Link Weight Tolerance

A study of betweenness centrality and data transmission in complex networks

Deeksha Pothirajan

TU Delft

Link Weight Tolerance

A study of betweenness centrality and data transmission in complex networks

by

Deeksha Pothirajan

to obtain the degree of Master of Science in Embedded Systems (Software and Networking)
at the Delft University of Technology,
to be defended publicly on Monday August 24, 2020 at 10:00 AM.

Student number:	4813472	
Thesis committee:	Ir. R. Noldus,	TU Delft, supervisor
	Dr. ir. J. S. S. M. Wong,	TU Delft
	Dr. M. Kitsak,	TU Delft

TU Delft

Abstract

Links play a significant role in the functioning of a complex network. The aim of this thesis is to study the links in a weighted network by introducing two new concepts. The link betweenness centrality of a link is defined as the fraction of shortest paths between all pairs of nodes in a graph that traverses that link. Although link betweenness is a widely known measure that characterizes the link, we introduce the concept, link weight tolerance, to understand the extent to which the weight of the link can be increased or decreased such that the shortest paths in the graph are unaffected, therefore the link betweenness of the links remain the same. We develop a method to generate the positive and negative tolerance of a link. We use examples to illustrate the algorithm and discuss the results. Prior to introducing this concept, in addition to surveying existing network theory measures, we also analyse the metric, betweenness centrality and describe the methods used to generate weighted and unweighted random graphs. To extend the concept of link betweenness, we introduce the second concept, link tension. Link tension provides the information related to the ability of the link to handle transmission of data and shows us the links that are important in a network.

Keywords: Weighted networks, link betweenness, link tolerance, link tension

Preface

Deeksha
Pothirajan
Delft

This thesis is written as completion to the Master of Science, at the Delft University of Technology. From November 2019, I have had the opportunity to conduct research on the topic, link weight tolerance. I have experienced this period as very thought-provoking and unique. This journey of nine months was a steep learning curve and gave me an opportunity to gather a lot of skills.

I would like to thank my supervisor from the University Ir. Rogier Noldus, whose valuable insights and directions gave me needful guidance to complete the research and write this thesis. His wise counsel and constant support has helped me in the journey of this thesis. Although the research felt difficult in the beginning, he was always available and willing to answer my queries. It has been a pleasure working with him.

My parents and sister deserve a particular note of thanks: their continuous encouragement and kind words have, as always, served me well. I would also like to thank my friends for their motivation and never-ending patience.

Contents

Symbols and Notations

G	Graph
$\mathcal{N}$	Set of Nodes
$\mathcal{L}$	Set of Links
n	Number of nodes
m	Number of links
A	Adjacency matrix
a_{ij}	Entry of an adjacency matrix, for corresponding node *i* and node *j*
B	Incidence matrix
W	Weighted Adjacency matrix
w_{ij}	Weight of a link between node *i* and node *j*
C	Clustering coefficient
C_w	Weighted Clustering coefficient
K_n	Complete graph with *n* nodes
C_B	Node Betweenness centrality
D	Degree Distribution
k	Degree of a node
p	Probability
$P(k)$	Probability of a node with degree *k*
Q	Modularity
K	Mean degree of a graph
G^c	Complement of a graph
s_i	Strength of a node *i*
C_n	Cycle graph with *n* nodes
S_n	Star graph with *n* nodes
X	Shortest path matrix
Z_p	Positive link weight tolerance
P	Positive link weight tolerance matrix for link *l*
ZP	Positive link weight tolerance matrix for all the links in a graph
Z_n	Negative link weight tolerance
N	Negative link weight tolerance matrix for link *l*
ZN	Negative link weight tolerance matrix for all the links in a graph
T_l	Link Tension of link *l*
ρ	Assortativity
ρ_D	Degree Assortativity
ρ_w	Weighted Assortativity coefficient
π_i	Neighbourhood of a node *i*

1 Introduction

Various models in nature and society can be visualized as a graph network. A graph is simply a representation of real-world system as a mathematical model, where quest for a path between two nodes in a graph is few of the problems that is usually observed. The criteria for optimality are quite often evaluated in terms of the weights that are associated with the links of the graph. When data flows through the network, it causes load on each of the nodes and links called betweenness. The betweenness of a node or a link depends on the shortest path between each node pair in a graph where, the shortest paths rely on the link set and the link weights. When a link weight changes, this may affect the shortest path between the node pairs. Each link has a tolerance zone and the link weight may vary within the tolerance boundaries without affecting the shortest path between the node pairs.

Nodes have always been the most common entity to be studied in a network compared to the links. Identifying and understanding critical nodes and links is an important issue. Based on the location and who they are connected with, nodes and links can have different importance. Studying the importance of links in a network can help in identification of links that could be susceptible or vulnerable to attacks. In this thesis therefore, we introduce two new concepts related to links in a weighted network: weight tolerance of a link and link tension.

1.1. Research Question

The aim of the thesis is to define and develop a method to determine the link weight tolerance for a single link. The link weight tolerance for a single link indicates how much the weight of that link may vary without affecting the shortest paths for all node pairs. The thesis is developed by building a strong graph theoretic study by constructing various models of random graphs and obtaining various distribution of few graph measures. As discussed earlier, links in a weighted network are least explored. A change in the link weight may affect a shortest path in the network, therefore affecting its link betweenness measure. To understand the extent to which the weight of a link can be changed to maintain the same shortest paths, brings us to the following research questions:

- How can we determine the positive and negative link weight tolerance for a single link?
- What is the relation between link betweenness and link weight tolerance? When we have the link betweenness, does that tell us how weight-tolerant that link is.

Further, to understand the transmission of data through links in a weighted network, we define a new metric, link tension. Link tension is the product of link betweenness and weight of a link, which brings us to the secondary research question

- What does the measure link tension tell us about a link ?

1.2. Thesis Outline

Chapter 2 contains a review and a discussion of few graph measures from complex network theory for unweighted and weighted graphs. This chapter also covers the previous work done on the measures including, betweenness centrality. Chapter 3 is about graph theory, which introduces various terminology related to a graph. The first section, Section 3.1, gives a formal definition of a graph and types of graphs. In section 3.2, types of regular graphs is discussed. The discussion of regular graphs is followed by directed and undirected graphs, weighted and non-weighted graphs in section 3.3 and 3.4 respectively. Section 3.5 is an exposition of a few other graph measures, which characterizes the graph. In section 3.6, three classical random graph models - Erdos Renyi random graph, Small world graph, Scale free graphs are reviewed prior to their generation which is discussed in chapter 5. Chapter 4 provides an extensive description of the measure betweenness centrality. Section 4.1 provides the definition for node betweenness. Section 4.2 explains link betweenness and section 4.3 describes betweenness centrality in weighted networks. In Section 4.4, variants of betweenness centrality are discussed. Its application in various fields are also discussed. Chapter 5 is dedicated to the generation of graphs. This chapter has three main sections describing the construction of three important graph models as three types - unweighted graphs, directed graphs and weighted graphs. Alongside its construction various distributions for the three classes of graphs are also generated. Chapter 6 introduces a new concept, link weight tolerance. This chapter describes the algorithm used to obtain the weight tolerance of a link in a graph and uses an example to describe it. Chapter 7 introduces another new concept - link tension. A formal definition is provided for the metric. Distribution for this metric is obtained for all three classes of the graph. Lastly, Chapter 8 provides the conclusions which consists of the contributions made through this thesis and the possible future work.

2

Literature Survey

The purpose of this chapter is to provide some background and insight into the metrics of non-weighted and weighted complex networks, with emphasis on its existing work alongside its relationship between the topology and dynamical behavior of such complex networks. Betweenness centrality is one such measure that is often used in social and computer communication networks to analyze and understand the behaviour of the network. In this chapter, along with betweenness centrality, a few other metrics are also discussed.

Complex networks come under the territory of graph theory and have been extensively studied in the past few years. In the past decade, research in complex networks has taken various directions and has witnessed massive growth as networks with regular, irregular shapes and various structures, sizes with thousands to millions of nodes have been analyzed. Watts and Strogatz's (WS) interpretation on small world networks which was published in 1998 [13] followed by Barabási and Albert's work on scale-free networks [12] have paved the way for studies on properties of real world networks. Some of the real world networks include the World Wide Web, neural networks, metabolic networks, social networks, telephony networks and transportation networks. A set of unexpected results have been obtained when networks from areas were compared and analysed, one of the significant issues is related to the structure of the network. Research has been conducted to define new measures and concepts to distinguish the topology of real world networks. To execute this, unifying and common principles besides the statistical properties of the real world networks were identified. Correlations in node degree also helps in distinguishing real world networks which can be achieved with shortest paths between any two nodes. Degree of a node is the number of links that are connected to it from the other nodes. For a node *i*, its degree can be given as

$$k_i = \sum_{i,j \in N} a_{ij} \tag{2.1}$$

where

$$a_{ij} = \begin{cases} 1, & \text{if a link exists between nodes } i \text{ and node } j \\ 0, & \text{otherwise} \end{cases}$$

where a_{ij} is an element of adjacency matrix *A* which is used to represent a finite graph (a graph with finite number of nodes and links) *G*.

Probability distribution of these degree over the entire network is the degree distribution (*D*) of the graph. Each node has various connections. The degree of a node tells us whether it is central or not, i.e. whether the node is more connected or not. However, nodes with smaller degree could also be crucial as such nodes could connect different regions of a network. Network characterization is affected by the most central nodes making their identification necessary. Network modelling has become significant after such empirical findings, leading to modelling of network growth in real world topology to have a better understanding of its evolution and dynamic behaviour. The distribution function P(*k*) characterizes the spread in the node degree which provides a probability for a node with *k* links that was selected at random. Maximum of the nodes in a random graph have the same degree approximately similar to the average degree *k* due to the random placement of the links. For random graphs in particular, the degree distribution shows a Poisson distribution making a peak at *k* (average degree). But the degree distribution for some other complex networks like the internet, World Wide Web deviates from the Poisson distribution to a power law tail. For a Watts Strogatz small world model, each node has the same degree *k* given *p*=0. When *p* is non zero, a disorder is introduced in the network which in turn broadens the degree distribution but the average degree remains equal to *k*.

In some models [22], [16], [17], [18], there is no systematic method for analysing degree distribution as they assume the node number with degree to be continuous. For networks with multiple links and loops, Bollobas *et al.* [24] provided a rigorous method to solve degree distribution. However, the method is applicable only for networks with multiple links and loops. For uncorrelated networks, its statistical properties are determined by the degree distribution. But, in several cases of real world networks, they are correlated, making introduction of conditional probability $P(k' \mid k)$ necessary [25]. This conditional probability can be defined as the probability of a link from a node with degree *k* pointing to a node of degree *k'*.

In social networks, each link has a particular strength and weighted links carry essential information related to a network. The weight of a link has the potential to affect the connectivity of a network which is further explored through this thesis. Cases of complex networks where their link weights are known, helps to establish a better overview of the system. Weight or cost of a link is its associated numerical value which aids in representing a graphical structure. Yook *et al.* [65] presented a weighted network model where preferential attachment drives the connection weights, connectivity of the nodes and the structure of the network. A weighted evolving network model was proposed by Zhang *et al.* It [66] suggests assignment of stochastic weights to the links. For a weighted network, the degree of a node (k_i) is protracted to weighted degree or strength (s_i) of a node which is the sum of the weights of all links incident to node *i*.

$$s_i = \sum_{i \in \pi(i)} w_{ij} \tag{2.2}$$

where w_{ij} is the weight of the link between node *i* and node *j*. The connectivity and weight of the links is taken into account by this metric. The degree distribution of a weighted network is similarly extended to strength distribution P(*s*), which can be defined as the probability of a node that its strength is *s*. Barrat *et al.* in [74] indicates that the strength distribution also obeys power law, $P(s) \sim s^{-a}$, where a is a constant.

Watts and Strogatz [13] introduced the concept of clustering coefficient (*C*) in 1998 to "quantify the structural properties" of a graph which can be further defined as the measure of the extent to which the nodes tend to cluster together. Let us consider a graph *G=(N,L)* where *N* is the set of all the nodes in the graph and *L* is the set of all the links $(x \sim y)$ in the graph. Let us define three nodes *x, y* and *z* where $(x \sim y)$ belongs to *L* and $(y \sim z)$ belongs to *L*, the clustering coefficient gives the likeliness that $(x \sim z)$ also belongs to *L*. Newman defined another version of this metric known as the global clustering coefficient [42]. Triplets of nodes make up this metric. This triplet is formed by three nodes that are connected either to two or three links that are undirected. Clustering coefficient is the ratio of the number of triplets that are connected by three links to the total number of the triplets. Luce and Perry (1949) made the first attempt to measure this metric [44].

$$C = \frac{3xNumberofTriangles}{NumberofTriplets} \tag{2.3}$$

On the other hand, the local clustering coefficient is developed on the basis of local density [45] [46]. To determine whether a graph exhibits small world network properties, Watts and Strogatz used this metric to measure it. The clustering coefficient in random graphs is *C*=*p* when the links are distributed at random. In comparison to the random networks, the clustering coefficient of the real world networks is higher [47]. In random networks, probability that two neighbours of a particular node are connected is equal to the probability that two nodes are selected at random. Clustering coefficient in the case of regular lattice (*p*=0) depends only on the topology and not on the size of the lattice. Barrat in 2000 [14] introduced a different definition of *C* based on the dependence of clustering coefficient on *p*. The definition of clustering coefficient *C*(*p*) is described in [14] as the ratio of number of links between the neighbours of a node to the average number of links possible between those neighbours.

Variants of Weighted Clustering Coefficient	Author
C^L	Lopez-Fernandez *et al.* [69]
C^B	Barrat *et al.* [74]
C^Z	Zhang *et al.* [67]
C^O	Onnela *et al.* [70]
C^S	Serrano *et al.* [71]
C^H	Holme *et al.* [72]

Table 2.1: List of variants of weighted clustering coefficients

Although clustering coefficient have been defined only for non-weighted networks, few authors have generalized clustering coefficient to take link weights into consideration. Table 2.1 lists the variants of weighted clustering coefficients and their respective authors. Lopez-Fernandez *et al.* [69] in 2004 defined a clustering coefficient for weighted networks without normalizing the weights.

$$C^L_{w,i} = \sum_{j,k\in\pi(i)} \frac{w_{jk}}{k_i(k_i-1)} \tag{2.4}$$

where w_{jk} is the weight of the link connecting the nodes *j* and *k*, k_i is the degree of node *i*. Barrat *et al.* (2004) [74] proposed a formula which was used for scientific collaboration networks and airports. The weights in [74]'s formula is not normalized.

$$C^B_{w,i} = \frac{1}{s_i(k_i-1)} \sum_{j,k} \frac{w_{ij}+w_{ik}}{2}(a_{ij}a_{jk}a_{ki}) \tag{2.5}$$

where $\frac{w_{ij}+w_{ik}}{2}$ is the average of the weights of the links between node *i* and its neighbours *j* and *k*. In the definition provided by Zhang *et al.* (2005) [67], the weights used are normalized.

$$C^Z_{w,i} = \frac{\sum_j \sum_k w_{ij}w_{jk}w_{ki}}{(\sum_j w_{ij})^2 - \sum_j (w_{ij})^2} \tag{2.6}$$

As per Onnela *et al.* (2005) [70], the definition of clustering coefficient is,

$$C^O_{w,i} = \frac{w_{ij}w_{jk}w_{ki}^{\frac{1}{3}}}{k_i(k_i-1)} \tag{2.7}$$

where $(w_{ij}w_{jk}w_{ki})^{\frac{1}{3}}$ is the intensity of the triangle *ijk*. Geometric mean of the link weights is put to use in this formula. A different version of Lopez-Fernandez formula was proposed by Li *et. al.* in 2005 [73]. In the Serrano *et al.*'s (2006) version [71],

$$C^S_{w,i} = \frac{\sum_j \sum_k w_{ij}w_{ik}a_{kj}}{(s_i)^2(1-Y_i)} \tag{2.8}$$

where $Y_i = \sum_j (\frac{w_{ij}}{s_i})^2$ is named as the disparity. Similar to non-weighted clustering coefficient, this formula has a probabilistic interpretation as it makes use of average weights along with the degree of the node. Holme *et al.* [72] proposed a similar definition as Zhang *et al.* [67] except, in [72], it is divided by max(w_{ij}) i.e.

$$C^H_{w,i} = \frac{\sum_j \sum_k w_{ij}w_{jk}w_{ki}}{max(w_{ij}) \sum_j \sum_{k\neq j} w_{ij}w_{ik}} \tag{2.9}$$

Newman and Girvan [26] introduced the definition of modularity (*Q*) which is used to measure the strength of modules of a network. Modularity is the fraction of the links which comes under a given group subtracting the likely fraction if the links were randomly distributed with a defined probability *p*. Modularity defined by Newman and Girvan is to evaluate community structures in binary networks. The modules could be of the form of graphs, clusters or communities. The value of modularity could be either positive or negative where in positive and large values of modularity indicate a presence of community structure. The partitioning is better when the value of modularity is large due to the deviation from the null case. This problem has been addressed by several authors who have proposed various optimization heuristics [28], [29], [30], [31] as the the number of partitions grow exponentially typically equal to the Bell or exponential numbers [32]. Modularity optimization is a non-deterministic polynomial-time hard (NP-hard) problem [27]. Modularity optimization tends to provide only single partition which is not sufficient for multi-scale systems and it fails to detect smaller communities since it undergoes a resolution limit. This resolution limit does not allow all modules to be compatible with the system architecture.

As real systems are expressed better through weighted networks, it is necessary for the community structures to take link weights into consideration. Newman suggested that modularity can be generalized to weighted networks [82] in the following way:

$$Q^w = \frac{1}{2w}\sum_{ij}[w_{ij} - \frac{s_i s_j}{2w}\delta[c_i c_j]] \tag{2.10}$$

where w is the summation of all the link weights in the network and s_i, s_j are the strength of nodes *i, j* respectively. The nodes *i, j* belong to the community c_i, c_j respectively. A structure is called a community when it consists of a group of nodes that are closely connected to each other but sparsely connected to other densely connected groups in the network [83]. The formula 2.10 takes into account link existence, describing the community in weighted networks. It can be interpreted from formula 2.10 that, within the group the relations of nodes are close but between the groups, the relations of nodes are distant. Through generalized modularity and community structure, definitive divisions of networks into communities becomes possible.

When high degree nodes in a network are connected to various other nodes with high degree and low degree nodes are connected to other low degree nodes, the network is said to be assortative. Newman *et al.* introduced assortativity for undirected and non-weighted graphs in [52]. When high degree nodes are connected to low degree nodes and low degree nodes are connected to higher degree nodes, the network is termed as disassortative. Degree assortativity (ρ_D) is expressed in a scalar form which falls in the range $[-1 \le \rho \le 1]$. Information about robustness, structure and dynamic behaviour of a network is provided by this metric.

