float info_size_max = 2.0;
    for (int i=1; i < 301; i++){ //Change Number of Iteration
        Generate_node(node_queue, 0, NUMBER, info_size_max, i, G, infoSize,
resExMatrix);
        Send_Node(node_queue, Bandwidth, i);
        Printtofile(node_queue);
    }
};


void Printtofile(Node_queue&node_queue){
    ofstream queue_size_file;
    queue_size_file.open ("Output/" + to_string(networkIndex) + "_Queue_Size_"
+ to_string(resMatrixIndex) + "_" + brandWidthName + ".txt",ios::app);

    //node_queue.Queue_Size.size() = NUMBER for now
    for(int i=0; i<node_queue.Queue_Size.size(); i++){
        queue_size_file << node_queue.Queue_Size[i]<<",";
    }

    queue_size_file<<"\n";
    queue_size_file.close();

    //success_send_counter print out
```

```
    ofstream counter;
    counter.open ("Output/" + to_string(networkIndex) + "_Counter_Pa_De_Ab_"
+ to_string(resMatrixIndex) + "_" + brandWidthName+ ".txt",ios::app);
    counter << node_queue.success_pass_counter +
node_queue.success_deliver_counter<<","<<node_queue.success_deliver_counte
r<<","<<node_queue.aborted_counter<<"\n";
    counter.close();
};

void Printtofile_node(Node_Info_Passing&info, int time_step){
    //Node passing
    ofstream info_delivery_time;
    info_delivery_time.open ("Output/" + to_string(networkIndex) +
"_Info_Deliver_Time_" + to_string(resMatrixIndex) + "_" + brandWidthName+
".txt",ios::app);
    info_delivery_time <<time_step<<","<<time_step - info.Initial_Sim_Step;
    info_delivery_time<<"\n";
    info_delivery_time.close();
};

// const string fileName = "Network Topology/" + networkName + ".txt";
// string networkNames[4] = {"StarLike_50_97", "SF_50_100",
"Random_50_100", "Ring_50_100"};
// string networkName = networkNames[networkIndex];
```

c. Path.cpp

```
#include "Path.h"

#define INFINITY 999

using namespace std;



void Dij::trys(string  fileName){
    ifstream file(fileName);

    int col_read = NUMBER;
    int row_read = NUMBER;
```

```
   for(int row = 0; row < row_read; ++row)
   {
      string line;
      getline(file, line);

      stringstream iss(line);
      for (int col = 0; col <col_read; ++col)
      {
         string val;
        getline(iss, val, ',');

         int conn;
         conn=atoi(val.c_str());
         adjMatrix[row][col] = conn;
      }
   }
}

void Dij::initialize(){
   for(int i = 0; i < numOfV; i++){
      mark[i] = false;
      predecessor[i] = -1;
      distance[i] = INFINITY;
   }
   distance[source] = 0;
}


int Dij::getClosestUnmarkedNode(){
   int minDistance = INFINITY;
   int closestUnmarkedNode;
   for(int i = 0; i < numOfV; i++){
      if((!mark[i]) && (minDistance >= distance[i])){
         minDistance = distance[i];
         closestUnmarkedNode = i;
      }
   }
   return closestUnmarkedNode;
}

void Dij::calculateDistance(){
   initialize();
   int minDistance = INFINITY;
   int closestUnmarkedNode;
```

