Network Research: Exploration of Centrality Measures and Network Flows Using Simulation Studies

Written by: Zhipeng Luo,
Supervisors: Dr. Chintan Amrit, Dr. Klaas Sikkel

November 5, 2018
Enschede

Abstract

The study of network centrality has many real-world applications (e.g. targeted marketing strategy in business, identification of vital players in a criminal network, identification of keystone proteins in bio-research.) According to Borgatti (2005), networks are differentiated by the flow mechanism (i.e. Parallel replication, Serial replication or Transfer) and the flow trajectory (i.e. Geodesic, Paths, Trails or Walks). However, the classification of networks showed that Freemans centrality measures are not applicable to various types of networks and flows. The aim of this research is to determine centrality measures that are applicable to varying network types and flows. An in-depth literature research was conducted to gain a deep understanding of various networks in terms of structure and information flow within the networks. In the process of deriving plausible centrality measures to address networks other than geodesic transfer network, betweenness-like (Flow centrality, Random-walk betweenness), diffusion, key players set and closeness-like measures were considered. The derived centrality approaches were validated using Entropy measures (Connectivity and Centrality Entropy). A simulation study was conducted to determine the impact of network properties (size, power-law/linkage parameter, clustering coefficient)(random vs. scale-free network) on all the proposed centrality measures. The results of different experiments showed that network cohesion is the main centrality factor. Network cohesion determines whether a network has a single central node or a set of key nodes. Furthermore, the network cohesion factor also proved that certain networks are comparable to some of the proposed centrality measures under certain network settings. However, the experiments also showed that the randomness factor had some impact on the results. Future research direction is aimed to address some of the limitations found in this research.

1 Introduction

Network theory is the study of graphs as a representation of symmetric relations, which provides explanations for a myriad of social phenomena, from individual creativity to corporate profitability (Borgatti et al., 2009). In social science, sociologists saw concrete relations between people love, hate, support, and so on as the basis of community, and they used network analysis to represent community structure. In business studies, network study is a popular focus. It enables companies to develop targeted strategies that can be used in different areas (such as: marketing or product development) (Landherr et al., 2010).

Network analysis consists of characterizing network structures (e.g., small-worldness), node positions (e.g., centrality) and relating these to group and node outcomes (Borgatti and Halgin, 2011). The most widely studied concept is centrality, which entails the properties and characteristics at the node level relating to the structural importance or prominence of a node in the network (Borgatti et al., 2009). These properties and characteristics include the number of nearest neighbors, the average distance among all other nodes, the number of go-between between two other nodes, and more. The knowledge and technique of determining the most central nodes of a network could be very beneficial for the real world applications. The research by De Laat (2002) in finding the central actors of a drug-related network was done so by using Freemans degree (the number of nearest neighbors) and betweenness (the number of go-between between two other nodes) centrality. Much research has been conducted and posited that the more centralized structures (star structure), outperformed decentralized structures (circle structure) in information flow, even though it could be shown mathematically that the circle structure had, in principle, the shorter time needed for information to transfer (Borgatti et al., 2009). In biology and ecology, centrality measures become an important tool in identifying essential proteins, keystone species, information hubs and/or valuable infrastructures (Enriquez, 2010).

1.1 Motivation

The development of centrality measure research by Freeman (1978) was groundbreaking. The measures were developed according to three concepts: one absolute, one relative of the centrality of positions in a network and, one reflecting the degree of centralization of the entire network. The Freemans centrality has led to the work by Borgatti (2005) in the attempt of mapping appropriate centrality measures to the classification of network flow topologies. However, the work started by Borgatti (2005) was never finished. This is simply because Freemans centralities could not be applied to all types of network flow. One possible explanation is that Freemans centralities are built on the premise that information flows along the shortest path (Borgatti, 2005). This is however not the case for all types of network. As result, Freemans centralities would fail in those situations. Over the years, many researchers have tried to fill in the blanks. For instance, the random-walk betweenness proposed by Newman

(2005) was developed to account for non-geodesic flows, it was considered reliable in some cases, others have tried to consider multiple vital positions (key player set) (Ortiz-Arroyo and Hussain, 2008), and random-walk betweenness were derived to account for walks types of network flows. With that being said, deriving appropriate centralities for various types of network is a vital part of network research and may be contributive to various real-world applications.

1.2 Objectives

Freemans centralities (1979) were derived in the attempt of determining the most central person in a social network analysis (SNA). A person is considered the most central when he/she has the most contacts, well connected or a go-between. However, there are other non-social networks that Freeman had not accounted for. Few examples of these networks are E-mail broadcast network, mitotic reproduction, Internet name-server network, and more. Since the work started by Freeman, other centralities include eigenvector and Katz centrality have been developed in the continuation of his work. Though the work of Borgatti (2005) could not shed light on networks other than transfer networks that follow the shortest path (Geodesic). With that being said, the main objective of this research is to explore the possible centrality measures for the other types of network. This will be achieved by means of utilizing centrality insights of various authors and data scientists. The derived centrality metrics will be closely studied through a simulation study. The results obtained from the simulation analysis will help to finish the work started by Freeman and Borgatti. In order to arrive at this phase, it is important to have a deep understanding of how these networks behave and what they have in common. Furthermore, it may also be insightful to examine the types of network that are a mixture of multiple flow types. With these objectives in mind, a list of research questions is formulated.

1.3 Research Questions

1. How does the flow process differ between different types of network or network settings?
2. What possible approaches could be used to determine the centrality of networks that are not geodesic?
3. How can simulation be used to determine centrality of various network types?
4. How do network properties impact the outcomes of different centrality measures?

1.4 Approach

In order to obtain information to address the above-mentioned areas of focus, a systematic review of network centrality will be conducted as well as simulation experiments. During the systematic review, different ideas, perspectives,

approaches and findings from different authors will be taken into consideration. The reason behind such a literature review is to gain a deep understanding of the topic and knowledge of what researchers found that may have a deep contribution to my research. Based on the found theories, test conditions will be created. Finally, simulation algorithms will be developed to test those conditions and validate the simulation findings by comparing them to the related work.

2 Background of Network Theory

A network consists of nodes (also called vertices) and edges. Each node can be interpreted as a person, organization or an individual component in a system. Each edge can be interpreted as a relationship that connects one node with another. An example of the edges in a family network would be love, each family member possesses a certain degree of love towards each member in the family. However, this love may not be mutual or balance. Wasserman and Faust (1993) stated that social network can be classified as a one-sided or two-sided relationship ((un)directed network), and/or if the relationship is with different intensities ((un)weighted network). An example of undirected love would be the love between the brother and sister, and an example of directed love would be the love the family has for their family pet. A weighted family network may refer to the different degree of love each member holds. The entire network represents the workings real-life system. Travers and Milgram (1969) argued that everyone is connected to everyone else on average of six contacts, as it is called the small world phenomenon. Within a social network, there is always a person or set of persons that play a significant role or have the most influence on the network. The fact is social networks are mostly scale-free in which the number of contacts per person is not evenly distributed (Barabási, 2009). Rather, social networks are made up of many interconnected individual groups (i.e. family and friends) and only a few highly integrated members (i.e. middleman). The existence of such person (i.e. middleman) is what makes the network functional. Removing this person would drastically impact on the coherence of the network communication, relation as well as the information exchange. The study of the centrality of a network deals with identifying the key players in a network. Such study plays a crucial part in our daily life, knowing who or what plays a vital role in a network enables us to take a more directed approach in addressing the issue, whether it is improving information exchange within a network or the disruption of a terror cell network.

2.1 Classification of Network

Before delving into what the centrality measures are and how they work, it is important to understand the characteristics of different networks. Borgatti (2005) stated that the variations of networks are differentiated by three factors. The concept of information mentioned in each factor refers to the object that is

being circulated in a particular network. This information differs based on the types of network.

2.1.1 Transfer Mechanism

The dyadic diffusion mechanism refers to how information moves from one node to another. That is, whether diffusion occurs via replication mechanism or transfer mechanism. An example of replication could be a virus spreading from host to others by replicating the virus. In this example, the information that flows within the network refers to the contagious virus. An example of transfer could be trading of a used good where an object is passed from person to person without replicating. In this case, the information being transferred within the network refers to the used good.

2.1.2 Flow Mechanism

The rate of replication refers to the form in which information is being replicated. This factor is only applicable to replication mechanism. That is, whether the replication happens one at a time (Serial) or simultaneously (Parallel). An example of a serial replication could be gossip that takes place between two people, the information in such a network refers to the topic and content of the gossip. An example of a simultaneous replication could be radio broadcast, the information in such a network refers to the news or songs that is being broadcast.

2.1.3 Graph Trajectory

According to graph theory, the flow of information in a network can follow a path, a walk, a trail or geodesic. A path is a flow from node (s) to (t) via a set of nodes without repeating any node. This implies that a path will also not have any repeated edges. A path can be open (line segment) or closed (polygon). That is, whether the end node of the path equals the start node. A trail is a flow from node (s) to (t) via a set of nodes that may contain repeated nodes but without having any edge repeated. A walk is a set of nodes that may contain repeated nodes and edges. Lastly, geodesic refers to the shortest path (geodesic). An example of geodesic flow could be traffic navigation that determines a route based on earliest arrival time.

Borgatti (2005) combined and collapsed these factors and formed table 1 with two dimensions that describe all the networks. On one dimension, the relationship of node-to-node transmission (Parallel, Serial, Transfer) is described. As for the other dimension, the trajectory of the network flow (Geodesic, Paths, Trails, Walks) is described.

Table 1: The table was originally created by Borgatti (2005), which describes the different network types with a real world example.

	Parallel Duplication	Serial Duplication	Transfer
Geodesic	<no process>	Mitotic reproduction	Package delivery
Paths	Internet name-server	Viral infection	Mooch
Trails	E-mail broadcast	Gossip	Used goods
Walks	Attitude influence	Emotional support	Money exchange

3 Measures of Network

Centrality measure has been considered as an attribute of an actor's position in the network, measurable without regard to how it is connected to others and who and, in turn, how these others are positioned in the network. According to Freeman (1978): "There is certainly no unanimity on exactly what centrality is or on its conceptual foundations, and there is little agreement on the proper procedure for its measurement. Similarly, Borgatti and Everett (2006) also stated that there is no uniform understanding of an actors central network. There is, however, some very different context-specific interpretations of node centrality which may be the result of the different objectives for the use of centrality measures. These authors posited that determining the centrality of a network is not bounded to one specific measure. Instead, Centrality can be calculated in numerous ways, which depends entirely on the objectives of the search. Furthermore, there is a major distinction between finding one central node and finding a set of central nodes. Borgatti and Everett (2006) argued that the problem of getting an optimal set of k-players is different from the problem of selecting k individuals that are each, individually optimal. This insight originated from the fact that a network may contain multiple nodes that in spite of not having a high degree, have a greater impact in disrupting the network structure when removed (Ortiz-Arroyo and Hussain, 2008).

3.1 Centrality measures for a key player

Freeman (1978) argued that the approaches to centrality are based on three ideas about what being central means: (1) being active within the network, that is, maintaining many ties, (2) being efficient, that is, closer to all other nodes in the network, and (3) being an important go-between, that is, being part of many paths between other nodes in the network. The Following centrality measures have all been derived from an unweighted and undirected network. A distinction was made by Borgatti and Everett (2006) between radial and medial measures. Radial refers to metrics that start or end at a given node, whereas medial refers to the amount of information passes through a node. Furthermore, these mentioned measures only cover a portion of Borgattis network classification matrix (Table 1).

3.1.1 Degree

Degree Centrality (DC) is a type of radial centrality measure that counts the number of edges a node is attached to. For comparison purposes, DC must be standardized by dividing by the maximum possible value $\big(n-1\big)$. The rule is, the node with the highest DC value is the most central node.

DC is one of the easiest measures. It is also considered as a highly effective measure. In many social settings, the people with more connections tend to have more power and more visible. Moreover, the nodes with high degree hold the network cluster together. It is an appropriate measure for the walk-based transfer processes due to the fact that the proportion of times a node is visited is degree dependent (Borgatti, 2005). Nieminen (1974) considered DC as an indicator for determining the interconnectedness of a network member. Landherr et al. (2010) had a different view of such matter. They argued that DC does not sufficiently differentiate the interconnectedness of individual member as it only considers the number of immediate contacts but does not consider their further interconnectedness.

3.1.2 Closeness

Closeness centrality (CC) measures the average distance of a node to all other nodes in the network. Scientifically, CC is denoted as:

$$closeness\big(v\big) = \frac{1}{\sum_{i=1,i\neq v}^{n} d\big(v,i\big)} \tag{1}$$

The equation above sums up all the distances of node v, the $d\big(v,i\big)$ refers to the number of the distance of 1 from the node v to the node i. For comparison purposes, CC must be standardized by dividing the maximum possible value of $\frac{1}{\big(n-1\big)}$. The rule is, the node with the highest CC value is on average closest to all other nodes (central node). CC is considered as a measure of how long it takes for information to spread from a source node. It can be applied to both parallel and transfer flows, but it is more accurate when applied to processes that flow along the shortest paths. Both CC and BC neglect network communications that occur along reachable and non-geodesic pathways.

3.1.3 Betweenness

Betweenness centrality (BC) is a type of medial centrality that can be regarded as a measure of the importance of a node as a controller of the information which is flowing between the other nodes in the network. BC measures the number of times a node acts as a bridge along the shortest path (geodesic) between two other nodes. Scientifically, BC is denoted as:

$$betweenness(v) = \sum_{s\neq t\neq v} \frac{d_{s,t}\big(v\big)}{d_{s,t}} \tag{2}$$

The equation above sums up all the all the ratios of (v). (s, t, v) are distinct nodes in the graph, $d_{s,t}(v)$ refers to the number of geodesic paths from the node (s) to (t) that via the node v, and $d_{s,t}$ refers to the total number of geodesic paths from the node (s) to (t). For comparison purposes, BC must be standardized by dividing the number of pairs of nodes that does not include v, that is, $(\frac{(n-1)*(n-2)}{2})$. The rule is, a node with the highest BC is considered the most central node.

BC is considered as a measure of volume of traffic moving from each node to every other node that would pass through a given node Borgatti (2005). It is suitable for the transfer types of the process due to its indivisible path that transfers from one node to another along the shortest (geodesic) paths. However, a major downfall with BC and CC is that they do not take into account of flows that are non-geodesic. Furthermore, another problem with BC is in its calculation. De Meo et al. (2012) posited that computing the exact value of BC for each node-edge is almost unfeasible as the graph size grows. Decomposition of the network was considered in avoiding the computational problem, though it will compromise the network infrastructure subtlety.

3.1.4 Other Variations of Betweenness

Due to the limitations of Freemans betweenness, numerous researchers have been building on top of Freemans betweenness over the years and derived various betweenness-like metrics.

3.1.4.1 Flow Betweenness

Flow betweenness (Freeman et al., 1991) is based on the idea of maximum flow a node holds. Imagine the edges in the network are pipes that liquid flows through, the maximum possible flow at a node (v) from source (s) to target (t) is calculated to determine the importance of that node (v). Needless to say that there are numerous paths in which the liquid could take, the ratio of the maximum possible paths from (s) to (t) that via (v) is the flow betweenness of node (v). One major advantage of flow betweenness is that it takes into account all types of paths (geodesic and non-geodesic). However, just like Freemans betweenness, flow betweenness also seems to be an unrealistic definition of many practical situations (Newman, 2005). The properties of flow betweenness may be valuable in formulating a plausible measure for networks with serial and/or parallel replications that are not geodesic (paths, trails, and walks). The notion of liquid flowing in pipes and forking into multiple pipes is similar to replication of the virus. Furthermore, the notion of liquid that flows away from the source due to flow pressure from source is similar to geodesic and paths flows trajectories. In addition to that, just like liquid, networks like virus do not spread via an ideal path.

3.1.4.2 Random-Walk Betweenness

Freemans betweenness measure has the condition that information flowing in the network follows the shortest (or geodesic) paths. However, Estrada et al. (2009) argued that most of the information is likely to flow along non-shortest paths. In this case, the information flow may follow a walk, a trail or a path. In the attempt of overcoming this flaw, Freeman et al. (1991) has introduced flow betweenness centrality which supports both geodesic and non-geodesic paths. However, Newmans finding showed that the flow betweenness was counterintuitive, which led him to his own alternative concept of random-walk betweenness centrality.

To understand what random-walk betweenness is, it is necessary to understand what the definition of random-walk is. Suppose a message that originates at source (s), and it is intended to reach target (t), having no idea where (t) is, simply get passed around until it reaches (t). This means, at each step, the message moves from the current position in the network to one of its adjacent nodes, chosen uniformly at random. Therefore, random-walk betweenness (Noh and Rieger, 2004) is similar to flow betweenness, it measures the frequency of node (v) in between a source node (s) and a target node(t).

The major distinction is that random-walk betweenness is the frequency of node (v) occurring in a random-walk from (s) to (s), whereas flow betweenness comparing the maximum possible paths with paths contain node (v). This distinction provides crucial importance to networks that follow walks trajectory. In the liquid flowing in pipes analogy, the liquid flows in the directions away from the source, thus it is illogical to consider the flow current is in the direction of the source. Unlike trails and paths, walks may contain repeated nodes and edges. When dealing with randomness-walk, it is unlikely that a target node will ever be visited as the size of the network increases. In this case, random-walk betweenness and the betweenness of Freeman (1978) are at the opposite ends of the spectrum (Newman, 2005), where one end represents information has no idea where it is heading and the other end represents information know exactly where it is going. There is one minor yet crucial remark regarding random-walk betweenness. That is, to resolve the biased high betweenness score caused by traversing the same nodes back and forth multiple times, those repeated nodes will be canceled out. Estrada et al. (2009) conducted a simulation on the Strozzi family and correlation analysis on the variety of betweenness measure results. The analysis showed that the lower correlation are observed for the flow betweenness and the shortest path betweenness, whereas the random-walk betweenness exhibited higher correlation (Estrada et al., 2009).

3.1.5 Eigenvector Centrality

DC, CC, and BC measures were derived from the principle of the node to node relationship. Unlike the rest, Eigenvector Centrality (EC) is considered as a radial centrality measure that measures the prestige of a node in relation to its neighbors. According to Borgatti et al. (2009), if two nodes have ties to the

same others, they face the same environmental forces and are likely to adapt by becoming increasingly similar. This means that a node is also considered central if it is directly connected to other well-connected nodes. Bonacich (2007) stated that the node that has a high eigenvector score is the one that is adjacent to nodes that are themselves, high scorers. The EC is denoted as:

$$C_E(v) \propto \sum_{i \epsilon N(v)} A(v,i) C_E(i) \tag{3}$$

The equation above sums up all the eigenvector centrality values of neighbors of node v. $C_E(v)$ refers to the Eigenvector centrality of node v, $i \epsilon N(v)$ refers to node i being one of node v neighbors, and $A(v,i)C_E(i)$ refers to the eigenvector centrality value of node i.

The mechanism of EC affects all nodes and their neighbors simultaneously, as in a parallel duplication process (Borgatti, 2005). EC does have its limitations, EC neglects multiple shared paths between points that CC and BC do not.

3.1.6 Comparison and Usage

Radial measures are concerned with the position of a node in the network. The direct relationships of a node summarize a node's connectedness with the rest of the network (Borgatti and Everett, 2006). However, the radial centralities are not suitable in network with multiple dense clusters. Radial centralities make sense in the network which has, at most, one center. Relating to the multiple cluster (two or more centers) networks, medial measures come to play. Unlike radial measures, medial centrality assigns high centrality scores to nodes that serve as a bridge between subgroups (clusters) (Borgatti and Everett, 2006).

When choosing between radial (e.g. DC, EC) and medial (e.g. BC, CC) centrality measures, one must consider the conception and cohesion of the network. For example, if one is studying the risk of receiving information flow through the network, then logic dictates that the length measure (medial) would be more suitable. However,if one is studying the package delivery certainty, then the volume measure would be a more obvious choice (Borgatti, 1995). This centrality distinction helps to provide insight into the appropriate centrality measure given the network properties (flow, graph, and transfer mechanism). The previously mentioned centrality metrics from various researchers have been assigned to the network classification developed by Borgatti (2005) according to his research. This is shown in Table 2.

3.2 Diffusion

According to Valente (1996), network influence is captured by an exposure or contagion model. The likelihood of each individual of adoption increase as the number of individuals in his or her network increase (high degree) (Iribarren and Moro, 2011). For example, a person with a high degree centrality in his/her network may have a higher chance of being infected with a virus. The exposure of an individual can be measured by social influence processes. The

Table 2: The table describes the appropriate centrality metrics for the different network types according to Borgatti (2005).

	Parallel duplication	Serial duplication	Transfer
Geodesic		Freeman Closeness	Freeman Betweenness Freeman Closeness
Paths	Freeman Degree Freeman Closeness		
Trails	Freeman Degree Freeman Closeness		
Walks	Freeman Degree Freeman Closeness Bonacich Eigenvector		

Table 3: The table describes the different factors that contribute toward social influence. These factors are divided into three classes: relational, positional and central)

Relational	Positional	Central
Direct ties Indirect ties Joint participation	Percent Positive matches: (tie overlap) Euclidean distance: (distance between two nodes) Regular equivalence: (two nodes tie to same 3rd node)	Degree Closeness Betweenness Flow Information Power

social influence processes refer to the number of direct contacts (degree) and position (closeness) of the individual within the network (Burt, 1987). The social influence processes can be modeled with three different classes of network weight matrices (relational, positional, and central) (Table 3). In addition to that, the exposure is also influenced by the social distance and in-degree. For undirected and unweighted networks, the positional and central matrices may be useful in determining the social influence processes. According to Valente (1996), an accelerated diffusion is dependent on the network structure. Diffusion reaches pockets of interconnectivity (within the cluster) and spreads rapidly within these pockets, but slow between network subgroups (between clusters). Based on that remark, a starting node that yields the quickest diffusion may be considered as a central source. With that being said, the centrality of the replication processes can be determined by comparing the time needed to reach full diffusion of all nodes.

3.3 Centrality measures for a set of key players

Borgatti (2006) defines the Key Player Problem Positive (KPP-Pos) as consisting of identifying these k-players that could be used as seeds in diffusing optimally some information on the network. The Key Player Problem Negative

(KPP-Neg) goal consists of identifying those k-players that, if removed, will disrupt or fragment the network. In tackling the Key player problem, Ortiz-Arroyo and Hussain (2008) have developed a method that is based on the entropy concept by Shannon (1948). Entropy is defined as a measure to quantify the amount of information that could be transmitted in a noisy communication channel. The basic idea is to find those nodes that produce the largest changes in the connectivity and/or the centrality entropy when removed from the graph.

$$X\big(v\big) = \frac{deg\big(v_i\big)}{2N} \tag{4}$$

The equation above measures the connectivity of node v, where $deg\big(v_i\big)$ refers to the number of edges attached to node V_i, and N refers to the total number of nodes in the network.

$$Y\big(v\big) = \frac{paths\big(v_i\big)}{paths\big(v_1, v_2, v_3 ... v_m\big)} \tag{5}$$

The equation above measures the centrality probability of node v, where $paths\big(v_i\big)$ refers to the number of paths from node v_i to all other nodes, and $paths\big(v_1, v_2, v_3 ... v_m\big)$ refers to all the paths exist in the networks.

These two equations are used to obtain the different entropy measures (H_{co} and H_{ce}), by modifying the entropy equation of Shannon (1948). Connectivity entropy measure (H_{co}):

$$H_{co}\big(G\big) = -\sum_{i=0}^{n} X\big(v_i\big) * Log_2 X\big(v_i\big) \tag{6}$$

Centrality entropy measure (H_{ce}):

$$H_{ce}\big(G\big) = -\sum_{i=0}^{n} Y\big(v_i\big) * Log_2 Y\big(v_i\big) \tag{7}$$

These two derived entropy equations ($H_{co}\big(G\big)$ and $H_{ce}\big(G\big)$) help to determine the impact of a node in the graph before and after the removal of that node.

Connectivity entropy measure ($H_{co}\big(G\big)$) determines the degree of connectedness of a node within the network. In a fully connected graph, the removal of a node will result in a decrease in the total entropy of the graph, in the same proportion as if any other node is removed. However, in a less connected graph, the removal of a node with many incident edges will have a greater impact on the total connectivity entropy of the system, compared to a node with a smaller connectivity degree.