Chang *et al.* [78] studied assortativity for weighted networks. Chang *et al.* in [78] used the strength of the node for defining assortativity of weighted networks. This indicates that the nodes with similar or opposing strength tend to bond with one another. Leung and Chu [79] also studied assortativity for weighted networks and defined a definition for it, ρ_w. As observed in [79], weighted assortativity and assortativity for the same network when weights are removed can be different. For a single node, distribution of link weight was studied by Wang *et al.* [80].

There are not many studies that have proposed a quantity which can measure directly the correlations of degree while including link weight. Leung *et al.* in [81] introduces weight evolution model where they define the weighted assortativity coefficient, ρ_w which can measure the tendency of having a high weighted link between two nodes having similar degrees. This model also takes into consideration the nonlinear growth of number of links, strength preferential attachment and evolution of weights in exiting links.

Centrality is another important aspect in social networks analysis, betweenness being the most prominent measure among them which was introduced by Anthonisse (1971) [64] and Freeman (1977) [33]. Borgatti and Everett (2006) [62] described betweenness as a measure of mediation. Among the few centrality measures, betweenness centrality is one of the important classes which is used to measure the extent to which a node comes on the path between other nodes. Freeman introduced the simplest and popular form of betweenness measure (termed simply as 'betweenness') that is widely used [33]. In a shortest path dependent network for flow of information, betweenness metric can aid in the measurement of how much information will flow through a certain node or link. Practically, in real life networks, information or anything else does not necessarily follow a shortest path [35]. Usually, information that is flowing though a network like a message or news follows a random path and not an ideal route to reach a destination. There has been no proof for the participants in the experiment of the well known small world experiment of Milgram [38] and the experiment by Dodds *et al.* 2003 [39] to have successfully taken a direct possible route even though they were instructed to do so. It is therefore necessary to include a non-shortest path while measuring betweenness for a realistic approach. When all paths or walks in a graph are considered, there is a high probability of double counting certain links as few paths can share the same set of links. This problem was addressed by Freeman *et al.* [35] by considering only link-disjoint paths, i.e. paths that share no links.

Instead of shortest paths, betweenness centrality based on random walks was proposed by Newman [85]. Variants of betweenness centrality like link betweenness, group betweenness, distance-scaled betweenness and bounded-scaled betweenness was reviewed by Brandes [86]. Apart from reviewing betweenness centrality based on shortest paths, Brandes [86] mentions algorithms to efficiently compute each of the variants and identifies the necessity to develop an efficient algorithm for recomputation of betweenness centrality.

Wang *et al.* [40] studied the betweenness centrality in weighted networks and proved that the links with smaller weight tend to carry greater traffic. Alongside this, they showed that the extreme value index, topology structure and the strength of link weight disorder influences the negative correlation between link betweenness and link weight. Kwangho Park *et al.* [41] used betweenness measure to characterize the weighted network and introduced exponential and algebraic scaling laws which governs betweenness as the function link weight and degree. In complex networks, these laws help in identifying nodes that are influential in terms of physical function.

Joong Lee *et al.* [87] provided an efficient algorithm to update betweenness centrality by identifying a set of nodes whose betweenness centrality is most likely to change and to update the betweenness centralities by using the betweenness centrality of the nodes in the set and the number of nodes not in the set.

Although betweenness measure considers the global network and is capable of assessing whether a node lies on the shortest path between two nodes, it has its limitations like at instances when the majority of the nodes does not lie on the shortest path between any two nodes, the betweenness score for that node is zero. Measures defined by Freeman's work in 1978 [36] are specified only for the binary networks (i.e. a link either exists between a a pair of nodes or not). Barrat *et al.* 2004 [20], Brandes 2001 [34], Newman 2001 [43], have made several attempts to generalize Freeman's (1978) [36] measure which is based on the three node centrality. But the attempts made have focused only on the link weights and not on the number of links based on which the initial measure is based. Barrat *et al.* (2004) [20] extended the degree to weighted networks and defined it as the sum of the weights of the connections to the node. Newman (2001) [43] and Brandes (2001) [34] introduced extensions of the closeness and betweenness centrality measure which were dependent on Dijkstra's algorithm for the shortest path which gives a least costly path. Since these three attempts only consider the link weight, it skips considering the number of links (ties) which is the basis of Freeman's work. To extend the work of betweenness, in this thesis we extend the work by introducing the concepts link weight tolerance and link tension. Weight tolerance of a link in a weighted network could possibly have an effect on the shortest path between two nodes and thereby on the betweenness centrality of a node and a link.

3 Graph Theory

This chapter provides a formal definition of graphs and introduces network models and basic terminology related to graph theory. Graph theory is a sub-field of discrete mathematics where graphs are studied, which can be further applied to describe pairwise relationships between objects. Leonhard Euler in 1735 proved the first theorem of graph theory, who also provided the solution to the Seven Bridges of Konigsberg problem [61]. Graph theory can be used to study various real life applications, for example - social networks, biological networks, computer networks, transportation networks etc.

3.1. Graph

A graph can be defined as $G = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ constitutes the set of elements called nodes (vertices), where $|\ \mathcal{N}\ |$ = number of nodes N and $\mathcal{L} \subseteq x, y \mid (x, y) \in N^2 \wedge x \neq y$ constitutes the set of links which connects the nodes, also known as the edges, where $|\ \mathcal{L}\ |$ = number of links L. The nodes x and y of a link $x \sim y$ are called the endpoints of the link. When a graph has an empty set of nodes and thereby empty set of links, it is called an empty graph or a null graph. The nodes correspond to

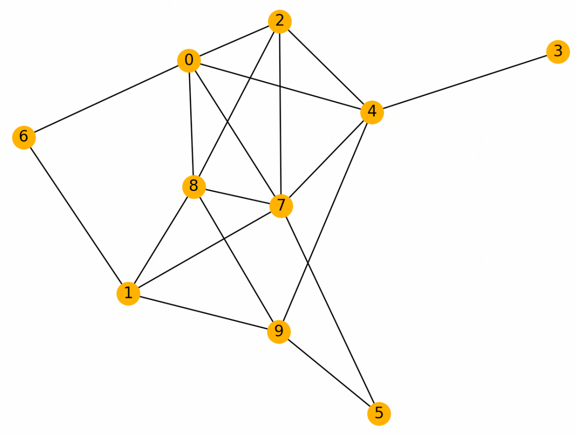


Figure 3.1: Example of a graph

the dots in Figure 3.1, and the links correspond to the lines, with 10 nodes and 19 links. The graph in figure 3.1 can be expressed in the form of $G = (\mathcal{N}, \mathcal{L})$) as

$\mathcal{N}$ = {0,1,2,3,4,5,6,7,8,9}
$\mathcal{L}$ = { (0,2), (0,4), (0,6), (0,7), (0,8), (1,6), (1,7), (1,8), (1,9), (2,4), (2,7), (2,8), (3,4), (4,7), (4,9), (5,7), (5,9), (7,8), (8,9) }

A simple graph is a graph that is unweighted, undirected and has no self-loops or multiple links between a node pair. A loop is a link that connects a node to itself. A multi-graph allows self loops and multiple links i.e. links that have same end nodes, two nodes may have multiple links connecting them. A generalized term for referring probability distribution over graphs is called a random graph. Description of random graphs can be easily provided by the random process that generates the graphs or its probability distribution. Random graphs are generated by initially defining a set of *n* isolated nodes and then successively adding links between the nodes in a random manner. There are various models of random graphs which have different probability distribution on graphs which is explained in later part of this chapter.

3.1.1. Complete graph

A graph consisting of *n* nodes, where every single node is connected to every other node through a link is called a complete graph (K_n). For a graph with *n* nodes, each node has a degree of $n-1$, and there are $\frac{n(n-1)}{2}$ links. The connectivity of the nodes is maximum for a complete graph as the set of nodes that can disconnect the graph is the complete set of nodes. An empty graph is the complement of a complete graph.

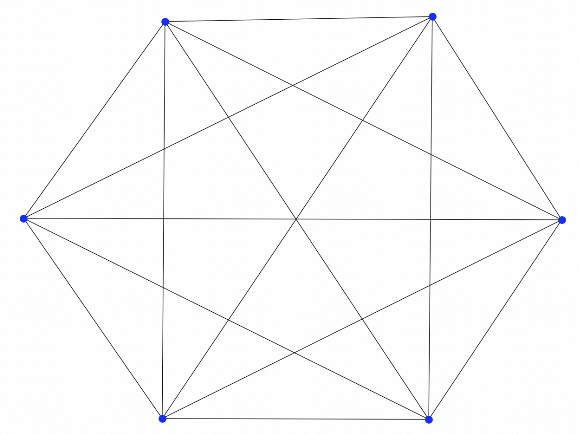


Figure 3.2: Example of a complete graph

Figure 3.2 displays an example of a complete graph with 6 nodes, where each node is connected to five other nodes, i.e. each node has a degree of 5.

3.1.2. Bipartite graph

A graph where the set of nodes is decomposed into two independent and disjoint sets such that the links do not connect any two nodes from the same set is called a bipartite graph. No two nodes of the same set in a bipartite graph are adjacent to each other. In a bipartite graph when every pair of graph nodes in the two sets are adjacent, it is called a complete bipartite graph.

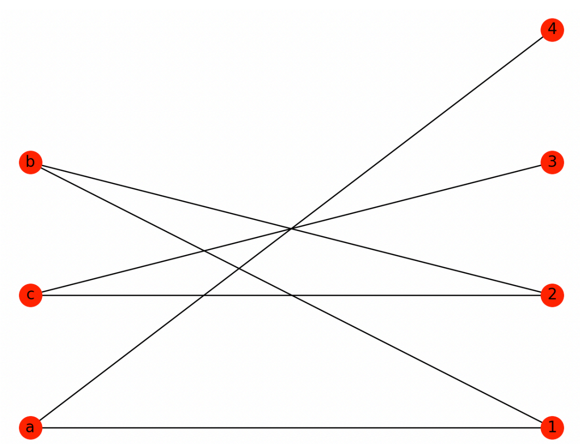


Figure 3.3: Example of a bipartite graph

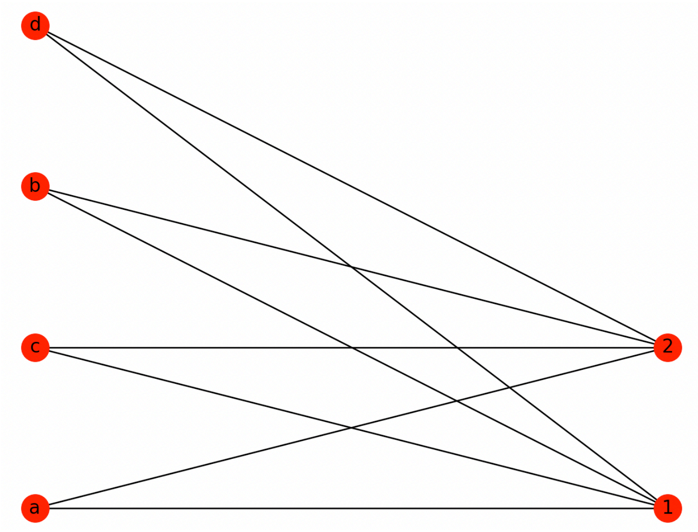


Figure 3.4: Example of a complete bipartite graph

In figure 3.3, there are two set of nodes - {*a,b,c*} and {1,2,3,4}. The nodes from the set {*a,b,c*} are only connected to the nodes from the set {1,2,3,4} and not to the nodes from the same set. Figure 3.4 displays a complete bipartite graph with two nodes in one set and four in another. Each node from one set is connected to all the nodes from the other set.

3.1.3. Complement of a graph

A complement or inverse of a graph *G* is a graph G^c consisting of the same set of nodes as *G*, but two nodes in G^c are adjacent if and only if the same two nodes are not adjacent are in *G*. To construct a complement of a graph, G^c, all the missing links to form a complete graph is filled and all the links that exist in *G* are removed. In the first subplot of figure 3.5, it represents a graph *G* with 6 nodes, the second subplot represent the complement of the graph *G*, consisting of links between the nodes which was absent in *G*.

L in *G* = { (0,1), (0,2), (0,3), (0,4), (0,5), (1,3), (2,3), (2,4), (4,5) }
L in G^c = { (1,2), (1,4), (1,5), (2,5), (3,4), (3,5) }

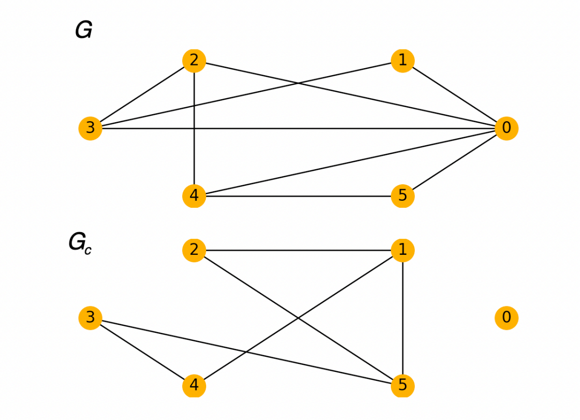
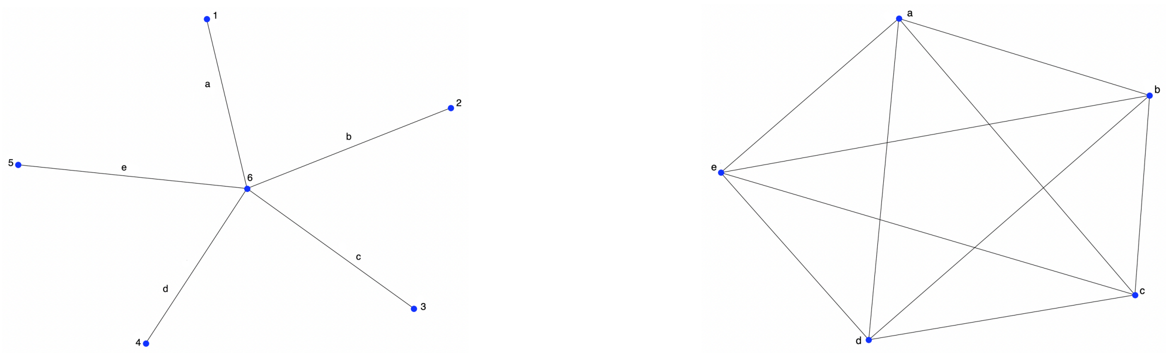


Figure 3.5: Example of an undirected graph *G* and its complement G^c

3.1.4. Line graph

A line graph is derived from graph *G* by associating a node with each link of the graph i.e links in graph *G* become nodes in the corresponding line graph. For a graph *G*, the line graph *L*(*G*) is generated by connecting two nodes with a link only when the corresponding links of *G* have a node in common. In a line graph, each node represents a link of *G* and two nodes become adjacent in *L*(*G*) only when the corresponding links in *G* have a common endpoint. For a directed line graph *L*(*G*), the nodes belong to the link set of *G* where links are directed from l_1 to l_2, where the head of l_1 is touching the tail of l_2. For an undirected graph in figure 3.6 (a), figure 3.6 (b) is the undirected line graph *L*(*G*). For an directed graph in figure 3.7 (a), figure 3.7 (b) is the directed line graph *L*(*G*).



(a) Example of a graph *G* (b) Line graph *L(G)*

Figure 3.6: Example of a Line graph with 5 nodes

(a) Example of a directed graph G (b) Example of a directed line graph L(G)

Figure 3.7: Example of a directed line graph

3.2. Regular graphs

A regular graph can be described as a graph where every node has the same degree. When a regular graph has nodes with degree *k*, is termed as *k*-regular graph. A regular graph where every adjacent and non-adjacent pair of nodes have same number of neighbours in common, it is called a strongly regular graph. Figure 3.8 shows a 2-regular graph with 6 nodes. Figure 3.9 shows a strongly regular graph with 6 nodes and each node has 3 links connected to it.

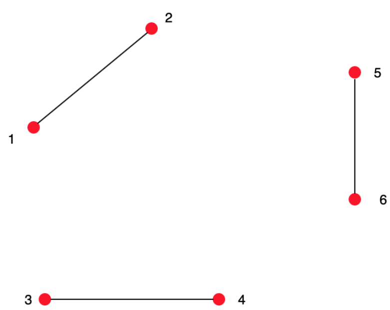


Figure 3.8: Example of a k-regular graph, *k*=2

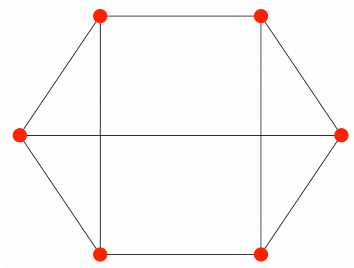


Figure 3.9: Example of a strongly regular graph with 6 nodes

3.2.1. Tree graph

A tree graph is a simple, undirected, connected and a graph with no cycle with a set of line segments that are connected at their ends without any closed loops. Tree graphs are bipartite graph with *n* nodes and *n-1* links. The points at which the links connect are called fork and the segments are called branches. The last nodes and segments at their ends are called tree leaves. A tree graph where each node has same degree is called as a regular tree graph, while when a tree graph has nodes with various degrees, it is called non-regular tree graph.

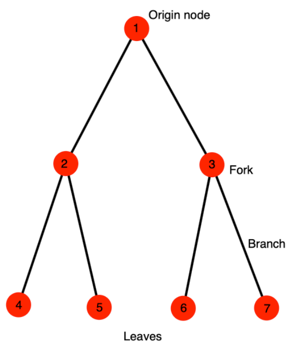


Figure 3.10: Example of a regular tree graph with 7 nodes

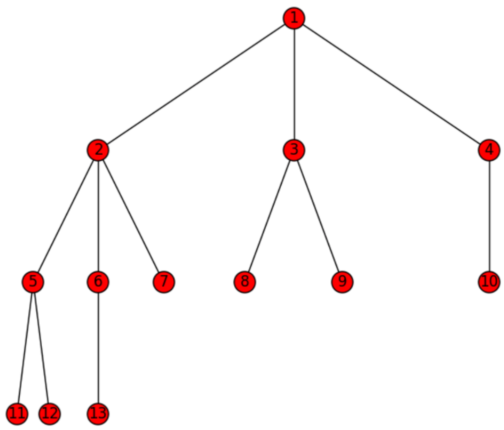


Figure 3.11: Example of an non-regular tree graph with 13 nodes

The height of a node is the number of links on the longest path from the root node (origin node) to a leaf. A leaf node has a height of 0. The height of a tree is same as the height of its root node. Figure 3.10 display a regular graph with height 2. Figure 3.11 displays a tree graph with 13 nodes, 12 links and 7 leaves. The nodes that make leaves of the graph are { 11, 12, 13, 7, 8, 9, 10 }. The height of this non-regular tree graph is 3.