```
   int count = 0;
   while(count < numOfV){
      closestUnmarkedNode = getClosestUnmarkedNode();
      mark[closestUnmarkedNode] = true;
      for(int i = 0; i < numOfV; i++){
         if((!mark[i]) && (adjMatrix[closestUnmarkedNode][i] > 0)){
            if(distance[i] > distance[closestUnmarkedNode] +
adjMatrix[closestUnmarkedNode][i]){
               distance[i] = distance[closestUnmarkedNode] +
adjMatrix[closestUnmarkedNode][i];
               predecessor[i] = closestUnmarkedNode;
            }
         }
      }
      count++;
   }
}

void Dij::printPath(int node){
   if(node == source){
      //cout<<node<<"..";
   }
   else if(predecessor[node] == -1){
      //cout<<"No path from “<<source<<”to "<<node<<endl;
   }
   else {
      printPath(predecessor[node]);
      //cout<<node<<"..";
      shortestPath.push(node);
     //  cout<<shortestPath.back()<<"..";
   }
}

void Dij::output(){
   if(dest == source){
      //cout<<source<<".."<<source;
   }
   else
      printPath(dest);

   //cout<<"->"<<distance[dest]<<endl;
}

void Init_Graph(Dij*G){
```

```
    for (int networkIndex = 0; networkIndex < 3; networkIndex++){
            string networkNames[4] = {"ExampleProblemAdj", "SF_10_40",
"Rand_10_40", "Ring_10_40"}; //Change network name. YD
            string networkName = networkNames[networkIndex];
            ofstream myfile;
            //myfile.open("ShortestDistance/Dij_" + networkName + ".txt");
            G[networkIndex].trys("Network Topology/" + networkName + ".txt");
        }
}


queue<int> Shortest_Path(int source, int dest, Dij &G){

    G.source = source;
    G.dest = dest;
    G.calculateDistance();
    G.output();

    return G.shortestPath;
};

void Print_Path(queue<int> a){
    while (!a.empty()){
    //std::cout << ' ' << a.front();
    a.pop();
    }

    //std::cout << '\n';
};

void Clear_Path(queue<int> *a){
    while (!a->empty()){
    a->pop();
    }
};
```

d. Queue.cpp

```
#include "Queue.hpp"
int Node_queue::success_pass_counter=0;
int Node_queue::success_deliver_counter=0;
int Node_queue::aborted_counter=0;
```

```
std::vector<unsigned>
Node_queue::Queue_Conjested(static_cast<size_t>(NUMBER));

Node_queue::Node_queue():Info_Queue(static_cast<size_t>(NUMBER)),
                         Queue_Size(static_cast<size_t>(NUMBER)){};


 void Node_queue::addToQueue(const Node_Info_Passing& node){

 Info_Queue[node.position].push(node);

 };

  unsigned Node_queue::queueSize(const Node_Info_Passing& node ) const{
    return Info_Queue[node.position].size();
  };


  const Node_Info_Passing& Node_queue::viewFrontNodeInfo(const
Node_Info_Passing& node) const
  {
     return Info_Queue[node.position].front();
  };

  Node_Info_Passing Node_queue::getFrontNode(const Node_Info_Passing&
node)
  {
      Node_Info_Passing front_node(Info_Queue[node.position].front());
      Info_Queue[node.position].pop();
      return front_node;
  };


 void Node_queue::increaseGroupSize(const Node_Info_Passing& node){
    Queue_Size[node.position]+=1;
 };

 void Node_queue::decreaseGroupSize(const Node_Info_Passing& node)
 {
    Queue_Size[node.position]-=1;
  };
```

```
unsigned Node_queue::groupSize(const Node_Info_Passing& node) const
{
 return Queue_Size[node.position];
};

void Node_queue::Set_Conjested(const Node_Info_Passing& node)
{
   Queue_Conjested[node.position]=false;
};

void Node_queue::Clear_Conjested_state(const Node_Info_Passing&node)
{
  Queue_Conjested[node.position]=true;
};

bool Node_queue::isConjested(const Node_Info_Passing& node)
{
    return Queue_Conjested[node.position];
};
```

e. nodeevet.hpp

```
#ifndef _NODEEVENT_HPP_
#define _NODEEVENT_HPP_

#include "Path.h"