Centrality entropy measure ($H_{ce}\big(G\big)$) determines the degree of centrality of a node within the network. The node that drastically reduces the number of viable paths to reach other nodes when removed will have a greater impact on the total centrality entropy of a graph.

The simulation model based on the Algorithm 1 of Ortiz-Arroyo and Hussain (2008) has proven to be a simple yet effective method. Furthermore, It also revealed that the entropy method also identifies redundant nodes in the network.

Algorithm 1: The algorithm describes the machine code of calculating the importance of each node and returns a set of nodes that exceed certain significance level(d_1, d_2).

```
Calculate initial total entropy Hco0(G) and Hce0 (G);
for all nodes in graph G do:
    Remove node vi, creating a modified graph G;
    Recalculate Hcoi(G) and Hcei (G), store these results;
    Restore original graph;
end for;
To solve the KPP-Pos problem select those nodes where Hco0 - Hcoi >d1;
To solve the KPP-Neg problem select those nodes where Hce0 - Hcei >d2;
```

4 Network

4.1 Information Flow

The centrality measures determine the central players in the network but do not provide much insight regarding the information flow of the network. In the research of Granovetter (1973), the strength of a relationship (tie) is dependent on four components: the frequency of contact, the history of the relationship, the contact duration, and the number of transactions. Granovetter (1973) observed that as the frequency of interactions between two people increases, their sentiment of the relationship becomes stronger. The history of the relationship also determines which tie is selected. In a given context environment, a contact with which a person has interacted over a longer period of time may be more important than a newly formed contact. The contact duration and the number of transactions refer to the interaction recency. Granovetter (1973) believes that recency may influence the intensity of the relationship between two people. These indicators can be used for information flow to determine which contact has the strongest social relationship from origin to destination. Tie strength theory provides insight into how information flows in a socio-network. Daly and Haahr (2009) hypothesized that tie strength is a good measure of whether a tie will be chosen or not. Since strong ties are typically more readily available and as result, more frequent interactions may occur. However, unlike strong ties, connecting to a weak tie has its own benefit. The connection to a weak tie enables the access of that circle (subgroup with their own strong ties).

The network analysis focuses on the structural properties of the system from a topological point of view. That is, considering mainly on the connectivity properties of the network and not the actual flow through it (Zio and Piccinelli,

2010). There are three drawbacks associated with the measure of network performance that is based on (Zio and Piccinelli, 2010):

1. Binary links (link or no link) in the network, which neglects the strength of the connection. This has been pointed out as a limitation in both social and engineering networks. Zio and Piccinelli (2010) believe that the strength of the interpersonal relationship is relevant in path selection decision.

2. The simplified representation of the social network assumes information flow along the shortest path (geodesic). Based on the context of the information flow, it is very much possible that information will take a more circuitous route.

3. The simplified representation of the network also neglects the possibility of failure in the interconnection of linked nodes. This could be an associate that is no longer available and it is not been informed to anyone. However, this is particularly relevant for the engineering network that is made of fallible hardware and software.

4.2 Network Structure Properties

Different networks come in different shapes and sizes, each has its unique features that impacts the information flow. Guzman et al. (2014) have captured three aspects of network structural properties: size, scale-free parameter and clustering coefficient.

4.2.1 Size

Size refers to the number of nodes within a network; the size of a network is context specific. The level of detail determines what actors should be included in the network. The size of the network dictates what the centrality measure is appropriate. For instance, as size increases, it is unwise to use Freemans betweenness due to the scaling of the computational power required.

4.2.2 Scale-free Parameter

Scale-free networks are centralized structure networks with very few dominant nodes. The degree of nodes in such a network follows a power law distribution ($\frac{1}{x^2}$). A power law distribution does not have a peak, instead, it is described by a continuously decreasing function. That is, there are a few core nodes with many connections and a trailing tail of nodes with a very few connections. The World Wide Web, semantic maps, and academic citations are typical examples of a scale-free network.

4.2.3 Clustering coefficient

Clustering refers to the grouping of nodes in a network. In a realistic environment, a social network is not centralized. In fact, they are formed by many subgroups each with a significant actor in the middle that is bridged with other subgroups in a larger network. As for this reason, various betweenness measures are more suitable in determining the central nodes in this type of network. However, this is not the case for all types of network. For instance, Freemans closeness centrality is more suitable for the networks that have one cohesive cluster.

4.3 Network Structure

According to Daly and Haahr (2009), Freemans centrality metrics are based on the analysis of a complete and bounded network, which is sometimes referred to as a socio-centric network. However, these metrics become difficult to evaluate when the network size scales up as they require a complete knowledge of the network topology. Due to this reason, the Ego-Network were created. The Ego-Network analysis can be performed locally without the need of complete network knowledge. Alongside Ego-network, other types of network were also introduced. These networks differ in the structural properties that consequently affect information flow drastically.

4.3.1 Ego Network

In an Ego-network (Figure 1) setting by Burt (1987) with equal nodes (ties), different information flow outcomes were discovered. In his discovery, two premises were established:

1. People connected by strong ties (by a specific criterion) are considered similar and tend to overlap (e.g. personality, mentality, etc.) (McPherson et al., 2001). Example, the idea generation of two people that are connected by a strong tie are more comparable than the idea generation of two people that are connected by a weak tie.

2. People that are connected by the bridge ties (weak ties) tend to spark novel ideas. The benefit of the bridge ties enables the access of other dense groups (groups with their own strong ties).

Two different structures (left and right) of Figure 1 were analyzed in terms of information exchange efficiency and idea novelty. The structure on the right have actors that are closely connected by the strong ties which induce faster information flow and greater motivation to be of assistance. However, it is more likely to have similar ideas (Premise 1). Granovetter (1973) argued that the power of an actor is further affected by who their "other-ties" are connected so on and so forth. Oppositely, the structure on the left with fewer strong ties and more weak ties (structural holes) but has the capability of better idea generation or being perceived as the source of new ideas due to more non-redundant

Figure 1: Ego Network (Borgatti and Halgin, 2011): The figure consists of two different network structure based on the same nodes (ties) and same tie strengths. However, due to the difference in node-to-node structure, different information flow effects are produced.

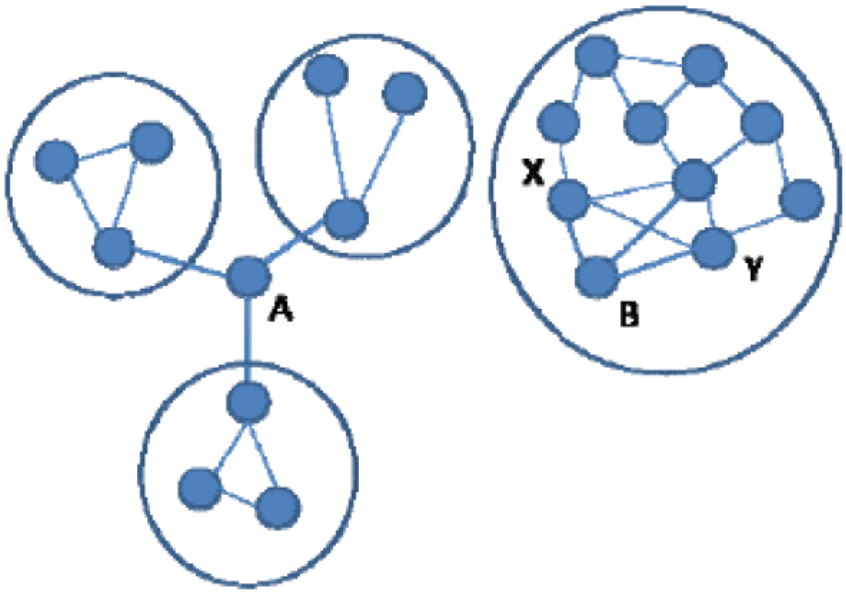
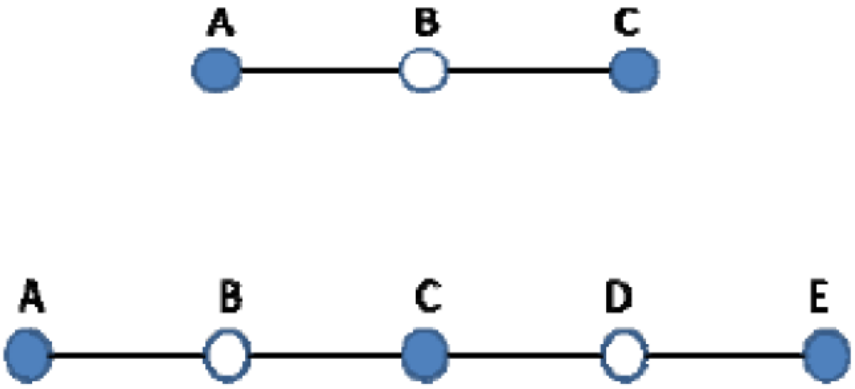


Figure 2: Exchange Network (Borgatti and Halgin, 2011): The figure displays two versions of the exchange network. The different between the two structures has resulted in the power and centrality difference.

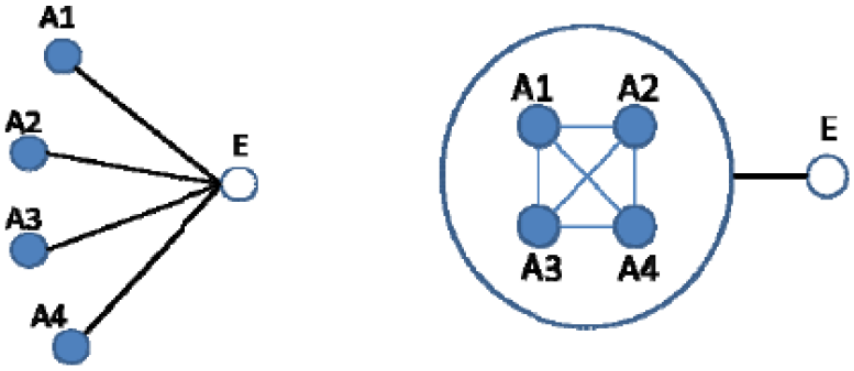


information received at any given time than the structure on the left. Weak ties can be very useful in some settings, they provide bridge connections between different network clusters, allowing subgroups to share access and capabilities (Enriquez, 2010). Granovetter (1973) emphasized that the weak ties can lead to information dissemination between groups. He stated that information can reach a large number of people and traverse a greater social distance when passes through weak ties rather than strong ties. In addition to that, weak ties also provide people with access to information and resources beyond those available in their own social cluster.

4.3.2 Exchange Network

Cook and Emerson (1978) pioneered the experimental study of power in exchange networks (Figure 2). They argued that the network structure and location within that structure are fundamental for the network power and centrality. Consider an exchange network structure (Top network) with three nodes, both centrality and power phenomena suggest that (B) is the node, due to (B) is positioned in the middle which has the possibility of making a better trade out of the two (A and C). However in the network structure (Bottom network) setting with five nodes, (C) is the most central node but node (B) and (D) have the most power. This is because, (B) can choose who to make deal with (A or C), similar case for (D) (choice between C and E), while (C) have to deal with (B) and (D). Since (B and C) can choose (A and E), leaving (C) with no trade partner. As Cook and Emerson (1978) theorized that in the centrality phenomena, being connected to well-connected other implies greater centrality. In power phenomena it can be the other way around, being connected to weak others makes one powerful and being connected to powerful others makes one

Figure 3: Unionization Network (Borgatti and Halgin, 2011): The figure displays how similar nodes in a network can be interpreted as on unit.



weak (Willer, 1992).

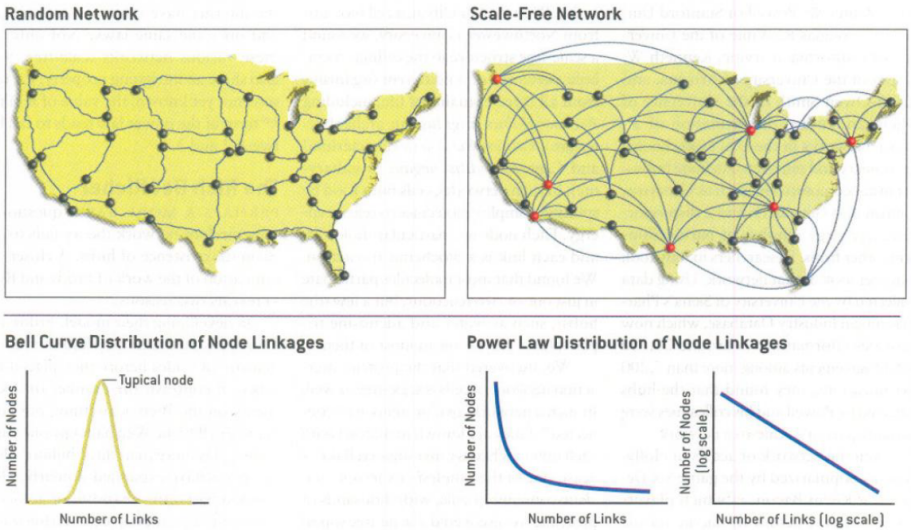
4.3.3 Unionization Network

Consider a network setting (Figure 3) in which multiple of similar actors (A1, A2, A3, and A4) in communication with one other actor (E). Like in a factory with multiple low-level workers, who report to the same supervisor. Reporting to the same supervisor individually can be time-consuming and difficult. The principle of unionization is that if multiple actors have similar goals, then they can be treated as one unit. The key point in a unionized network is that nodes with the same interests and capabilities working together can accomplish more than they could alone. This is the so-called "network organization", in which a set of autonomous unit coordinate closely as if comprising a single, superordinate entity (Powell, 2003). The bonding function between the actors is called bond model which is the analog of the flow function in the flow model.

4.4 Random Network vs. Scale-free Network

The understanding of network structure properties helps to provide insights to network cohesion and centralization. One of the three structural properties briefly described in the previous chapter is the scale-free parameter. This property describes how the nodes within the network are connected. According to Barabási and Bonabeau (2003), a complex network can be classified as random or scale-free (Figure 4). The characteristics of a random network are that despite the random placement of links, most nodes have approximately the same number of links. The links of nodes in the network follow a Poisson distribution with a bell shape. So, it is extremely rare to find nodes with higher and and lower total links. A real-life example of a random network is the US highway system. The US highway network consists of intersections with a similar

Figure 4: Random vs. Scale-free Network (Barabási and Bonabeau, 2003): The simplified highway intersections of US is shown on the left. The simplified US airline on the right. They correspond to the specific type of network shown in the figure.



number of connections to neighbor states. In this example, it is indeed rare to find interactions with fewer and/or many connections. As for the scale-free network, the links of nodes in the network follow the power-law distribution, which is the probability of a node connecting to k other nodes is proportional to $\frac{1}{k^n}$. A scale-free network has an abundant of nodes with fewer links and few nodes with high links. A node with many connections in a scale-free network is called a hub. Networks with central hubs are robust against accidental failure, but vulnerable to coordinated attack (fragility) (Grassi, 2010). Understanding such characteristic is vital to the development of centrality metrics for parallel and serial replication. Few life examples of a scale-free network are the Swedish sexual relationship, while many people have few sexual partners, a few had hundreds in their lifetime. Published scientific papers with citations, Internet systems with routers and the World Wide Web with web pages are examples of the power-law network.

Erdos and Renyi (Barabási and Bonabeau, 2003) revealed why random-network theory fails in a social network environment and two reasons were concluded. Firstly, growth, the number of actors in networks is growing. This means that the older nodes in the network are more likely to establish more connections than the newly joined nodes. Lastly, the preferential attachment describes that a new node is more likely to establish a connection with the node that has more ties than nodes with fewer ties. This can be seen in the web search, people can choose from billion of the search result, but instead, only the top web pages are selected. Likewise, the most cited articles in the scientific

literature stimulate even more researchers to read and cite them, a phenomenon that noted sociologist Robert K. Merton called the Matthew effect, after a passage in the New Testament: "For unto every one that hath shall be given, and he shall have abundance." These two reason help to explain the existence of hubs and help in the construction of networks.

4.5 Cohesion and Clustering

Since all the networks are not identically shaped. The question is, at what point a network is considered as a cohesive group or a multiple cohesive subgroups (clustering) network? One of the first intuitions in SNA concerns the tendency of human beings to form cohesive subgroups. The main characteristic is that the nodes in a subgroup are similar in attributes. People adjust their behavior and attitudes, opinions, and beliefs to the behavior of other members of the social system in which they participate. Over time, people decide on which ties to establish, maintain, or terminate (social selection model). The key player set theory (Ortiz-Arroyo and Hussain, 2008) suggests that some networks have multiple vital players. These key players may hold some strong positions in the network. In such case, how would one determine the cohesive subgroups in a network? The clustering coefficient (CCoe) of the network determines how cohesive the nodes in the network are. The measure of CCoe is based on the triplet of nodes. A triplet is three nodes that are connected to each other forming a triangle. The global clustering coefficient is the average of triplet ratios of all the nodes. The end result is a value from a scale between 0 and 1. The networks with low CCoe value are structured like a star shape (hub - like), whereas the networks with high CCoe value are structured like a clique (close-knit group).

In the history of SNA, there are four ways of identifying cohesive subgroups at the level of overall network structure have been formulated (Crowston et al., 2010). The first and strictest approach, a cohesive subgroup is defined as a set of nodes in which all the nodes are adjacent, that is, directly linked to one another. The second approach is based on the notion of reachability and closeness of members within a subgroup. The shorter the geodesics between them, the closer the nodes are in a graph, the more cohesive the network structure subgroup is. However, this would not perform so well in a less dense network. The third approach focuses on the minimum number of ties among subgroup members. The number of neighbors within that sub-graph is called k-cores and a maximal subgraph with respect to the maximum number of nodes in the subgraph that are not adjacent are known as k-plex (Seidman, 1983). And the final approach is based on the relative frequency of ties among subgroup members in comparison to non-members. The cohesive subgroups are relatively denser than the sections outside of the subgroups. Ultimately, the cohesion of the network determines which centrality measure (closeness-like, betweenness-like or other) is most appropriate.

5 Development of Centrality Metrics

The in-depth knowledge of the network structures and their unique characteristics along with a variety of centrality measures from the previous chapters provided a solid foundation in the formulation of suitable centrality approaches for other network types. With that being said, this chapter is aimed at developing appropriate methods according to the network examples of Borgatti (2005) (Figure 1).

5.1 Centrality

Centrality is an abstract property of a nodes position in a network (Enriquez, 2010). Centrality corresponds to the overall network property and it is defined as the variation of centrality scores of all nodes. The variation of centrality scores describes where the central nodes and/or peripheries are. The star and ring networks are considered respectively the most and the least centralized networks (Borgatti et al., 2009). The fundamental metrics (Degree, Closeness, Betweenness, Eigenvector Centrality) are not applicable to all types of network flow. Degree and Eigenvector metrics focus on the immediate relationships. Both Betweenness and Closeness are based on the idea of information flow along the shortest path (Freeman, 2004). There are other networks in which information does not flow along the geodesic path (Stephenson and Zelen, 1989). News, rumors or messages do not know the ideal route to take. They simply moves from one place to another; more likely it wanders around more randomly (Newman, 2005). The random-walk betweenness approach of Newman (2005) and flow betweenness approach of Freeman et al. (1991) are promising solutions in addressing such problem. Despite all that, there is another concern regarding what centrality metrics are appropriate for the network with different flow mechanisms (parallel, serial and/or transfer). In the example of a viral infection (Path - Serial duplication) that virus spreads around randomly, infecting one person at a time. Determining the centrality of a single node in the network using conventional metrics may not yield useful results. Instead, it may be wise to inspect such phenomenon differently. The work of key player set by Borgatti (2006) mentioned in the earlier chapter determines the centralization of the network by weighing the importance of each node, neglecting the flow mechanism. Furthermore, the key player set may also be used as a validation tool in determining the reliability and validity of the proposed methods.

5.2 Formulation of Methods

According to the typology of flow processes (Figure 1) by Borgatti (2005), many of the flow processes have not been assigned a set of proven centrality measures. In the attempt of continuing the work started by Borgatti (2005), the following subsections describe the workings of the proposed methods correspond to the network examples given by Borgatti (2005). The key assumption of all the following examples is that information is assumed to be immutable. That is, the

state of the information does not vary under any circumstances. For instance, the condition of the used good (example by Borgatti (2005)) that is passing around within the network does not change over time.

5.2.1 Serial and Parallel Replication

In the scenario of serial and parallel duplication, it would take some time to reach full diffusion starting from source S and end at multiple targets Ts. Assuming the information spread takes place in incremental steps, then the number of reached nodes after each intermediate incremental step is an addition to the current step. At some point in time, all the nodes will have a copy of the information that was transmitted from the source S. Based on this premise, the nodes require the least amount of time to reach full diffusion would be considered as central (definition of centrality). This idea is based on Freemans closeness centrality, flow betweenness and key player set (Borgatti et al., 2009; Ortiz-Arroyo and Hussain, 2008). Freemans closeness was discovered to determine the centrality of a geodesic transfer network. Instead of closeness based on geodesic paths, all possible paths should be taken into account (flow betweenness). The main objective is to find a suitable source position (central node), such that the minimal time interval is needed to reach full diffusion (Iribarren and Moro, 2011; Valente, 1996; Burt, 1987).

5.2.1.1 Serial: Paths

In a serial duplication, only one copy of the information per node is replicated per time. For path trajectories, each of the possible adjacent neighbors of a node is a potential receiver of the replication which means that each selection of a receiver leads to a completely different path. Let's use the viral infection example (Figure 1) by Borgatti (2005), a person (node) cannot infect someone who has already been infected (already received a copy of the virus) or someone who he received the virus from. At each time interval, the maximum number of possible virus infections (number of copies) is equal to the number of infected people. This virus spreads until all the people in the network are infected (all the people received a copy of the virus). The proposed approach is a closeness-like measure (Borgatti and Everett, 2006), which assesses the length of the walks that a node is involved in.

Assuming the network (Figure 5) is a virus network (Serial-Paths) that spreads via physical contacts, then average time intervals (TI) required to reach full diffusion with person 1 as the original virus carrier of is $TI = 3.5$. Let's look at this in detail, at $TI = 0$, the maximum possible virus replication is 1, so person 1 infects person 2 or person 3 at random. Assuming person 1 infects person 2 (Table 4: Path 1), then the maximum people can be infected at $TI = 2$ is 2. So, person 1 and 2 infect their neighbors, person 3 and 5, respectively. At $TI = 3$, only person 3 and person 5 can spread the virus to new hosts, since the adjacent neighbors have already been infected (person 1 and person 2). Person 3 and 5 infect person 4 and person 6 respectively and reached full diffusion run

Figure 5: Simple Network Example: The network consists of six nodes. The node to node connection is shown. This network example is used in the following sections. (Source: Stackoverflow Forum)

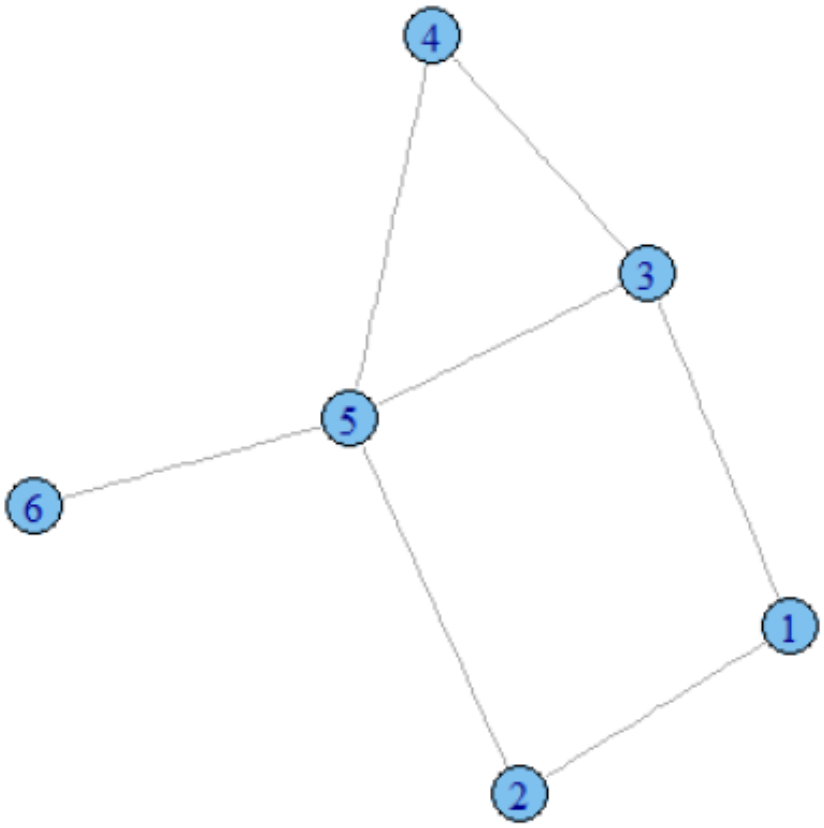


Table 4: The table described the spread of the virus based on the Simple Network Example (Figure 5). The table contains two of many possible diffusions with person 1 as the source. During each time interval, newly infected people are appended.