3.2.2. Star graph

A bipartite graph where one node is part of one set and the rest of the nodes belong to the other set, it forms a star graph S_n. In a star graph that comprises *n* nodes, *n-1* nodes are connected to a single node. A star graph is also a tree graph since it has one internal node and (n-1) leaves.

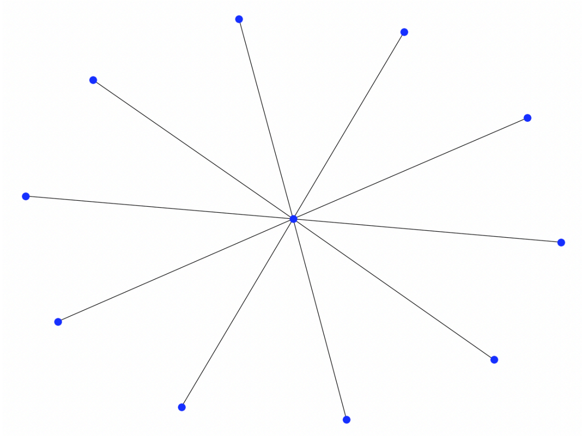


Figure 3.12: Example of a star graph with 11 nodes

In figure 3.12, a star graph consists of 11 nodes with 10 nodes connected to one node. Ten nodes have its degree as 1 and one node has a degree as 10.

3.2.3. Circle graph

A circle or a cycle graph C_n is a graph that has minimum of three nodes connected to each other in a chain, with the starting node and last node connected to each other. The number of links in the graph is equal to the number of nodes where each node has its degree as two since every node has two links emerging from it. Figure 3.13 displays a circle graph with 8 nodes. This graph consists of a single cycle connecting all the nodes. A directed circle graph has a directed trail as shown in figure 3.14.

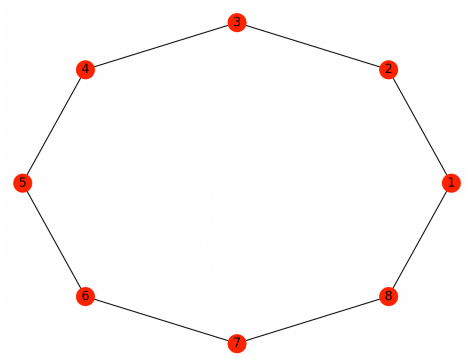


Figure 3.13: Example of a cycle graph with 8 nodes

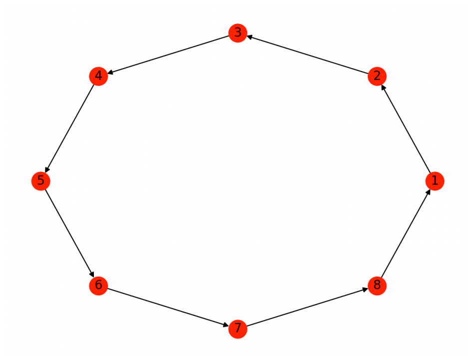


Figure 3.14: Example of a directed cycle graph graph with 8 nodes

3.2.4. Lattice graph

A lattice graph is a regular structure constituting of nodes that are connected to specific number of neighbours. There are various types of lattice - square lattice, ring lattice, spherical lattice, cubic lattice. A square lattice of size n is a $(n \times n)$ two-dimensional grid which contains $N = n^2$ nodes. Each node in a square lattice except the ones in the four boundaries, have four incident links to their respective four closest neighbours. Figure 3.15 displays a square lattice graph with *N*=100.

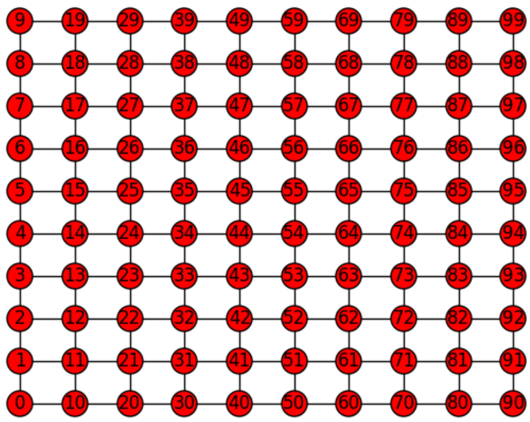
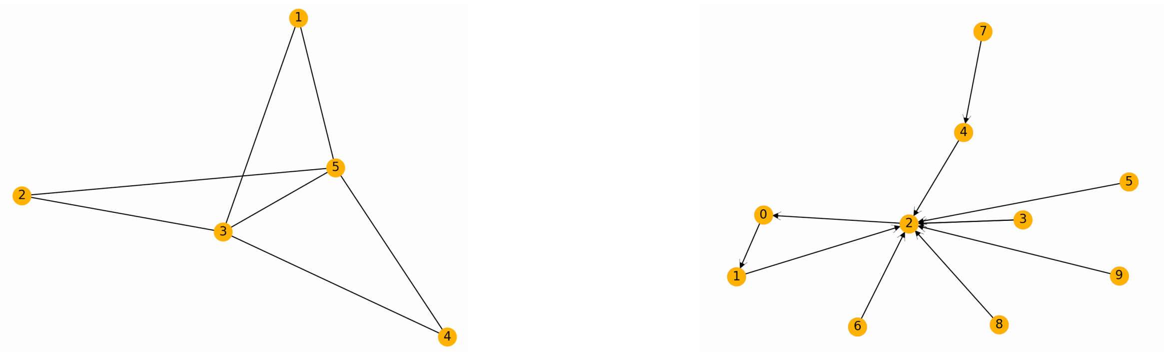


Figure 3.15: Example of a square lattice graph with *N*=100

3.3. Directed and Undirected graphs

On the basis of the link orientation, graphs can be classified either as directed or as undirected. These graphs are significant in the process of communication in a network. Let's consider a pair of nodes *i* and *j* to be connected by a link, and the nodes are indexed from 1 to *N*. The orientation of the link, i.e. the ordering of the the pair of nodes that defines a link is not significant for an undirected graph. If there is an undirected link connecting the nodes *i* and *j*, then the nodes *i* and *j* can communicate with each other from either one of the nodes. The ordering of this pair is important for the directed graph as the links have an orientation. If the link connecting the nodes *i* and *j* is directed towards *j*, the communication is possible only one way from node *i* to node *j* and not in the other direction. However, if there exists another link connecting the same nodes, directed towards node *i*, then node *j* can communicate with node *i*. Figure 3.16 (a) displays an undirected graph with 5 nodes, 3.16 (b) displays a directed graph with 10 nodes where each link is directed towards a node.

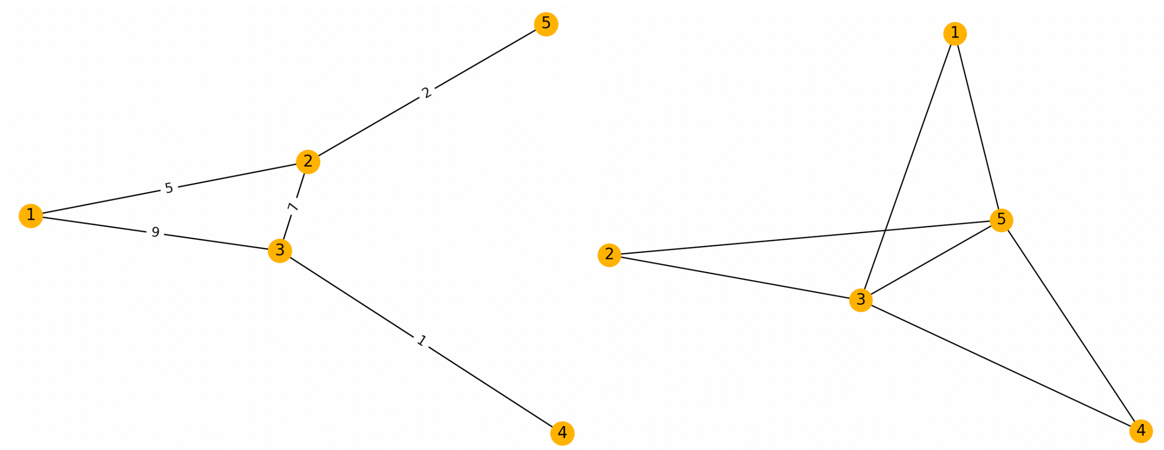


(a) Example of an Undirected graph (b) Example of a Directed graph

Figure 3.16: Directed and Undirected graphs

3.4. Weighted and Non-weighted graphs

A graph that has links without weight associated to it, is called a non-weighted graph. Non-weighted graphs display the relationship between two nodes in a binary form, i.e. the two nodes are either connected or not connected. Social network is a prominent example of non-weighted graphs. When links in a graph have a numerical value associated to it, known as weight, it is a weighted graph. The relationship among the nodes have a magnitude in a weighted graph. A weighted graph with every link having a non-negative number is represented in figure 3.17 (a). Figure 3.17 (b) displays a non-weighted graph



(a) Example of a weighted graph (b) Example of a Non-weighted graph

Figure 3.17: Weighted and Non-weighted graphs

3.5. Neighbourhood

In an undirected network, two nodes are said to be adjacent if there exists a link that connects the two nodes. In an undirected graph *G*, neighbourhood of node *i*, π_i is the set of nodes in *G* that are adjacent to node *i*. If there are two nodes *i* and *j* in a directed network, node *i* is said to be adjacent to node *j* if there is a directed link from node *i* to *j* connecting them. If a directed link is absent from node *j* to *i*, *j* is not adjacent to node *i*. Every node present in the neighbourhood π_i is a neighbour of of node *i*. In the case of a directed network, only when there is a directed link from node *i* to the set of nodes *j*, the nodes become part of the neighbourhood π_i. Links that are directed from node *j* to *i* does not become part of the neighbourhood π_i. Figure 3.18 illustrates adjacent nodes in case of a directed link. Figure 3.19 (a) shows neighbourhood of node *i* in an undirected graph, figure 3.19 (b) shows neighbourhood of node *i* in a graph with directed links.

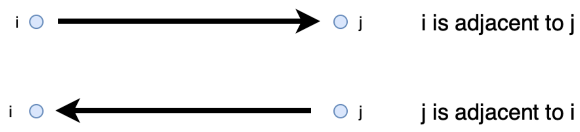


Figure 3.18: Illustration of adjacent nodes

(a) Neighbourhood of node *i* in undirected links

(b) Neighbourhood of node *i* with directed links

Figure 3.19: Neighbourhood

3.6. Graph Metrics

A graph metric can be defined as topological information defined over a graph which helps in understanding and analysing of the graphs when they are subjected to constraints.

3.6.1. Degree, Mean degree and Degree distribution

The degree of a node *i*, k_i is the number of links that are incident to node *i*. In a graph of *n* nodes, a node with degree $n-1$ (maximum degree) is called the dominating node. When a node has zero degree, it is isolated. For an undirected graph, degree of a node *i* can be defined as the number of links between node *i* and various other nodes in the graph. For a directed graph, degree can be of two types - in-degree and out-degree. If a link emerging from node *i* is going away from the node i.e. it is directed away from node *i*, it is called an out-link; if the link is directed towards node *i*, it is known as an in-link. In the case of a regular graph, each node has the same degree, while in a complete graph (a simple undirected graph where every pair of distinct nodes is connected by a unique link), all the nodes have maximum degree (*n-1*). To study real networks such as internet and social networks, degree distribution plays an important role. Degree distribution provides the probability distribution of the nodes over the entire network. The degree distribution of a network P(*k*) can be defined as the fraction of nodes in the network that have degree *k*. In a network, among *n* nodes, if n_k nodes have degree as *k*, then P(*k*) = $\frac{n_k}{n}$. For a random network modelled by Erdos and Renyi, the degree distribution behaves like a binomial distribution [3]. Some popular networks like world wide web and internet are found to follow a degree distribution very similar to that of Power law, $P(k) \sim K^{-\gamma}$, where γ is a constant, with $2 \leq \gamma \leq 3$.

3.6.2. Connectivity, Walk, Path

A sequence of distinct nodes such that two consecutive nodes are adjacent, makes a path. When graph *G* has any two nodes in it linked by a path, it is termed as connected. A walk in the graph *G* is a sequence of finite or infinite steps which connects a sequence of nodes. Here, a finite walk is a sequence of steps $(l_1, l_2, ...l_{n-1})$ for a sequence of nodes $(n_1, n_2..n_n)$ whereas the infinite walk has a sequence of links and nodes without the first and last node. A walk with all distinct links is known as a trail. A path can also be defined as a trail with all distinct nodes. In a directed graph, a sequence of links pointing in the same direction connecting a sequence of nodes forms a directed path.

Connectivity defines whether a graph is connected or disconnected. A graph is said to be connected if there exists a path between every pair of node. There must exist a path to traverse from one node to any other node in the graph. This is called the connectivity of a graph. In general, connectivity can be of two types: node connectivity and link connectivity. The node connectivity of an incomplete graph can be defined as the minimum number of nodes that are to be removed to make it disconnected. Similarly the minimum number of links that needs to be removed to disconnect the graph is called as link connectivity. In the case of a complete graph, going by the above definition, node connectivity cannot be determined since deleting of the nodes does not disconnect the graph. The robustness of the graph increases with node or link connectivity. But both these connectivity measures do not consider the significance of the deleted nodes or links. Even though a graph is disconnected into two or more components, it could function well if the amount of traffic is low. Therefore, two graphs may not be equally robust though they have same the node or link connectivity.

3.6.3. Shortest path and Average path length

Shortest path in a network provides the fastest and optimal path from a source to the destination making it a significant aspect for effective communication in comparison to various other paths that connect the source to the destination. All the shortest path lengths of a graph *G* can be represented in a shortest path matrix X where each entry is the length of the shortest path from node *i* to node *j*. The largest value of the entry in the matrix is called the diameter of the graph. Mean or average shortest path length is the mean of the shortest path lengths among the over all couple of nodes which provides a typical measure of separation between two nodes. Network topology is quantified by the metric average path length which aids in serving various inferences for neural networks and their evolution. It can be simply defined as the average number of links that must be traversed in the shortest path between any two pair of nodes *i* and *j*. This metric tends to lose its meaning for graphs where each node is accessible from any other node in the network.

3.7. Incidence matrix and Adjacency matrix

For a graph *G*, the incidence matrix is a *n* x *m* matrix *B* with b_{ik} as its elements, *n* and *m* are the number of nodes and links respectively. Here, each row corresponds to a node and each column corresponds to a link such that if l_k is a link joining node *i* and node *j*, then for undirected graphs b_{ik} and b_{jk} value becomes 1 and rest of the elements of column *k* are 0. The matrix in equation 3.1 shows an example of an incidence matrix for the simple graph (a graph that is unweighted, undirected without any loops or multiple links) displayed in figure 3.20.

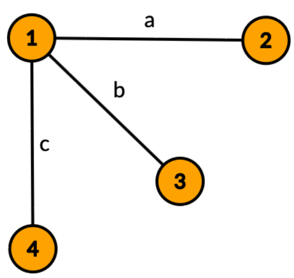


Figure 3.20: Example of a simple graph

$$B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

An adjacency matrix or a connection matrix is a square matrix which is used to represent a finite graph. The elements of this matrix, a_{ij}, indicate whether pairs of nodes are adjacent to each other or not. For an undirected graph, the adjacency matrix is symmetric, i.e. $a_{ij} = a_{ji}$ and $A = A^T$. For a graph *G*, the adjacency matrix *A* is a *n* x *n* matrix where a_{ij} = 1 if there is a link that connects node *i* and node *j*, otherwise a_{ij} = 0. This matrix consists of zeroes in the diagonal for all simple graphs with no self loops. If the graph is weighted, a weighted adjacency matrix *W* is defined where the elements of the adjacency matrix are the weights of the link w_{ij} instead of a_{ij}. The matrix in equation 3.2 is an example of the adjacency matrix for the displayed non-weighted graph in figure 3.20.

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.2}$$

For the weighted graph shown in figure 3.16 (a), the matrix *W* in equation 3.3 is its weighted adjacency matrix.

$$W = \begin{bmatrix} 0 & 5 & 9 & 0 & 0 \\ 5 & 0 & 7 & 0 & 2 \\ 9 & 7 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \end{bmatrix} \tag{3.3}$$

3.8. Graph models

This section briefly introduces the various models of Random graphs. Construction of the graphs is explained in detail in Chapter 5.

3.8.1. Erdos Renyi Random Graphs

In the year 1959, Paul Erdös and Alfrèd Reńyi were the pioneers to study Random graphs in detail to provide a probabilistic construction for these type of graphs, popularly know as Erdös-Reńyi (ER) random graph models [1]. Random networks stand as the starting juncture in terms of structural complexity. Random network is characterized by *n* number of nodes and probability *p* defining the existence of a link between any pair of nodes and has no articulate structure. Gilbert [2] also made independent contribution related to random graphs. Random graphs have several applications, which are defined in the network theory work of Newman 2002 [4]. Percolation theory also makes use of random graphs. Random graphs have been compared to percolation bonds in *The probabilistic methods* by Alon and Spencer [5]. The Erdös-Reńyi random graph model consists of two models. Gilbert [2] introduced the first model, represented by *G*(*N*,*p*). This includes all the links *L* of the link set $[N^2]$ with probability *p* in the random graphs. In the second model, the *G(N,L)* model has an empty graph with *n* nodes and *m* links from $[N^2]$ are added at random to the graph. Development of a random graph goes through an evolution beginning with a set of *n* isolated nodes and then the graph is further constructed by addition of links in a random manner. A connected graph is eventually obtained by the collection of graphs produced at different stages connected by links with larger probability *p*.

Despite few similarities, random graphs vary from the real-world networks in two particular ways [47], [15]. First one being clustering, which is observed in the real world networks is absent in the Erdos-Renyi model as pointed out by Watts and Strogatz [8], [6]. Clustering is seen in the network only if the probability of two nodes being connected by a link is greater when the nodes in question have a common neighbour. This can be comprehended as there exists another node in the network to which they are both attached. Clustering coefficient was introduced by Watts and Strogatz to measure this clustering. Another difference that is observed between random graphs and the real-world networks is the degree distribution. As pointed out by Albert and Barabassi [47], the degree distribution of random graphs is binomial which is a type of poisson distribution which is not seen in the case of the real-world networks. The degree distribution of real world networks is seen to be a power law distribution for few and exponential distribution for the others.

3.8.2. Small world networks

The classical model of small world networks was studied by Watts and Strogatz in 1998 [6], typically known as the WS model, consists of an undirected graph with n nodes that are connected to k nearest neighbours, for a defined value k. The average path length and clustering coefficient of regular graphs are high as they are well connected to the neighbours only. The clustering coefficient is high in regular graphs due to the formation of triads by the well connected neighbors. Nodes that are distant lie far apart from other nodes making the average path length higher in regular graphs [77]. The regular graph is then introduced with randomness by rewiring a few edges to connect distant nodes, this reduces the average path length and generates the small world network.