#include <cstdio>
#include <iostream>
#include <queue>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <cmath>
#include <fstream>
#include <random>


int rand(int a,int Range);
```

```
void send_to_Next(Node_Info_Passing&node,Node_queue&node_queue,int i,int
simu);
void prepare_node_to_send(Node_queue&node_queue,float Bandwidth,int
Num_node,int simu);
void Send_Node(Node_queue & node_queue,float Bandwidth,int simu);
void Generate_node(Node_queue& node_queue,bool a,int n,double
info_size_max,int simu,Dij&G, float infoSize, float ** resExMatrix);
void Compute_path( Node_Info_Passing&node,Dij &G);
void simulation(Node_queue&node_queue,Dij &G, float infoSize, float **
resExMatrix, float brandWidth);
void Printtofile(Node_queue&node_queue);
void Printtofile_node(Node_Info_Passing&node,int time_step);
int uniIntRand(int n) ;
#endif
```

f. path.hpp

```
#ifndef DISTANCE_H_INCLUDED
#define DISTANCE_H_INCLUDED

#include "Queue.hpp"

#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <iomanip>
#include <stdlib.h>
#include <vector>
#include <queue>

using namespace std;


class Dij{
   public:
      queue <int> shortestPath;
      static const int numOfV = NUMBER;
      int predecessor[numOfV], distance[numOfV];
```

```
        int adjMatrix[NUMBER][NUMBER];
        void trys(string);
        int tree[numOfV][numOfV];
        bool mark[numOfV];
        int source;
        int dest;
        void initialize();
        void calculateDistance();
        void output();
        void printPath(int);
        int getClosestUnmarkedNode();

};
void Init_Graph(Dij*G);
queue<int> Shortest_Path(int, int , Dij&);
void Print_Path(queue<int> a);
void Clear_Path(queue<int> *a);


#endif // DISTANCE_H_INCLUDED
```

g. Queue.hpp

```
#ifndef QUEUE_HPP_
#define QUEUE_HPP_

#include <queue>
#include <vector>
#include <iostream>
#include <string>

#include <sstream>

namespace patch
{
   template < typename T > std::string to_string( const T& n )
   {
      std::ostringstream stm ;
      stm << n ;
      return stm.str() ;
   }
}

using namespace std;
using namespace patch;

static const int NUMBER = 7;
static const int resMatrixIndex = 3;
static const int networkIndex = 0;
static const float brandWidth = 1;
static const string brandWidthName = "1";

struct Node_Info_Passing{
  int Origin;
  int Destination;
  int position;
  std::queue<int> Path;
  float info_size;
  int Simul_step;
  int Initial_Sim_Step;
};
```

```
class Node_queue {
public:
  enum queue_static {Normal,Conjested,Empty};

  Node_queue();

  static int success_pass_counter;
  static int success_deliver_counter;
  static int aborted_counter;

public:
  void addToQueue(const Node_Info_Passing&);

  unsigned queueSize(const Node_Info_Passing&) const;

  const Node_Info_Passing& viewFrontNodeInfo(const Node_Info_Passing&)
const;

  Node_Info_Passing getFrontNode(const Node_Info_Passing&);

  void increaseGroupSize(const Node_Info_Passing&);

  void decreaseGroupSize(const Node_Info_Passing&);

  unsigned groupSize(const Node_Info_Passing&) const;


public:
  static void Set_Conjested(const Node_Info_Passing&);

  static void Clear_Conjested_state(const Node_Info_Passing&);

  static bool isConjested(const Node_Info_Passing&);

//private:
  std::vector<std::queue<Node_Info_Passing> > Info_Queue;
  std::vector<float> Queue_Size;

//private:
  static std::vector<unsigned> Queue_Conjested;
```

```
}; // class QueueState

#endif // QUEUE_HPP_
```

M. The "Link Minus" Network Topology Optimization Code (Matlab)

This program is used to generate the optimized network topoglogy for a given network node number based on the process discussed in Section 6.3. The inputs for this program are the number of netwowrk nodes and the set of constrained network links that cannot be removed during the "Link Minus" process.