Time Interval	**Paths 1**	**Paths 2**
0	[1]	[1]
1	[1], [2]	[1], [3]
2	[1], [2], [3, 5]	[1], [3], [2, 5]
3	[1], [2], [3, 5], [4, 6]	[1], [3], [2, 5], [4]
4	<Finished>	[1], [3], [2, 5], [4], [6]

in 3-time intervals. However, if the virus replication had taken a different path (Table 4: Path 2), at $TI = 3$, only person 3 and person 5 could have the possibility to spread the virus. If person 5 infects person 4, then the last person in this small network (person 6) can only be infected by person 5 in the next time interval ($TI = 4$). In such case, the virus diffusion took 4-time intervals. Since there are countless virus diffusion possibilities, the only way to obtain a plausible measure is by taking the average score of many runs. In this brief example, 2 runs (Path1 and Path 2) were mentioned, thus the average centrality value of person 1 is 3.5 ($(3 + 4)/2 = 3.5$). This procedure is applied to all other nodes in the network to determine which nodes are the most central. The workings of the example above are translated into machine code (Algorithm 2).

Algorithm 2: The algorithm represents the machine code that is used to determine the serial-paths centrality value of each node in the network.

```
Iterate over simulation runs:
    Create an empty queue to to store all possible paths combinations;
        Store the first node;
        While queue is not empty:
            Get first path sequence from the queue;
            Iterate over the neighbors of each node in the path sequence:
                Store one neighbor node if not been visited to a temporary list;
            If the temporary list is not empty:
                Create new possible path sequence and add it to queue;
            Else:
                Determine and add length of current path sequence;
Determine average time interval of a node;
```

5.2.1.2 Serial: Trails

Similar to serial paths, a trail trajectory also transmits a copy of the information per time interval. The definition of trails trajectory states that the nodes can

be repeated but the edges cannot. Lets use the example of gossip by Borgatti (2005), a person (A) can tell a rumor to person (B) in a gossip, even though person (B) has already heard it from person (D). But, person (B) cannot tell that very same rumor back to person (D) or person (A). This unique property may impact the course of gossip spread. Instead of picking a person who is not aware of the rumor, picking someone who may have heard of the rumor is entirely possible. This results in more node selection possibilities and thus longer time needed to reach full diffusion.

Table 5: The table describes the spread of a rumor in a gossip based on the Simple Network Example (Figure 5). The table describes two possible rumor diffusions with person 1 as the source. During each time interval, the newly gossip occurred between two people (a and b) are described in form of e(a,b).

Time Interval	**Trails 1**
0	(1)
1	[e(1,2)]
2	[e(1,2)], [e(1,3),e(2-5)]
3	[e(1,2)], [e(1,3),e(2-5)],[e(3,5),e(5,6)]
4	[e(1,2)], [e(1,3),e(2-5)],[e(3,5),e(5,6)],[e(5,4),e(3,4)]
Time Interval	**Trails 2**
0	(1)
1	[e(1,2)]
2	[e(1,2)], [e(1,3),e(2-5)]
3	[e(1,2)], [e(1,3),e(2-5)],[e(3,5),e(5,3)]
4	[e(1,2)], [e(1,3),e(2-5)],[e(3,5),e(5,3)],[e(3,4),e(5,4)]
5	[e(1,2)], [e(1,3),e(2-5)],[e(3,5),e(5,3)],[e(3,4),e(5,4)],[e(5,6)]

Assuming both gossip diffusions (Table 5: Trails 1 and Trails 2) occurred the same prior $TI = 3$, the random gossip partner selection have led to different outcomes. In Trails 1, assuming during $TI = 3$, person 3 and person 5 have selected person 5 and person 6 respectively, then the remaining person (person 4) would have to be picked by either person 5 or 3 in the next time interval ($TI = 4$). In comparison to Trails 2, the gossip partner selection from $TI = 3$ had led to a different diffusion outcome. Due to person 3 and person 5 had selected each other in $TI = 3$, this resulted in an additional turn in completing the rumor diffusion ($TI = 5$). Similar to serial paths, there are countless ways in which how the diffusion could have progressed. Thus,it is a must to take the average score of all the runs. In this particular example, the average centrality score for person 1 is 4.5 ($(4 + 5)/2 = 4.5$). This procedure is applied to all other nodes in the network to determine which nodes are the most central. The workings of the example above are translated into machine code (Algorithm 3).

5.2.1.3 Serial: Geodesic

Algorithm 3: The algorithm represents the machine code that is used to determine the serial-trails centrality value of each node in the network.

```
For every node in th network:
Iterate over simulation runs:
    Create an empty queue to to store all possible paths combinations;
        Store the first node;
        While queue is not empty:
            Get first path sequence from the queue;
            Iterate over the neighbors of each node in the path sequence:
                Form new edge with node and its neighbors;
                Store the neighbor node if newly formed edge is not in temporary list;
            If the temporary list is not empty:
                Create new possible path sequence and add it to queue;
            Else:
                Determine and add length of current path sequence;
    Determine average time interval of a node;
```

Serial geodesic is uniquely different from paths and trails. A real-world example of such network is a mitotic reproduction (Figure 6) thought by Borgatti (2005), where a cell divides into two at each iteration (i.e. 1 becomes 2, 2 become 4, so on and so forth). One would consider this manifestation behavior like a tree structure with 2 children at each level. At the nth level, the number of children is equal to 2^n. Based on the analogy of mitotic reproduction, the total number of cells that can reproduce at a time interval (TI) is the twice of a total number of cells in the previous time interval (TI - 1). Using the network example (Figure 5), each node represents a possible cell with its unique traits, and each edge represents the potential of producing cells with a specific mutation. For example, cell 5 (node 5) has the potential to produce cells 2, 3, 4 and 6. However, cell 1 can only be produced by either cell 2 or 3. In this particular scenario, a geodesic serial replication can be interpreted as the shortest time needed to produce a full range of cells with different traits. This is only possible by possible by maximizing the number of newly produced cells during each time interval. such that the total number of replications is maximized. With that being said, the starting cell that is able to produce a full range of cell with the least amount of time is considered the most important cell (central node).

If the mitotic reproduction can be controlled, then the aim is to maximize the number of distinct cell traits. Referring to the network example (Figure 5), in the mitotic reproduction scenario (Table 6), at $TI = 3$, cell 5 has the potential to produce cell with traits 4 (cell 4) and traits 6 (cell 6), but cell 3 only has the potential to produce cell with traits 4 (cell 4). So, the ideal reproduction would be for cell 5 to produce a cell with traits 6 (cell 6), such that cell 3 can produce a cell with traits 4 (cell 4). This way, cell 5 does not hinder cell 3, which in turn, speeds up the mitotic reproduction process. Obviously, it is not possible to control how the mitotic reproduction progresses, but it is possible to determine

Figure 6: The figure above describes how a cell that follows a mitotic mechanism divides. During each iteration, the number of cells doubles.(Source: www.goldiesroom.org)

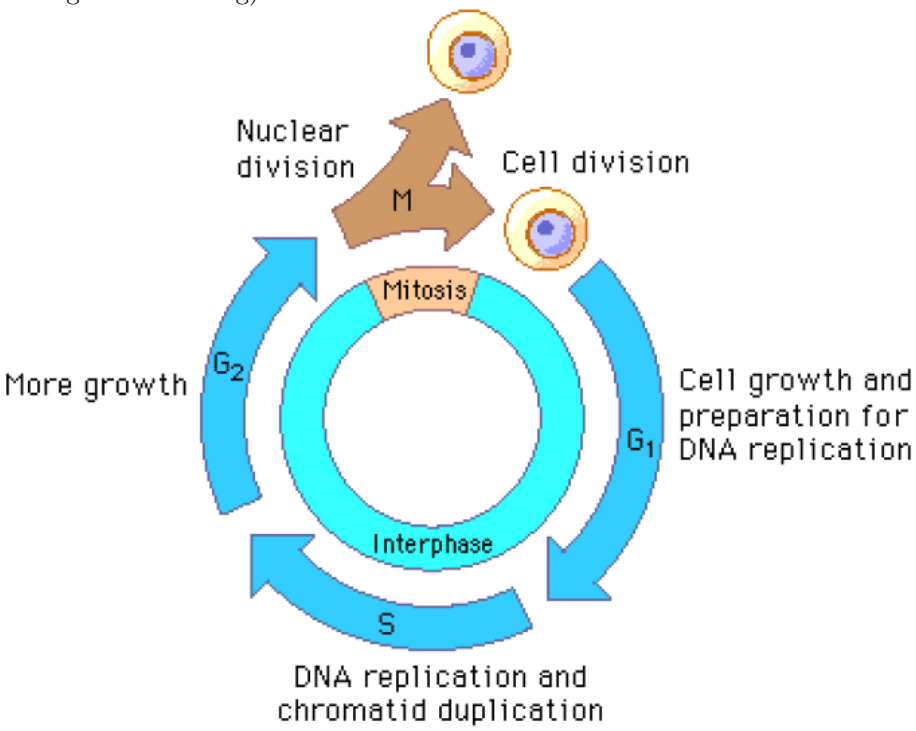


Table 6: The table described the mitotic reproduction based on the network (Figure 5), and with parent cell 1 (node 1). It maximizes the production of the distinct cell during each iteration.

Time Interval	**Geodesic**
0	[1]
1	[1], [2]
2	[1], [2], [3, 5]
3	[1], [2], [3, 5], [4, 6]
4	<Finished>

which parent cells are able to produce a large variety of cell traits in the least amount of time. This is achieved by determining the cell with minimal time interval needed to produce a wide range of distinct cells for each source cell. The above-mentioned approach is translated into machine code (Algorithm 4).

Algorithm 4: The algorithm represents the machine code that is used to determine the serial-geodesic centrality value of each node in the network.

```
For every node in the network:
    Iterate over simulation runs:
        Create an empty queue to store all possible path combinations;
            Store first node;
            While the queue is not empty:
                Get first path sequence from queue;
                Iterate over the neighbors of each node in the path sequence;
                    Store nodes that have not been visited in a temporary list;
                Sort temporary list according to minimal remaining neighbors;
                While temporary list is not empty:
                    Get first node from list, and add to addOn list;
                    Remove node from the temporary list;
                    Loop over every node in the temporary list is account for;
                If addOn list is not empty:
                    Create new path sequence with nodes in addOn list;
                Else:
                    Add up all the path sequences lengths;
            Determine average time interval of a node;
```

5.2.1.4 Parallel: Paths and Trails

In a parallel replication process, multiple replications take place during each time interval. The method of measuring centrality in a parallel replication is similar to the proposed method for the serial replication. That is, measuring the time interval needed to reach full diffusion for each source node. Although

paths and trails of a parallel replication differs in the number of replications of information a node receives, there is no difference in terms of time needed to reach full diffusion. The nodes of trails may have multiple copies of the information. In the example of Email broadcast, Email is sent from source (S) and are received by multiple recipients (Rs) at each time interval. In the next time interval, the recipients who have already gotten that email may receive receive another copy from another sender. Borgatti (2005) proposed Freemans degree and closeness to measure centrality for paths, trails, and walks. The time step approach similar to Freemans closeness centrality, which determines the network centrality by finding the node that is closest to all other nodes.

Table 7: The table describes the email broadcast based on the Simple Network Example (Figure 5). The table describes 2 possible email broadcasts with different points of origin. During each time interval, the new recipients appended.

Time Interval	**Paths / Trails 1**	**Paths /Trails 2**
0	[5]	[1]
1	[5], [2, 3, 4, 6]	[1], [2, 3]
2	[5], [2, 3, 4, 6], [1]	[1], [2, 3], [4, 5]
3	<Finished>	[1], [2, 3], [4, 5], [6]

The two parallel replication processes with different source nodes yielded different results. The parallel replication example (Table 7) showed that the broadcast with source sender 5 took 2-time intervals ($TI = 2$), while the broadcast with source sender 1 needed 3 ($TI = 3$). Since all the nodes simultaneously received a copy of the replication, there is no need to conduct multiple runs. The above-mentioned approach is translated into machine code (Algorithm 5).

Algorithm 5: The algorithm represents the machine code that is used to determine the parallel - paths and/or trails centrality value of each node in the network.

```
For every node in the network:
    Create an empty queue to store all possible path combinations;
      Store first node;
      While the queue is not empty:
        Get first path sequence from queue;
        Iterate over the neighbors of each node in the path sequence;
          Store the nodes that have not been visited in a temporary list;
        While temporary list is not empty:
          Create new path sequence with the nodes in temporary list;
        Else:
          Add up all the path sequences lengths;
      Determine average time interval of a node;
```

5.2.2 Transfer

As for the transfer mechanism, measuring centrality for geodesic, paths, trails, and walks would not work with the approach designed for serial and/or parallel replication. Because the transfer mechanism states that the information is passed onto the adjacent node without leaving a copy. Furthermore, the concept of diffusion does not apply in the networks with transfer mechanism. One possible way to determine centrality is by looking at how much each node has contributed to the transfer flow. Of course, there are four types of transfer mechanisms: geodesic, trails, paths, and walks. Each of them is uniquely different and thus a custom-made approach for each of them is a must. The centrality metrics for the geodesic types already existed. Freemans betweenness and closeness centrality are specifically designed for geodesic flows. With that being said, the following proposed approaches are aimed at other types of transfer process.

5.2.2.1 Transfer: Paths

Freemans closeness and betweenness are designed to measure geodesic (i.e. package delivery). With package delivery example, the assumption is that the shortest route is known and chosen, while paths (i.e. mooch) takes a random path. Based on that premise, an effective approach in measuring centrality is by determining the involvement of the nodes. The proposed approach is a betweenness-like measure (Borgatti and Everett, 2006), which measures the number of walks that passes through a given node. In a transfer paths flow, with starting node (S) and target node (T), the set of nodes that have been traversed would be counted as nodes in between (S) and (T). Since there are multiple nodes in the network, picking a fixed starting point and ending point would produce bias results. Therefore, every single combination of size 2 nodes (start and target) must undergo the same process. The centrality measure of a node (I) is equal to the number of different paths that node (I) is involved in. Since a node is chosen at random during each traverse and not possible for a node to be visited twice, it is unlikely that all the possible information flow combinations can be achieved. Therefore, the in-between nodes of the incomplete paths (unable to each target node from start) are excluded. At the end of all information flow combinations, the centrality measure node (I) is the ratio of node occurrence and flow combination size. The node that has the highest proposed betweenness-like value is considered as the most central node.

The mooch example by Borgatti (2005) in shown in Figure 8. In the flow combination $[1, 2]$, person (1) will not ask person (3) for favor ever again once person (1) has mooch off person (3). Thus, person (1) may attempt to mooch off one of person (3)'s closest contacts. The mooch continues until person (1) has exhausted all the possible options. The proposed method should take into account of all the possible mooch flow combinations. Since the selection of mooch partner is at random, the results of some flow combinations may be excluded. This can be seen in flow combination 2 and 3. Flow combination

Table 8: The table describes the mooch between people based on the Simple Network Example (Figure 5). The flow combination column represents the starting and ending person of the mooch event. The transfer paths column shows all the intermediate people that have been asked for favors. The between nodes column displays all people who were in between the starting person and ending person.

Flow Combination	**Transfer Paths**	**Between Nodes**
[1, 2]	[1], [3], [5], [2]	[3], [5]
[1, 3]	[1], [3]	-
[1, 4]	[1], [2], [5], [6]	-
...		

2 $[1, 3]$ is rejected due to no in-between node, and flow combination 3 $[1, 4]$ is rejected due to the requirement of ending at node 4 is not met. So, based on this particular example, the transfer paths centrality measure for both person 3 and person 5 is $\frac{1}{n}$ (1 occurrence in n different flow combinations). The above-mentioned approach is translated into machine code (Algorithm 6).

Algorithm 6: The algorithm represents the machine code that is used to determine the transfer - paths centrality value of each node in the network.

```
Get all (start, end) tuples in a list;
    Iterate over simulations runs:
    Iterate over all (start, end) tuples:
        Create an empty list to store nodes;
        Add the start node in the list;
        While:
            Get last node from the list;
            Pick a neighbor that has not been visited randomly;
            If last node is end node:
                Terminate;
        If last of the list is end node:
            Count each in-between node;
    Iterate over all the nodes in the network:
        Determine average result of a node;
```

5.2.2.2 Transfer: Trails

Similar to transfer paths, determining the centrality of a node in trails also adopt the betweenness-like measure (Borgatti and Everett, 2006). Just like paths, node selection in trails (passing of a used good) is also at random, as long as the selected node forms a new edge. The proposed approach also analyze the node involvement but focusing on the edges instead. Since a node can be

repeated, reaching the target node from the starting node is more feasible. On the other hand, the repeated nodes property also increases the node selection pool and thus, it is highly likely that the size of in-between nodes of trails is much larger than the size of in-between nodes of paths. The centrality measure of a node is the ratio of the node occurrence and the flow combinations.

Table 9: The table describes the transfer of a used good between people based on the Simple Network Example (Figure 5). The flow combination column represents the starting person and the ending person of the used good transfer. The transfer trails column shows all the transfers between two people in the form of $e(a, b)$. The between nodes column displays all people who were in-between the starting person and the ending person.

Flow Combination	**Transfer Trails**	**Between Nodes**
[1, 6]	[e(1,2), e(2,5), e(5,4), e(4,3), e(3,5), e(5,6)]	[2, 5, 4,3, 5]
...		

Consider the used goods example by Borgatti (2005). The passing of the used good (Table 9) strictly showed that no edges can be repeated. Since once you have passed the used good onto another person, it is unlikely that you want it back. Though, you may accept the used good if it was originating from another person. The above example (Table 9) showed that person 5 is has accepted the same used good twice, but by from two different people (person 2 and person 3), thus it is completely acceptable. The occurrence of each in-between person is stored. In this case, the occurrence of person 5 is counted twice. Once all the different flow combinations have been processed, the average occurrence is the total occurrence (O_t) of each person divided by the total number of flow combinations ($\frac{O_t}{n}$). The above-mentioned approach is translated into machine code (Algorithm 7).

5.2.2.3 Transfer: Walks

Unlike geodesic, paths and trails, walks (i.e. money exchange) has no traverse restrictions which means that a person can exchange money with anyone as long as that person it is reachable. It is also possible that the money exchange happens between two people multiple times. Because of that, all the centrality metrics of Freeman (1978) and proposed metrics in the previous sections are not suited for such case. Instead, the random-walk betweenness by Newman (2005) is used. Newmans random-walk betweenness (2005) is a concept of betweenness-like (Borgatti and Everett, 2006), which is similar to the approach of paths and trails but with one significant difference. In walks, it is highly possible for a node to get high betweenness score simply random-walk 'back and forth' between two nodes. In such case, these two nodes will get ridiculously high betweenness score and not actually go anywhere. Such phenomena do not exist in any other transfers processes. Newmans approach of addressing such issue is by canceling out the going and coming edges, so that nodes traverse back and

Algorithm 7: The algorithm represents the machine code that is used to determine the transfer - trails centrality value of each node in the network.

```
Get all (start, end) tuples in a list;
    Iterate over simulations runs:
    Iterate over all (start, end) tuples:
        Create an empty list to store nodes;
        Add the start node in the list;
        While:
            Get last node from the list;
            Create new edge with last node and one random neighbor node;
            Add node, if new edge not yet visited;
            If last node is end node:
                Terminate;
        If last of the list is end node:
            Count each in-between node;
    Iterate over all the nodes in the network:
        Determine average result of a node;
```

forth are neglected.

Table 10: The table describes the money exchange between people based on the Simple Network Example (Figure 5). The flow combination column represents the starting and ending person of the money exchange. The walks sequence before shows all the intermediate people involved in the money exchange. The walks sequence after removes any 'back and forth exchanges' between two people.

Case	**Flow Combination**	**Walks Sequence Before**	**Walks Sequence After**
1	[1, 6]	[1, 2, 5, 4, 5, 6]	[1, 2, 5, 6]
2	[1, 6]	[1, 2, 5, 4, 3, 5, 6]	[1, 2, 5, 4, 3, 5, 6]
...			

The money exchange example (Table 10) describes two money exchange scenarios from person 1 to person 6. Case 1, every intermediate exchange partner is shown in Walks Sequence Before. Person 4 and person 5 in that sequence are excluded as it is a type of 'back and forth' walk (the same edge between person 4 and person 5 occurred directly after each other), that must be corrected. The corrected result is shown in "Walks Sequence After". Based on that, person 2 and 5 are the in-between nodes in this particular money exchange example. However, Case 2 has somewhat similar (from the edge(5,4) to the edge(4,3), then to the edge(3,5)), such example does not count as a type of 'back and forth' walk, and thus no need to apply the edge cancellation procedure. Newman (2005) strictly stated that the cancellation only applies to the nodes of the same edge. This means that node 2, 5, 4, 3, 5 are all accepted as in be-

tween node. Once all the flow combinations have been processed, the node with the highest ratio is considered as the most central node. The above-mentioned approach is translated into machine code (Algorithm 8).

Algorithm 8: The algorithm represents the machine code that is used to determine the transfer - walks centrality value of each node in the network.

```
Get all (start, end) tuples in a list;
    Iterate over simulations runs:
    Iterate over all (start, end) tuples:
        Create an empty list to store nodes;
        Add the start node in the list;
        While:
            Get last node from the list;
            Randomly pick a neighbor node of the last node;
            If picked node is the same as second last node:
                Remove last two nodes
            Else:
                Add node to list;
            If last node is end node:
                Terminate;
        If last of the list is end node:
            Count each in-between node;
    Iterate over all the nodes in the network:
        Determine average result of a node;
```

6 Realization

In order to determine the validity and reliability of the proposed methods for measuring different types of network process, a stable simulation environment is needed. The simulation environment includes the basis of the network with dynamic network property input (size, power-coefficient, cluster, etc), and the different measures that were mentioned in the previous chapter. The construction of the simulation is based on Python programming language on Spyder IDE with few imported libraries which will be mentioned later in this chapter. Python is chosen due to the capability of processing large data set, and the marvelous built-in functions, all of which may contribute greatly to this research.

6.1 Construction of Network

Before the proposed measures can be tested, a proper network foundation must be constructed. The network is constructed using user-defined class Node with

user-defined attributes (Table 11). Object-oriented programming (OO) is selected over the adjacency matrix, simply because OO is more flexible with the future modifications and scaling. The different variables within the Node class store the centrality measures and other relevant data. Once the class is constructed, the next step is to create nodes and forming edges that comply with different network types (random or scale-free) and different network properties (size, power-parameter, clustering coefficient).

Table 11: The tables describes all the relevant attributes of each node object. The attributes include a set of neighboring nodes, variables to store different centrality measures and node identifier.

Class: Node	
Variable	**Data Type**
id	string
links	Integer
neighbors	list
degree	float
closeness	float
betweenness	float
eigenvector	float
Transfer_Paths	float
Transfer_Trails	float
Transfer_Walks	float
closeness_Serial_Geodesic	float
closeness_Serial_Paths	float
closeness_Serial_Trails	float
closeness_Parallel_Paths	float

6.1.1 Creation of Nodes and Formation of Edges

Barabási and Bonabeau (2003) explained the difference between random network and scale-free network. The machine code (Algorithm 9) and (Algorithm 10) describe how the node objects are created and formed. The distribution of the nodal links in a random network can be characterized by a bell curve, meaning that most of the nodes will have links equal to the mean and few with on the either opposite sides of the bell curve. On the other hand, the distribution of the nodal links in a free-scale network can be characterized by the power-law, where there are few nodes with high linkages but the majority of the nodes have low linkages. Once the number of expected links of the nodes is defined, they are then instantiated one by one and forming one connection with the existing nodes at each iteration, such that all the nodes are reachable within the network. Given the specific type of network, the formation of edges differs greatly. In a random network, the remaining expected linkages are randomly selected from the rest of the nodes in the network. The selection is at random,

the chosen node must have free links and it is not linked with the node. With scale-free, the selection of node is not random, instead, it follows the concept of preferential selection or 'The Rich gets Richer' analogy. The preferential selection states that a node is more likely to form a connection with nodes that have a higher connection. This phenomenon is why Barabási and Bonabeau (2003) referred to it as The Rich gets Richer. Similar to random, the chosen node must not violate the conditions.