Traditionally, small world networks have average path length similar to a random network and clustering coefficient similar to that of a lattice network. But this definition is not always successful in differentiating a true small world network from those that are very similar to lattice or random structures. Qawi *et al.* proposed a metric [7], omega (small-world metric) which compares the path length to random network alongside comparing clustering coefficient to an equivalent lattice network to identify a small world network. Watts and Alpha published another model on small world networks which takes into account two diverse graph models - ordered and random [8]. As per the author in [8], the ordered world graphs consists of disconnected cliques called caves. This type of graph is clustered and consists of $n/(k+1)$ caves or simply cliques that are isolated, thereby naming them caveman graphs. Each of these caves is a complete graph, consisting of several nodes connected to one another but no link exists to connect nodes from two different caves. Whenever a new node enters a cave, it instantly gets connected with every other node in the cave. As the nodes in the cave form triads, it results in high clustering coefficient. In the world of random graph models, every node can communicate with each other regardless of the connectivity it formerly had. The probability of two nodes connecting is equal in the model, making the connections that are newly formed to appear in a random manner. The way these two models interact, new set of rules could be established to build a network gradually where connections among nodes are made based on one of the two rules of interaction. Among these two diverse world of networks, few sets of networks exhibit a manner of interaction which falls in the equilibrium of randomness and orderly behaviour with high clustering coefficient as well as low average path length due to random connection of nodes. This results in a balance between a random graph and an ordered graph, generating small world networks.

Besides the classical model by Watts and Strogatz, Newman and Watts introduced a modified model of the classical WS model where new links are added between a pair of nodes in a random way without disconnecting the existing links, instead of just rewiring the links between the nodes [9]. This modified model ensures that all regions stay connected and is simpler to analyse and derive mathematical conclusions.

Kasturirangan also proposed a new method to generate small world networks [10]. Unlike the classical WS model, the proposed idea by the author suggests that small world networks can be generated due to multiple scales of network that are formed by nodes with high degree. This in turn reduces the average path length of the network. High degree nodes are randomly introduced to a regular network to generate small world networks. This model is different from the WS model due to the fact that well defined boundaries are not formed between the well connected nodes when random edges are introduced. A more realistic model was proposed by Kleinberg [11] which tries to understand given a regular network, how short paths occur. Kleinberg argues that the WS model is a poor representation of few real world situations as shortcuts connect nodes promptly which falls far apart with uniform probability. In the model proposed by Kleinberg, the links are added randomly to a regular network but there is a decrease in the probability of the link being added with respect to the distance as measured in the regular graph.

3.8.3. Scale free networks

Erdos-Renyi graph model and small world networks exhibit a common feature which is the homogeneity of the degree distribution of a network which undergoes an exponential decay, exhibits power law, . Many of the complex networks like social networks, software dependency networks, protein-protein interaction network are few of the example of a scale free network. Barabási and Albert in 1999 did exceptional work on describing scale free networks [12]. The Barabási and Albert model (BA) encompasses the power law tail of degree distribution. The authors argued that many of the existing models fail to consider two significant factors. The first factor, existing models are static unlike the real world networks which are dynamic. When links are added or rearranged in a network, the number of nodes remain static/fixed through the forming process making it dissimilar to the real world networks. Second factor, probabilities assumed for small world and random graphs when new links are created are uniform which is also unlike the real world networks. The authors propose that in a scale free network, for the purpose of self-organization, the two main elements are preferential attachment and growth.

Given a same size of network while considering a same average degree, the average path length of scale free networks is observed to be smaller compared to that of random networks. Bollobas *et al.* proposed a model for directed scale-free graphs that considers the two factors which is described in [12] and is dependent on the in- and out-degrees.

Model	Reference
BA model	[12]
BA tree	[12]
Huberman–Adamic model	[48]
Static model	[49]
Accelerated growth model	[18]
Fitness model	[50]
PIN model	[51]
Hierarchical Network	[75]

Table 3.1: List of various scale free networks

Table 3.1 lists the diverse model of scale free networks alongside its network size, *N*. Fitness model, another model of scale free network, has its emphasis on fitness and eliminates the preferential attachment rule [76]. For every node *i*, fitness x_i is introduced which is a measure of importance or rank of a node. Instead of assigning a link between two nodes *i* and *j* with a probability *p*, which is similar for all the pair of node, probability $p(x_i, x_j)$ is used.

Hierarchical network model, a model of scale free networks proposed by Ravasz *et al.* [75], considers that many networks are modular in nature as they can be identified by groups of nodes which are interconnected and have sparsely connected nodes outside the group of nodes they belong to. These modules are combined with each other to form a hierarchical network.The preferential attachment for the BA model was expanded to nonlinear by Krapivsky *et al*. [16]. In a non preferential attachment, hubs are absent or have small hubs compared to the scale free network with linear preferential attachment. Linear preferential attachment is the only case for which power law degree distribution can be obtained with the rate equation method. Preferential attachment in which the probability of attachment is proportional to the degree of the target node is called linear preferential attachment. Dorogovtsev *et al.* [18] also established a simple model on the basis of selection of nodes. Beginning with three nodes that are connected, this model selects a link at random from the existing set of links and then its ends are tied to a newly generated node. This model's degree distribution displayed power-law form with the method of master equation. All the links in the models of [22], [16], [17], [18] are equivalent which is the common property exhibited among them. For a network to carry on its basic functionalities, it is necessary to have variation in interaction strengths [19]. Combination of new links and nodes, and the dynamic evolution of the weights is proposed by Barrat *et al.* [20] for the growth of the weighted networks in a model which is based on weight driven systems in real life that exhibits scale free properties. Scale free networks are generated with varying power-law exponents [21] by generalizing the model [20] through weight driven preferential attachment of new nodes to existing nodes.

4
Betweenness Centrality

This chapter explains the metric betweenness centrality. Section 4.1 provides the definition for node betweenness centrality and section 4.2 describes link betweenness. Betweenness centrality for weighted graphs is explained in section 4.4. The various variants of betweenness is described in section 4.5.

4.1. Betweenness for unweighted and undirected graphs

In the analysis for various network data models, *betweenness centrality* plays an important role. Bavelas [37] in 1948 introduced this concept of betweenness centrality which can be defined as a measure based on the shortest paths in a graph. The principle of the measure betweenness centrality is based on the concept that an individual could be connected weakly and distant from others but could also be a significant intermediary. For example, let us consider exchange of information between individuals in a network. The individual can have control over the communication depending on the strength of his/her intermediary role. This individual can influence the network by disrupting or filtering the information flowing in the network. The coordination of the network is assured by this individual. A node or a link has the potential to control the flow of information in the network, making betweenness centrality important.

Betweenness measure also provides a measure of the load carried by the nodes and links in a network. The conventional definition of betweenness centrality was provided by Anthonisse (1971) [64] and Freeman (1977) [33] independently. Anthonisse defined the amount of flow through a node or link as ”rush” due to a unit flow that is induced on all pairs of nodes. As described by Anthonisse, the distribution of unit flow is uniform among all the shortest path connecting the two nodes. Importance of a link or a node is indicated by the betweenness value of the respective node or link.

The distribution of node and link betweenness centrality is determined by the topology of the network. The distribution could be either uniform or non-uniform. When the node betweenness distribution is uniform, it implies that every node has same value of betweenness centrality. While, when the node betweenness distribution is non-uniform, few nodes may have higher values of betweenness centrality compared to others. This implies that the nodes that have high value of betweenness centrality are more vulnerable than others.

In a connected graph, for each pair of nodes, there exists at least one shortest path between the nodes in a way that either the number of links that the path encounters or the sum of the weight of the links is kept to a minimum. The number of these shortest paths that passes through a particular node defines its betweenness centrality. When there is only one shortest path that connects each pair of nodes, determining betweenness centrality becomes fairly simple as the intermediate nodes can have control over the communication between the pair of other nodes. But in the cases when there are multiple shortest paths linking a pair of nodes, the intermediate nodes tend to lose their control.

4.2. Node betweenness

Freeman [33] describes betweenness centrality of a graph as the mean difference between the centrality measure of the most central node and other nodes. It quantifies the amount of times a particular node has to be traversed to connect a pair of nodes. Betweenness centrality of a node *v* in a graph *G*=(*N*,*L*) as described in [33] can be calculated in the following manner :

- Considering each pair of nodes (*x*,*y*) in the graph and finding all shortest paths between them.
- For each of these pairs, finding the number of shortest paths that passes through the node *v*.
- The ratio of number of shortest paths passing through the node *v* to the total number of shortest paths between the pair of node (*x*,*y*) is obtained .

Expression for betweenness centrality is given as below,

$$C_B = \sum_{x \neq y \neq v \in N} \frac{\sigma_{xy}(v)}{\sigma_{xy}} \tag{4.1}$$

where σ_{xy} represents the number of shortest paths with nodes *x* and *y* as their end nodes and $\sigma_{xy}(v)$ is the number of shortest paths that include node *v* in the path [33].

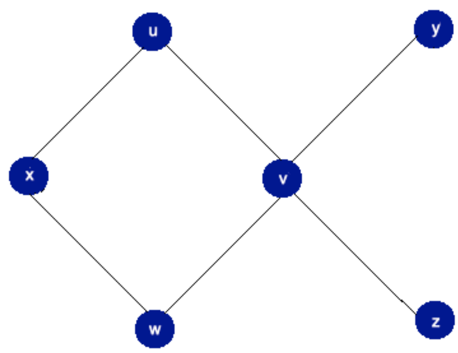


Figure 4.1: Example of an undirected graph (G)

In the graph from the figure 4.1, for the node pair (*x*,*y*), node *v* plays a significant role as it is required for any path that leads to node *y*. For a complete graph K_n, the node betweenness centrality is zero since no node lies in the path of a shortest path as the length of each shortest path is one. To get C_B in the range [0,1], as the betweenness centrality increases with the number of nodes in the network, division by the number of pair of nodes excluding the node *v* can be done, where for undirected graphs the expression is $\frac{(n-1)(n-2)}{2}$ and $(n-1)(n-2)$ for directed graphs, here n is the number of nodes in the giant component. Figure 4.2 displays an undirected graph with betweeness centrality value of each node, which shows that few nodes are more likely to be in the path of communication between other nodes.

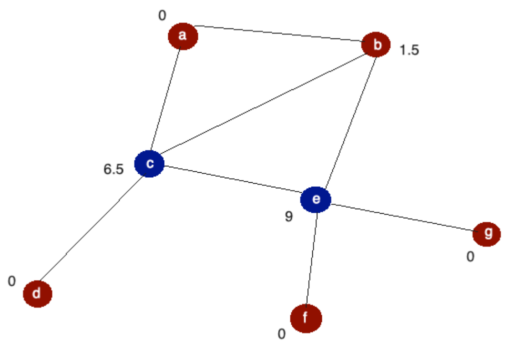


Figure 4.2: Example of an undirected graph (G) with betweeness value of each node

Figure 4.3, 4.4, 4.5, shows node betweenness distributions for three classes of random graphs.

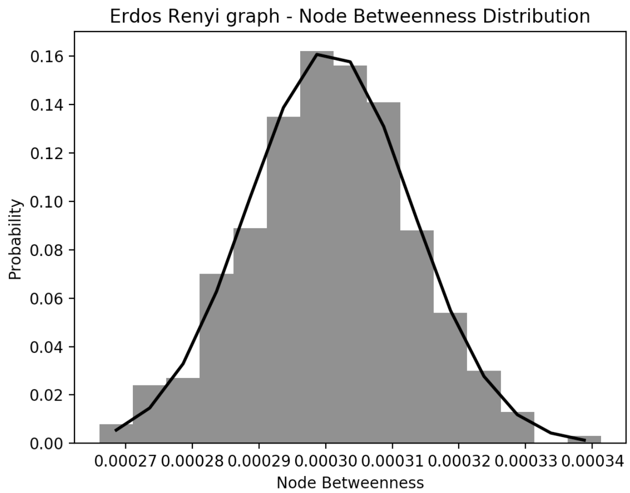


Figure 4.3: Node betweenness distribution of Erdos-Renyi graph with 1000 nodes

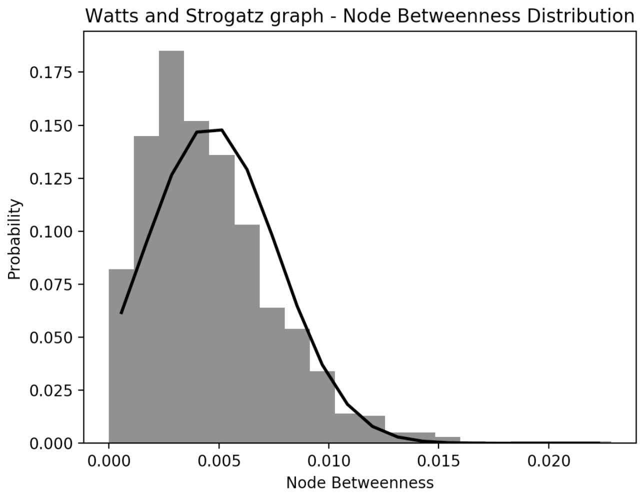


Figure 4.4: Node betweenness distribution of Watts-Stogartz smallworld graph with 1000 nodes

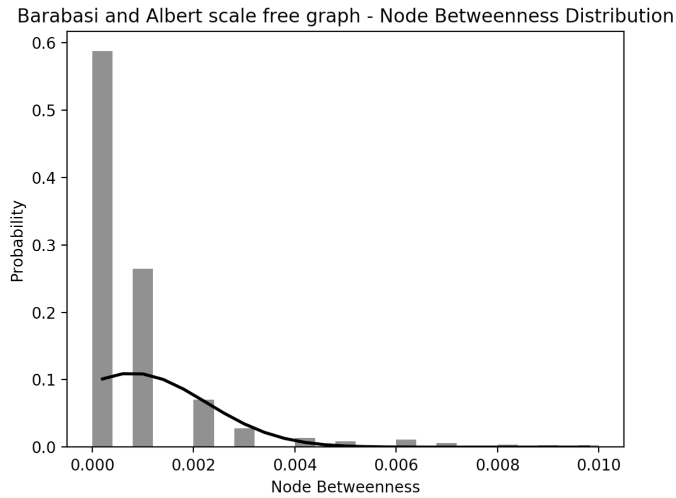


Figure 4.5: Node betweenness distribution of Barabasi-Albert scale free graph with 1000 nodes

4.3. Link betweenness centrality

Similar to node betweenness, Girvan and Newman [84] extended this concept to the cases of links defining it as “link betweenness“. Link betweenness of a link is the number of shortest paths between a pair of nodes that passes through it. Link betweenness is obtained by replacing $\sigma_{xy}(v)$ in the definition of node betweenness by $\sigma_{xy}(l)$, where *l* is a link [34]. In a graph containing groups and communities that are connected loosely by few links, these links have greater chances of having high link betweenness since many shortest paths will traverse these links. When one of these links which connects two groups is removed, the groups get separated. Link betweenness in normalized by multiplying the value by the factor $\frac{2}{n(n-1)}$, where n is the number of nodes in a graph.

$$C_B^l = \sum_{x \neq y \neq v \in N} \frac{\sigma_{xy}(l)}{\sigma_{xy}} \tag{4.2}$$

Figure 4.6, 4.7, 4.8, shows link betweenness distributions for three classes of random graphs.

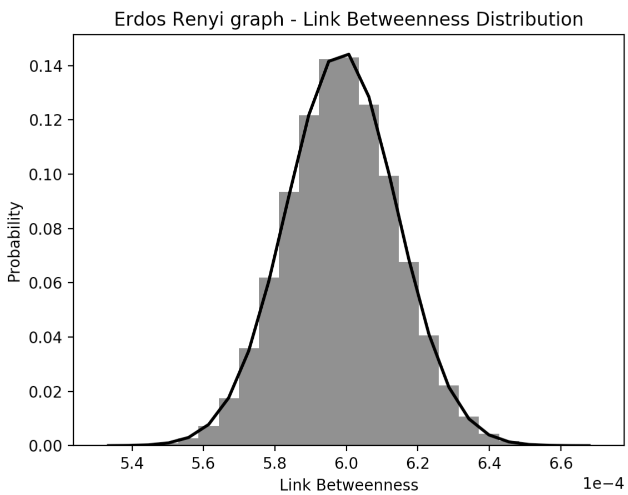


Figure 4.6: Link betweenness distribution for Erdos-Renyi graph with n=1000 and p=0.5

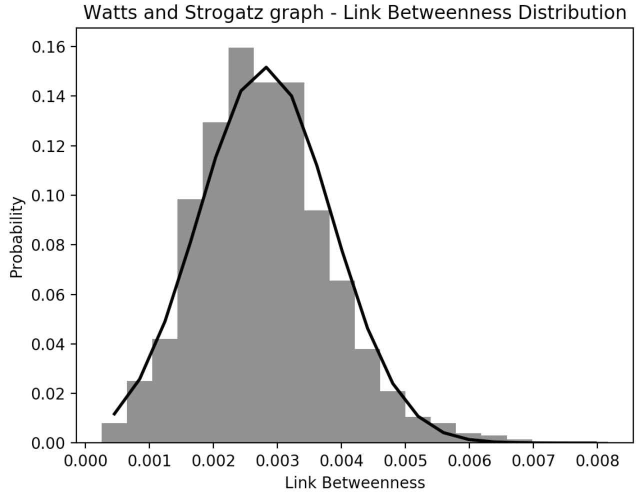


Figure 4.7: Link betweenness distribution for Watts-Stogartz smallworld graph n=1000 and p=0.5

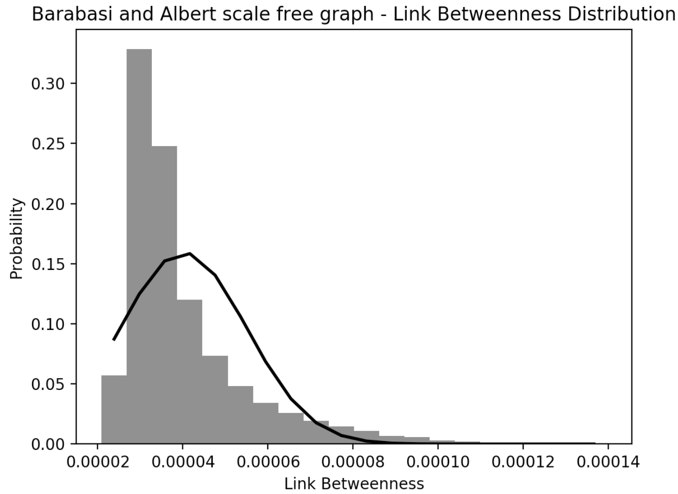


Figure 4.8: Link betweenness distribution for Barabasi-Albert scale free graph with final nodes, n=1000 and p=0.5

4.4. Betweenness centrality for weighted graphs

In a weighted graph, nodes connected by links are not treated as binary (i.e. either a link is present or absent) interactions. Another dimension of heterogeneity is added to the network by weighing the links in terms of capacity, influence. Newman (2001) [43] in his model used weighted node betweenness to identify most important persons in a social network. Instead of physical distances between the two persons, in Newman's model the link weight corresponds to the inverse strength of the relation between the two of them. It can be inferred from here that information is passed by people having strong relations/connections and does not necessarily follow the shortest path. In addition to betweenness based on shortest paths, flow betweenness is one of the significant extensions to it. Flow betweenness of a node v can be defined as the fraction of path through node v when maximum flow is transmitted from source x to destination y over all pairs of (x,y). Unlike shortest-path and flow based betweenness, which takes into account only a particular set of available path, random walk betweenness on the other hand considers all paths between nodes which is useful for identifying nodes that have high centrality even when they do not lie on the shortest path or on the the path of maximum flow.