```
clc;
clear;
nodeNum = 8;
adjMatrix = importdata('adj_design.csv', ',' , 0);
count = 1;
results = [];

adjMatrixCopy = adjMatrix;
results(count,1) = 0;
results(count,2) = 0;
results(count,3) = 0;
results(count,4) = ER_Cal(8,adjMatrix,1,8);

for inter = 1:8
%     h = view(biograph(sparse(adjMatrix)));
   temp = adjMatrix.*linkIMP_2(8, adjMatrixCopy, 1, 8);
   tempCopy = temp;
   tempCopy(:,8) = 0;
   tempCopy(8,:) = 0;

   S = 2;
```

```
    while (S>=2)
        [value, index] = max(reshape(tempCopy, numel(tempCopy), 1));
        [i,j] = ind2sub(size(tempCopy), index);
        adjCopy = adjMatrix;
        adjCopy(i,j) = 0;
        adjCopy(j,i) = 0;
        [S, C] = graphconncomp(sparse(adjCopy(1:7,1:7)));
        if(S>=2)
            tempCopy(i,j) = 0;
            tempCopy(j,1) = 0;
        end

        if(sum(sum(tempCopy))==0)
            break;
        end
    end

    if(sum(sum(tempCopy))>0)
        count = count + 1;
        results(count,1) = i;
        results(count,2) = j;
        results(count,3) = temp(i,j);
        adjMatrix(i,j) = 0;
        adjMatrix(j,i) = 0;
        results(count,4) = ER_Cal(8,adjMatrix,1,8);
    end
end
```