Algorithm 9: The algorithm represents the machine code that is used to construct node according to random and/or scale-free network.

```
If random network:
    Get random linkage distribution and store them in a linkage list;
    Iterate over network size:
        Create new node with specific amount of links;
        Pick a random node from the existing network;
        If chosen node neighbor is not full:
            Form connection with chosen node;
        Add this new node to the network;
Else if scale free network:
    Get power law linkage distribution and store them in a linkage list;
        Sort linkage list, descending order;
            Iterate over network size:
                Create new node with specific amount of links;
                Choose a random node based on links of the node;
                If chosen node neighbor not full:
                    Form connection with the chosen node;
                Add this new node to the network;
```

The machine code (Algorithm 9) (Algorithm 10) are capable of specifying network property parameters (size and power-law parameter). However, it is not possible to specify the cluster, but it is possible to determine the clustering coefficient of the network. All of which will come in handy when conducting the simulation studies. The clustering coefficient is built based on the working principle of triangle formations in the A Clustering Coefficient Network Formation Game by Brautbar and Kearns (2011). It is determined by taking the average of the triangle-degree ratio of all the nodes (See Algorithm 11).

6.2 Construction of Centrality Metrics

The construction of each centrality measure is based on the proposed approaches in the previous chapter. Few libraries were used during the realization. The Itertools library is used specifically for returning a combination list of size 2 (starting and ending node), which is used for determining the geodesic and non-geodesic flows. The random library is used specifically for choosing the random

Algorithm 10: The algorithm represents the machine code that is used to establish connections among nodes.

```
Iterate over every node in the network:
    Store nodes that need neighbors in the possible list;
    While current node has free neighbor slot:
        If possible list is not empty:
            Choose a random node from the possible list;
            If chosen node and current node are not neighbor:
                Form connection between the two nodes;
            Else:
                Remove chosen node from possible list;
        Else:
            Choose a random node from the entire network;
            If chosen node and current node are not neighbors:
                Form connection between the two nodes;
```

node object, which was vital to the non-geodesic simulation scenarios. For each specific Python code, see Appendix: Python Core Code.

7 Simulation Study

A simulation study is an experiment that is conducted on a small representation of the real world setting. A simulation study is often applied in practice as the main benefit of the simulation study is cost saving. Depending on the type of experiment, live testing can be expensive and maybe not feasible. The simulation study is capable of producing a high volume of reliable results that may be difficult to achieve in real-world testing. These difficulties stem from time, ethics and other practical reasons (Bandini et al., 2009). Generally, a simulation study serves two areas: predictive research and explanatory research (Figure 7). However, there is a crucial factor of a simulation study, that is, how representative is the simulation model? If results are obtained from an unreliable simulation model, then the produced results are rendered useless. Therefore, a validation procedure must be conducted before the simulation can be begin.

7.1 Validation

The purpose of the validation process in a simulation study is to determine whether a simulation model is an accurate representation of the system, for the particular objective of the study (Law, 2003). Experimentation of different test scenarios can only begin once the simulation model is said to have face validity. If the simulation model produces inconsistent results, then the model must be revised. The validation process is based on the entropy measure of Ortiz-Arroyo

Algorithm 11: The algorithm represents the machine code that is used to determine the cluster coefficient of the network. CC_graph refers to the cluster coefficient of the entire network, while CC_node refers triangle ratio of particular node.

```
CC_graph counter set to 0;
Iterate over all the nodes in the network:
    CC_node counter set to 0;
    If node has at least two neighbor nodes:
        Triangle counter set to 0;
        If any node with two other neighbors forms a triangle:
            Add one to triangle counter;
    CC_node = triangle/(node neighbors size*(node neighbors size -1));
    CC_graph += CC_node;
CC_graph = CC_graph/network size;
```

and Hussain (2008). Ortiz-Arroyo and Hussain (2008) determine the importance of nodes in a network by analyzing the degree of connectivity and centrality of each node in the network. Despite the lack of existing models to compare with, it is possible to validate the proposed measures by comparing their results. If the central nodes produced from the proposed methods are aligned with the results of key players set, then the proposed measures are considered valid. Before applying Ortiz-Arroyo and Hussain (2008) entropy validator on the proposed measures, the validator model should also be validated. The entropy validator will be applied to the proposed centrality measures on the "Florentine Family" network, once it has been validated.

7.1.1 Validation of the Entropy Validator

To validate the entropy validator, a simple fully connected network of size 5 is constructed. Ortiz-Arroyo and Hussain (2008) state that the removal of any node in a fully connected network will have the same impact (Figure 8).

The graph (Figure 9) refers to the difference of impact each node has on the entire network. The ID of each node is displayed by integers on the x-axis (i.e. 0 on the x-axis is referred to node 0, so on and so forth.) and level of node removal impact is represented by the y-values. It can be seen that the removal of each has an equal amount of influence on the entire network. This finding is in line with the characteristics of a fully connected network. This result indicates that the entropy validator is valid in itself and can thus be used in this research.

7.1.2 Validation of the Proposed Measures

In this validation process, the Florentine Family network has been chosen for the sake of easy hands-on comparison and analysis. This model is widely used in social network analysis (SNA). The objective of this validation procedure

Figure 7: The figure describes the different phases of a traditional simulation study. (Image Source: Bandini et al. (2009))

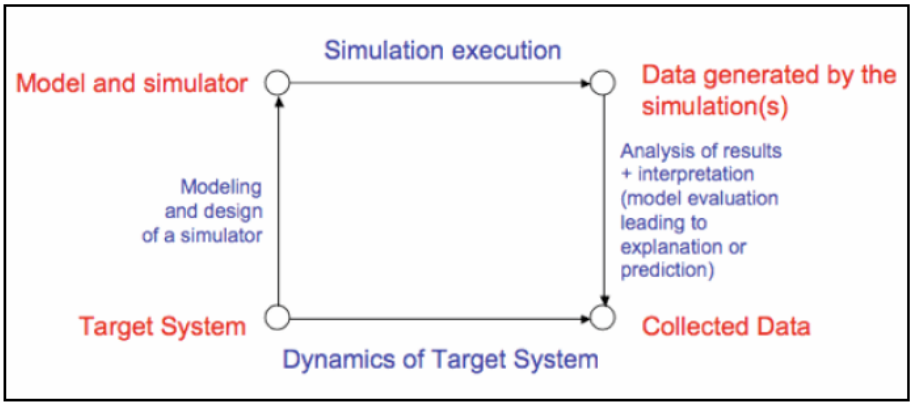


Figure 8: This is an example of a fully connected network of size 5.

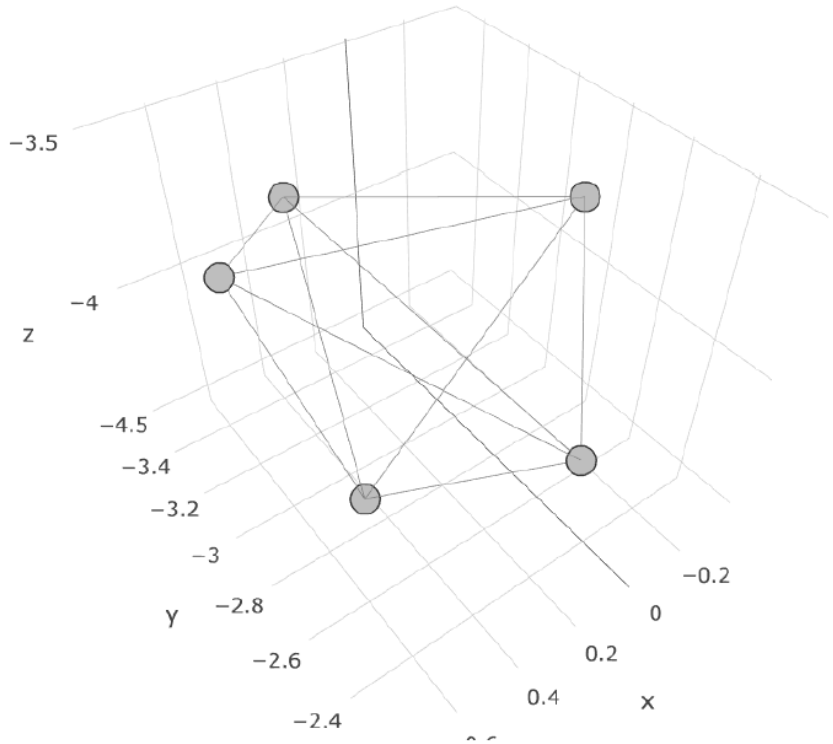


Figure 9: The graph describes the impact of node removal has on the entire network. The node IDs are represented on the x-axis, while their connectivity and centrality entropy values are displayed on the y-axis.

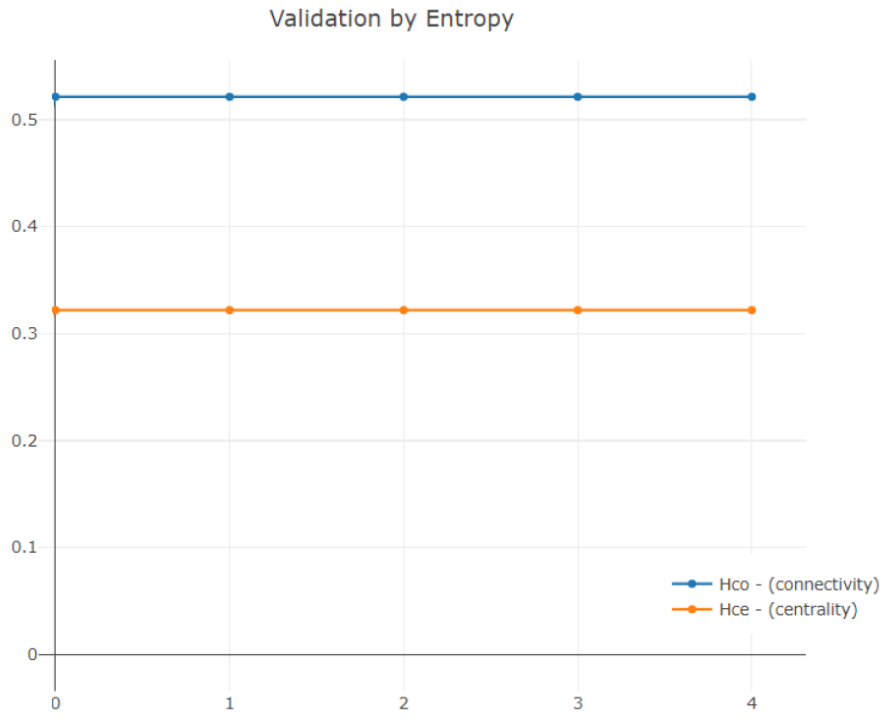
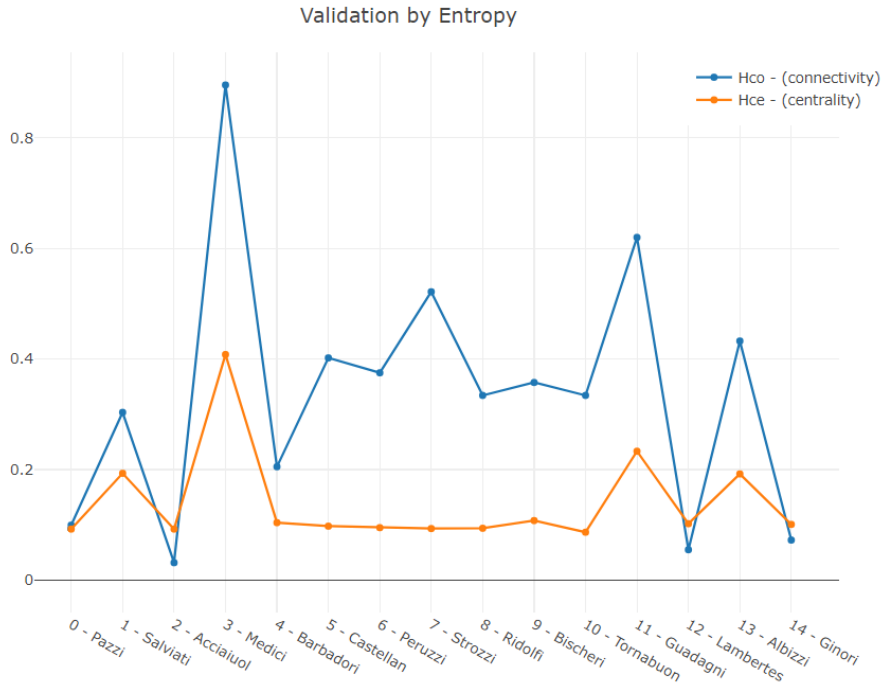


Figure 10: The graph presents the importance of each node of the Florentine family. The individual family represented on the x-axis while y-axis measures the connectivity and centrality entropy.



is to compare and analyze the results of entropy measures with the results of the proposed methods. The main focus is to determine whether each proposed methods is considered valid. Keep in mind that, the connectivity entropy is based on the degree of individual nodes, and this may not lead to higher values of measures that are closeness-like or betweenness-like (i.e. Serial-Paths, Serial-Trails, Serial-Geodesic). That said, it is not meaningful to simply match the node output between models.

Table 12: The table describes the simulation results based 100 runs. Each column represents different centrality measure, the values are measured according to the various centrality metrics. Each centrality measure has its own value scale (i.e. Values of Betweenness and Closeness should not be used in comparison).

Node ID	**Entropy Connectivity**	**Entropy Centrality**	**Parallel Paths/Trails**	**Serial Geodesic**	**Serial Paths**	**Serial Trails**	**Transfer Paths**	**Transfer Trails**	**Transfer Walks**
0 - Pazzi	993	922	2,33	1,72	1,69	1,69	0,00	0,00	0,00
1 - Salviati	3,034	1,931	2,80	1,83	1,80	1,80	0,05	0,05	10,51
2 - Acciaiuol	314	922	2,80	1,94	1,90	1,90	0,00	0,00	0,00
3 - Medici	**8,957**	**4,081**	**3,50**	**2,10**	**2,00**	**2,00**	**0,19**	**0,20**	**111,76**
4 - Barbadori	2,050	1,038	2,80	2,01	1,95	1,94	0,07	0,07	39,04
5 - Castellan	4,020	975	2,80	1,91	1,85	1,85	0,12	0,11	67,40
6 - Peruzzi	3,750	948	2,33	1,88	1,87	1,87	0,11	0,09	68,87
7 - Strozzi	5,216	932	2,80	1,92	1,89	1,89	**0,14**	**0,14**	**98,99**
8 - Ridolfi	3,341	937	**3,50**	**2,12**	**2,05**	**2,05**	0,11	0,10	74,36
9 - Bischeri	3,575	1,076	2,33	1,88	1,89	1,89	0,11	0,09	68,84
10 - Tornabuon	3,341	866	**3,50**	**2,15**	**2,08**	**2,07**	0,10	0,09	71,55
11 - Guadagni	**6,199**	**2,331**	2,80	1,93	1,93	1,93	0,11	0,10	72,18
12 - Lambertes	548	1,019	2,33	1,82	1,75	1,75	0,00	0,00	0,00
13 - Albizzi	**4,325**	**1,920**	**3,50**	1,99	1,93	1,93	0,06	0,06	44,19
14 - Ginori	723	1,007	2,80	1,89	1,80	1,80	0,00	0,00	0,00

Table 12 shows the result of all the proposed centrality approaches based on 100 simulation runs. The general finding regarding the top results of each approach is that Medici is the most central node. The parallel and serial duplication methods incorporates Freemans closeness-like and flow-betweenness principles. As result, Medici, Ridolfi and Tornabuon are the top 3 candidates in the serial and parallel replication rankings. A plausible explanation as to why Ridolfi and Tornabuon do not produce a significant impact in the centrality entropy and connectivity entropy measures, is that the positions of Ridolfi, Tornabuon and even Medici are situated closest to all other nodes. This effect is shown in Figure 10 as the removal of Ridolfi or Tornabuon do not produce a significant impact in reducing the number of viable paths for other nodes. This implies that the absence of one of those nodes will not influence the diffusion rate. Furthermore, since Ridolfi and Tornabuon do not have high degree centrality, they do not stand out in the connectivity entropy measure.

The proposed transfer methods are based on the random-walk betweenness, which also produced similar results. The top results of all the transfer approaches are supported by the connectivity entropy measure. Since the information is immutable and cunreplicated, it exists in one place at any point in time. This means that the nodes that are well connected in the network are more likely to be chosen. Strozzi is the balance between high connectivity and centeredness, he is positioned on the paths of many nodes and the fact that he has high degree centrality makes Strozzi a strong candidate for random node selection. Meanwhile, both Guadagni and Albizzi have high connectivity measure, but they are not considered in the proposed transfer measures. As they are located outside of the core center, making them less accessible.

Based on the findings and analysis above, it is conclusive to say that the proposed methods are well constructed and produced consistent results. This implies that the proposed methods can be used to test the different network property settings.

7.2 Experimentation

In the experimentation phase different centrality measures including the proposed ones will be tested on a variety of networks. Guzman et al. (2014) classified network properties into three types: size, power-law parameter, and clustering coefficient. Barabási and Bonabeau (2003) made a significant distinction between random and scale-free network. Once all the factors have been tested, data analysis will be conducted in the attempt to answer the proposed research questions. The aim of the simulation study is not to produce a predictive result, but rather examine how the centrality measures are influenced by the network properties.

7.2.1 Experiment settings

To obtain a clear understanding of the network properties effects, one property will be focused at any given time, while keeping others fixed. With that being

said, the independent variable is one of the factors mentioned by Guzman et al. (2014) and Barabási and Bonabeau (2003), the dependent variable is the centrality scores of each node in the network (rankings), and control variables are all other factors of Guzman et al. (2014). Confounding variables such as the state of the information is assumed to be immutable (e.g. The state of the used good in a transfer simulation does not change over time.), and simulation time is not the focus in this research and therefore is not included in the simulation analysis.

Furthermore, the insights of the increase in process complexity due to the increase in network size found during the realization phase was taken into consideration. When simulating other independent variables, the size of the network is set to 100. This decision is made due to the performance issues of the computer equipments.

7.3 Results

7.4 Data Analysis

To determine what effect each network property has on the outcomes of the centrality measures (rankings), data analysis will be conducted. The data analysis checks which network centrality measures are alike or different under the different network settings. The analysis will not be conducting a direct cross comparison between experiments (i.e. The value of node A in Exp 1 and the value of node A in Exp: 2 should not be compared.). The reason for this is that each experimental setup involves an entirely different network and it would be invalid to assume node A is the same node in both experiments (e.g. Exp: 1 and 2). Instead, the data analysis will be conducted on the rankings of the different centrality measures within the same experimental setup. The correlation analysis (Figure: 11, 12, 13, 14) is based on the number of similar nodes between two different centrality measures. For example, if there are 5 similar nodes between the T_Degree rankings and the T_Closeness, then the correlation value between T_Degree and T_Closeness is 5. If both T_Degree and T_Closeness rankings are identical, then the correlation values is 10. Oppositely, the correlation value is 0 when there is no similar nodes.

For the experiments where the network size is large, finding the same nodes in different centrality measures is less about coincidence and more about the connectivity and the position of the node. Of course, this does not mean that the factor of randomness should be ignored. The data analysis is divided into two parts: Scale-Free networks and Random networks. For the sake of convenience, the different centrality measures on the result tables (Table 14) have been re-coded. T_Degree refers to Degree Centrality, T_Walks refers to Transfer-Walks Centrality, T_Trails refers to Transfer-Trails Centrality, T_Paths refers to Transfer-Paths Centrality, T_Betweenness refers to Betweenness Centrality, T_Closeness refers to Closeness Centrality, T_Eigen refers to Eigenvector Centrality, S_Trails refers to Serial-Trails Centrality, S_Paths refers to Serial-Paths Centrality, S_Geodesic refers to Serial-Geodesic Centrality, and P_P&T refers

Table 13: The overview of the simulation experiments. Each experiment setup consists of distinctive network properties and is coupled with a unique experiment ID. Each table describes the top 10 rankings and simulation scores of different centrality measures. The distinctions among the tables are the experimental factors (i.e. network size, power-law parameter, random, and/or scale-free))

Simulation Exp		**Settings**		
ID	**Type**	**Size**	**Power / Linkage**	**Clustering Coefficient**
1	Random	10	10	1
2	Random	20	10	0.4652
3	Random	50	10	0.1599
4	Random	100	10	0.0777
5	Random	150	10	0.0551
6	Random	100	3	0.0167
7	Random	100	5	0.0347
8	Random	100	15	0.1243
9	Random	100	25	0.233
10	Random	100	35	0.3398
11	Scale-Free	10	5	0.3933
12	Scale-Free	20	5	0.01
13	Scale-Free	50	5	0.001
14	Scale-Free	100	5	0.0001
15	Scale-Free	150	5	<0.0001
16	Scale-Free	100	5	0.0001
17	Scale-Free	100	10	0.026
18	Scale-Free	100	15	0.0927
19	Scale-Free	100	20	0.1357
20	Scale-Free	100	25	0.1697

to Parallel Centrality that is used for both Parallel-Paths and Parallel-Trails.

7.4.1 Scale-Free Network

The results of Exp: 11 to 20 revealed that both network size and power-law parameter have huge impacts on the cohesion of the network. The results of Exp: 11 to 15 revealed that cohesion and network size are negatively correlated. That is, as the network size increases (while keeping power-law parameter fixed), the network cohesion drastically decreases (Table 13). As for the power-law parameter (Exp: 16 to 20), the network cohesion and network power-law parameter

Table 14:

are positively correlated. That is, as the network power-law parameter increase (while keeping network size fixed), the network cohesion increases. The network cohesion determines how the nodes are ranked for each centrality measure. In scale-free network, the most favorable node tend to be the ones with high degree or closeness. This observation can be seen in both scale-free and random network. The following sub-sections provide an in-depth discussion as to how network cohesion influences the various centrality measures.

7.4.1.1 Factor: Network Size

[htb]

Table 15: Scale-free network result: The table contains the top 10 node rankings of different centrality measures. The table is divided into 5 sections according to the network size.