Betweenness takes a significant amount of time to be computed. An algorithm proposed by Brandes [34] calculates betweenness at a faster rate which needs $O(nm)$ time, where n is the number of nodes and m is the number of links. Apart from reducing the amount of time, this algorithm does not assume the network to be binary and also allows betweenness to be calculated for weighted network.

In the case of weighted graphs, time taken is $O(nm + n^2 logn)$. However, this algorithm by Brandes considers only the sum of link weights and fails to provide focus on the count of ties on paths. This algorithm is a generalized form of Freeman's (1991) [35] flow measure which takes into account the fact that in a weighted network, compared to the paths with few weakly connected intermediate nodes, paths with more intermediate nodes have better and quicker transaction between two nodes.

4.5. Variants of Betweenness centrality

There are various variants of betweenness centrality which are listed in table 4.1.

Variants of Betweenness Centrality	Author
k-betweenness $C_{B(k)}$	Borgatti and Everett [62]
Length-scaled betweenness $C_{B_{dist}}$	Borgatti and Everett [62]
Linear scaled betweenness C_{Blin}	Geisberger [54]
Group betweenness $C_B(C)$	Everett and Borgatti [63]

Table 4.1: List of variants of Betweenness Centrality

Borgatti and Everett (2006) [62] defined k-betweenness of a node to model a realistic network relations. It can be defined as the sum of dependencies of pairs that include only shortest paths of length that is bounded by a constant *k*. Formula for k-betweenness of a node *v* can be defined as below

$$C_{B(k)}(v) = \sum_{x,y \in : dist(x,y) \leq k} \frac{\sigma(x,y \mid v)}{\sigma(x,y)} \tag{4.3}$$

Variant of the above mentioned betweenness definition is the length scaled betweenness [62], which considers paths that are not the shortest paths but weighs in shortest path as inversely proportional to their length in the following manner:

$$C_{B_{dist}} = \sum_{x \neq y \in N} \frac{1}{dist(x,y)} . \frac{\sigma(x,y \mid v)}{\sigma(x,y)} \tag{4.4}$$

The principle behind this variant is that, the longer a length of a path, less valuable it could be to control it. Since the length of the shortest path is the only factor that is scaled here, it is termed as length scaled betweenness. In contrast to the length scaled betweenness where the dependency is on length of the shortest path, linear scale betweenness is dependent on the relative distance of a node *v* from the source *x* [54]. The node v gets more influential when it is far away from the source *x* and thereby closer to the target *y*. This linear scaled betweenness is given by

$$C_{Blin} = \sum_{x \neq y \in N} \frac{dist(x,v)}{dist(x,y)} . \frac{\sigma(x,y \mid v)}{\sigma(x,y)} \tag{4.5}$$

In 1999, Everett and Borgatti [63] introduced group betweenness

$$C_B(C) = \sum_{x<y} \frac{\sigma_{x,y}(C)}{\sigma_{x,y}} \tag{4.6}$$

where $x, y \notin C$, $C \subseteq N$ i.e. *C* is a subset of a graph with node set *N*, $\sigma_{x,y}$ is the number of shortest paths connecting *x* to *y*, $\sigma_{x,y}(C)$ counts the number of shortest paths connecting (*x*, *y*) passing through *C* (any node of *C*).

Group betweenness can be defined as the proportion of shortest paths that connect the pairs which are not part of the group, passing through the group. But this measure considers each individual in the group separately, thus giving more general results. A more accurate measure for group betweenness is provided by Kolaczyk *et al.* in [92], but the measure is applicable only for simple graphs and is not flexible to be used for other centralities measures.

For the betweenness centrality of a multigraph, i.e. a graph in which loops (which can be ignored) and multiple links connecting the same pair of nodes are allowed, the number of shortest paths connecting two nodes depends on the multiplicity of their links: tripling a link of a path results in three different paths of the same length, because either copy of the tripled link can be used. If more than one link has multiplicity larger than one, then any instance of one link combined with any instance of another link yields a different path, so that the total number of paths obtained from a generic path is the product of the multiplicities of its links.

When betweenness centrality is computed in the traditional manner [33], Brandes [34] points out that extra information than necessary is being determined which acts as its weakness. This is because the approach involves defining the pair dependency of a node pair (*x,y*) on an intermediary node *v* by the ratio of shortest path between the nodes (*x,y*) such that the intermediary node *v* lies on all shortest paths between (*x,y*). Brandes [34] aggregated path counts from various source nodes in the network to build a faster algorithm.

Although the algorithm proposed by Brandes [34] is faster than the algorithm by Freeman in [33], many researchers believe that the computation algorithm by Brandes is still costly for larger networks. To fasten the process of computation, an approximate value instead of an exact value of betweenness can be used as an alternative. An estimation method for betweenness was proposed by Brandes *et al.* [53], where computation of betweenness was based on different selection strategies of node sources in order to check the approximation quality. Another variant to compute approximate betweenness centrality was given by Geisberger *et al.* [54] with the help of bisection scaling algorithm. Following this, a linear time approximation algorithm was suggested by Makarychev [55] to find the ordering of the nodes that maximizes the constraint for betweenness. For scale free graphs, Bader *et al.* [56] presented a parallel algorithm to compute betweenness centrality.

Applications of betweenness centrality is diverse and can be implemented across several disciplines. The magnitude of a node's role in the flow of information across a network is provided by betweenness centrality. By using this, the most influential and prominent nodes in the network are identified which is significant in identification of critical junctions in transportation, nodes in biological networks, documents in World Wide Web. Betweenness centrality in real life networks was studied by Holme [57] who described the relationship of traffic density to betweenness; Jin *et al.* [58] used betweenness centrality to identify nodes that are potentially harmful in an electric grid; to detect the most central residues in protein-protein complex structures, Leydesdorff [59] also used betweenness centrality.

Betweenness centrality plays an important role in detection of communities also. A community detection technique was proposed by Newman *et al.* [26] which involves repeated removal of links with highest betweenness centrality value from the network. As an alternate to this, Pinney *et al.* [60] devised an algorithm for community detection where node betweenness determines the network decomposition instead of link betweenness. The weakness in these algorithms as pointed out by Newman *et al.* [26] is the cost involved in computation and repeated calculation of all-pair shortest paths when the links are removed. For the unweighted graph in figure 4.9 with 8 nodes, figure 4.10 is the visualization of node degree and node betweenness of all its nodes. Figure 4.11 shows the value trends of link betweenness and link degree, where link degree can be defined as the number of incident links to the nodes on the end-points of the link, subtracted by two. From figure 4.10 and 4.11, we can observe that the nodes and links with high degree have high betweenness.

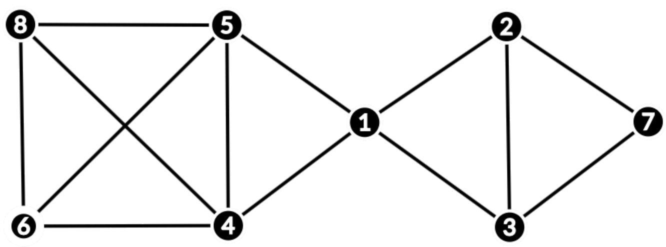


Figure 4.9: Example of an unweighted graph with 8 nodes

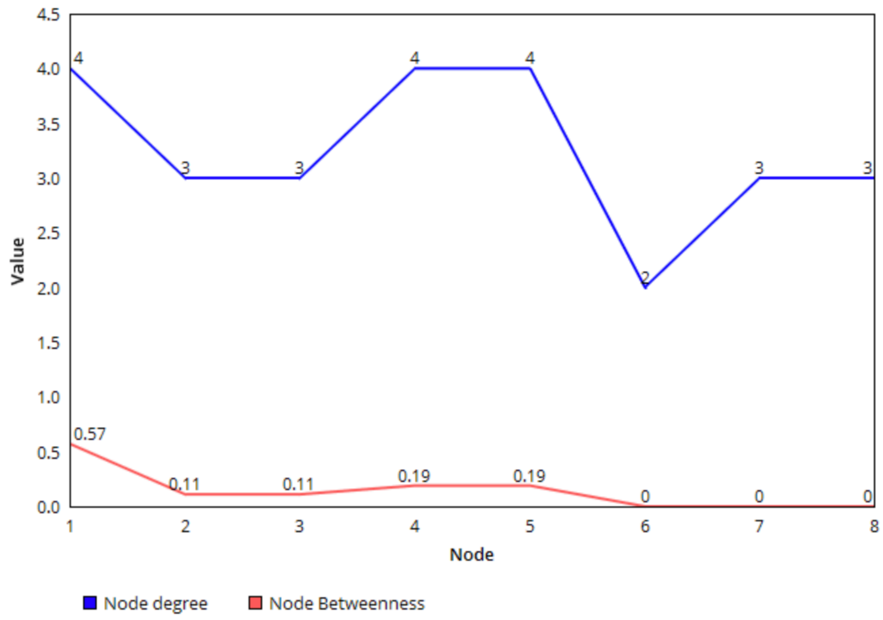


Figure 4.10: Node betweenness and Node degree for the nodes in figure 4.9

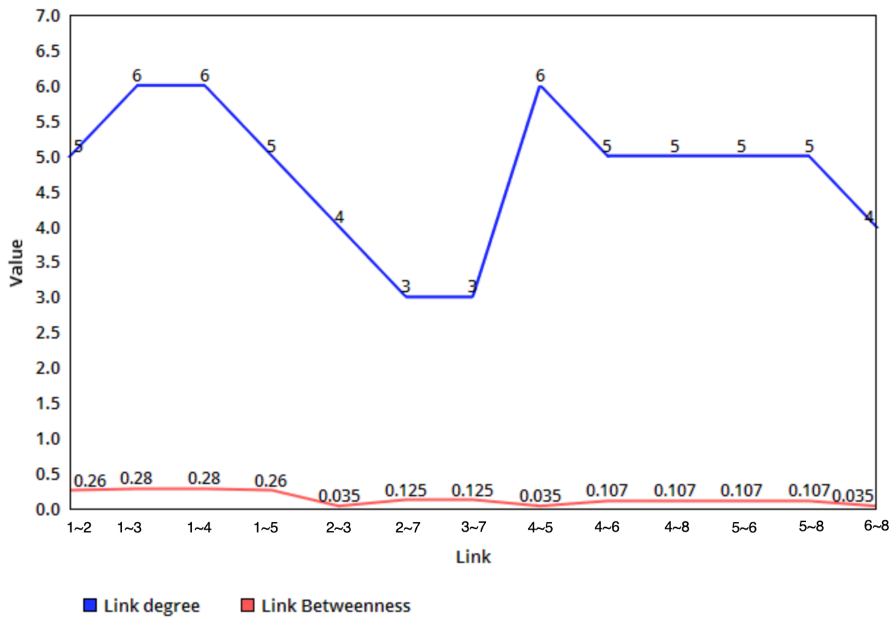


Figure 4.11: Link betweenness and Link degree for the links in figure 4.9

5

Generation of Graphs

This chapter explains the method and algorithm used to generate non-weighted graphs and weighted Erdos renyi random graph, Watts and Strogatz small world graph and Barabasi Albert scale free graph.

Python language is used to generate the graphs in this thesis, as it is a powerful programming language which can be flexibly used to build network algorithms and represent networks. Graphs are represented using a package called NetworkX, the most commonly used network library in Python for the manipulation, creation and studying of structures, studying the function and dynamics of complex networks. Matplotlib is another library of Python that has been used for graphical representation of the graphs generated. For generation of non-weighted and directed graphs stochastic graph generators has been used. The concept used for generation of weighted networks has also been described in the chapter.

5.1. Non-weighted and non-directed graphs

When a graph in which a link does not have any cost or weight associated with it, it is called an unweighted graph. All the links in the unweighted graph have unit weight. This section describes the functions used to generate unweighted random graphs.

5.1.1. Non-weighted Erdos Renyi Graph

Erdos Renyi model of the form $G(n,p)$ is a random graph with n nodes where each possible link has probability p of existing. The "erdos_renyi_graph" function, part of the NetworkX library is used to generate a non-weighted ER graph which returns a $G(n,p)$ random graph and chooses each of the possible links with probability p. The parameters of the function include: number of nodes (n), probability of link creation (p). Figure 5.1 displays a generated unweighted Erdos Renyi graph with n=200 and p=0.5.

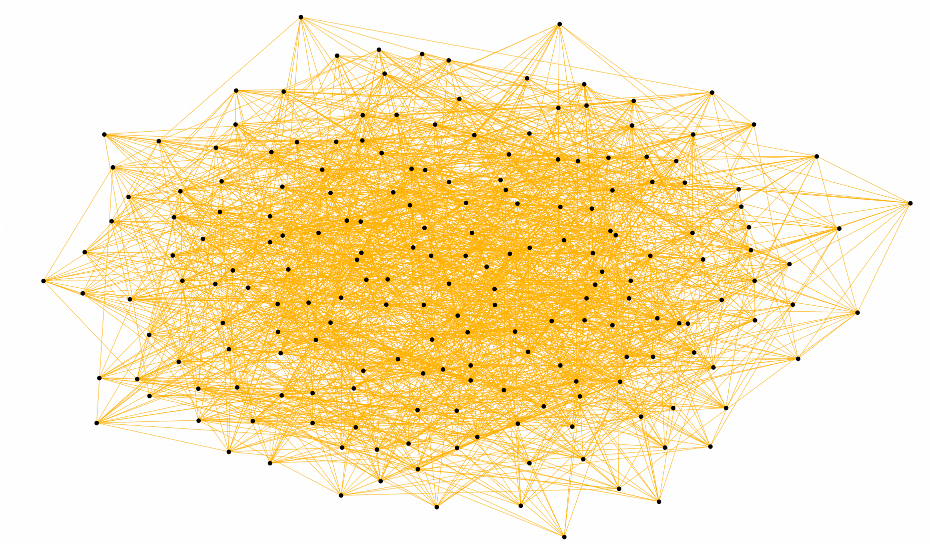


Figure 5.1: Example of the generated non-weighted Erdos Renyi graph with n=200 and p=0.5

5.1.2. Non-weighted Small World graph

Duncan Watts and Steven Strogatz in 1998 proposed the idea of small world phenomenon [6]. Watts and Strogatz consider two types of graphs in their model, namely: regular graphs and random graphs. Every node has exactly the same number of neighbours in the case of regular graphs, while nodes are connected at random in random graphs. Clustering coefficient and path length are two imperative properties that are taken into account to model their small world network similar to a social network. The following steps are followed to build a small world network from a regular graph as proposed by Watts and Strogatz:

1. Beginning with a regular graph constituting *n* number of nodes and *k* neighbours
2. Choosing a subset of links to rewire and replacing them with random links. Probability *p* controls the extent to which a graph can be random where *p* is the probability of a link to be rewired. If *p*=1, the graph is completely random; *p=0* results in a regular graph. An ideal small world network with high clustering and small average distance is attained with an intermediate value of *p*.

Small world networks by Watts and Strogatz which is a non-weighted graph by default is generated using "$watts\_strogatz\_graph$", a built-in function in NetworkX library. However, since this function can return a disconnected graph, "$connected\_watts\_strogatz\_graph$" is used. Unlike "$watts\_strogatz\_graph$", this function generates a connected network by repeated generation of Watts-Strogatz small-world graphs. The function requires a set of parameters to be user defined: number of nodes (*n*), number of nearest neighbours to be connected (*k*), probability of rewiring each link (*p*), the number of attempts to generate a connected graph (tries). Figure 5.2 displays a generated unweighted Watts Strogatz graph with n=200 and *p*=0.5 and *k*=2.

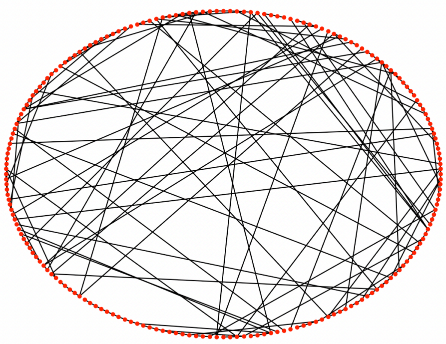


Figure 5.2: Example of the generated non-weighted Small world graph with n=200 and p=0.5

5.1.3. Non-weighted Scale free graph

The Barabasi–Albert model is one of the many proposed models that generate scale-free networks. Barabasi and Albert [12] characterized the structure of real world networks in their work “Emergence of Scaling in Random Networks” based on which scale free graph is generated in this section.

There are two essential properties of the scale free model proposed by Barabasi and Albert which differentiates them from the small world network by Watts and Strogatz, they are :

- Growth: Unlike regular graphs that start with fixed number of nodes, scale free graph by Barabasi and Albert begins with a small graph and nodes are added subsequently, one at a time. The number of nodes increase gradually with time.

- Preferential attachment: Whenever a new node is added, it is more likely to connect to a node which has connections to large number of nodes. The graph begins with m_0 initial nodes and adding a new node with $m \leq m_0$ links to connect the new node to *m* different nodes that are already existing in the graph. Nodes that have higher degree tend to have stronger ability to attract links added to a network. This phenomenon is called as ”rich gets richer” or the Matthew effect [88].

To generate scale free graph by Barabasi and Albert, Python’s NetworkX built-in function is used which returns a random graph using Barabási-Albert preferential attachment model. Through this function, a graph is grown with *n* new nodes attaching themselves to *m* links that are preferentially attached to existing nodes with high degree, where *n* is the number of nodes to be achieved and *m* is the number of links to attach from a new node to the existing nodes. The parameters *n* and *m* are user defined. Figure 5.3 displays the generated unweighted Barabasi and Albert scale graph with n=200 nodes and *m*=4.