```
view(biograph(adjMatrix))
```

REFERENCES

1. Bellinger, G., D. Castro, and A. Mills, *Data, information, knowledge, and wisdom.* URL: http://www. systems-thinking. org/dikw/dikw. htm, 2004: p. 47.
2. Dodds, P.S., D.J. Watts, and C.F. Sabel, *Information exchange and the robustness of organizational networks.* Proceedings of the National Academy of Sciences, 2003. **100**(21): p. 12516-12521.
3. Durugbo, C., et al., *Modelling collaboration using complex networks.* Information Sciences, 2011. **181**(15): p. 3143-3161.
4. White, A., *A Collaboration Network for Unmanned Aerial Vehicle Operation, Research and Education*. 2005, DTIC Document.
5. Frew, E.W. and T.X. Brown, *Networking issues for small unmanned aircraft systems.* Journal of Intelligent and Robotic Systems, 2009. **54**(1-3): p. 21-37.
6. Derya, A., G. Kelly, and N.M. Dimitri, *UAVs for Law Enforcement: A Case Study for Connectivity and Fuel Management*, in *14th AIAA Aviation Technology, Integration, and Operations Conference*. 2014, American Institute of Aeronautics and Astronautics.
7. Engh, C., M.A. Goodrich, and B.S. Morse, *UAV Video Coverage Quality Maps and Prioritized Indexing for Wilderness Search and Rescue.* 2010.
8. Yanmaz, E., et al. *A discrete stochastic process for coverage analysis of autonomous UAV networks*. in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*. 2010. IEEE.
9. Frew, E.W., et al., *Networked communication, command, and control of an unmanned aircraft system.* Journal of Aerospace Computing, Information, and Communication, 2008. **5**(4): p. 84-107.
10. Long, T., A. Bati, and D. Hilliard. *Radar Cross Section measurements of small Unmanned Air Vehicle Systems in non-cooperative field environments*. in *Antennas and Propagation, 2009. EuCAP 2009. 3rd European Conference on*. 2009.
11. Chao, H., Y. Cao, and Y. Chen. *Autopilots for small fixed-wing unmanned air vehicles: A survey*. in *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*. 2007. IEEE.
12. George, J., S. P. B, and J.B. Sousa, *Search Strategies for Multiple UAV Search and Destroy Missions.* Journal of Intelligent & Robotic Systems, 2011. **61**(1-4): p. 355-367.
13. Sun, Z., et al., *BorderSense: Border patrol through advanced wireless sensor networks.* Ad Hoc Networks, 2011. **9**(3): p. 468-477.
14. Nigam, N. and I. Kroo. *Control and design of multiple unmanned air vehicles for a persistent surveillance task*. 2008.
15. Beard, R.W., et al., *Decentralized Cooperative Aerial Surveillance Using Fixed-Wing Miniature UAVs.* Proceedings of the IEEE, 2006. **94**(7): p. 1306-1324.
16. Valenti, M., D. Dale, and J. How. *Mission health management for 24/7 persistent surveillance operations*. 2007.
17. Casbeer, D.W., A.L. Swindlehurst, and R. Beard, *Connectivity in a UAV multi-static radar network.* 2006.
18. Barrado, C., et al., *Wildfire monitoring using a mixed air-ground mobile network.* Pervasive Computing, IEEE, 2010. **9**(4): p. 24-32.
19. Zajkowski, T., S. Dunagan, and J. Eilers. *Small UAS communications mission*. 2006.

20. Cho, A., et al., *Wind estimation and airspeed calibration using a UAV with a single-antenna GPS receiver and pitot tube.* Aerospace and Electronic Systems, IEEE Transactions on, 2011. **47**(1): p. 109-117.
21. White, B.A., et al., *Contaminant Cloud Boundary Monitoring Using Network of UAV Sensors.* Sensors Journal, IEEE, 2008. **8**(10): p. 1681-1692.
22. Corrigan, C.E., et al., *Capturing vertical profiles of aerosols and black carbon over the Indian Ocean using autonomous unmanned aerial vehicles.* Atmos. Chem. Phys. Discuss., 2007. **7**(4): p. 11429-11463.
23. Curry, J., et al., *Applications of Aerosondes in the Arctic.* Bulletin of the American Meteorological Society, 2004. **85**(12): p. 1855-1861.
24. Maza, I., et al., *Experimental results in multi-UAV coordination for disaster management and civil security applications.* Journal of intelligent & robotic systems, 2011. **61**(1-4): p. 563-585.
25. Sofge, E. *Houston Cops Test Drone Now in Iraq, Operator Says*. 2008; Available from: http://www.popularmechanics.com/flight/drones/a2324/4234272/.