ID	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
11	1	0.5556	0	7.3330	0	0.3004	1	0.1791	1	0.6806	1	0.6923	1	15	1	1.5762	1	1.4950	1	1.5332	1	3.0000
	0	0.5556	1	6.1670	1	0.2756	0	0.1544	0	0.2361	0	0.6000	0	15	3	1.4803	3	1.4377	3	1.4658	5	2.2500
	3	0.3333	3	4.6190	2	0.1636	2	0.0991	5	0.2222	3	0.5294	3	12	2	1.4706	2	1.4286	2	1.4658	4	2.2500
	2	0.3333	2	4.5280	3	0.1569	3	0.0933	4	0.2222	2	0.5294	2	12	0	1.4423	8	1.4196	4	1.4196	3	2.2500
	8	0.2222	6	2.3340	6	0.1089	8	0.0656	3	0.0694	5	0.4737	8	8	4	1.4107	0	1.4041	6	1.4107	2	2.2500
	6	0.2222	8	2.2990	8	0.1024	6	0.0651	2	0.0694	4	0.4737	6	8	5	1.4085	6	1.3889	8	1.4041	0	2.2500
	5	0.2222	4	0.8270	5	0.0664	5	0.0482	9	0.0000	8	0.4091	5	6	8	1.3889	5	1.3720	0	1.3975	9	1.8000
	4	0.2222	5	0.7440	4	0.0589	4	0.0462	8	0.0000	6	0.4091	4	6	6	1.3740	4	1.3720	5	1.3804	8	1.8000
	9	0.1111	9	0.0000	9	0.0000	9	0.0000	7	0.0000	9	0.3333	9	2	7	1.3062	7	1.2730	7	1.2876	7	1.8000
	7	0.1111	7	0.0000	7	0.0000	7	0.0000	6	0.0000	7	0.3333	7	2	9	1.3006	9	1.2605	9	1.2857	6	1.8000
	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
12	0	0.2632	0	9.5190	1	0.0549	1	0.0508	0	0.6023	0	0.4634	0	14	1	2.4707	1	2.3171	4	2.3399	4	3.8000
	2	0.2105	1	8.6515	0	0.0514	0	0.0504	1	0.4269	1	0.4419	1	12	0	2.4580	4	2.2919	1	2.3227	1	3.8000
	1	0.2105	3	6.4140	8	0.0347	10	0.0337	3	0.3080	4	0.3878	3	10	8	2.4142	0	2.2809	0	2.2947	0	3.8000
	5	0.1579	4	5.7205	10	0.0347	2	0.0337	2	0.2885	3	0.3725	4	9	4	2.4112	8	2.2067	8	2.2459	16	3.1667
	4	0.1579	2	5.3470	2	0.0335	8	0.0329	4	0.2066	2	0.3725	2	9	3	2.3780	12	2.1689	3	2.2274	12	3.1667
	3	0.1579	8	4.1025	3	0.0323	3	0.0303	5	0.2047	6	0.3455	6	7	12	2.3515	3	2.1591	16	2.2093	8	3.1667
	14	0.1053	14	4.0280	14	0.0286	14	0.0285	6	0.1988	8	0.3333	14	6	16	2.2809	16	2.1566	12	2.1991	7	3.1667
	10	0.1053	10	3.9910	4	0.0267	4	0.0251	9	0.1053	12	0.3220	8	6	10	2.1915	10	2.0994	7	2.0994	6	3.1667
	9	0.1053	5	1.6350	6	0.0187	6	0.0193	7	0.1053	7	0.3220	12	5	6	2.1889	6	2.0948	6	2.0994	3	3.1667
	8	0.1053	6	1.5480	5	0.0162	5	0.0157	8	0.0799	14	0.2923	7	5	7	2.1689	14	2.0430	10	2.0765	2	3.1667
	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
13	0	0.1224	2	14.7000	4	0.0060	4	0.0061	2	0.6250	1	0.3101	0	17	1	3.8222	1	3.5559	1	3.5328	2	7.0000
	1	0.1020	1	14.2000	6	0.0057	6	0.0056	1	0.6037	2	0.3025	2	15	27	3.7376	27	3.4605	32	3.5100	1	7.0000
	11	0.0816	0	14.1800	2	0.0056	2	0.0055	0	0.6029	0	0.2952	1	15	32	3.6759	32	3.3793	27	3.4434	32	6.1250
	5	0.0816	4	8.9800	0	0.0049	0	0.0048	4	0.3818	4	0.2606	5	13	10	3.4605	2	3.2930	0	3.2386	27	6.1250
	4	0.0816	5	7.6000	7	0.0043	7	0.0041	5	0.3231	5	0.2500	4	10	0	3.4605	0	3.2886	10	3.2343	10	6.1250
	3	0.0816	7	5.3400	1	0.0041	1	0.0038	7	0.2270	7	0.2475	3	10	2	3.4580	10	3.2515	2	3.2258	9	6.1250
	2	0.0816	6	5.3400	11	0.0035	16	0.0032	6	0.2270	10	0.2426	11	9	29	3.3793	29	3.1922	47	3.1839	7	6.1250
	10	0.0612	11	4.5600	5	0.0034	11	0.0032	11	0.1939	3	0.2402	7	9	42	3.3539	37	3.1310	37	3.1572	4	6.1250
	9	0.0612	3	4.5400	16	0.0032	3	0.0032	3	0.1930	32	0.2379	6	8	37	3.3539	42	3.1111	29	3.1210	0	6.1250
	8	0.0612	10	1.9000	12	0.0031	5	0.0031	10	0.0808	27	0.2379	24	7	47	3.3108	47	3.0954	42	3.0760	47	5.4444
	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
14	7	0.0606	6	28.2700	7	0.0020	7	0.0021	6	0.5828	5	0.2357	1	19	5	5.8615	5	5.6218	5	5.6962	6	11.0000
	10	0.0505	5	28.0300	14	0.0018	14	0.0018	5	0.5778	2	0.2324	0	19	2	5.8201	80	5.5307	80	5.4908	5	11.0000
	9	0.0505	1	27.9600	1	0.0016	1	0.0015	1	0.5764	6	0.2292	6	16	80	5.7625	2	5.4098	2	5.3427	80	9.9000
	3	0.0505	2	26.8700	2	0.0015	2	0.0015	2	0.5539	1	0.2220	7	15	75	5.6539	75	5.2744	75	5.2856	58	9.9000
	2	0.0505	7	22.4000	0	0.0015	28	0.0014	7	0.4618	7	0.2089	4	15	70	5.6091	17	5.2133	17	5.2326	17	9.9000
	1	0.0505	0	16.3800	28	0.0014	30	0.0014	0	0.3377	17	0.1957	15	13	6	5.5587	6	5.1697	70	5.2270	10	9.9000
	0	0.0505	4	11.7200	30	0.0014	0	0.0014	4	0.2416	10	0.1957	5	13	17	5.4848	24	5.1322	6	5.1887	7	9.9000
	18	0.0404	10	10.8900	16	0.0013	4	0.0013	10	0.2245	0	0.1957	2	13	24	5.4726	70	5.1269	58	5.1482	2	9.9000
	15	0.0404	14	10.7900	10	0.0013	10	0.0013	14	0.2224	80	0.1911	10	12	58	5.3542	58	5.0821	24	5.1216	98	9.0000
	8	0.0404	15	8.3000	4	0.0012	16	0.0013	15	0.1711	24	0.1911	9	12	23	5.2298	23	4.9180	23	4.9699	97	9.0000
	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
15	4	0.0403	0	56.9533	0	0.0009	0	0.0009	0	0.7748	0	0.2685	0	27	0	9.3770	0	8.6931	0	8.8270	0	18.6250
	3	0.0403	3	35.4400	3	0.0008	3	0.0008	3	0.4821	3	0.2423	3	23	3	8.4755	3	8.2320	3	8.1066	13	16.5556

	1	0.0403	2	32.8867	13	0.0008	2	0.0008	2	0.4474	2	0.2392	2	23	13	8.3008	13	7.9594	2	7.9467	4	16.5556
	15	0.0336	13	22.3867	4	0.0008	4	0.0008	13	0.3046	13	0.2289	4	17	141	8.2048	141	7.9087	13	7.8711	3	16.5556
	14	0.0336	4	18.9267	2	0.0008	13	0.0007	4	0.2575	4	0.2254	1	16	2	8.1823	2	7.7042	141	7.7202	2	16.5556
	11	0.0336	15	15.4133	25	0.0007	24	0.0006	15	0.2097	1	0.2201	15	14	17	8.0323	4	7.5291	35	7.4874	1	16.5556
	9	0.0336	18	15.1600	24	0.0006	25	0.0006	18	0.2062	15	0.2044	7	14	1	7.8753	148	7.5101	4	7.4762	148	14.9000
	5	0.0336	7	13.7933	28	0.0006	1	0.0006	7	0.1876	7	0.2011	13	13	4	7.8462	35	7.4799	1	7.4575	141	14.9000
	2	0.0336	23	12.9200	15	0.0006	15	0.0006	23	0.1758	6	0.2005	6	13	35	7.7403	1	7.4724	17	7.4426	135	14.9000
	0	0.0336	1	12.1933	18	0.0006	23	0.0006	1	0.1659	17	0.1984	9	12	6	7.7362	17	7.4463	18	7.4019	11	14.9000

The results (Table 15) showed that the most central nodes of different centrality measures appeared to be the same in networks with higher cohesion. This can be seen in Exp: 11 where node 1 is ranked number one for each centrality measure. As the network cohesion decreases, the general pattern showed that the rankings of these centrality measures tend to diverge in ways that specifically match their flow characteristics.

All the transfer centrality measures rankings tend to lean toward degree and betweenness, while the rankings of all the replication centrality measures tend to lean toward closeness and eigenvector. This phenomenon can be seen in several areas. As the network size increases, T_walks is correlated (Figure 11) with T_Degree, T_Betweenness and T_Eigen (Exp: 12, 13, 14, 15), while T_Paths and T_Trails exhibited a positive correlation with T_Betweenness and T_Eigen (Exp: 13, 14, 15). The findings support of Borgatti (2005) theory where Freemans Degree centrality is used to determine the Transfer-Walks centrality. Logically speaking, a well-connected node is also a most randomly picked node. And therefore, a node with higher Eigenvector Centrality value has a higher chance of being selected.

As for the serial replication flows, S_Trails, S_Paths and S_geodesic show a strong correlation (Figure 11) with T_Closeness (Exp: 12, 13, 14, 15). The strong correlation to Freemans Closeness stems from the fact that S_Trails, S_paths and S_geodesic are derived from the Freemans Closeness centrality. Similarly, P_P&T showed a positive correlation (Figure 11) with T_Closeness (Exp: 12, 13, 14, 15). The positive correlation to the Freemans Closeness stems from the number of parallel replications at any point in time is maximized. Because the high T_Eigen rank nodes are connected to other well-connected nodes, this feature maximizes the number of possible replications during each iteration.

Furthermore, given the power-law parameter of Exp: 11 to 15 is fixed, the network of larger size (e.g. Exp: 15) produces ranking results more reliable than the network of smaller size (e.g. Exp: 11). This is shown in Exp: 15 (Table 15), the likelihood of selecting node 0, 1, 2, 5, 7 out of a 150 node network is highly improbable in comparison to selecting nodes 0, 1, 2 from a 10 node network. Because, the placements of edges in larger network have greater impacts in the overall routing. In low cohesion network, the existence of an edge determines how big of a detour it causes for all nodes. A detour is defined as the rerouted path between two points due to the absence of an edge between those tow points. The detour is considerably smaller in a denser (higher cohesion) network.

7.4.1.2 Factor: Power-law Parameter

Figure 11: This table describes the correlation values between two centrality rankings based on the results of Exp: 11 to 15. The number represents the total number of nodes that exists within the two rankings. Note: Value 10 means all the notes are exist in both rankings, while 0 means two rankings have no node in common.

Exp: 11	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	10	10	10	10	10	10	10	10	10	10
T_Walks		-	10	10	10	10	10	10	10	10	10
T_Trails			-	10	10	10	10	10	10	10	10
T_Paths				-	10	10	10	10	10	10	10
T_Betweenness					-	10	10	10	10	10	10
T_Closeness						-	10	10	10	10	10
T_Eigen							-	10	10	10	10
S_Trails								-	10	10	10
S_Paths									-	10	10
S_Geodesic										-	10
P_P&T											-

Exp: 12	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	9	9	9	8	7	7	6	7	6	6
T_Walks		-	10	10	8	8	8	7	8	7	7
T_Trails			-	10	8	8	8	7	8	7	7
T_Paths				-	8	8	8	7	8	7	7
T_Betweenness					-	8	8	7	6	7	8
T_Closeness						-	10	8	8	8	9
T_Eigen							-	8	8	8	9
S_Trails								-	9	10	9
S_Paths									-	9	8
S_Geodesic										-	9
P_P&T											-

Exp: 13	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	8	6	7	8	7	7	4	4	4	6
T_Walks		-	8	9	10	8	9	4	4	4	6
T_Trails			-	9	8	6	8	3	3	3	5
T_Paths				-	9	7	9	3	3	3	5
T_Betweenness					-	8	9	4	4	4	6
T_Closeness						-	7	6	6	6	8
T_Eigen							-	3	3	3	5
S_Trails								-	10	10	7
S_Paths									-	10	7
S_Geodesic										-	7
P_P&T											-

Exp: 14	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	6	5	5	6	5	7	1	1	1	3
T_Walks		-	7	7	10	7	9	3	3	3	5
T_Trails			-	10	7	5	6	1	1	1	3
T_Paths				-	7	5	6	1	1	1	3
T_Betweenness					-	7	9	3	3	3	5
T_Closeness						-	7	6	6	6	7
T_Eigen							-	3	3	3	5
S_Trails								-	10	10	6
S_Paths									-	10	6
S_Geodesic										-	6
P_P&T											-

Exp: 15	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	6	5	6	6	6	7	5	5	5	6
T_Walks		-	7	8	10	8	8	6	6	7	6
T_Trails			-	8	7	6	6	5	5	6	5
T_Paths				-	8	7	7	6	6	6	6
T_Betweenness					-	8	8	6	6	7	6
T_Closeness						-	9	8	7	7	6
T_Eigen							-	7	6	6	6
S_Trails								-	9	9	7
S_Paths									-	9	8
S_Geodesic										-	7
P_P&T											-

[htb]

Table 16: Scale-free network result: The table contains the top 10 node rankings for the different centrality measures. The table is divided into 5 sections according to the power-law parameter.

ID	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
16	3	0.0707	1	36.2400	0	0.0022	0	0.0022	1	0.7471	1	0.2955	1	25	1	7.1377	1	6.8041	1	6.8182	1	14.1429
	2	0.0505	0	27.9000	1	0.0021	1	0.0021	0	0.5751	0	0.2773	2	18	0	6.7347	0	6.2937	0	6.3625	6	12.3750
	1	0.0505	2	19.2800	5	0.0020	5	0.0020	2	0.3974	2	0.2558	0	18	31	6.5132	5	6.2225	6	6.1491	5	12.3750
	0	0.0505	7	14.8800	7	0.0020	7	0.0019	7	0.3067	5	0.2457	7	14	5	6.4875	6	6.0699	5	6.1338	3	12.3750
	38	0.0404	5	13.2800	2	0.0017	2	0.0018	5	0.2738	6	0.2386	3	14	2	6.4791	2	5.9567	31	6.0109	2	12.3750
	18	0.0404	8	9.9400	4	0.0014	4	0.0015	8	0.2049	3	0.2386	8	12	6	6.3995	51	5.9353	51	5.9567	0	12.3750
	12	0.0404	3	8.4200	11	0.0013	3	0.0013	3	0.1736	7	0.2352	6	12	51	6.2225	31	5.8999	2	5.9175	89	11.0000
	10	0.0404	4	8.3600	24	0.0013	11	0.0013	4	0.1723	8	0.2286	5	12	17	6.1875	7	5.7929	24	5.8167	88	11.0000
	9	0.0404	6	8.3000	29	0.0012	29	0.0012	6	0.1711	31	0.2205	4	12	28	6.1529	24	5.7895	28	5.7491	87	11.0000
	8	0.0404	9	6.6000	20	0.0012	24	0.0012	9	0.1361	17	0.2205	13	11	27	6.1187	27	5.7558	27	5.7259	76	11.0000
	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
17	0	0.1010	0	487.2448	0	0.0066	0	0.0053	13	0.2789	5	0.3113	0	57	19	6.9965	19	7.4046	19	7.3442	58	12.3750
	2	0.0909	2	457.1855	1	0.0061	1	0.0051	5	0.2613	2	0.3103	2	56	5	6.9279	20	7.2263	20	7.3279	19	12.3750
	1	0.0909	3	434.5786	3	0.0058	3	0.0046	0	0.2477	1	0.3103	5	49	20	6.7901	47	7.1532	5	7.3225	14	12.3750
	4	0.0808	5	388.5756	2	0.0053	2	0.0042	1	0.2269	0	0.3075	3	45	14	6.7855	14	7.1377	58	7.1895	5	12.3750
	3	0.0808	1	385.2996	5	0.0052	5	0.0041	7	0.2000	3	0.2861	1	44	47	6.7577	3	7.1223	47	7.1377	2	12.3750
	7	0.0707	6	349.4134	7	0.0041	7	0.0036	12	0.1840	7	0.2813	6	41	1	6.7577	1	7.1019	11	7.1223	1	12.3750
	6	0.0707	7	286.8510	6	0.0041	6	0.0034	17	0.1723	6	0.2789	11	37	3	6.7073	5	7.0765	14	7.1121	0	12.3750
	5	0.0707	11	282.1536	11	0.0033	11	0.0030	2	0.1419	9	0.2750	10	33	58	6.6622	6	6.9866	3	7.0968	99	11.0000
	10	0.0606	9	272.9464	9	0.0033	9	0.0029	30	0.1306	14	0.2720	9	33	2	6.6488	2	6.9817	1	7.0463	77	11.0000
	9	0.0606	8	259.7432	8	0.0032	8	0.0029	10	0.1246	8	0.2712	8	32	0	6.6221	11	6.9767	22	7.0313	49	11.0000
	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
18	0	0.1515	0	516.1885	0	0.0125	0	0.0082	0	0.2961	0	0.4041	3	84	17	6.8370	12	7.7953	12	8.1214	14	16.5000
	1	0.1414	3	475.5816	2	0.0120	2	0.0081	1	0.2483	1	0.3882	0	82	6	6.8276	0	7.7708	14	8.0032	12	16.5000
	2	0.1313	2	469.3834	1	0.0118	1	0.0074	2	0.1629	3	0.3837	4	75	3	6.8041	14	7.7404	0	7.9710	8	16.5000
	3	0.1212	1	454.5763	3	0.0110	5	0.0069	4	0.1565	4	0.3708	1	74	48	6.7577	6	7.6983	17	7.9073	4	16.5000
	4	0.1111	4	442.6382	5	0.0104	3	0.0069	3	0.1464	2	0.3667	6	60	12	6.7531	48	7.6625	4	7.8571	3	16.5000
	5	0.1010	5	411.0468	4	0.0096	4	0.0067	9	0.1068	12	0.3600	2	60	7	6.7210	7	7.6507	20	7.8261	2	16.5000
	7	0.0909	6	391.0610	6	0.0096	6	0.0065	10	0.1025	17	0.3462	12	56	0	6.7119	10	7.6330	3	7.8199	0	16.5000
	6	0.0909	7	339.7272	8	0.0082	7	0.0055	23	0.0999	6	0.3462	7	55	14	6.6892	3	7.6271	11	7.7586	95	14.1429
	9	0.0808	8	332.7414	7	0.0079	8	0.0053	5	0.0860	7	0.3426	5	55	59	6.6712	8	7.6037	38	7.7465	85	14.1429
	8	0.0808	12	304.6225	11	0.0065	12	0.0047	7	0.0803	14	0.3414	8	54	4	6.6622	17	7.5979	24	7.7465	84	14.1429
	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
19	0	0.2020	0	752.0977	0	0.0329	1	0.0163	0	0.2269	2	0.4361	3	155	30	6.7210	30	8.7302	30	9.0576	1	19.8000
	1	0.1919	1	748.9733	1	0.0322	0	0.0157	1	0.2139	1	0.4361	6	154	13	6.6937	13	8.6995	13	9.0246	56	16.5000
	2	0.1818	2	717.4170	2	0.0292	2	0.0144	2	0.1786	0	0.4323	4	154	5	6.6937	39	8.6766	39	8.9674	42	16.5000
	3	0.1717	3	706.0166	3	0.0263	3	0.0136	3	0.1241	3	0.4213	2	152	7	6.6802	4	8.6766	4	8.9270	41	16.5000
	4	0.1616	4	672.4891	4	0.0246	4	0.0129	10	0.1037	4	0.4125	1	151	26	6.6757	26	8.6463	9	8.9109	40	16.5000
	5	0.1515	5	591.5495	5	0.0245	5	0.0129	5	0.0955	5	0.4024	5	144	34	6.6577	15	8.6463	6	8.8472	39	16.5000
	6	0.1414	6	547.5180	7	0.0213	7	0.0117	4	0.0851	6	0.3992	0	138	1	6.6532	34	8.6087	26	8.8393	38	16.5000
	7	0.1313	7	535.5131	6	0.0202	6	0.0106	8	0.0846	9	0.3960	7	117	56	6.6443	2	8.6012	5	8.8314	33	16.5000
	8	0.1212	8	400.6401	11	0.0163	11	0.0091	6	0.0737	10	0.3898	9	116	4	6.6309	9	8.5938	3	8.8235	30	16.5000
	9	0.1111	9	385.6782	8	0.0150	8	0.0082	20	0.0675	7	0.3867	8	108	2	6.6265	7	8.5863	16	8.8157	29	16.5000
	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
20	0	0.2525	1	836.9929	0	0.0685	0	0.0236	0	0.2586	0	0.4853	4	246	14	6.1453	17	8.7766	17	9.1328	4	19.8000

	1	0.2424	3	798.9368	1	0.0676	1	0.0223	2	0.1811	2	0.4806	7	232	22	6.1414	29	8.6842	18	9.1160	3	19.8000
	2	0.2323	4	787.0763	3	0.0622	3	0.0219	1	0.1289	1	0.4692	6	220	33	6.1376	7	8.6614	22	9.0909	2	19.8000
	3	0.2222	2	776.6914	2	0.0555	2	0.0207	5	0.1254	7	0.4605	3	214	5	6.1376	18	8.6237	7	9.0742	84	16.5000
	4	0.2121	0	756.1464	4	0.0555	4	0.0195	11	0.0884	3	0.4583	10	208	3	6.1376	14	8.6237	36	9.0576	73	16.5000
	5	0.2020	7	683.7317	5	0.0482	5	0.0179	20	0.0802	5	0.4521	1	203	17	6.1262	36	8.6162	29	9.0494	70	16.5000
	6	0.1919	6	671.5227	7	0.0475	6	0.0176	3	0.0774	4	0.4500	5	202	7	6.1262	8	8.6162	38	9.0082	66	16.5000
	7	0.1818	5	655.6307	6	0.0459	7	0.0172	4	0.0727	6	0.4459	9	196	28	6.1111	38	8.5863	33	9.0000	63	16.5000
	8	0.1717	8	608.0947	8	0.0401	8	0.0157	6	0.0651	8	0.4381	8	195	32	6.1036	28	8.5863	4	9.0000	56	16.5000
	9	0.1616	9	550.4874	9	0.0366	9	0.0145	16	0.0637	13	0.4342	12	192	18	6.1036	33	8.5789	6	8.9837	55	16.5000

Based on the results (Table 16), the power-law parameter is also a significant factor for the network cohesion. As the power-law parameter increases, the average linkage per node is increased. This implies that a free-scale network is slowly transforming into a random network. From the networks with fixed network size and low power-law parameter, all the transfer centrality measures (T_Walks, T_Trails, T_Paths) showed a positive correlation (Figure 12) with Freemans Degree (Exp: 17, 18, 19, 20) and Eigenvector Centrality (Exp: 17, 18, 19).

The serial centrality measures do not show any significant correlation (Figure 12) with the transfer centrality measures (The correlation of the serial centrality measures and Freemans Closeness is only significant in Exp: 18. Instead, the correlation among the different serial centrality measures (S_Geodesic, S_Paths, S_Trails) is observed (Exp: 16, 17, 19, 20). P_P&T is positively correlated with Freemans Closeness and Eigenvector Centrality (Exp: 16, 17, 18).

Due to the increase of power-law parameter, the correlations (Figure 12) described above became weaker (Exp: 18, 19, 20). As a result, the selection of nodes during the simulation is effected. Instead of choosing the same nodes from a small-sized neighbor pool, it is more likely that a range of different nodes will be chosen from a larger neighbor pool. This is caused by the increase of edges between the nodes in the network. This means that the rankings will return multiple moderately central nodes (Key Player Set) instead of one significant central node. This phenomenon is as predicted, as the power-law parameter increases, the network flow has more traversal opportunities than the network with lower power-law parameter. Conversely, if the network were to decrease in size (while maintaining the same power-law parameter), the network would also become denser.

7.4.2 Random Network

Since the node linkage of random networks does not follow a power-law distribution, therefore, the term "linkage parameter" is referred. Similar to scale-free networks, the results of the experiments (Exp: 1 to 10) revealed that both the network size and the linkage parameter have a huge impact on the network cohesion. The first five experiments (Exp: 1 to 5) (Table 13) relate to the changes in the network size, whereas the following five experiments (Exp: 6 to 10) relate to the changes in the linkage parameter.

7.4.2.1 Factor: Network Size

Figure 12: This table describes the correlation values between two centrality rankings based on the results of Exp: 16 to 20. The number represents the total number of nodes that exists within the two rankings. Note: Value 10 means all the notes are exist in both rankings, while 0 means two rankings have no node in common.