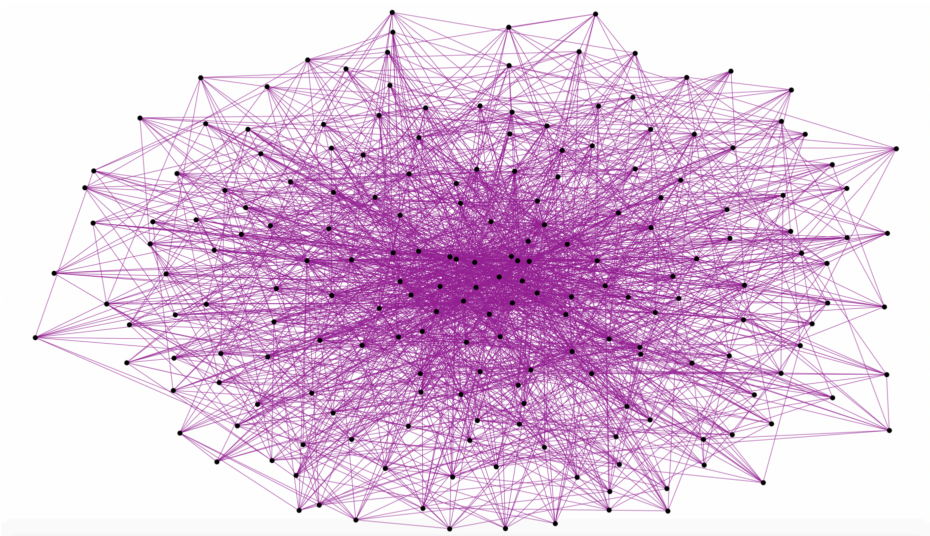


Figure 5.3: Example of the generated non-weighted Scale free graph with n=200 and m=4

5.2. Directed and non-weighted graphs

When a set of nodes is connected by directed links, where each link connects an ordered pair of nodes, it is called a directed graph or a digraph. In a directed graph, a link points from the first node in the pair and points to the second node in the pair. Simple directed graphs have no loops and no multiple arrows with same source and target nodes. This section describes the generation of directed random graphs.

5.2.1. Directed Erdos-Renyi Graph

To generate a directed Erdos-Renyi graph, the same function used to generate the unweighted Erdos-Renyi graph is used. The ”erdos_renyi_graph” function when used to generate directed Erdos-Renyi graph (by passing a ’true’ parameter for direction), uses a directed graph object to create a directed random graph of the form *G*(*n*,*p*), where *n* are the number of nodes and each possible link has probability *p* of existing. The random graph is built by connecting nodes randomly and each link is included in the graph with probability *p* independent from every other link. Figure 5.4 shows a generated directed Erdos Renyi graph with n=15 and *p*=0.5.

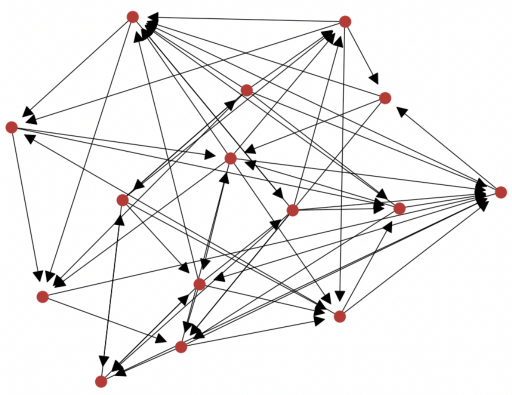


Figure 5.4: Example of a generated directed, non-weighted Erdos Renyi graph with n=15 and *p*=0.5

5.2.2. Directed Small World Graph

A directed Watts and stogartz small world network is constructed by first creating a directed ring lattice, using a directed graph object from NetworkX library. The number of nearest neighbours to be connected (*k*) and probability of rewiring each link (*p*) are user defined. Following this, the subset of links are chosen which are to be rewired and are replaced with random links to form a directed and a non-weighted small world graph. Figure 5.5 shows a generated directed and non-weighted small world graph with n=10 and k=4.

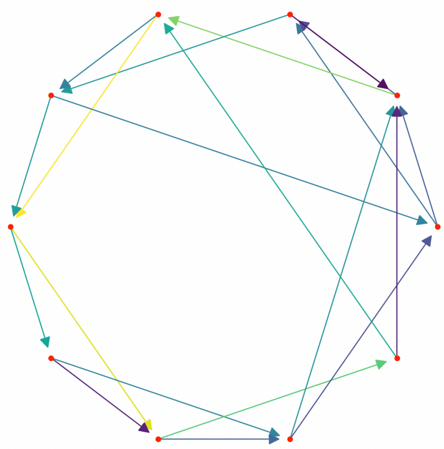


Figure 5.5: Example of a generated directed small world graph with n=10 and k=4

5.2.3. Directed Scale free Graph

A directed scale free graph is generated using the function scale_free_graph function from NetworkX, based on the paper by Bollobas *et al.* [90]. Bollobas *et al.* define three probabilities α, β, γ such that $\alpha + \beta + \gamma = 1$. Two other factors, $\delta_(in)$ and $\delta_(out)$ are also defined which are the bias for choosing nodes from in-degree distribution and out-degree distribution respectively. With α as probability, a new node *v* is added with a link from *v* to an existing node *w*. With β as probability, a link is added from an existing node *v* to an existing node *w*, where the nodes *v* and *w* are chosen independently.

With γ as probability, a new node *w* is added and a link from an existing node *v* to *w* is added. This model of scale free graph grow with preferential attachment depending on the in- and out-degrees, where the in- and out- degree distribution are power laws. Figure 5.6 shows the generated directed scale free graph with n=20.

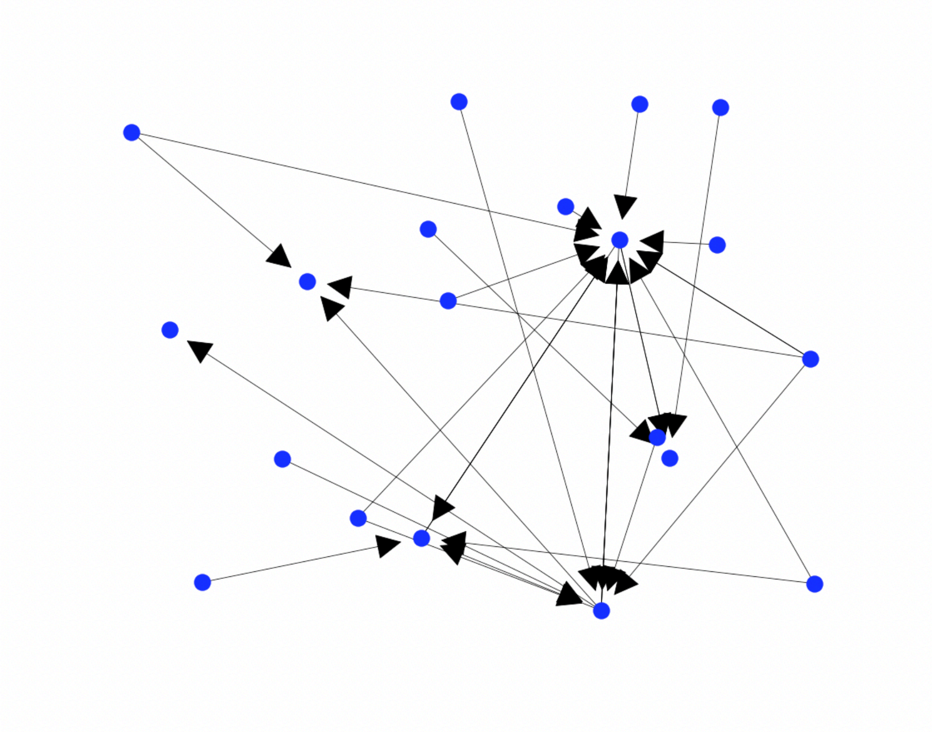


Figure 5.6: Example of a generated directed scale free graph with n=20

5.3. Weighted Graphs

Unlike a non-weighted graph which does not have relationships in terms of magnitude, a weighted graph in simple terms can be defined as a graph where each link is associated with a numerical value, called weight. A network grows by addition of new nodes and new nodes are preferentially attached to existing nodes among several neighbours. But all the links in the network cannot be viewed as the same since interaction of weights between nodes are different in many real-world networks. These weighted networks in reality have links with varying strengths. Such networks include social networks where connections between individuals could be either weak or strong; food web, metabolic networks, transportation network, business transactions, cardiovascular network are few examples of weighted networks in nature.

Newman [43] in his work has showed that better characterization can be achieved by assignment of weights to the links between scientists. Significance of weak links among species for stability of ecosystem is still being discussed [19]. Weighted networks are generally represented by a weighted adjacency matrix W which specifies the weight of the links in a network. Measure of the properties of the networks in terms of actual weight is obtained by the strength of the node which is the sum of all the link weights connected to the node. This measure provides all the information related to the node's connectivity and weight of its links. In this section, three classes of weighted random graphs are constructed and few distributions are also obtained, namely:

- Node degree distribution
- Link degree distribution (link degree is the number of incident links to the nodes on the end-points of the link, subtracted by two)
- Node strength distribution (node strength is the sum of the weights of its incident links)
- Link weight distribution

- Link strength distribution (Link strength is the sum of strength of the nodes on the end-points of the link, subtracted by two times the weight of the link)

5.3.1. Weighted Erdos Renyi Graph

The Erdos Renyi (ER) model acts as the prototype for all the unweighted graphs and is a reference for properties of real world networks. The Erdos Renyi graph of the form *G(N, L)* has *N* nodes and set of *L* links connected by a probability *p*. Most of the weighted random graphs that have been developed are either defined to imitate the properties of real networks by generalizing the rules of unweighted graphs or adding weights to the links by keeping few real world properties fixed and randomizing other aspects of the graph. Weighted Erdos Renyi graph in this thesis is also generated by adding weights to the links and aim to reproduce the properties of Erdos Renyi graph. The weighted Erdos Renyi graph is built by initially defining the required number of nodes (*n*) and desired probability (*p*) for link creation. The number of links (*m*) that is observed in the Erdos renyi graph is $m = p * n(n-1)/2$. Following this, the range for the link weights are set. Finally, using a graph object of NetworkX, an empty graph is created to add the desired nodes and links. To this empty graph, first the defined number of nodes are added. Links are then added as per probability *p* with their respective weights in a user defined range to the graph. Based on the number of links, a list of same number of weights are generated within the range that was set. Each link is then set a random weight from the list of weights that was generated. To generate a directed, weighted Erdos Renyi graph, it is constructed by following the same method but instead, a directed graph object from NetworkX is used. From the developed random graph, the above mentioned distributions are obtained which is seen in figure 5.9, 5.10, 5.11, 5.12, 5.13. The generated weighted non-directed Erdos Renyi graph is displayed in figure 5.7, with n=10 and *p*=0.5, figure 5.8 displays a weighted, directed Erdos Renyi graph with n= and p=0.5

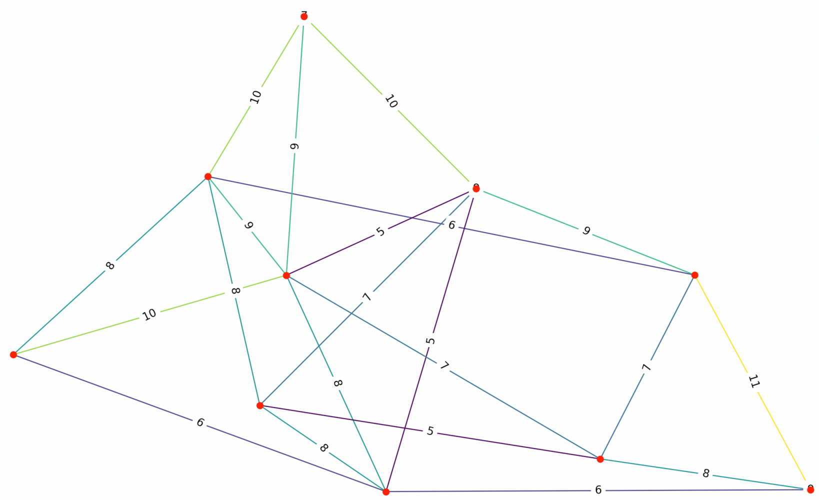


Figure 5.7: Example of a generated weighted, non-directed Erdos Renyi graph with n=10 and p=0.5

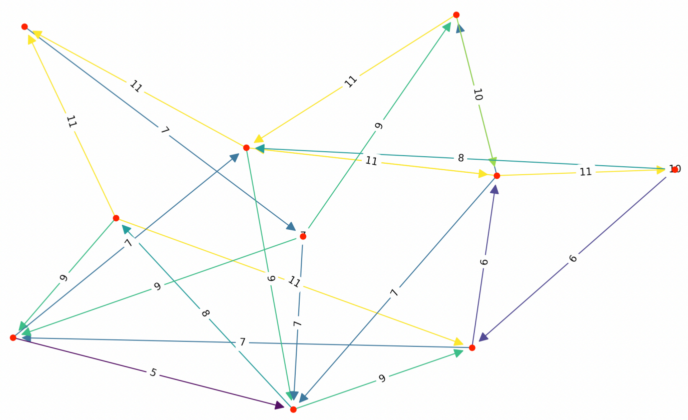


Figure 5.8: Example of a generated weighted, directed Erdos Renyi graph with n=10 and p=0.5

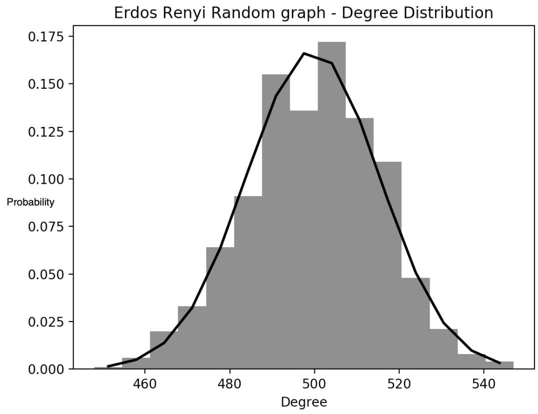


Figure 5.9: Degree distribution of Erdos Renyi graph with n=1000 and p=0.5

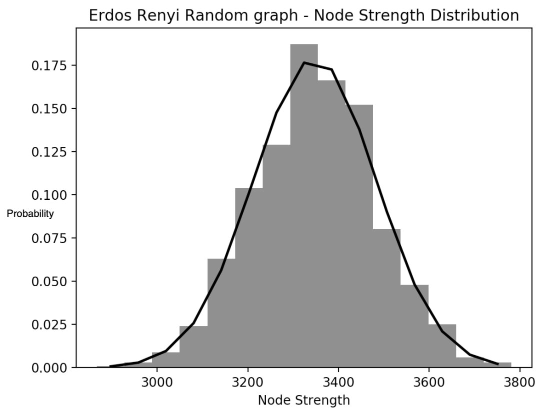


Figure 5.10: Node Strength distribution of Erdos Renyi graph with n=1000 and p=0.5

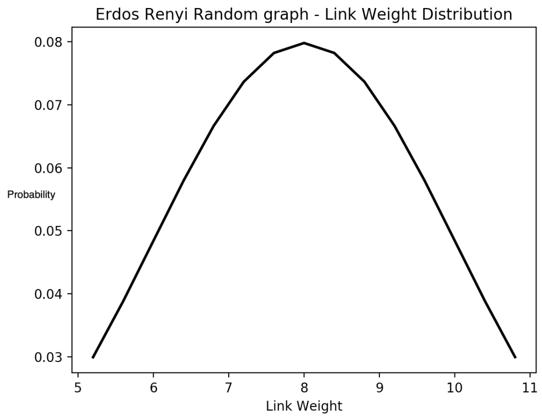


Figure 5.11: Link weight distribution of Erdos Renyi graph with n=1000 and p=0.5

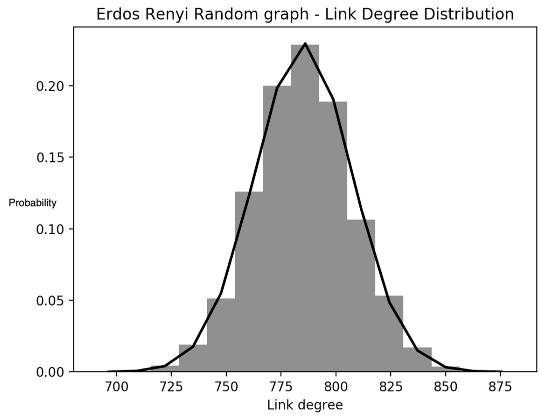


Figure 5.12: Link Degree distribution of Erdos Renyi graph with n=1000 and p=0.5

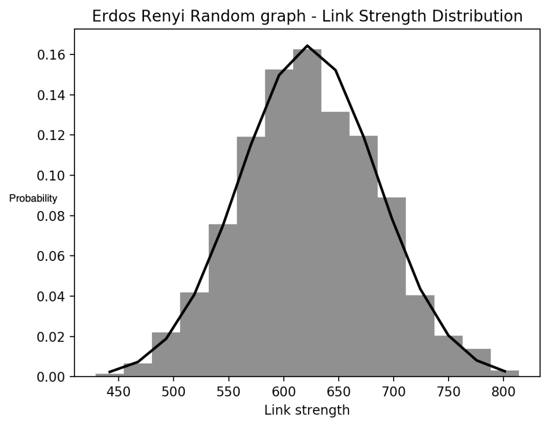


Figure 5.13: Link strength distribution of Erdos Renyi graph with n=1000 and p=0.5

5.3.2. Weighted Small world graph

In this thesis, the generation of weighted small world graph is inspired by the model explained by Watts and Strogatz following which weights are added to the links. A regular graph is constructed in the form of a ring lattice and rewired as per Watts and Strogatz. In this ring lattice consisting of n nodes, they are arranged in a circular form where each node is connected to k nearest neighbours. Watts and Strogatz in their work handle links in a particular order and rewire each of them with a probability p. When a link is rewired, the first node is left unchanged and the second node is chosen at random. Self loops or multiples links with more than one link between the same two nodes are avoided. When links are being rewired from node *u* to *v*, a replacement for node *v* is necessary. This is computed by subtracting *u* and its neighbours from the set of nodes. Following this, the new node is chosen randomly from the new node set using python's NumPy random module. Existing link between *u* and *v* is removed and the new link from *u* and the computed new node is added. Using the edge weight attribute from NetworkX library, each link is given a weight by mapping the weight to all the links in the graph from the edge weight dictionary. The edge weight dictionary consists of a list of weights that are randomly generated for all the links in the graph. In this manner, the weighted small world network is generated. To generate a weighted directed small world graph, similar procedure is followed, but the ring lattice is replaced with a directed ring lattice using a directed graph object from NetworkX library. Figure 5.14 displays a generated weighted, non-directed Watts and Stogartz small world graph with n=8 and k=4 and figure 5.15 shows a generated weighted, directed Watts and Stogartz small world graph with n=8 and k=4.