26. Government. *Systems Engineering Guide for Systems of Systems*. 2008.
27. Chakrabarti, D., et al., *Epidemic thresholds in real networks.* ACM Transactions on Information and System Security (TISSEC), 2008. **10**(4): p. 1.
28. Gorod, A., B.J. Sauser, and J.T. Boardman, *System-of-Systems Engineering Management: A Review of Modern History and a Path Forward.* IEEE Systems Journal, 2008. **2**(4): p. 484-499.
29. DoD, U., *DoDAF Architecture Framework Version 2.02.* Website, August, 2010.
30. Garstka, J., *Implementation of Network Centric Warfare.* Transformation Trends, 2004. **28**.
31. Noda, I. and M. Hatayama. *Common Frameworks of Networking and Information-Sharing for Advanced Rescue Systems*. in *Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on*. 2004. IEEE.
32. Portmann, M. and A.A. Pirzada, *Wireless mesh networks for public safety and crisis management applications.* Internet Computing, IEEE, 2008. **12**(1): p. 18-25.
33. Manzano, M., E. Calle, and D. Harle. *Quantitative and qualitative network robustness analysis under different multiple failure scenarios*. in *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2011 3rd International Congress on*. 2011. IEEE.
34. Purohit, A., F. Mokaya, and P. Zhang. *Collaborative indoor sensing with the sensorfly aerial sensor network*. in *Proceedings of the 11th international conference on Information Processing in Sensor Networks*. 2012. ACM.
35. Yanmaz, E., R. Kuschnig, and C. Bettstetter. *Channel measurements over 802.11 a-based UAV-to-ground links*. in *GLOBECOM Workshops (GC Wkshps), 2011 IEEE*. 2011. IEEE.
36. Bekmezci, I., O.K. Sahingoz, and Ş. Temel, *Flying ad-hoc networks (FANETs): A survey.* Ad Hoc Networks, 2013. **11**(3): p. 1254-1270.
37. Baillieul, J. and P.J. Antsaklis, *Control and communication challenges in networked real-time systems.* Proceedings of the IEEE, 2007. **95**(1): p. 9-28.
38. Li, J., et al., *Packet Delay in UAV Wireless Networks Under Non-saturated Traffic and Channel Fading Conditions.* Wireless Personal Communications, 2013. **72**(2): p. 1105-1123.

39. Criado, R., et al., *(ψ, p, q)-vulnerabilities: A unified approach to network robustness.* Chaos: An Interdisciplinary Journal of Nonlinear Science, 2009. **19**(1): p. 013133.
40. Albert, R., H. Jeong, and A.-L. Barabasi, *Error and attack tolerance of complex networks.* Nature, 2000. **406**(6794): p. 378-382.
41. Crucitti, P., et al., *Efficiency of scale-free networks: error and attack tolerance.* Physica A: Statistical Mechanics and its Applications, 2003. **320**: p. 622-642.
42. Latora, V. and M. Marchiori, *Efficient behavior of small-world networks.* Physical review letters, 2001. **87**(19): p. 198701.
43. Dunne, J.A., R.J. Williams, and N.D. Martinez, *Network structure and biodiversity loss in food webs: robustness increases with connectance.* Ecology letters, 2002. **5**(4): p. 558-567.
44. Dekker, A.H. *Simulating network robustness: two perspectives on reality*. in *Proceedings of SimTecT 2004 Simulation Conference*. 2004.
45. Balestrini Robinson, S., *A modeling process to understand complex system architectures.* 2009.
46. Aksaray, D., K. Griendling, and D. Mavris. *UAVs for Law Enforcement: A Case Study for Connectivity and Fuel Management*. in *14th AIAA Aviation Technology, Integration, and Operations Conference*. 2014.
47. Perry, W., et al., *Measurements of Effectiveness for the Information-Age Navy: The Effects of Network-Centric Operations on Combat Outcomes*. 2002, DTIC Document.
48. Ellens, W. and R.E. Kooij, *Graph measures and network robustness.* arXiv preprint arXiv:1311.5064, 2013.
49. Abraham, I., et al., *Alternative routes in road networks.* Journal of Experimental Algorithmics (JEA), 2013. **18**: p. 1.3.
50. Dekker, A.H. and B.D. Colbert. *Network robustness and graph topology*. in *Proceedings of the 27th Australasian conference on Computer science-Volume 26*. 2004. Australian Computer Society, Inc.