Exp: 16	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	6	3	4	6	5	5	3	3	3	4
T_Walks		-	6	7	10	8	9	5	6	5	6
T_Trails			-	9	6	5	6	4	6	5	4
T_Paths				-	7	6	7	4	6	5	5
T_Betweenness					-	8	9	5	6	5	6
T_Closeness						-	8	7	7	6	6
T_Eigen							-	5	6	5	6
S_Trails								-	8	9	5
S_Paths									-	9	5
S_Geodesic										-	5
P_P&T											-

Exp: 17	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	8	8	8	6	8	8	5	5	3	4
T_Walks		-	10	10	5	9	9	5	6	4	4
T_Trails			-	10	5	9	9	5	6	4	4
T_Paths				-	5	9	9	5	6	4	4
T_Betweenness					-	5	5	4	3	2	4
T_Closeness						-	8	6	6	4	0
T_Eigen							-	5	6	4	4
S_Trails								-	8	8	7
S_Paths									-	8	5
S_Geodesic										-	5
P_P&T											-

Exp: 18	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	9	9	9	8	7	9	5	5	3	5
T_Walks		-	9	10	7	8	10	6	6	4	6
T_Trails			-	9	7	7	9	5	5	4	5
T_Paths				-	7	8	10	6	6	4	6
T_Betweenness					-	6	7	4	4	3	4
T_Closeness						-	8	8	7	6	6
T_Eigen							-	6	6	4	6
S_Trails								-	8	6	5
S_Paths									-	5	5
S_Geodesic										-	5
P_P&T											-

Exp: 19	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	10	9	9	8	9	10	5	4	5	1
T_Walks		-	9	9	8	9	10	5	4	5	1
T_Trails			-	10	8	8	9	5	3	4	1
T_Paths				-	8	8	9	5	3	4	1
T_Betweenness					-	8	8	4	2	4	1
T_Closeness						-	9	5	4	5	1
T_Eigen							-	5	4	5	1
S_Trails								-	7	5	3
S_Paths									-	6	2
S_Geodesic										-	2
P_P&T											-

Exp: 20	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	10	10	10	7	9	8	3	2	3	3
T_Walks		-	10	10	7	9	8	3	2	3	3
T_Trails			-	10	7	9	8	3	2	3	3
T_Paths				-	7	9	8	3	2	3	3
T_Betweenness					-	7	5	2	0	2	3
T_Closeness						-	7	3	2	3	3
T_Eigen							-	3	2	3	2
S_Trails								-	6	5	1
S_Paths									-	7	0
S_Geodesic										-	1
P_P&T											-

[htb]

Table 17: Random network result: The table contains the top 10 node rankings for the different centrality measures. The table is divided into 5 sections according to the network size.

1	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
	9	1	1	2.635	0	0.6538	1	0.4058	9	0	9	1	9	81	1	1.0429	9	1.8	9	1.8	9	4.5
	8	1	0	2.632	2	0.6407	7	0.4031	8	0	8	1	8	81	7	1.0393	8	1.8	8	1.8	8	4.5
	7	1	3	2.57	1	0.6211	8	0.4022	7	0	7	1	7	81	5	1.0393	7	1.8	7	1.8	7	4.5
	6	1	6	2.568	4	0.608	3	0.4013	6	0	6	1	6	81	0	1.0393	6	1.8	6	1.8	6	4.5
	5	1	2	2.561	3	0.6038	4	0.4011	5	0	5	1	5	81	2	1.0369	5	1.8	5	1.8	5	4.5
	4	1	7	2.559	5	0.5967	5	0.3982	4	0	4	1	4	81	3	1.0345	4	1.8	4	1.8	4	4.5
	3	1	5	2.513	6	0.5809	0	0.398	3	0	3	1	3	81	8	1.0333	3	1.8	3	1.8	3	4.5
	2	1	4	2.48	7	0.5698	6	0.3964	2	0	2	1	2	81	9	1.0309	2	1.8	2	1.8	2	4.5
	1	1	9	2.47	9	0.5669	9	0.3918	1	0	1	1	1	81	4	1.0309	1	1.8	1	1.8	1	4.5
	0	1	8	2.456	8	0.5596	2	0.3884	0	0	0	1	0	81	6	1.0286	0	1.8	0	1.8	0	4.5
2	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
	0	0.5789	0	8.499	0	0.8648	0	0.4491	0	0.0375	0	0.7037	0	103	3	1.8429	19	3.1667	19	3.1667	19	6.3333
	9	0.5263	8	7.739	1	0.7928	2	0.4284	7	0.0347	9	0.6786	3	97	12	1.8393	18	3.1667	18	3.1667	18	6.3333
	8	0.5263	3	7.6855	3	0.7912	1	0.4278	9	0.033	8	0.6786	8	96	8	1.8322	17	3.1667	17	3.1667	17	6.3333
	7	0.5263	1	7.6255	4	0.7826	7	0.4278	6	0.0325	7	0.6786	4	95	2	1.8304	16	3.1667	16	3.1667	16	6.3333
	6	0.5263	2	7.617	2	0.7784	9	0.4264	3	0.0324	6	0.6786	7	94	19	1.8287	15	3.1667	15	3.1667	15	6.3333
	5	0.5263	4	7.5875	6	0.7751	6	0.4229	1	0.0319	5	0.6786	5	94	9	1.8287	14	3.1667	14	3.1667	14	6.3333
	4	0.5263	6	7.5735	5	0.7743	8	0.4229	2	0.0306	4	0.6786	2	94	7	1.8252	13	3.1667	13	3.1667	13	6.3333
	3	0.5263	7	7.5415	8	0.7728	3	0.4218	5	0.03	3	0.6786	1	94	18	1.8199	12	3.1667	12	3.1667	12	6.3333
	2	0.5263	5	7.448	7	0.7651	5	0.4206	8	0.0291	2	0.6786	9	93	16	1.8199	11	3.1667	11	3.1667	11	6.3333
	1	0.5263	9	7.4095	9	0.7554	4	0.4191	15	0.0273	1	0.6786	6	93	11	1.8199	10	3.1667	10	3.1667	10	6.3333
3	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
	12	0.2245	1	25.7492	1	0.9148	0	0.4055	0	0.0285	2	0.5506	1	110	20	4.0164	2	6.9602	49	7	49	12.25
	1	0.2245	0	25.4346	0	0.9089	12	0.4008	14	0.0276	1	0.5506	12	106	2	4.0131	26	6.9602	48	7	48	12.25
	0	0.2245	12	25.334	12	0.9013	1	0.4004	1	0.0273	14	0.5444	0	105	14	4.0098	24	6.9602	45	7	47	12.25
	35	0.2041	10	23.05	10	0.8264	14	0.3878	12	0.0249	0	0.5444	10	100	12	4.0065	49	6.9504	44	7	46	12.25
	33	0.2041	13	23.029	9	0.8249	2	0.3853	29	0.0248	29	0.5385	13	99	37	4.0033	43	6.9504	43	7	45	12.25
	29	0.2041	3	22.985	3	0.8239	9	0.3829	20	0.0239	17	0.5385	20	98	36	4.0033	29	6.9504	42	7	44	12.25
	22	0.2041	20	22.8986	2	0.8237	29	0.3826	17	0.0238	5	0.5385	11	98	41	4	20	6.9504	41	7	43	12.25
	21	0.2041	4	22.877	11	0.8231	16	0.3826	16	0.0238	33	0.5326	5	97	23	3.9967	14	6.9504	40	7	42	12.25
	20	0.2041	11	22.871	13	0.8213	7	0.3826	2	0.0237	20	0.5326	3	97	32	3.9935	9	6.9504	39	7	41	12.25
	19	0.2041	6	22.8074	6	0.8199	4	0.3826	33	0.0236	18	0.5326	35	96	30	3.9935	1	6.9504	38	7	40	12.25
4	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
	5	0.1212	5	58.792	5	0.9341	5	0.3798	5	0.0206	5	0.467	5	117	94	7.3117	43	12.2072	91	12.375	99	24.75
	20	0.1111	2	53.928	2	0.8602	20	0.365	3	0.0176	6	0.4605	2	113	8	7.3063	7	12.2072	90	12.375	98	24.75
	19	0.1111	1	53.669	1	0.8536	4	0.3633	19	0.0175	9	0.4583	1	112	4	7.3063	76	12.1921	89	12.375	97	24.75
	6	0.1111	0	53.265	0	0.8526	3	0.3626	20	0.0171	3	0.4583	6	108	85	7.3009	72	12.1921	81	12.375	96	24.75
	4	0.1111	6	53.242	6	0.8524	19	0.3616	34	0.0166	24	0.4562	0	108	78	7.3009	33	12.1921	79	12.375	95	24.75
	3	0.1111	3	53.167	3	0.8519	6	0.3614	9	0.0164	22	0.4562	3	107	95	7.2955	5	12.1921	75	12.375	94	24.75
	2	0.1111	4	52.949	4	0.8519	0	0.3598	13	0.0162	13	0.4562	19	105	34	7.2901	97	12.1771	73	12.375	93	24.75
	1	0.1111	19	52.856	20	0.8506	2	0.3585	0	0.0162	4	0.4562	4	105	89	7.2848	96	12.1771	70	12.375	92	24.75
	0	0.1111	20	52.655	19	0.8461	1	0.3571	6	0.0162	1	0.4562	20	103	54	7.2848	48	12.1771	67	12.375	91	24.75
	62	0.101	12	48.222	8	0.7773	34	0.3465	14	0.016	0	0.4562	36	102	53	7.2848	37	12.1771	66	12.375	90	24.75

5	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
	13	0.0738	2	80.192	11	0.821	12	0.341	5	0.0135	4	0.4245	2	111	40	10.434	149	16.556	149	16.556	144	37.25
	12	0.0738	11	80.153	0	0.820	8	0.340	0	0.0135	5	0.4221	6	110	30	10.434	148	16.556	148	16.556	143	37.25
	11	0.0738	0	80.053	2	0.819	5	0.339	8	0.0132	0	0.4209	4	109	2	10.434	147	16.556	147	16.556	139	37.25
	10	0.0738	6	79.966	9	0.819	3	0.339	13	0.0131	70	0.4197	1	109	127	10.427	146	16.556	146	16.556	135	37.25
	9	0.0738	1	79.946	6	0.819	0	0.338	4	0.0130	2	0.4197	0	109	126	10.427	145	16.556	145	16.556	133	37.25
	8	0.0738	4	79.923	3	0.818	13	0.337	12	0.0130	73	0.4185	11	108	111	10.427	143	16.556	144	16.556	129	37.25
	7	0.0738	3	79.768	4	0.818	9	0.337	42	0.0126	10	0.4185	10	108	98	10.420	142	16.556	143	16.556	127	37.25
	6	0.0738	7	79.718	5	0.818	11	0.337	10	0.0125	3	0.4185	7	108	89	10.420	141	16.556	142	16.556	125	37.25
	5	0.0738	5	79.705	7	0.818	4	0.336	3	0.0124	69	0.4174	13	107	72	10.420	140	16.556	141	16.556	124	37.25
	4	0.0738	10	79.690	10	0.817	7	0.336	2	0.0123	51	0.4174	9	107	63	10.412	139	16.556	140	16.556	121	37.25

Different network cohesions ranging from fully connected network (cohesion = 1) to less connected network (cohesion <0.1) were tested. In a fully connected network (Exp: 1), the concept of one unique central node does not exist, rather the concept of key player set is adopted. This is shown in Exp: 1 (Table 17), as none of the nodes is significantly different in comparison to others. As the network size increases, the one unique central node theory becomes more apparent (lower network cohesion). Similar to the scale-network, the traversal opportunity is drastically reduced in the low cohesion network, which implies that even a minor centrality difference between two nodes will stand out.

The transfer centrality measures (T_Walks, T_Trails and T_Paths) showed a positive correlation (Figure 13) with Freemans Betweenness and Degree, and Bonacichs Eigenvector Centrality (Exp: 2, 3, 4, 5). Furthermore, similar to the scale-free network findings, centrality rankings of the transfer walks, paths and trails tend to favor the nodes with high T_Degree and T_Eigen.

Regarding the result of the serial centrality measures, no significant result is produced. In the small size networks, S_Paths, S_Geodesic and P_P&T do not produce distinctive (useful) results. This is because the diffusion points of origin within these highly connected networks are not significantly different from each other (Exp: 1 to 5). However, the centrality rankings of S_Trails show signs of correlation (Figure 13) with T_Eigen, T_Closeness and T_Betweenness, but the correlation weakens as the network size increases. This can be seen in Exp: 3 and Exp: 4, the correlation value is decreased drastically. This can be explained partly by the randomness factor, but it is mainly caused by the influence of the network cohesion. In higher cohesion network, the positional and degree differences among the nodes impact the rankings in a smaller way. However, the positional and degree differences among nodes are more apparent in a lower cohesion network.

7.4.2.2 Factor: Linkage Parameter

Figure 13: This table describes the correlation values between two centrality rankings based on the results of Exp: 1 to 5. The number represents the total number of nodes that exists within the two rankings. Note: Value 10 means all the notes are exist in both rankings, while 0 means two rankings have no node in common.

Exp: 01	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	10	10	10	10	10	10	10	10	10	10
T_Walks		-	10	10	10	10	10	10	10	10	10
T_Trails			-	10	10	10	10	10	10	10	10
T_Paths				-	10	10	10	10	10	10	10
T_Betweenness					-	10	10	10	10	10	10
T_Closeness						-	10	10	10	10	10
T_Eigen							-	10	10	10	10
S_Trails								-	10	10	10
S_Paths									-	10	10
S_Geodesic										-	10
P_P&T											-

Exp: 02	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	10	10	10	9	10	10	5	0	0	0
T_Walks		-	10	10	9	10	10	5	0	0	0
T_Trails			-	10	9	10	10	5	0	0	0
T_Paths				-	9	10	10	5	0	0	0
T_Betweenness					-	9	9	5	1	1	1
T_Closeness						-	10	5	0	0	0
T_Eigen							-	5	0	0	0
S_Trails								-	5	5	5
S_Paths									-	10	10
S_Geodesic										-	10
P_P&T											-

Exp: 03	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	4	3	4	6	5	5	2	3	0	0
T_Walks		-	8	4	4	3	8	2	2	0	0
T_Trails			-	5	4	3	7	2	3	0	0
T_Paths				-	7	5	3	3	5	0	0
T_Betweenness					-	8	4	4	5	0	0
T_Closeness						-	4	3	5	0	0
T_Eigen							-	2	2	0	0
S_Trails								-	3	1	1
S_Paths									-	2	2
S_Geodesic										-	8
P_P&T											-

Exp: 04	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	9	9	9	6	6	9	1	1	0	0
T_Walks		-	9	9	6	6	9	1	1	0	0
T_Trails			-	9	6	6	9	2	1	0	0
T_Paths				-	7	6	9	2	1	0	0
T_Betweenness					-	6	6	1	1	0	0
T_Closeness						-	6	1	1	0	0
T_Eigen							-	1	1	0	0
S_Trails								-	0	1	2
S_Paths									-	0	2
S_Geodesic										-	2
P_P&T											-

Exp: 05	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	6	7	8	6	3	7	0	0	0	0
T_Walks		-	9	6	6	6	8	1	0	0	0
T_Trails			-	7	6	6	8	1	0	0	0
T_Paths				-	7	4	6	0	0	0	0
T_Betweenness					-	6	5	1	0	0	0
T_Closeness						-	4	1	0	0	0
T_Eigen							-	1	0	0	0
S_Trails								-	0	0	1
S_Paths									-	9	2
S_Geodesic										-	2
P_P&T											-

[htb]

Table 18: Random network result: The table contains the top 10 node rankings for the different centrality measures. The table is divided into 5 sections according to the linkage parameter.

ID	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
6	11	0.0404	4	123.3758	4	0.0267	24	0.0317	1	0.2733	3	0.2340	3	16	1	7.2901	1	7.3770	1	7.4717	34	9.9000
	10	0.0404	1	122.2864	9	0.0266	0	0.0293	3	0.2358	1	0.2340	1	15	0	7.2741	5	7.1739	5	7.2848	17	9.9000
	9	0.0404	3	121.3353	0	0.0260	9	0.0278	2	0.2192	0	0.2308	4	14	17	7.2210	14	7.1070	0	7.2105	8	9.9000
	8	0.0404	7	114.7239	5	0.0246	20	0.0276	0	0.1953	4	0.2255	8	13	90	7.1172	0	7.0866	17	7.1895	5	9.9000
	7	0.0404	8	111.4023	1	0.0240	1	0.0274	4	0.1843	5	0.2230	7	13	5	7.1121	17	7.0664	34	7.1791	3	9.9000
	6	0.0404	6	110.5628	7	0.0233	4	0.0274	11	0.1656	2	0.2166	5	13	12	7.0563	34	7.0563	14	7.1172	1	9.9000
	5	0.0404	0	109.3058	3	0.0230	5	0.0269	6	0.1422	8	0.2120	2	13	14	6.9866	8	7.0413	8	7.1172	0	9.9000
	4	0.0404	9	106.6989	6	0.0223	7	0.0258	5	0.1338	7	0.2111	0	13	34	6.9817	89	7.0362	89	7.1019	90	9.0000
	3	0.0404	5	106.1381	24	0.0221	33	0.0256	7	0.1333	6	0.2089	9	12	72	6.9376	12	7.0313	12	7.0463	89	9.0000
	2	0.0404	23	72.3252	8	0.0219	23	0.0252	9	0.1226	9	0.2050	6	12	8	6.9328	90	6.9571	90	6.9866	88	9.0000
	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
7	9	0.0606	1	65.1189	9	0.2579	4	0.2067	1	0.0581	1	0.3667	1	35	1	8.9674	7	10.9878	72	11.0000	72	19.8000
	8	0.0606	5	63.6507	1	0.2578	9	0.2059	2	0.0575	2	0.3626	5	32	23	8.8949	44	10.9756	79	10.9878	13	19.8000
	7	0.0606	2	63.3490	0	0.2563	8	0.2046	4	0.0514	7	0.3474	2	32	7	8.8314	23	10.9272	50	10.9878	7	19.8000
	6	0.0606	0	62.0783	3	0.2558	0	0.2038	7	0.0499	9	0.3438	7	30	79	8.8157	47	10.9151	44	10.9878	5	19.8000
	5	0.0606	7	61.9710	2	0.2549	7	0.2032	9	0.0448	5	0.3414	3	30	44	8.8157	1	10.9151	9	10.9878	4	19.8000
	4	0.0606	9	61.6689	5	0.2509	6	0.2027	0	0.0447	0	0.3402	0	30	22	8.8157	50	10.9031	7	10.9878	2	19.8000
	3	0.0606	3	61.1986	8	0.2482	2	0.2025	6	0.0384	47	0.3322	9	29	97	8.7844	79	10.8791	57	10.9756	1	19.8000
	2	0.0606	8	59.8925	7	0.2480	5	0.1986	3	0.0361	29	0.3289	8	27	47	8.7844	19	10.8791	47	10.9756	99	16.5000
	1	0.0606	4	58.9728	6	0.2469	1	0.1986	26	0.0359	14	0.3289	6	27	72	8.7766	14	10.8672	38	10.9756	97	16.5000
	0	0.0606	6	58.7672	4	0.2463	3	0.1942	22	0.0355	4	0.3289	4	27	0	8.7766	97	10.8553	29	10.9756	96	16.5000
	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
8	45	0.1616	0	50.9530	0	0.9731	45	0.4190	45	0.0128	2	0.5351	0	235	19	6.1453	99	12.3750	99	12.3750	99	24.7500
	2	0.1616	2	50.8008	2	0.9683	1	0.4184	1	0.0127	1	0.5351	2	231	1	6.1338	98	12.3750	98	12.3750	98	24.7500
	1	0.1616	1	50.4432	1	0.9677	2	0.4159	2	0.0113	45	0.5323	1	229	69	6.1300	96	12.3750	97	12.3750	97	24.7500
	0	0.1616	45	50.3537	45	0.9560	0	0.4159	14	0.0112	41	0.5323	45	227	44	6.1300	91	12.3750	96	12.3750	96	24.7500
	44	0.1515	4	47.7739	4	0.9132	42	0.4048	17	0.0112	17	0.5323	43	223	98	6.1262	90	12.3750	95	12.3750	95	24.7500
	43	0.1515	8	47.7087	8	0.9122	21	0.4041	33	0.0112	44	0.5294	39	221	85	6.1262	89	12.3750	94	12.3750	94	24.7500
	42	0.1515	22	47.5902	3	0.9110	12	0.4039	22	0.0111	62	0.5266	38	221	81	6.1262	88	12.3750	93	12.3750	93	24.7500
	41	0.1515	43	47.5810	13	0.9098	29	0.4038	34	0.0109	33	0.5266	41	220	56	6.1262	85	12.3750	92	12.3750	92	24.7500
	40	0.1515	36	47.4870	23	0.9096	31	0.4037	42	0.0108	14	0.5266	26	220	47	6.1262	84	12.3750	91	12.3750	91	24.7500
	39	0.1515	3	47.4733	9	0.9093	28	0.4032	12	0.0108	93	0.5238	22	220	16	6.1262	83	12.3750	90	12.3750	90	24.7500
	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
9	6	0.2727	6	51.2062	6	1.0104	6	0.4661	6	0.0093	6	0.5789	6	665	26	4.6154	99	12.3750	99	12.3750	99	33.0000
	30	0.2626	30	49.5195	7	0.9758	3	0.4584	8	0.0087	30	0.5756	7	646	41	4.6132	98	12.3750	98	12.3750	98	33.0000
	12	0.2626	5	49.5162	4	0.9750	30	0.4580	37	0.0086	12	0.5756	5	646	40	4.6089	97	12.3750	97	12.3750	97	33.0000
	8	0.2626	2	49.4874	0	0.9743	12	0.4580	30	0.0086	8	0.5756	4	644	37	4.6089	96	12.3750	96	12.3750	96	33.0000
	7	0.2626	12	49.4104	5	0.9742	5	0.4579	0	0.0086	7	0.5756	2	644	35	4.6089	95	12.3750	95	12.3750	95	33.0000
	5	0.2626	4	49.3647	1	0.9741	8	0.4578	41	0.0086	5	0.5756	0	644	27	4.6089	94	12.3750	94	12.3750	94	33.0000
	4	0.2626	1	49.3621	2	0.9737	0	0.4577	12	0.0086	4	0.5756	1	643	23	4.6089	93	12.3750	93	12.3750	93	33.0000
	3	0.2626	7	49.3424	12	0.9730	1	0.4577	4	0.0085	3	0.5756	8	641	4	4.6089	92	12.3750	92	12.3750	92	33.0000
	2	0.2626	0	49.3252	8	0.9723	4	0.4574	3	0.0085	2	0.5756	3	640	92	4.6068	91	12.3750	91	12.3750	91	33.0000
	1	0.2626	8	49.3186	3	0.9713	2	0.4568	11	0.0085	1	0.5756	12	639	73	4.6068	90	12.3750	90	12.3750	90	33.0000
	N	T_Degree	N	T_Walks	N	T_Trails	N	T_Paths	N	T_Betw	N	T_Close	N	T_Eigen	N	S_Trails	N	S_Paths	N	S_Geo	N	P_P&T
10	50	0.3636	2	48.6885	0	0.9761	18	0.4758	36	0.0075	50	0.6111	50	1253	75	3.7134	99	12.3750	99	12.3750	99	33.0000

	36	0.3636	6	48.6224	4	0.9751	36	0.4756	50	0.0074	36	0.6111	2	1252	2	3.7120	98	12.3750	98	12.3750	98	33.0000
	24	0.3636	50	48.6138	3	0.9747	8	0.4754	18	0.0073	24	0.6111	0	1252	34	3.7106	97	12.3750	97	12.3750	97	33.0000
	19	0.3636	4	48.6091	2	0.9745	24	0.4752	0	0.0073	19	0.6111	8	1251	76	3.7065	96	12.3750	96	12.3750	96	33.0000
	18	0.3636	3	48.6062	1	0.9741	2	0.4751	4	0.0072	18	0.6111	7	1251	68	3.7065	95	12.3750	95	12.3750	95	33.0000
	8	0.3636	8	48.6014	5	0.9740	7	0.4750	19	0.0072	8	0.6111	6	1251	31	3.7065	94	12.3750	94	12.3750	94	33.0000
	7	0.3636	7	48.5855	6	0.9735	1	0.4748	5	0.0072	7	0.6111	4	1250	96	3.7051	93	12.3750	93	12.3750	93	33.0000
	6	0.3636	0	48.5579	8	0.9732	6	0.4742	1	0.0072	6	0.6111	3	1250	13	3.7051	92	12.3750	92	12.3750	92	33.0000
	5	0.3636	19	48.5319	7	0.9725	19	0.4740	24	0.0072	5	0.6111	24	1249	87	3.7037	91	12.3750	91	12.3750	91	33.0000
	4	0.3636	5	48.5141	24	0.9712	5	0.4740	8	0.0072	4	0.6111	1	1249	59	3.7037	90	12.3750	90	12.3750	90	33.0000

The centrality rankings (Table 18) show that influence of the linkage parameter is similar to the power-law parameter of the scale-free networks. From Exp: 6 (linkage parameter = 3), the rankings of P_P&T, S_Geodesic, S_Paths and S_Trails showed a strong correlation (Table: 14) with T_Closeness and T_Eigenvector (Exp: 6, 7).