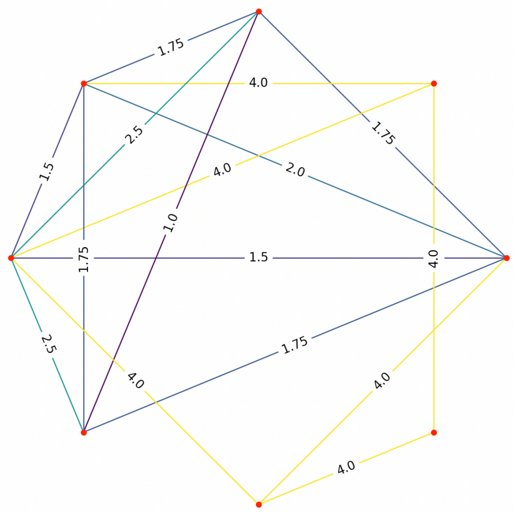


Figure 5.14: Example of a generated weighted, non-directed Watts and Stogartz small world graph with n=8 and k=4

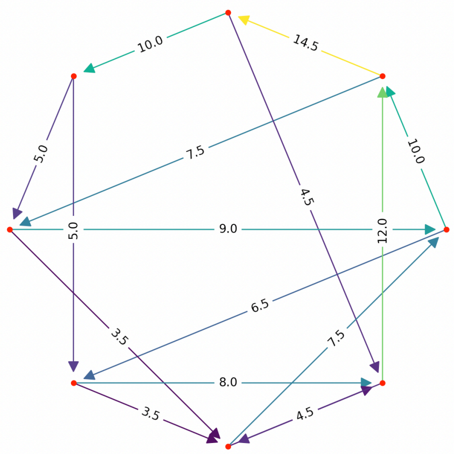


Figure 5.15: Example of a generated weighted, directed Watts and Stogartz small world graph with n=8 and k=4

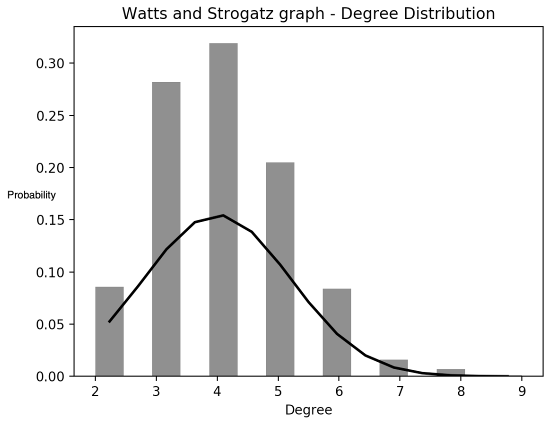


Figure 5.16: Degree distribution of Watts and Stogartz model with n=1000 and k=4

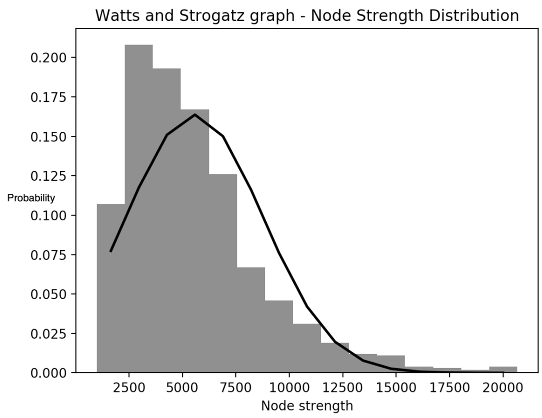


Figure 5.17: Node strength distribution of Watts and Stogartz model with n=1000 and k=4

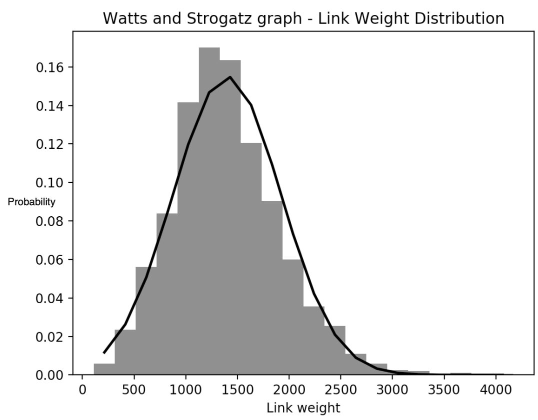


Figure 5.18: Link weight distribution of Watts and Stogartz model with n=1000 and k=4

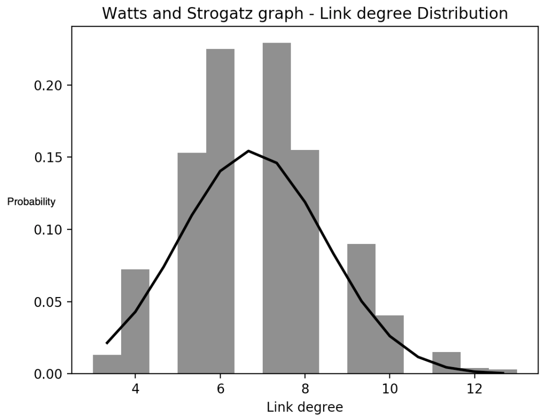


Figure 5.19: Link degree distribution of Watts and Stogartz model with n=1000 and k=4

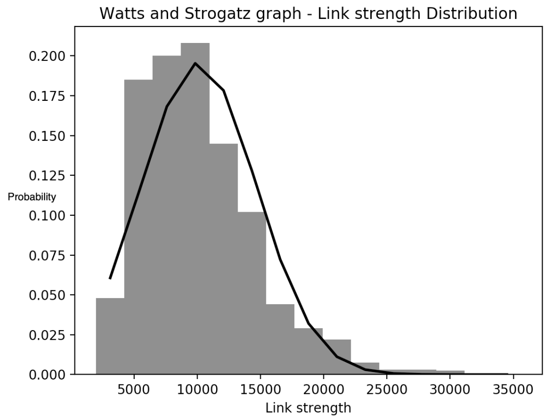


Figure 5.20: Link strength distribution of Watts and Stogartz model with n=1000 and k=4

Figures 5.16, 5.17, 5.18, 5.19, 5.20 show various distributions that are generated for the weighted small world graph.

5.3.3. Weighted Scale free graph

An algorithm was built based on Barabasi and Albert [12], to model the scale free graphs which is originally a non-weighted graph. Since we aim to generate a weighted scale free graph, weights are added to the links as a modification. The network begins with a complete graph with an initial number of nodes, n_0, using a graph object. This parameter is user defined. Final number of nodes to be achieved (*n*), the number of links to be attached (*m*) are also defined in prior. New nodes are added to the network one at a time. Each time, a new node is connected with $m \leq n_0$ links which connects the newly added node to m existing nodes with a probability that is proportional to the number of links that the existing nodes already have. Formally, the probability p_i that the new node is connected to node *i* is

$$p_i = \frac{k_i}{\sum_j k_j}$$

where k_i is the degree of node i and the sum is made over all pre-existing nodes *j* (i.e. the denominator results in twice the current number of links in the network). Heavily linked nodes (hubs) tend to quickly accumulate even more links, while nodes with only a few links are unlikely to be chosen as the destination for a new link. The new nodes have a "preference" to attach themselves to the already heavily linked nodes. When links are added, its weight attribute is also added to form a weighted scale free graph. This is done using the edge attribute function from NetworkX which maps the weight to all the links in the graph from the edge weight dictionary. To develop a directed and weighted scale free graph, similar procedure is followed except, a directed graph object is used to generate a weighted, directed scale free graph. Figure 5.21 and 5.22 shows a generated weighted, non-directed and directed Barabasi Albert Scale free graph with n=10 and m=2, respectively. Figures 5.23, 5.24, 5.25, 5.26, 5.27 show various distributions that are generated for the weighted scale free graph.

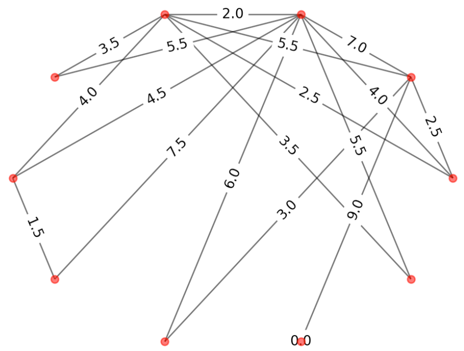


Figure 5.21: Example of a generated weighted, non-directed Barabasi Albert Scale free graph with n=10 and m=2

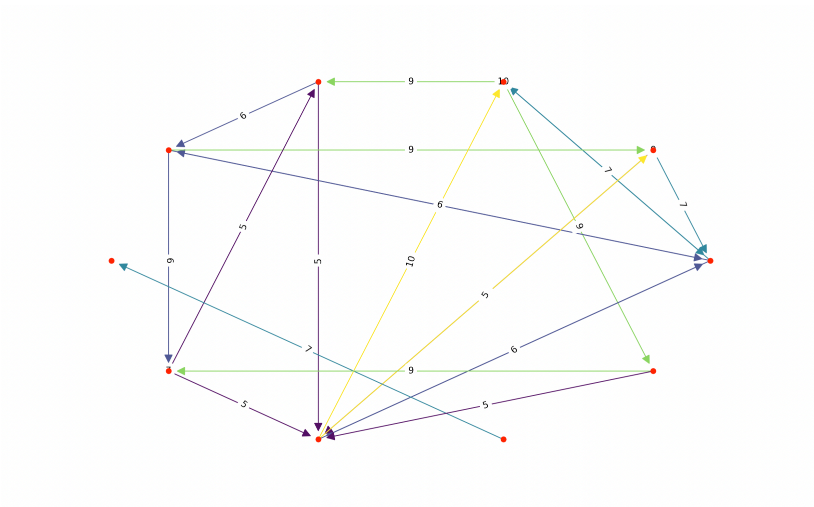


Figure 5.22: Example of a generated weighted, directed Barabasi Albert Scale free graph with n=10 and m=2

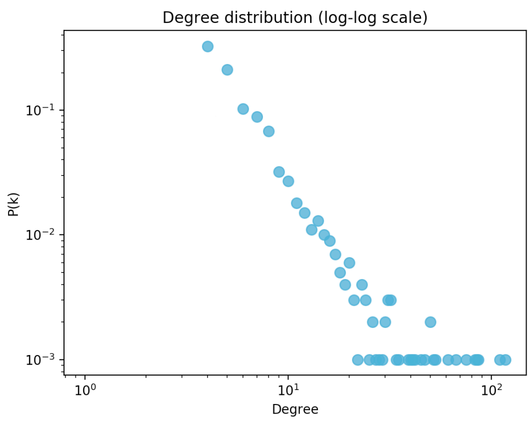


Figure 5.23: Degree distribution of Barabasi Albert Scale free graph with n=1000 and m=4

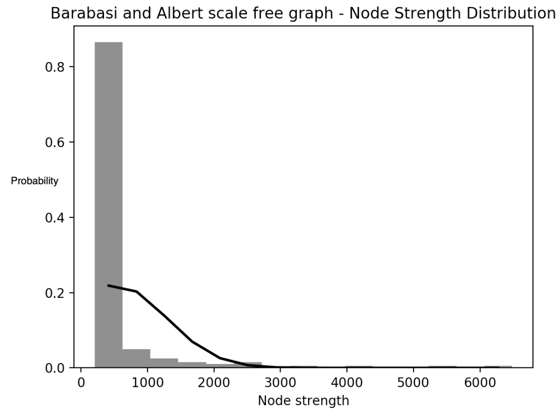


Figure 5.24: Node strength distribution of of Barabasi Albert Scale free graph with n=1000 and m=4

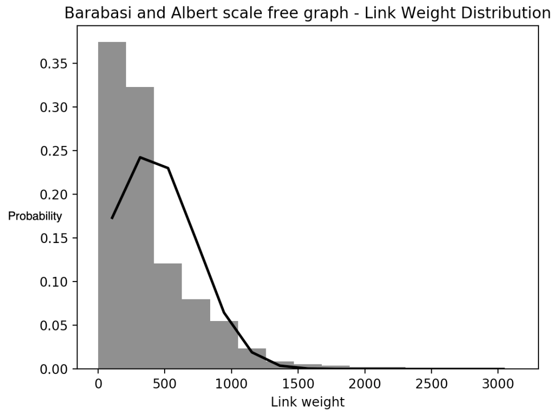


Figure 5.25: Link weight distribution of of Barabasi Albert Scale free graph with n=1000 and m=4

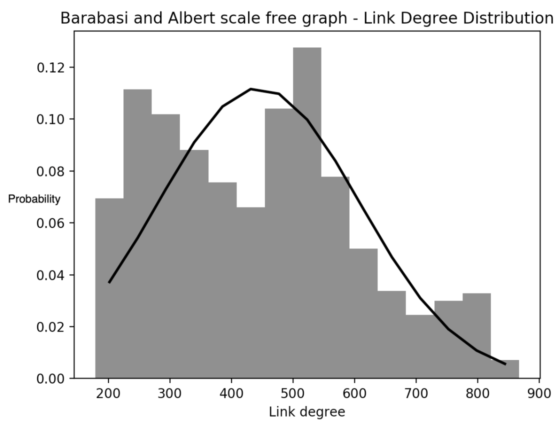


Figure 5.26: Link degree distribution of of Barabasi Albert Scale free graph with n=1000 and m=4

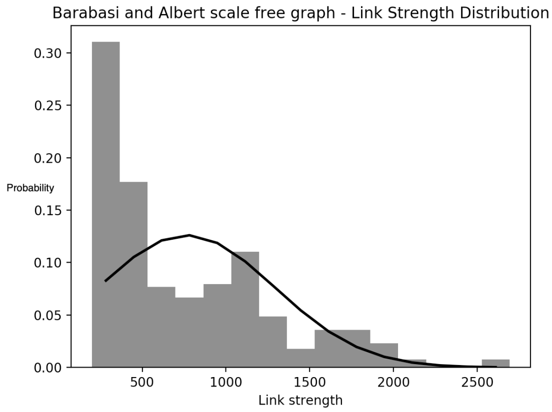


Figure 5.27: Link strength distribution of of Barabasi Albert Scale free graph with m=1000 and m=4

6
Link Weight Tolerance

This chapter introduces the concept *Link weight tolerance* in weighted graphs with positive weights. Section 6.1 provides the definition of the link weight tolerance. Section 6.2 explains the algorithm that is used to calculate the tolerance. Following this, section 6.3 illustrates the algorithm with two examples. Section 6.4 contains the observations related to link weight tolerance.

6.1. Definition

Link weight Tolerance is defined as the tolerance for the weight of a link such that, as long as the weight of the link stays within the tolerance, the shortest paths for all node pairs in the graph *G* are not affected. Since the shortest path for all node pairs are not affected, the betweenness of the nodes and the links remain unchanged. Link weight tolerance comprises two values namely- positive tolerance Z_p, the value that can be added to the link weight, and negative tolerance Z_n, which gives us the value that can be decreased from the weight of the link.

6.2. Algorithm

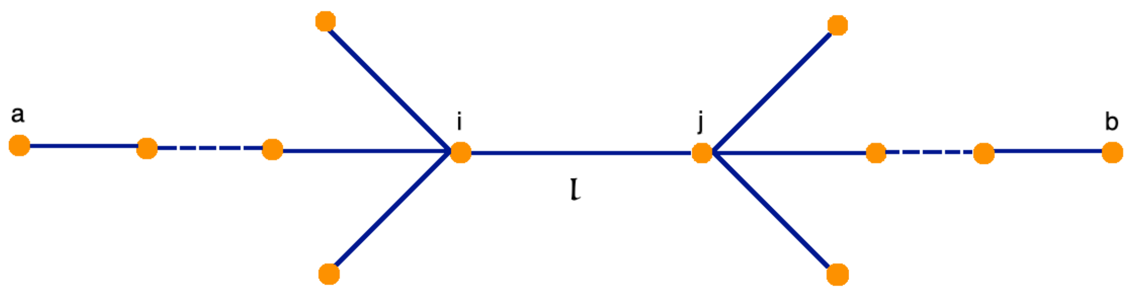


Figure 6.1: Example of a graph

To obtain the link weight tolerance of a link, we consider a weighted undirected graph *G* in figure 6.1. For a link *l*, that connects a node pair $(i \sim j)$, the number of shortest paths that traverse the link *l*, governs the betweenness of the link. Each shortest path that traverses the link *l* connects two nodes in the network. We know that the betweenness of link *l* is the accumulation of all betweenness contributions caused by shortest paths between all node pairs in the network. The process of calculating the link weight tolerance of a link is divided into two parts:

1. Positive link weight tolerance Z_p

2. Negative link weight tolerance Z_n

6.2.1. Positive Link weight tolerance

To calculate positive link weight tolerance of link *l*, let us consider a shortest path between an arbitrary node pair, node *a* and node *b* in graph *G*. The shortest path between the node pair may or may not traverse link *l*. We are interested only in the shortest path that traverses *l*, the principle is that when the weight of link *l* increases, at certain point the shortest path between node *a* and node *b* will no longer traverse link *l*, but will take another path, which does not include link *l*, provided that such other path exists. If a shortest path runs through link *l* to connect an arbitrary node pair, positive link weight tolerance, Z_p tells us the value that can be added to the weight of link *l*, to maintain the same shortest path. For a link *l*, the shortest path connecting an arbitrary node pair continues running through that link when the weight of the link is increased with a value less than or equal to Z_p.

Let us consider the network in figure 6.1 as graph *G*. To calculate the positive link weight tolerance for link *l* ($i \sim j$), the shortest path between node *a* and node *b* is calculated. The shortest path between the node pair is calculated using the Dijkstra algorithm. The Dijkstra algorithm is used to find a shortest path from a source node to a target node in a weighted graph. The algorithm creates a tree of shortest path from the source node to all other nodes in the graph [91]. We term the length of the shortest path between node *a* and node *b* in graph *G* as X_p.

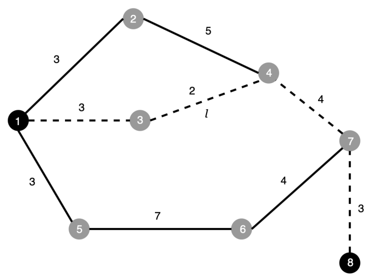
To measure the shortest path between node *a* and node *b* when the shortest path does not traverse link *l*, we remove the link *l* from graph *G*. This graph is now termed as *G*', where $G' = G - l$. The length of the shortest path between node *a* and node *b* in graph *G*' is termed as Y_p where $Y_p \geq X_p$. The difference in the length of the shortest paths in graph *G* and *G*' is $Z_p = Y_p - X_p$, which gives the positive link weight tolerance for link *l*, relative to node pair (*a*,*b*).