51. Newman, M.E., *The structure and function of complex networks.* SIAM review, 2003. **45**(2): p. 167-256.
52. Bozzo, E., *The Moore–Penrose inverse of the normalized graph Laplacian.* Linear Algebra and its Applications, 2013. **439**(10): p. 3038-3043.
53. Spielman, D.A. *Algorithms, graph theory, and linear equations in Laplacian matrices*.
54. Abbas, W. and M.B. Egerstedt, *Robust graph topologies for networked systems.* 2012.
55. Wang, X., et al., *Improving robustness of complex networks via the effective graph resistance.* The European Physical Journal B, 2014. **87**(9): p. 1-12.
56. Ellens, W., et al., *Effective graph resistance.* Linear algebra and its applications, 2011. **435**(10): p. 2491-2506.
57. Young, G.F., L. Scardovi, and N.E. Leonard, *A New Notion of Effective Resistance for Directed Graphs-Part I: Definition and Properties.* arXiv preprint arXiv:1310.5163, 2013.
58. Li, Y. and Z.-L. Zhang, *Digraph laplacian and the degree of asymmetry.* Internet Mathematics, 2012. **8**(4): p. 381-401.
59. Chung, F. and W. Zhao, *PageRank and Random Walks on Graphs*, in *Fete of Combinatorics and Computer Science*, G.H. Katona, et al., Editors. 2010, Springer Berlin Heidelberg. p. 43-62.

60. Boley, D., G. Ranjan, and Z.-L. Zhang, *Commute times for a directed graph using an asymmetric Laplacian.* Linear Algebra and its Applications, 2011. **435**(2): p. 224-242.
61. Young, G.F., L. Scardovi, and N.E. Leonard, *A New Notion of Effective Resistance for Directed Graphs-Part II: Computing Resistances.* arXiv preprint arXiv:1310.5168, 2013.
62. Mahadevan, P., et al., *The Internet AS-level topology: three data sources and one definitive metric.* ACM SIGCOMM Computer Communication Review, 2006. **36**(1): p. 17-26.
63. Jamakovic, A. and P. Van Mieghem, *On the robustness of complex networks by using the algebraic connectivity*, in *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*. 2008, Springer. p. 183-194.
64. Fiedler, M., *Algebraic connectivity of graphs.* Czechoslovak Mathematical Journal, 1973. **23**(2): p. 298-305.
65. Jamakovic, A. and S. Uhlig. *Influence of the network structure on robustness*. in *Networks, 2007. ICON 2007. 15th IEEE International Conference on*. 2007. IEEE.
66. Baras, J.S. and P. Hovareshti. *Efficient and robust communication topologies for distributed decision making in networked systems*. in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*. 2009. IEEE.
67. Ellens, W., *Effective resistance.* 2011.
68. Klein, D.J. and M. Randić, *Resistance distance.* Journal of Mathematical Chemistry, 1993. **12**(1): p. 81-95.
69. Barabási, A.-L. and R. Albert, *Emergence of scaling in random networks.* science, 1999. **286**(5439): p. 509-512.
70. ERDdS, P. and A. WI, *On random graphs I.*
71. Fouss, F., et al., *Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation.* Knowledge and data engineering, ieee transactions on, 2007. **19**(3): p. 355-369.
72. Ranjan, G., *Understanding (Inter-) dependencies and vulnerabilities in static and dynamic networks*. 2013, University of Minnesota.
73. Newman, M., *Networks: an introduction*. 2010: Oxford University Press.
74. Criado, R., et al., *Understanding complex networks through the study of their critical nodes: efficiency, vulnerability and dynamical importance.* New Trends and Tools in Complex Networks, 2007: p. 23.
75. Boccaletti, S., et al., *Complex networks: Structure and dynamics.* Physics reports, 2006. **424**(4): p. 175-308.
76. Farkas, I.J., et al., *Spectra of "real-world" graphs: Beyond the semicircle law.* Physical Review E, 2001. **64**(2): p. 026704.
77. Faloutsos, M., P. Faloutsos, and C. Faloutsos. *On power-law relationships of the internet topology*. in *ACM SIGCOMM computer communication review*. 1999. ACM.
78. Freeman, L.C., *A set of measures of centrality based on betweenness.* Sociometry, 1977: p. 35-41.
79. Freeman, L.C., *Centrality in social networks conceptual clarification.* Social networks, 1979. **1**(3): p. 215-239.
80. Stephenson, K. and M. Zelen, *Rethinking centrality: Methods and examples.* Social Networks, 1989. **11**(1): p. 1-37.