The transfer centrality measures (T_Walks, T_Trails and T_Paths), they are strongly correlated (Figure 14) with Freemans Degree, Betweenness and Closeness (Exp: 6, 7, 9, 10). Furthermore, the concept of key player set is adopted due the increase of linkages. This can be seen from the results of Exp: 6 to Exp: 10, where the correlation values are significantly higher in Exp: 10 than the correlation values in Exp: 6.

However, Exp: 8, Exp 9 and Exp 10 result in high network cohesion, Which cause the serial and parallel centrality measures (S_Geodesic, S_Paths, S_Trails and P_P&T) failed to produce any useful result. This finding is similar to the Exp 1 findings. This is because the diffusion points of origin within these highly connected networks are not significantly different from each other.

7.4.3 Summation of Findings

The random node selection during each simulation run, and as a result, the produced data should not be taken at face value. The random factor did indeed produce some uncorrelated results. However, there are clearly numerous correlations between the size, linkage parameter, and cohesion in both scale-free and random networks. The correlation of network size and network cohesion must also take into account the linkage parameter. If the linkage parameter increases or the network size decreases (high network cohesion), then a set of central nodes can be expected.

Alternatively, fewer central nodes are expected in a network with lower cohesion. This is because a network with low cohesion has fewer direct connections, which means that each relationship (link) is valued more. Between two points A and B, lower cohesion network has fewer paths between the two than in a network with higher cohesion. Thus, the result of one central node is more apparent.

However, these findings are highly volatile. The differences among the top 10 centrality values are so small that the inconsistent rankings is considered insignificant. The differences between the top 10 rankings can be explained by the randomness factor. Despite the inconsistency in the node centrality rankings, the overall patterns cannot be ignored. In both types of network (Random and Scale-Free), the most central nodes of Transfer-Walks, Transfer-Trails and Transfer-Paths tend to be the ones with high Degree, Betweenness and Closeness for the transfer centrality measures.

Regarding the serial and parallel centrality measures, the most central nodes tend to the ones with high Closeness and Eigenvector. Based on the simulation and correlation analysis showed that the Eigenvector Centrality can also be used for serial and parallel replication network flows. Furthermore, the Flow Betweenness and Random-Walks Betweenness are also applicable to transfer

network flows (Transfer-Walks, Transfer-Trails and Transfer-Paths).

The proposed transfer measures are derived from Freemans Betweenness Centrality, while the variation of replication measures are derived from Freemans Closeness Centrality. The serial and parallel replication findings matched the social influence process of the diffusion model by (Burt, 1987; Valente, 1996). The central nodes of the serial and the parallel replication flow favored those nodes that are well-positioned and Eigenvector. A well-positioned node is the one where the distance among all other nodes is the shortest, like Freemans Closeness Centrality. Though only the parallel replication type of network may be significantly influenced by Eigenvector Centrality. This is because being connected to other high Eigenvector centrality enables that node to propagate rapidly. The high Eigenvector Centrality nodes have high Degree which provide a great assistance in parallel diffusion.

7.4.4 Experiment Limitations

Due to the shortcomings of the hardware, the experiment environment could not handle large variables. That is, larger network size, higher power-law parameter, and higher linkage parameter, all of which contribute toward the network cohesion. As a result, the range of network cohesions produced by the specified variables were not even (Table 13 Exp: 11 to 20: Clustering Coefficient). Furthermore, to handle an increase in network size, it may be necessary to increase the simulation runs. In a large network, there are many more flow possibilities having an insufficient number of runs could produce results favoring a subset of nodes. Consider the example of throwing a die 3 times with results: 3, 5, 3. The result based on 3 throws is not enough to truly represent the even probability of a die. Furthermore, there are other centralities (e.g. Katz Centrality) that were not considered in the simulation experiment. Perhaps, those centrality measures might have a better description of the network. Unlike the typical centrality measures, Katz Centrality considers a total number of walks between a pair of nodes. This is similar to random-walks based measure without the randomness factor.

8 Conclusion

This chapter concludes the research by anwsering the research questions as defined in chapter 1. Furthermore, suggestions for the future research directions will be described in this chapter.

8.1 Research Questions

1. How does the flow process differ between different types of network or network settings?

   The flow process of a network behaves according to the specific network type. According to Borgatti (2005), networks are differentiated by the

node to node transmission (i.e. serial and parallel replication and transfer) and flow trajectory (i.e. Geodesic, Paths, Trails and Walks) (See: Table 1). Each type of network is uniquely different from the other. The transfer types of network are commonly applied to the passing of an object, while the different flow trajectories dictate how the flowing object should behave. In geodesic flow, the knowledge of the shortest path between nodes is assumed. While the flows along paths, trails, and walks behave randomly. Serial and parallel replication types of network are commonly applied to a duplication of data. The distinction between serial and parallel replication is that parallel replication produces duplicates for all adjacent nodes, whereas serial replication merely duplicates the information one node at a time. Similar to transfer networks, the geodesic, paths, trails, and walks of serial and parallel networks dictate how the replication processes behave.

The flow process of a network also depends on the weight of each relationship (edge). In a social network, this is referred to as tie strength. According to the work of Daly and Haahr (2009), the selection of people whom to transfer information to is not purely random. People are chosen according to the strength of the relationship (Tie strength). Tie strength is determined by the frequency of contact, relationship history, contact duration and the number of transactions. While the flow process of other types network may depend on other types of drivers (specific to the particular network). Based on the network structural analysis by Zio and Piccinelli (2010), flow process may follow a circuitous route regardless of the strong relationship strength.

Furthermore, the flow process is also influenced by the structural properties. Guzman et al. (2014) have captured size, scale-free parameter and clustering coefficient as the key properties of a network. Barabási and Bonabeau (2003) emphasized the distinctions between random and scale-free network, and how random network-theory cannot be implemented in a social network environment. The ego network from Burt (1987) is an example of how different power-law/linkage parameter can have a different impact on the flow process. High cohesion networks that have strong ties enables faster information flow and greater motivation, while clustered networks that have weak ties tend to spark novel ideas.

2. What possible approaches could be used to determine the centrality of networks that are not geodesic?

   The Freemans centralities (betweenness, closeness, and degree) (Freeman, 1978) and Bonacichs Eigenvector Centrality (Bonacich, 2007) were derived from a social network based on the premise that information flows along the shortest (geodesic) paths. However, according to the network classification of Borgatti (2005) and Newman (2005), there exists a variety of network that does feature flows along geodesic paths. Therefore, the above-mentioned centralities may not be suitable in determining the

centrality of networks that are non-geodesic.

The flow betweenness (Freeman et al., 1991) type of betweenness-like centrality specifically derived for the purpose of determining the centrality of both geodesic and non-geodesic paths. The concept of flow betweenness is based on the idea of maximum liquid flow a node can hold. Assuming the liquid flows freely in a piping system network, the amount of liquid flows via a junction (node) determines its importance. In such an example, the forking of liquid in the piping system is similar to the replication mechanisms. That being said, it may contribute to formulating centrality for replication networks and networks which follow paths and trails. Similarly, the random-walk betweenness (Noh and Rieger, 2004) is also for non-geodesic paths and specifically for walks. In random-walk betweenness, the major drawback of back and forth traversal is eliminated by counting multiple back and forth traversal as 1, to avoid biased results. Furthermore, the network influence can be captured by exposure or contagion diffusion model. The social influence of a node can be determined by three factors: relational, positional and central. The quickest influence diffusion is based on the network structure and the position of a node within the network.

3. How can simulation be used to determine the centrality of various network types?

   Simulation experimentation is able to simulate the various unique flow processes that other methods can not. The connectivity entropy and centrality entropy methods may be useful in validating the simulation models, but it lacks various test scenarios. The entropy method is designed based on the degree and closeness of the nodes, but it does not take into account the different flow types. The conventional mathematical solution is prone to mistake, hard to visualize and difficult to validate. Based on the finding of the research, it has been found that the conventional mathematical solution is difficult to picture the random node selection. And two, if problem arises, it would be a difficult process to determine where the problem lies.

   Furthermore, the use of simulation allows the flexibility to change in input parameters and quick output. The key point with every simulation model is the validity and reliability. The validation process involved a simulation run on the Florentine Families network. It showed that the results of the proposed centrality methods are in-line with the results of the entropy results. In terms of reliability, the overall patterns of the experimental results show that the central nodes of the transfer processes are correlated with high degree, betweenness, and closeness, while the central nodes of the replication processes are correlated with high closeness and eigenvector. The few inconsistencies throughout the different experiments can be explained by the factor of randomness. However, the randomness may also be caused by the trade-off made. The experimental runs were

set out to test different levels of network properties. First, certain actions (i.e. setting a smaller network size, low number of simulation runs, etc) were taken to ensure the obtained results are obtained within a reasonable amount of time. Second, in order to see the extreme effects, input settings may have been set beyond reasonable bounds, as a result, some of the produced data are not very fruitful. Additionally, the number of simulation runs should also increase in situations where the network size is large. All these factors may have affected with the reliability of the simulation.

4. How do network properties impact the outcomes of different centrality measures?

   The results of the different simulation experiments have posited that the network type and the existence of network cohesion has a high impact on the centrality outcomes. In both random and scale-free networks, the increase in the number of nodes (network size) directly reduces the total network cohesion. Conversely, the increase in the number of edges (power-law/linkage parameter) directly increases the total network cohesion. In a highly cohesive network, there are numerous paths to get from source S to target T. As result, each node is less significant in terms of connectivity and centrality, which implies that the centrality values of the top central nodes are the same. Therefore, a large set of key nodes is expected. Conversely, a low cohesive network consists of few edges, as such, each edge has a great impact on centrality and connectivity. As a result, one specific central node is expected. The randomness factor of the simulation experiments has produced results that do not show a strong correlation. Nevertheless, it is undeniable that the centrality measures are correlated in some ways. Based on all the simulation results, the random transfer types (T_Walks, T_Trails, T_Paths) produced central nodes that correlate with high degree, high betweenness and/or high closeness, while the replication types (S_Paths, S_Geodesic, S_Trails and P_Paths&Trails) of network are correlated with high closeness and eigenvector. These findings are in line with the findings of Borgatti (2005). Furthermore, the replication centrality findings reflect the diffusion by contagion model (Burt, 1987; Valente, 1996). Indeed, a rapid diffusion depends on the selection of the starting node, which is determined by the node position and direct contacts.

8.2 Future work

This research set out to explore the impact of different network properties on the various network centrality measures. Based on the related work findings, various sources indicate that network centrality and network flow are far more important than where the node is located in the network (position), how many links it has (links), or how in-between it is (hub). In the area of social network analysis (SNA), the network flow of such a network is heavily dependent on strong and weak ties, which others refer to as tie strength. This tie strength corresponds to the weighted network. In addition to this, directed and undirected networks

determine the flow paths. These two aspects were not explored as they are outside the scope of the research. Furthermore, the state of the information in the network flow is assumed to be immutable, but how does it look like when the state mutates over time or over each pass? Further, research could explore the possible effect of the above-mentioned factors on network flow and network centrality.

9 Appendix: Python Core Code

9.1 Script: Main Control

```
# This is the main run file, including creation of network and simulation run
# Created by: Zhipeng Luo
# Last Modify: 17-07-18

import classNode
import measuresTransfer
import measuresSerial
import measuresParallel
import validateMeasure
import plotGraph as pg

# declare simulation configurations
simRuns = 100
topN = 10

# declare plot types
plotType = [
        "t_degree",
        "t_walks",
        "t_trails",
        "t_paths",
        "t_betweenness",
        "t_closeness",
        "t_eigenvector",

        "s_trails",
        "s_paths",
        "s_geodesic",

        "p_paths_trails"
        ]
# declare connection types
networkType = ["ScaleFree", "Random"]
testMode = False
```

```
#initialize network
if not testMode:
    net = classNode.network
    print("create node objects...")
    classNode.createNetworkObjects(networkType[0])
    print("create node links...")
    classNode.setConnections(net)
    print("network generation completed...")
else:
    net = classNode.network
    net.append(classNode.node("0 - Pazzi", 1 ,"semi"))
    net.append(classNode.node("1 - Salviati", 2 ,"semi"))
    net.append(classNode.node("2 - Acciaiuol", 1 ,"semi"))
    net.append(classNode.node("3 - Medici", 6 ,"semi"))
    net.append(classNode.node("4 - Barbadori", 2 ,"semi"))
    net.append(classNode.node("5 - Castellan", 3 ,"semi"))
    net.append(classNode.node("6 - Peruzzi", 3 ,"semi"))
    net.append(classNode.node("7 - Strozzi", 4 ,"semi"))
    net.append(classNode.node("8 - Ridolfi", 3 ,"semi"))
    net.append(classNode.node("9 - Bischeri", 3 ,"semi"))
    net.append(classNode.node("10 - Tornabuon", 3 ,"semi"))
    net.append(classNode.node("11 - Guadagni", 4 ,"semi"))
    net.append(classNode.node("12 - Lambertes", 1 ,"semi"))
    net.append(classNode.node("13 - Albizzi", 3 ,"semi"))
    net.append(classNode.node("14 - Ginori", 1 ,"semi"))

    net[0].neighbores = [net[1]]
    net[1].neighbores = [net[0],net[3]]
    net[2].neighbores = [net[3]]
    net[3].neighbores = [net[1],net[2],net[4],net[8],net[10],net[13]]
    net[4].neighbores = [net[3],net[5]]
    net[5].neighbores = [net[4],net[6],net[7]]
    net[6].neighbores = [net[5],net[7],net[9]]
    net[7].neighbores = [net[5],net[6],net[9],net[8]]
    net[8].neighbores = [net[3],net[7],net[10]]
    net[9].neighbores = [net[6],net[7],net[11]]
    net[10].neighbores = [net[3],net[8],net[11]]
    net[11].neighbores = [net[10],net[13],net[9],net[12]]
    net[12].neighbores = [net[11]]
    net[13].neighbores = [net[3],net[14],net[11]]
    net[14].neighbores = [net[13]]

# determine network properties
print(classNode.networkProperty(net,testMode))
```

```
# get all simulation data
def simRun(n):
    for typ in plotType:
        data = []
        print(typ + " ..........................")
        if typ is "t_degree":
            measuresTransfer.degree(net)
            for node in net:
                data.append([node.id, node.degree])
        elif typ is "t_walks":
            measuresTransfer.walks(net, simRuns)
            for node in net:
                data.append([node.id, node.Transfer_Walks])
        elif typ is "t_trails":
            measuresTransfer.trails(net, simRuns)
            for node in net:
                data.append([node.id, node.Transfer_Trails])
        elif typ is "t_paths":
            measuresTransfer.paths(net, simRuns)
            for node in net:
                data.append([node.id, node.Transfer_Paths])
        elif typ is  "t_betweenness":
            measuresTransfer.betweenness(net)
            for node in net:
                data.append([node.id, node.betweenness])
        elif typ is "t_closeness":
            measuresTransfer.closeness(net)
            for node in net:
                data.append([node.id, node.closeness])
        elif typ is "t_eigenvector":
            measuresTransfer.eigenvector(net)
            for node in net:
                data.append([node.id, node.eigenvector])
        elif typ is "s_geodesic":
            measuresSerial.geodesic(net,simRuns)
            for node in net:
                data.append([node.id, node.closeness_Serial_Geodesic])
        elif typ is "s_paths":
            measuresSerial.paths(net, simRuns)
            for node in net:
                data.append([node.id, node.closeness_Serial_Paths])
        elif typ is "s_trails":
            measuresSerial.trails(net, simRuns)
            for node in net:
                data.append([node.id, node.closeness_Serial_Trails])
        elif typ is "p_paths_trails":
```

```
            measuresParallel.paths_trails(net)
            for node in net:
                data.append([node.id, node.closeness_Parallel_Paths])
        else:
            None

        # select top n results
        res = classNode.getTopNodes(data, n)
        for d in range(n):
            print(res[n-d-1][0])
        for d in range (n):
            print(res[n-d-1][1])

# plot the graph
pg.plotGraph(net,plotType[6])
pg.plotValidate(validateMeasure.validate(net))

# execute simulation experiements
simRun(topN)
```

9.2 Script: Class Node

```
# This is the  network file, including creation of node, formation of edges, det
# Created by: Zhipeng Luo
# Last Modify: 17-07-18

import random as r
import numpy as np
import math as ma
from operator import itemgetter

# set network properties
#total network size
size = 100
#random
meanLinks = 35
stdLinks = 0.7
#scale free
k = 5
kn = float(2.0)

#  declare network object
network = []

# class node
class node:
```

```
    def __init__(self, *args):
        if args[2] is "auto":
            self.id = args[0]
            self.links = args[1]
        elif args[2] is "semi":
            self.id = args[0]
            self.links = args[1]
        self.neighbores = []
        #Transfer
        self.degree = float(0)
        self.closeness = float(0)
        self.betweenness = float(0)
        self.eigenvector = float(0)

        self.Transfer_Paths = float(0)
        self.Transfer_Trails = float(0)
        self.Transfer_Walks = float(0)

        #Serial Replication
        self.closeness_Serial_Geodesic = float(0)
        self.closeness_Serial_Paths = float(0)
        self.closeness_Serial_Trails = float(0)
        #Parallel Replication
        self.closeness_Parallel_Paths = float(0)

# create node objects
def createNetworkObjects(netType):
    if netType is "Random":
        # return number of links for each node
        res = np.random.normal(meanLinks, stdLinks, size)
        res = sorted(res, key=float, reverse=True)
        for i in range(size):
            lnk = res[i]
            if lnk < 1:
                lnk = 1
            elif lnk >= size-1:
                lnk = size-1
            else:
                lnk = int(lnk)
            # create new node and form one edge with other node
            newNode = node(str(i), lnk, "auto")
            if len(network) > 0:
                while len(newNode.neighbores) is 0:
                    rand = r.randint(0,len(network)-1)
                    if len(network[rand].neighbores) < network[rand].links:
                        newNode.neighbores.append(network[rand])
```

```
                        network[rand].neighbores.append(newNode)
            network.append(newNode)
    elif netType is "ScaleFree":
        # return number of links for each node
        bunch = []
        for ki in range(k):
            links = ma.ceil(1/((ki+1)**kn)*size)
            #print(links)
            for l in range(links):
                bunch.append(ki+1);
        bunch.reverse()
        # create new node and form one edge with other node
        for i in range(size):
            newNode = node(str(i),bunch[i], "auto")
            if len(network) > 0:
                while len(newNode.neighbores) is 0:
                    viable = []
                    for nod in network:
                        if len(nod.neighbores) < nod.links:
                            viable.append(nod)
                    if len(viable) > 0:
                        select = r.choice(viable)
                        newNode.neighbores.append(select)
                        select.neighbores.append(newNode)
                    else:
                        pool = []
                        for nod in network:
                            for lk in range(nod.links):
                                pool.append(nod)
                        sel = r.choice(pool)
                        newNode.neighbores.append(sel)
                        sel.neighbores.append(newNode)
                        sel.links = len(sel.neighbores)
            network.append(newNode)

# setup links for nodes
def setConnections(net):
    for node in net:
        viable = []
        # get all possible node that can make connects
        if node.links > len(node.neighbores):
            for v_node in net:
                if v_node is not node:
                    viable.append(v_node)
        # set connections
        while node.links > len(node.neighbores):
```

```
            if len(viable) > 0:
                rand = r.randint(0,len(viable)-1)
                if viable[rand] not in node.neighbores and viable[rand].links >
                    node.neighbores.append(viable[rand])
                    viable[rand].neighbores.append(node)
                    viable.remove(viable[rand])
                else:
                    viable.remove(viable[rand])
            else:
                rand = r.randint(0,len(net)-1)
                if node is not net[rand]:
                    if net[rand] not in node.neighbores:
                        node.neighbores.append(net[rand])
                        net[rand].neighbores.append(node)
                        net[rand].links += 1



# determine cluster coefficent, size and power parameter
def networkProperty(net, test):
    CC_graph = float(0)
    for node in net:
        CC_node = float(0)
        if len(node.neighbores) >= 2:
            Nv = 0
            for neighNode in node.neighbores:
                for neighNeigh in neighNode.neighbores:
                    if neighNeigh in node.neighbores:
                        Nv += 1
            CC_node = Nv/(len(node.neighbores)*(len(node.neighbores)-1))
        CC_graph += CC_node
    CC_graph = CC_graph/len(net)
    if test:
        return [len(net), 0, CC_graph]
    else:
        return [len(net), k, CC_graph]

# return top n nodes
def getTopNodes(data, size):
    return sorted(data, key=itemgetter(1))[-size:]
```