81. Noh, J.D. and H. Rieger, *Random walks on complex networks.* Physical review letters, 2004. **92**(11): p. 118701.
82. Freeman, L.C., S.P. Borgatti, and D.R. White, *Centrality in valued graphs: A measure of betweenness based on network flow.* Social networks, 1991. **13**(2): p. 141-154.
83. Lovász, L., *Random walks on graphs: A survey.*
84. Wang, C.-h. and L. Pham. *Resilience and robustness*. in *Australasian Structural Engineering Conference 2012: The past, present and future of Structural Engineering*. 2012. Engineers Australia.
85. Okoh, P. and S. Haugen, *Improving the robustness and resilience properties of maintenance.* Process Safety and Environmental Protection, 2015. **94**: p. 212-226.
86. Hu, Z., K. Thulasiraman, and P.K. Verma, *Complex Networks: Traffic Dynamics, Network Performance, and Network Structure.* American Journal of Operations Research, 2013. **3**: p. 187.
87. Tizghadam, A. and A. Leon-Garcia. *On congestion in mission critical networks*. in *INFOCOM Workshops 2008, IEEE*. 2008. IEEE.
88. Hu, Z. and P.K. Verma. *Impact of network structure on latency in complex networks*. in *Sarnoff Symposium (SARNOFF), 2012 35th IEEE*. 2012. IEEE.
89. Hu, Z., P.K. Verma, and K. Thulasiraman. *Interplay Between Traffic Dynamics and Network Structure*. in *ICONS 2013, The Eighth International Conference on Systems*. 2013.
90. De Martino, D., et al., *Congestion phenomena on complex networks.* arXiv preprint arXiv:0808.0584, 2008.
91. *Packet Switching*. [Website] 2016 01/10/2016 [cited 2016 01/26/2016]; Available from: https://en.wikipedia.org/wiki/Packet_switching.
92. Tizghadam, A. and A. Leon-Garcia. *On traffic-aware betweenness and network criticality*. in *INFOCOM IEEE Conference on Computer Communications Workshops, 2010*. 2010. IEEE.
93. Conrad, K., *Probability distributions and maximum entropy.* retrieved November, 2013. **14**: p. 2013.
94. Jaynes, E.T., *Prior probabilities.* Systems Science and Cybernetics, IEEE Transactions on, 1968. **4**(3): p. 227-241.
95. Good, I.J., *Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables.* The Annals of Mathematical Statistics, 1963: p. 911-934.
96. Ranjan, G., Z.-L. Zhang, and D. Boley, *Incremental computation of pseudo-inverse of laplacian*, in *Combinatorial Optimization and Applications*. 2014, Springer. p. 729-749.
97. Courrieu, P., *Solving time of least square systems in Sigma-Pi unit networks.* arXiv preprint arXiv:0804.4808, 2008.
98. Courrieu, P., *Fast computation of Moore-Penrose inverse matrices.* arXiv preprint arXiv:0804.4809, 2008.
99. Wyatt, E.J., K. Griendling, and D.N. Mavris. *Addressing interoperability in military systems-of-systems architectures*. in *Systems Conference (SysCon), 2012 IEEE International*. 2012. IEEE.
100. Tran, H.T., J.C. Domercant, and D. Mavris, *Trade-offs Between Command and Control Architectures and Force Capabilities Using Battlespace Awareness*. 2014, DTIC Document.

VITA

Yuqian Dong grew up in Shenyang, China and later wend to Beijing, China for her undergraduate studies. She obtained her Bachelor of Engineering from Beijing University of Aeronautics and Astronautics in 2010. Later that year she began her graduate school in the school of Aerospace Engineering at Georgia Institute of Technology. She started her study as a Graduate Research Assistant in the Aerospace Systems Design Laboratory under the advisement of Dr. Dimitri Mavris. In 2011, she joined the ARCHITECT research team, which major focus was conducting fundamental researches for the Office of Naval Research. Her responsibility in the team is to help understand and address the issues related to the uncertainties and robustness of System of Systems Architectures. The researches she conducted led to this thesis, which is to be parlayed into at least one journal paper.

During the process of pursuing her Ph.D., Yuqian finished her Masters of Science in Aerospace Engineering in 2013 and in Operations Research in 2015. Currently she works at UPS at a senior analyst.