9.3 Script: Transfer Measures

```
# This is the main transfer centrality file
# Created by: Zhipeng Luo
# Last Modify: 17-07-18
```

```
import itertools as it
import random as ra

# return length of geodesic path from start to end
def any_geodesic(net, start, end):
    queue = []
    queue.append([start])
    while len(queue) > 0:
        qCurrent = queue.pop(0)
        node = qCurrent[-1]
        # returns the first shortest path found
        if node is end:
            return qCurrent
        # check node neighbores and create new paths
        for neigh in node.neighbores:
            if neigh not in qCurrent:
                newPath=list(qCurrent)
                newPath.append(neigh)
                queue.append(newPath)

# return all the gedoesic paths from start to end
def multi_geodesic(net, start, end):
    queue = []
    total = []
    queue.append([start])
    while len(queue) > 0:
        qCurrent = queue.pop(0)
        node = qCurrent[-1]
        for i in range(len(node.neighbores)):
            # add new path sequence in queue
            if node.neighbores[i] not in qCurrent:
                newPath=list(qCurrent)
                newPath.append(node.neighbores[i])
                # end node operations
                if node.neighbores[i] is end:
                    if len(total) > 0:
                        # clear the total list based on header length
                        len_head = len(total[0])
                        if len_head > len(newPath):
                            total[:] = []
                            total.append(newPath)
                        # append when path with same lengths (shortest)
                        elif  len_head == len(newPath):
                            total.append(newPath)
                        else:
```

```
                        break
                    else:
                        total.append(newPath)
                else:
                    queue.append(newPath)
            # remove all imcomplete queuess where the length exceeds total[0] length
            if len(total) > 0:
                len_total = len(total[0])
                for q_obj in queue:
                    if len(q_obj) >= len_total:
                        queue.remove(q_obj)
            # return the geodesic paths
            if len(queue) is 0:
                return total

#-----------------------------------
# calculate eigenvector
def eigenvector(net):
    for node in net:
        #print("Eigenvector node start..." + str(node.id))
        calcEC = 0
        # add up all neighbore influences
        for neigh in node.neighbores:
            calcEC += neigh.links
        node.eigenvector = calcEC

# calculate betweenness
def betweenness(net):
    # get all possible combinations from start to end
    combinations = list(it.combinations(net, 2))
    all_g_combinations =[]
    for combo in combinations:
        all_g_combinations.append(multi_geodesic(net, combo[0], combo[1]))
    for node in net:
        calcBC = float(0)
        # calculate centrality of node
        for g_group in all_g_combinations:
            g_paths = 0
            g_path_node = 0
            g_path_ratio = float(0)
            first = g_group[0]
            head = first[0]
            tail = first[-1]
            if head is not node and tail is not node:
                g_paths = len(g_group)
                # count the amount of path contains node x
```

```
                for one_g in g_group:
                    if node in one_g:
                        g_path_node += 1
            if g_path_node is 0:
                g_path_ratio = float(0)
            else:
                g_path_ratio = g_path_node/g_paths
            calcBC += g_path_ratio
        node.betweenness = calcBC/((len(net)-1)*(len(net)-2)/2)

# calculate closeness
def closeness(net):
    # get all possible combinations from start to end
    combinations = list(it.combinations(net, 2))

    for node in net:
        totalNodeDistance = 0
        # add up all paths distances with node x
        for group in combinations:
            # get a object with geodesic path
            if node in group:
                totalNodeDistance += (len(any_geodesic(net,group[0],group[1]))-1
        # assign CC with normalization
        node.closeness = float((len(net)-1)/totalNodeDistance)

# calculate degree
def degree(net):
    for node in net:
        # assign DC with normalization
        node.degree = node.links/(len(net)-1)

# calculate paths
def paths(net, iterations):
    combinations = list(it.combinations(net, 2))
    for i in range(iterations):
        for oneComb in combinations:
            pool = [oneComb[0]]
            while True:
                last = pool[-1]
                possible = []
                # add 1 of neighbor node to sequence
                for neigh in last.neighbores:
                    if neigh not in pool:
                        possible.append(neigh)
                if len(possible) > 0:
                    select = ra.choice(possible)
```

```
                    pool.append(select)
                else:
                    break
                if pool[-1] is oneComb[-1]:
                    break
            # add count 1 to all between nodes
            if pool[-1] is oneComb[-1]:
                for node in pool:
                    if node is not pool[0] and node is not pool[-1]:
                        node.Transfer_Paths += 1
    # normalize
    for node in net:
        node.Transfer_Paths = node.Transfer_Paths/iterations/len(combinations)

#calculate trails
def trails(net, iterations):
    combinations = list(it.combinations(net, 2))
    #combinations = [[net[0],net[5]]]
    for i in range(iterations):
        #print("starting....." + str(i))
        for oneComb in combinations:
            #print([x.id for x in oneComb])
            pool = [[oneComb[0]]]
            while True:
                last = pool[-1][-1]
                possible = []
                #check for repeated edges
                for neigh in last.neighbores:
                    newStep = [last, neigh]
                    r_newStep = [neigh, last]
                    if newStep not in pool and r_newStep not in pool:
                        possible.append(newStep)
                #add edge randomly
                if len(possible) > 0:
                    select = ra.choice(possible)
                    if len(pool[0]) is 1:
                        pool = [select]
                    else:
                        pool.append(select)
                else:
                    break
                if pool[-1][-1] is oneComb[-1]:
                    break
            # calculate centrality of node
            if pool[-1][-1] is oneComb[-1]:
               for edge in pool:
```

```
                    if edge is not pool[-1]:
                        edge[1].Transfer_Trails += 1
    # normalize
    for node in net:
        node.Transfer_Trails = node.Transfer_Trails/iterations/len(combinations)


# calculate walks
def walks(net, iterations):
    combinations = list(it.combinations(net, 2))
    for i in range(iterations):
        for oneComb in combinations:
            walk = []
            walk.append(oneComb[0])
            while walk[-1] is not oneComb[1]:
                last = walk[-1]
                last2 = None
                possible = []
                # add 1 of neighbor node
                for neigh in last.neighbores:
                    possible.append(neigh)
                select = ra.choice(possible)
                 # if check back and forth
                if len(walk) > 1:
                    last2 = walk[-2]
                    if select is last2:
                        walk.remove(last)
                    else:
                        walk.append(select)
                else:
                    walk.append(select)
            # calculate centrality of node
            for node in walk:
                if node is not walk[0] and node is not walk[-1]:
                    node.Transfer_Walks += 1
    # normalize
    for node in net:
        node.Transfer_Walks = node.Transfer_Walks/iterations/len(net)
```

9.4 Script: Serial Replication Measures

```
# This is the main serial centrality file
# Created by: Zhipeng Luo
# Last Modify: 17-07-18

import random as ra
```

```
# calculate geodesic
def geodesic(net, iterations):
    for node in net:
        avgResult = float(0)
        for i in range(iterations):
            # add first node to queue
            pool = []
            pool.append([[node]])
            while len(pool) > 0:
                # get first node from the first sequence object
                current = pool.pop(0)
                base_set = [y for x in current for y in x]
                cAddon = []
                allPossible = []
                for cPack in current:
                    for cNode in cPack:
                        nPossible = []
                        for cNeigh in cNode.neighbores:
                            if cNeigh not in base_set:
                                # add possible neighbors
                                nPossible.append(cNeigh)
                        if len(nPossible) > 0:
                            allPossible.append(nPossible)
                # sort them by size of groups
                allPossible.sort(key = lambda group: len(group))
                while len(allPossible) > 0:
                    pCurrent = allPossible.pop(0)
                    select = ra.choice(pCurrent)
                    cAddon.append(select)
                    base_set.append(select)
                    for pGroup in allPossible:
                        if select in pGroup:
                            if len(pGroup) > 1:
                                pGroup.remove(select)
                            else:
                                allPossible.remove(pGroup)
                    allPossible.sort(key = lambda group: len(group))
                # check if finished
                if len(cAddon) > 0:
                    newPath = list(current)
                    newPath.append(cAddon)
                    pool.append(newPath)

                else:
                    avgResult += len(current)
```

```
        # calculate centrality for node
        node.closeness_Serial_Geodesic = (len(net)-1)/(avgResult/iterations)


# calculate Paths
def paths(net, iterations):
    for node in net:
        avgResult = float(0)
        for i in range(iterations):
            # add first node to queue
            pool = []
            pool.append([[node]])
            while len(pool) > 0:
                # get first sequence object
                current = pool.pop(0)
                base_set = [y for x in current for y in x]
                cAddon = []
                for cPack in current:
                    for cNode in cPack:
                        possible = []
                        # add all possible neighbor nodes to possible queue
                        for cNeigh in cNode.neighbores:
                            if cNeigh not in base_set:
                                possible.append(cNeigh)
                        if len(possible) > 0:
                            select = ra.choice(possible)
                            cAddon.append(select)
                            base_set.append(select)
                # check if finished
                if len(cAddon) > 0:
                    newPath = list(current)
                    newPath.append(cAddon)
                    pool.append(newPath)
                else:
                    avgResult += len(current)
        # calculate centrality of node
        node.closeness_Serial_Paths = (len(net)-1)/(avgResult/iterations)


# calculate trails
def trails(net, iterations):
    for node in net:
        for i in range(iterations):
            # add first node to queue
            pool = []
            pool.append([[[node]]])
```

```
            while len(pool) > 0:
                pCurrent = pool.pop(0)
                baseEdges = [y for x in pCurrent for y in x]
                baseNodes = []
                # add possible neighbors
                for be in baseEdges:
                    for bn in be:
                        if bn not in baseNodes:
                            baseNodes.append(bn)
                allAddons = []
                for bNode in baseNodes:
                    possible = []
                    for neigh in bNode.neighbores:
                        newEdge = [bNode, neigh]
                        r_newEdge = [neigh, bNode]
                        if newEdge not in baseEdges and r_newEdge not in baseEdg
                            possible.append(newEdge)

                    if len(possible) > 0:
                        select = ra.choice(possible)
                        allAddons.append(select)
                # check if finished
                if len(allAddons) > 0:
                    if len(pCurrent[0][0]) is 1:
                        newSequence = []
                        newSequence.append(allAddons)
                        pool.append(newSequence)
                    else:
                        newPath = list(pCurrent)
                        newPath.append(allAddons)
                        pool.append(newPath)
                else:
                    node.closeness_Serial_Trails += len(pCurrent)
        # calculate centrality of node
        node.closeness_Serial_Trails =(len(net)-1)/(node.closeness_Serial_Trails
```

9.5 Script: Parallel Replication Measures

```
# This is the geodesic centrality measure file
# Created by: Zhipeng Luo
# Last Modify: 17-07-18

# calculate geodesic  path and trail
def paths_trails(net):
    for node in net:
        # add starting node to queue
```

```
        queue = []
        queue.append([[node]])

        while len(queue) > 0:
            # get first node of the firs sequence object
            qCurrent = queue.pop(0)
            base_set = [y for x in qCurrent for y in x]
            addon_set = []
            # get all neighbor nodes
            for qPack in qCurrent:
                for qNode in qPack:
                    for neigh in qNode.neighbores:
                        if neigh not in base_set:
                            addon_set.append(neigh)
                            base_set.append(neigh)
            # check if finished
            if len(addon_set) > 0:
                # add new sequences to queue
                newQueueObj = list(qCurrent)
                newQueueObj.append(addon_set)
                queue.append(newQueueObj)
            else:
                # calculate centrality for node
                node.closeness_Parallel_Paths = float((len(net)-1)/len(qCurrent)
```

9.6 Script: Plot Network

```
# This is the ploy network file
# Created by: Zhipeng Luo
# Last Modified: 17-07-18


from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
import igraph as ig
import plotly.plotly as py
from plotly.graph_objs import *
import plotly.graph_objs as go




def plotValidate(result):

    names = [str(row[0]) for row in result]
    Hco = [row[1] for row in result]
    Hce = [row[2] for row in result]
```

```
    trace1 = go.Scatter(
            x = names,
            y = Hco,
            mode = 'lines+markers',
            name = 'Hco - (connectivity)'
            )
    trace2 = go.Scatter(
            x = names,
            y = Hce,
            mode = 'lines+markers',
            name = 'Hce - (centrality)'
            )

    layout=Layout(
            title = "Validation by Entropy",
            width=1000,
            height=600,
            showlegend=True,
            margin=Margin(
                t=50
                ),
            hovermode='closest',
            )

    data = [trace1, trace2]
    fig=Figure(data=data, layout=layout)
    plot(fig, filename='scatter-mode')



def plotGraph(net, measure):

    # get edges and nodes and labels
    group =[]
    Edges=[]
    labels=[]
    for i in range(len(net)):
        value = float(0)
        if measure is "t_degree":
            value = net[i].degree
        elif measure is "t_betweenness":
            value = net[i].betweenness
        elif measure is "t_closeness":
            value = net[i].closeness
```

```
    elif measure is "t_eigenvector":
        value = net[i].eigenvector
    elif measure is "t_paths":
        value = net[i].Transfer_Paths
    elif measure is "t_trails":
        value = net[i].Transfer_Trails
    elif measure is "t_walks":
        value = net[i].Transfer_Walks

    elif measure is "s_geodesic":
        value = net[i].closeness_Serial_Geodesic
    elif measure is "s_paths":
        value = net[i].closeness_Serial_Paths
    elif measure is "s_trails":
        value = net[i].closeness_Serial_Trails

    elif measure is "p_paths_trails":
        value = net[i].closeness_Parallel_Paths
    else:
        None
    labels.append(
            "ID: " + net[i].id + "<br>" +
            "Score: " + str(value) + "<br>"
            )
    group.append(value)
    for j in range(len(net)):
        if net[i] is not net[j]:
            if net[j] in net[i].neighbores:
                Edges.append((i,j))


G=ig.Graph(Edges, directed=False)

# generate node positions
layt=G.layout('kk', dim=3)
Xn=[layt[k][0] for k in range(len(layt))]# x-coordinates of nodes
Yn=[layt[k][1] for k in range(len(layt))]# y-coordinates
Zn=[layt[k][2] for k in range(len(layt))]# z-coordinates
Xe=[]
Ye=[]
Ze=[]

for e in Edges:
    Xe+=[layt[e[0]][0],layt[e[1]][0], None]# x-coordinates of edge ends
    Ye+=[layt[e[0]][1],layt[e[1]][1], None]
    Ze+=[layt[e[0]][2],layt[e[1]][2], None]
```

```
# draw edges
trace1=Scatter3d(
        x=Xe,
        y=Ye,
        z=Ze,
        mode='lines',
        line=Line(color='rgb(125,125,125)', width=1),
        hoverinfo='none'
        )
# draw nodes
trace2=Scatter3d(
        x=Xn,
        y=Yn,
        z=Zn,
        mode='markers',
        name='actors',
        marker=Marker(
                symbol='dot',
                size=10,
                color = group,
                colorscale='thermal',
                line=Line(color='rgb(50,50,50)', width=1.0)
                ),
        text=labels,
        hoverinfo='text'
        )

# setup layout
layout=Layout(
        title = measure,
        width=1000,
        height=600,
        showlegend=False,
        margin=Margin(
            t=50
            ),
        hovermode='closest',
        )

# plot
data=Data([trace1, trace2])
fig=Figure(data=data, layout=layout)
plot_url = plot(fig, filename='newGraph')
```

9.7 Script: Validate Measures

```
import itertools as it
import math as ma
import copy

#calculate entrophy of removal of each node
def validate(net):
    allChanges = []
    base = getHcoHce(net)
    allChanges.append(["-1", 0, 0])

    for i in range(len(net)):
        newNet = copy.deepcopy(net)

        for neigh in newNet[i].neighbores:
            neigh.neighbores.remove(newNet[i])
        newNet.remove(newNet[i])

        res = getHcoHce(newNet)
        allChanges.append([net[i].id, base[0]-res[0], base[1]-res[1]])

    #print([x for x in allChanges])
    return allChanges



#given the result of paths, calculate Hco, Hce
def getHcoHce(net):
    combinations = list(it.combinations(net, 2))
    allPathsResults = []
    totalPathsCounts  = 0

    for node in net:
        pathCount = getPathsFromNodeV(net, node, combinations)
        #pathCount = getPathsFromNodeV2(net, node, combinations)
        allPathsResults.append([node, pathCount])
        totalPathsCounts += pathCount

    Hco = float(0)
    Hce = float(0)

    for res in allPathsResults:
        if len(res[0].neighbores) > 0:
            Hco += (len(res[0].neighbores)/(2*len(net)))*(ma.log2(len(res[0].nei
            Hce += (res[-1]/totalPathsCounts)*(ma.log2(res[-1]/totalPathsCounts)
```

```
        else:
            Hco += 0
            Hce += 0

    return [Hco*-1, Hce*-1]

# get all the different possible paths from node v
def getPathsFromNodeV(net, node, startEnd):
    nodeVCount = 0
    nodeCombo = []
    for combo in startEnd:
        if node in combo:
            if node is combo[0]:
                nodeCombo.append(combo)
            else:
                nodeCombo.append([combo[-1],combo[0]])
    for combo in nodeCombo:
        pool = [[combo[0]]]
        while len(pool) > 0:
            current = pool.pop(0)
            last = current[-1]
            for neigh in last.neighbores:
                if neigh not in current:
                    newPath = list(current)
                    newPath.append(neigh)
                    pool.append(newPath)
            if last is combo[1]:
                nodeVCount += 1
    return nodeVCount


# get all the geodesic paths from node v
def getPathsFromNodeV2(net, node, startEnd):
    nodeVCount = 0
    nodeCombo = []
    for combo in startEnd:
        if node in combo:
            if node is combo[0]:
                nodeCombo.append(combo)
            else:
                nodeCombo.append([combo[-1],combo[0]])
    for combo in nodeCombo:
        bests = []
        pool = [[combo[0]]]
        while len(pool) > 0:
            current = pool.pop(0)
```

```
            last = current[-1]

            if last is combo[1]:
                if len(bests) > 0:
                    if len(current) < len(bests[0]):
                        bests[:] = []
                        bests.append(current)
                    elif len(current) is len(bests[0]):
                        bests.append(current)
                else:
                    bests.append(current)
            else:
                for neigh in last.neighbores:
                    if neigh not in current:
                        newPath = list(current)
                        newPath.append(neigh)
                        pool.append(newPath)
        nodeVCount += len(bests)
    return nodeVCount
```

References

Bandini, S., Manzoni, S., and Vizzari, G. (2009). Agent based modeling and simulation: an informatics perspective. *Journal of Artificial Societies and Social Simulation*, 12(4):4.

Barabási, A.-L. (2009). Scale-free networks: A decade and beyond. *Science*, 325(5939):412–413.

Barabási, A.-L. and Bonabeau, E. (2003). Scale-free networks. *Scientific american*, 288(5):60–69.

Bonacich, P. (2007). Some unique properties of eigenvector centrality. *Social networks*, 29(4):555–564.

Borgatti, S. and Halgin, D. (2011). Network theorizing. 22.

Borgatti, S. P. (1995). Centrality and aids. *Connections*, 18(1):112–114.

Borgatti, S. P. (2005). Centrality and network flow. *Social Networks*, 27(1):55 – 71.

Borgatti, S. P. (2006). Identifying sets of key players in a social network. *Computational & Mathematical Organization Theory*, 12(1):21–34.

Borgatti, S. P. and Everett, M. G. (2006). A graph-theoretic perspective on centrality. *Social networks*, 28(4):466–484.

Borgatti, S. P., Mehra, A., Brass, D. J., and Labianca, G. (2009). Network analysis in the social sciences. *Science*, 323(5916):892–895.

Brautbar, M. and Kearns, M. (2011). A clustering coefficient network formation game. In *International Symposium on Algorithmic Game Theory*, pages 224–235. Springer.

Burt, R. S. (1987). Social contagion and innovation: Cohesion versus structural equivalence. *American journal of Sociology*, 92(6):1287–1335.

Cook, K. S. and Emerson, R. M. (1978). Power, equity and commitment in exchange networks. *American sociological review*, pages 721–739.

Crowston, K., Howison, J., and Wiggins, A. (2010). Validity issues in the use of social network analysis for the study of online communities. *Retrieved November*, 7:2002.

Daly, E. M. and Haahr, M. (2009). Social network analysis for information flow in disconnected delay-tolerant manets. *IEEE Transactions on Mobile Computing*, 8(5):606–621.

De Laat, M. (2002). Network and content analysis in an online community discourse. In *Proceedings of the conference on computer support for collaborative learning: Foundations for a CSCL community*, pages 625–626. International Society of the Learning Sciences.

De Meo, P., Ferrara, E., Fiumara, G., and Ricciardello, A. (2012). A novel measure of edge centrality in social networks. *Know.-Based Syst.*, 30:136–150.

Enriquez, J. G. (2010). Fluid centrality: a social network analysis of socialtechnical relations in computermediated communication. *International Journal of Research & Method in Education*, 33(1):55–67.

Estrada, E., Higham, D. J., and Hatano, N. (2009). Communicability betweenness in complex networks. *Physica A: Statistical Mechanics and its Applications*, 388(5):764–774.

Freeman, L. (2004). The development of social network analysis. *A Study in the Sociology of Science*, 1.

Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239.

Freeman, L. C., Borgatti, S. P., and White, D. R. (1991). Centrality in valued graphs : A measure of betweenness based on network flow.

Granovetter, M. S. (1973). The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380.

Grassi, R. (2010). Vertex centrality as a measure of information flow in italian corporate board networks. *Physica A: Statistical Mechanics and its Applications*, 389(12):2455–2464.

Guzman, J. D., Deckro, R. F., Robbins, M. J., Morris, J. F., and Ballester, N. A. (2014). An analytical comparison of social network measures. *IEEE Transactions on Computational Social Systems*, 1(1):35–45.

Iribarren, J. L. and Moro, E. (2011). Affinity paths and information diffusion in social networks. *Social networks*, 33(2):134–142.

Landherr, A., Friedl, B., and Heidemann, J. (2010). A critical review of centrality measures in social networks. *Business & Information Systems Engineering*, 2(6):371–385.

Law, A. M. (2003). How to conduct a successful simulation study. In *Proceedings of the 35th Conference on Winter Simulation: Driving Innovation*, WSC '03, pages 66–70. Winter Simulation Conference.

McPherson, M., Smith-Lovin, L., and Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444.

Newman, M. J. (2005). A measure of betweenness centrality based on random walks. *Social Networks*, 27(1):39 – 54.

Nieminen, J. (1974). On the centrality in a graph. *Scandinavian journal of psychology*, 15(1):332–336.

Noh, J. D. and Rieger, H. (2004). Random walks on complex networks. *Phys. Rev. Lett.*, 92:118701.

Ortiz-Arroyo, D. and Hussain, D. M. A. (2008). An information theory approach to identify sets of key players. In Ortiz-Arroyo, D., Larsen, H. L., Zeng, D. D., Hicks, D., and Wagner, G., editors, *Intelligence and Security Informatics*, pages 15–26, Berlin, Heidelberg. Springer Berlin Heidelberg.

Powell, W. (2003). Neither market nor hierarchy. *The sociology of organizations: classic, contemporary, and critical readings*, 315:104–117.

Seidman, S. B. (1983). Network structure and minimum degree. *Social networks*, 5(3):269–287.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Stephenson, K. and Zelen, M. (1989). Rethinking centrality: Methods and examples. *Social networks*, 11(1):1–37.

Travers, J. and Milgram, S. (1969). An experimental study of the small world problem. 32:425–443.

Valente, T. W. (1996). Network models of the diffusion of innovations. *Computational & Mathematical Organization Theory*, 2(2):163–164.

Wasserman, S. and Faust, K. (1993). Social network analysis methods and applications. 8.

Willer, D. (1992). Predicting power in exchange networks: a brief history and introduction to the issues. *Social Networks*, 14(3-4):187–211.

Zio, E. and Piccinelli, R. (2010). Randomized flow model and centrality measure for electrical power transmission network analysis. *Reliability Engineering and System Safety*, 95(4):379–385.

Figure 14: This table describes the correlation values between two centrality rankings based on the results of Exp: 6 to 10. The number represents the total number of nodes that exists within the two rankings. Note: Value 10 means all the notes are exist in both rankings, while 0 means two rankings have no node in common.

Exp: 06	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	7	7	4	8	8	8	2	2	2	3
T_Walks		-	9	7	8	9	9	4	4	4	5
T_Trails			-	7	8	9	9	4	4	4	5
T_Paths				-	6	6	6	3	3	3	3
T_Betweenness					-	9	9	3	3	3	4
T_Closeness						-	10	4	4	4	5
T_Eigen							-	4	4	4	5
S_Trails								-	9	9	7
S_Paths									-	10	8
S_Geodesic										-	8
P_P&T											-

Exp: 07	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	10	10	10	8	7	10	3	2	2	5
T_Walks		-	10	10	8	7	10	3	2	2	5
T_Trails			-	10	8	7	10	3	2	2	5
T_Paths				-	8	7	10	3	2	2	5
T_Betweenness					-	6	8	4	2	2	4
T_Closeness						-	7	4	4	4	5
T_Eigen							-	3	2	2	5
S_Trails								-	7	5	4
S_Paths									-	5	3
S_Geodesic										-	2
P_P&T											-

Exp: 08	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	5	4	5	4	5	7	2	0	0	0
T_Walks		-	7	4	4	3	6	1	0	0	0
T_Trails			-	4	3	3	4	1	0	0	0
T_Paths				-	5	3	4	1	0	0	0
T_Betweenness					-	6	4	1	0	0	0
T_Closeness						-	4	2	0	1	1
T_Eigen							-	1	0	0	0
S_Trails								-	2	1	1
S_Paths									-	5	5
S_Geodesic										-	10
P_P&T											-

Exp: 09	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	9	9	9	6	10	9	1	0	0	0
T_Walks		-	9	9	6	9	9	1	0	0	0
T_Trails			-	9	6	9	10	1	0	0	0
T_Paths				-	7	9	9	1	0	0	0
T_Betweenness					-	6	6	3	0	0	0
T_Closeness						-	9	1	0	0	0
T_Eigen							-	1	0	0	0
S_Trails								-	1	1	1
S_Paths									-	10	10
S_Geodesic										-	10
P_P&T											-

Exp: 10	*T_Degree*	*T_Walks*	*T_Trails*	*T_Paths*	*T_Betweenness*	*T_Closeness*	*T_Eigen*	*S_Trails*	*S_Paths*	*S_Geodesic*	*P_P&T*
T_Degree	-	9	9	9	6	10	9	1	0	0	0
T_Walks		-	9	9	6	9	9	1	0	0	0
T_Trails			-	9	6	9	10	1	0	0	0
T_Paths				-	7	9	9	1	0	0	0
T_Betweenness					-	6	6	3	0	0	0
T_Closeness						-	9	1	0	0	0
T_Eigen							-	1	0	0	0
S_Trails								-	1	1	1
S_Paths									-	10	10
S_Geodesic										-	10
P_P&T											-