The positive link weight tolerance Z_p for link *l* is determined relative to all node pairs (*a,b*) in the graph. The lowest Z_p value that is obtained for link *l* when calculated for all the node pairs (*a,b*) which traverses link *l* is the resultant positive link weight tolerance respective to it. Positive link weight tolerance matrix P is defined where each element in the matrix represents the positive link weight tolerance for the respective node pair relative to link *l*. The matrix ZP defines positive weight tolerance for all the links in a graph, where the index of the matrix corresponds to the respective link. While calculating positive link weight tolerance we consider three cases:

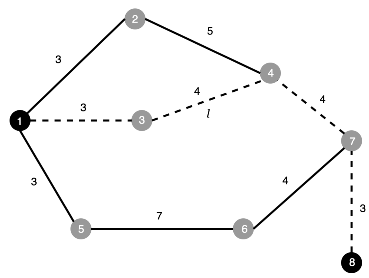
1. Case 1: The shortest path connecting a node pair (*a,b*) does not traverse link *l*. In such cases, the shortest path between the node pair (*a,b*) does not contribute to the betweenness of link *l*. Therefore, increase in the weight of link *l* does not affect the shortest path between the node pair. Positive link weight tolerance value Z_p for such node pairs relative to link is assigned as ∞.

2. Case 2: There are more than one unique shortest paths connecting a node pair (*a,b*) - i.e. at least one shortest path traverses link *l*, and the shortest paths do not traverse link *l*. In such cases, the positive link weight tolerance value Z_p relative to link *l* is zero since, if the weight of link *l* increases even by 1 (the smallest positive unit of weight in weighted graphs), the shortest path between the node pair will not traverse link *l* anymore and will take the other shortest path(s).

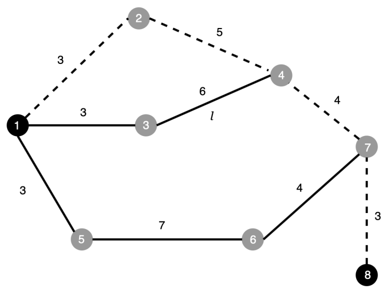
3. Case 3: There is only one shortest path between the node pair (*a,b*) and it traverses link *l*. In such cases, the positive link weight tolerance value Z_p is calculated using the method $Z_p = Y_p - X_p$ as explained.



(a) Example of a weighted graph *G*, where the shortest path X_p is indicated by a dotted path



(b) Example of a weighted graph *G*, where the shortest path remains unchanged when the weight of link *l* is increased with a value less than Z_p



(c) Example of a weighted graph *G*, where the shortest path changes when the weight of link *l* is increased with a value beyond Z_p

Figure 6.2: Example of how the shortest path changes in a weighted graph when the weight of link *l* increases

When the weight of link *l* increased by adding a value above Z_p, the shortest path between node *a* and node *b* no longer runs through link *l*. Figure 6.2 (a) displays an example of the shortest path X_p (indicated as a dotted path) between the node pair (1,8) which runs through the link $(3 \sim 4)$. Figure 6.2 (b) shows us an unchanged shortest path between the node pair (1,8) when the weight of link *l* $(3 \sim 4)$ is increased within the value of Z_p. Figure 6.2 (c) shows a different shortest path between (1,8) when the weight of link *l* $(3 \sim 4)$ is increased with a value beyond the value of Z_p, the shortest path does not traverse link *l* $(3 \sim 4)$ anymore. When weight of a link is increased with a value less than Z_p, the betweenness centrality of that link is maintained.

6.2.2. Negative link weight tolerance

To calculate the negative link weight tolerance, we consider graph *G* again where the shortest path between the node pairs (*a,b*) does not traverse link *l*. The principle behind negative link weight tolerance is that when the weight of link *l* decreases, then at a certain point the shortest path between node *a* and node *b* starts to run through link *l*. Therefore, it is necessary to calculate the extent to which the weight of link *l* has to be decreased such that the shortest path between node *a* and node *b* begins to run through link *l*. For an arbitrary node pair (*a,b*), the shortest path between them may or may not traverse link *l*. We are interested only in the shortest path that does not traverse link *l*, so that we can determine the extent to which the weight of the link *l* has to be decreased, so that the shortest path between a node pair (*a,b*) which did not traverse *l* previously, begins to traverse it. For the node pairs whose shortest path already traverses *l*, negative link weight tolerance is not necessary since decreasing the weight of link *l* will not affect their shortest path.

This process is carried out by determining the shortest path Y_n between any node pair node (*a,b*) traversing link *l*. This shortest path Y_n can be calculated in two ways-

1. Y_n = (the shortest path between node *a* and node *i*) + (the weight of link *l*) + (the shortest path between node *j* and node *b*), where $(i \sim j)$ is link *l*.

   i.e. The path is -> a-i-j-b

2. Y_n = (the shortest path between node *a* and node *j*) + (the weight of link *l*) + (the shortest path between node *i* and node *b*), where $(i \sim j)$ is link *l*. i.e. The path is -> a-j-i-b

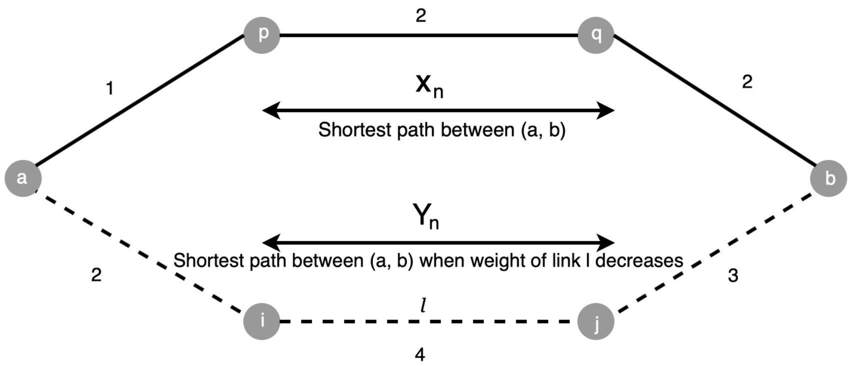


Figure 6.3: Illustration of X_n and Y_n in a weighted graph to calculate negative link weight tolerance

Figure 6.3 displays a weighted graph where the shortest path Y_n via link *l* is shown as the dotted path and the shortest path X_n is shown in black between the node pair (*a,b*). Both these values are calculated, the least of the two value is set as Y_n. Following this, the shortest path X_n is calculated between node *a* and node *b* in graph *G* using the Dijkstra's algorithm. The negative link weight tolerance Z_n is the difference in the lengths Y_n and X_n, $Y_n \geq X_n$ is calculated for all the node pairs (*a,b*) in the graph *G*, $Z_n = Y_n - X_n$. The negative weight tolerance values relative to each node pair for link *l* are represented in a negative link weight tolerance matrix, N. The lowest value obtained is the negative link weight tolerance with respect to link *l*. Matrix ZN is defined to represent negative link weight tolerance for all the links in a graph, where the index of the matrix corresponds to the respective link.

When the value for Z_n for a particular link, is calculated for all node pairs in the graph, some values can be greater than the weight of link *l* itself. This tells us that for some node pairs, the shortest path between them will traverse link *l* only when the weight of link *l* becomes negative. For such cases, when Z_n is greater than the weight of link *l*, the value of Z_n for that node pair is assigned the value of the weight of link *l*. So, the negative weight tolerance of a link can never be greater than the weight of the link itself.

6.3. Illustration with an example

The illustration of the algorithm is carried out with two examples of weighted graphs.

6.3.1. Example 1

Let us a consider a weighted graph with 6 nodes to calculate the weight tolerance of the link $(3 \sim 4)$ from the weighted graph displayed in figure 6.4. The link $(3 \sim 4)$ has the highest link betweenness compared to other links in the graph, from the table 6.1. The first part of the process is to calculate the positive link weight tolerance. Let us consider the weighted graph in figure 6.4 as graph *G*, and obtain the length of the shortest path between each node pair in *G*, which we term as X_p. The shortest path matrix is also computed, which is a square matrix representing shortest path between two nodes as each element in the matrix *X*, as shown below-

$$X = \begin{bmatrix} 0 & 1 & 8 & 4 & 3 & 9 \\ 1 & 0 & 7 & 3 & 2 & 8 \\ 8 & 7 & 0 & 4 & 5 & 1 \\ 4 & 3 & 4 & 0 & 1 & 5 \\ 3 & 2 & 5 & 1 & 0 & 6 \\ 9 & 8 & 1 & 5 & 6 & 0 \end{bmatrix} \tag{6.1}$$

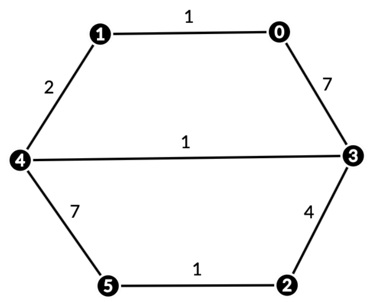


Figure 6.4: An undirected weighted graph with 6 nodes

Link	Link betweenness
(0,1)	0.333
(0,3)	0.0
(1,4)	0.533
(2,3)	0.533
(2,5)	0.333
(3,4)	0.6
(4,5)	0.0

Table 6.1: Link betweenness for graph in figure 6.4

After calculating all the shortest path length X_p for graph *G*, the link $(3 \sim 4)$ for which the tolerance is being calculated is removed from *G*. We call the new graph in which the link $(3 \sim 4\ )$ is absent as graph *G*'. The length of the shortest path is calculated again for all node pairs with the absence of the link $(3 \sim 4)$, termed as Y_p. Table 6.2 tabulates the shortest paths for the graphs *G* and *G*'.

Node pairs	Does the shortest path traverse link $(3 \sim 4)$	X_p	Y_p
(0,1)	No	1	1
(0,2)	Yes	8	11
(0,3)	Yes	4	7
(0,4)	No	3	3
(0,5)	Yes	9	10
(1,2)	Yes	7	10
(1,3)	Yes	3	8
(1,4)	No	2	2
(1,5)	Yes	8	9
(2,3)	No	4	4
(2,4)	Yes	5	8
(2,5)	No	1	1
(3,4)	Yes	1	10
(3,5)	No	5	5
(4,5)	Yes	6	7

Table 6.2: Positive link weight tolerance calculation

In Table 6.2, we can notice that not all node pairs depend on link $(3 \sim 4)$ to acquire the shortest path between them. Since not every node pair depends on link $(3 \sim 4)$ to achieve the shortest path, we consider only the node pairs whose shortest path traverses link $(3 \sim 4)$ to understand the extent to which the weight of the link $(3 \sim 4)$ can be increased to maintain the same shortest path and therefore the same betweenness. The node pairs that have same value of X_p and Y_p do not have their shortest path traverse link $(3 \sim 4)$ and therefore do not contribute to the link betweenness centrality for $(3 \sim 4)$. Thus, the positive link weight tolerance for such node pairs with respect to link $(3 \sim 4)$ is ∞

Node pairs	X_p	Y_p	Z_p
(0,2)	8	11	3
(0,3)	4	7	3
(0,5)	9	10	1
(1,2)	7	10	3
(1,3)	3	8	5
(1,5)	8	9	1
(2,4)	5	8	3
(3,4)	1	10	9
(4,5)	6	7	1

Table 6.3: Positive link weight tolerance calculation

Table 6.3 tabulates the Z_p, which is the difference between Y_p and X_p for the node pairs that are affected by the removal of link $(3 \sim 4)$. The lowest value of Z_p from table 6.3 is 1, which tells us that the weight of the link $(3 \sim 4)$ can be increased by 1 unit to retain all the shortest paths passing through it. However, link betweeness centrality of a link is maintained when the weight of the link is increased by a value less than Z_p. Therefore, the weight of the link $(3 \sim 4)$ can be increased by a value less than 1, to maintain the same link betweenness. The positive tolerance for each node pair respective to link $(3 \sim 4)$ is represented through a square matrix as below-

$$P = \begin{bmatrix} 0 & \infty & 3 & 3 & \infty & 1 \\ \infty & 0 & 3 & 5 & \infty & 1 \\ 3 & 3 & 0 & \infty & 3 & \infty \\ 3 & 5 & \infty & 0 & 9 & \infty \\ \infty & \infty & 3 & 9 & 0 & 1 \\ 1 & 1 & \infty & \infty & 1 & 0 \end{bmatrix} \tag{6.2}$$

To calculate negative link weight tolerance, the first step is to calculate the length of the shortest path Y_n through link $(3 \sim 4)$. The value of Y_n for link $(3 \sim 4)$ is calculated for all the node pairs in the graph. Following this, we calculate the shortest path X_n among the same node pairs, which may or may not traverse link $(3 \sim 4)$. The table 6.4 summarizes the calculated X_n, Y_n for all the node pairs in the graph.

Node pairs	Does the shortest path traverse $(3 \sim 4)$	Y_n	X_n
(0,1)	No	7	1
(0,2)	Yes	8	8
(0,3)	Yes	4	4
(0,4)	No	5	3
(0,5)	Yes	9	9
(1,2)	Yes	7	7
(1,3)	Yes	3	3
(1,4)	No	4	2
(1,5)	Yes	8	8
(2,3)	No	6	4
(2,4)	Yes	5	5
(2,5)	No	11	1
(3,4)	Yes	1	1
(3,5)	No	7	5
(4,5)	Yes	6	6

Table 6.4: Negative link weight tolerance calculation

From table 6.4, we are interested in the node pairs whose shortest path does not traverse the link $(3 \sim 4)$. Table 6.5 summarizes the list of those node pairs and the calculated Z_n. Since the difference between Y_n and X_n is greater than the weight of link *l* $(3 \sim 4)$, Z_n for the node pairs is assigned the value of the link weight, which is one. From table 6.5 we can observe that, the lowest Z_n value is one, which becomes the negative link weight tolerance respective to link $(3 \sim 4)$.

Node pairs	Y_n	X_n	Z_n
(0,1)	7	1	1
(0,4)	5	3	1
(1,4)	4	2	1
(2,3)	6	4	1
(2,5)	11	1	1
(3,5)	7	5	1

Table 6.5: Negative link weight tolerance calculation

Since we consider the weights of the links to be positive, in cases when the difference between Y_n and X_n is greater than the weight of link, we assign value of Z_n as weight of the link itself. This is because, if the difference between Y_n and X_n is greater than the weight of link, the shortest path will begin to traverse the link only when the weight of the link is negative. Therefore, to retain the value of link weight as positive, as seen in table 6.5, Z_n for this case is not defined as $Y_n - X_n$, but is given the value of the weight of link $(3 \sim 4)$, which is 1. The negative link weight tolerance for all node pairs respective to link $(3 \sim 4)$ is represented as elements of a square matrix is given as follows-

$$N = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{6.3}$$

The node pairs whose shortest path traverse link $(3 \sim 4)$, their Z_n is zero since their respective X_n and Y_n values are the same. The node pairs whose shortest does not traverse link $(3 \sim 4)$, and when the difference between their Y_n and X_n is greater than the weight of link $(3 \sim 4)$, their Z_n is one (weight of link $(3 \sim 4)$). If the difference between their Y_n and X_n is less than the weight of link $(3 \sim 4)$, then their $Z_n = Y_n - X_n$, but such node pairs do not exist in this example. Table 6.6 lists Z_p and Z_n values for all node pairs for the graph in figure 6.4 relative to link $(3 \sim 4)$

Node pairs	Z_p	Z_n
(0,1)	∞	1
(0,2)	3	0
(0,3)	3	0
(0,4)	∞	1
(0,5)	1	0
(1,2)	3	0
(1,3)	5	0
(1,4)	∞	1
(1,5)	1	0
(2,3)	∞	1
(2,4)	3	0
(2,5)	∞	1
(3,4)	9	0
(3,5)	∞	1
(4,5)	1	0

Table 6.6: Summary of Z_p and Z_n values for all node pairs for the graph in figure 6.4 relative to link $(3 \sim 4)$

Table 6.7 further summarizes the maximum and minimum values of positive and negative link weight tolerance for all the links present in the graph displayed in figure 6.4. Matrix ZP and ZN are defined for all the links in the graph.

Link	Link betweenness	Z_p Maximum	Z_p Minimum	Z_n Maximum	Z_n Minimum
(0,1)	0.333	9	3	1	0
(0,3)	0.0	∞	∞	7	3
(1,4)	0.533	7	3	2	0
(2,3)	0.533	5	1	4	0
(2,5)	0.333	11	1	1	0
(3,4)	0.6	9	1	1	0
(4,5)	0.0	∞	∞	7	1

Table 6.7: Link betweenness, Z_p and Z_n values for all the links for the graph in figure 6.2

$$ZP = \begin{bmatrix} . & 3 & . & \infty & . & . \\ 3 & . & . & . & 3 & . \\ . & . & . & 1 & . & 1 \\ \infty & . & 1 & . & 1 & . \\ . & 3 & . & 1 & . & \infty \\ . & . & 1 & . & \infty & . \end{bmatrix} \tag{6.4}$$

$$ZN = \begin{bmatrix} . & 0 & . & 3 & . & . \\ 0 & . & . & . & 0 & . \\ . & . & . & 0 & . & 0 \\ 3 & . & 0 & . & 0 & . \\ . & 0 & . & 0 & . & 1 \\ . & . & 0 & . & 1 & . \end{bmatrix} \tag{6.5}$$

Figure 6.5 and 6.6 display the maximum and minimum values of positive and negative link weight tolerance for the graph in figure 6.4, respectively. Maximum Z_p for a link is the maximum tolerance value which can be added to the weight of a link, such that at least one shortest path between any node pair traverses it. Minimum Z_p for a link is the minimum value that can be added to the weight of the link, such that all the shortest paths that initially traversed the link, continues to traverse it. Minimum Z_p for a link is the positive weight tolerance of that link. Similarly, minimum Z_n for a link is the negative tolerance weight tolerance for that link, such that when this value is reduced from the weight of the link, the node pairs for which the shortest path did not traverse the link previously, at least one of the shortest path begins to traverse the link. Maximum Z_n for a link is the weight of the link itself since the negative tolerance cannot exceed the weight of the link.

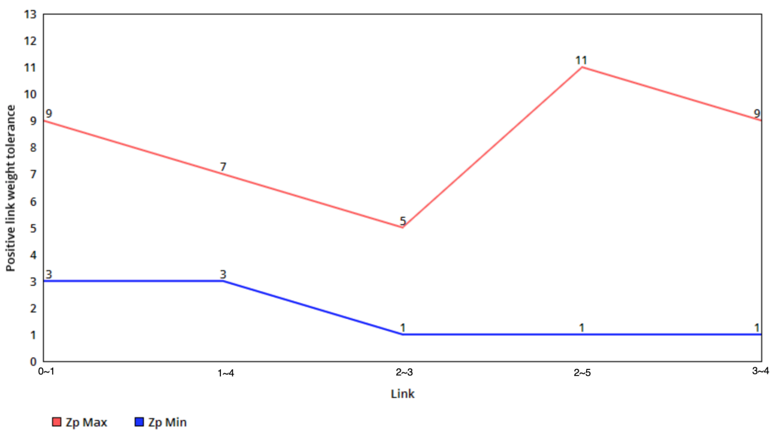


Figure 6.5: Maximum and minimum values of Z_p for figure 6.4

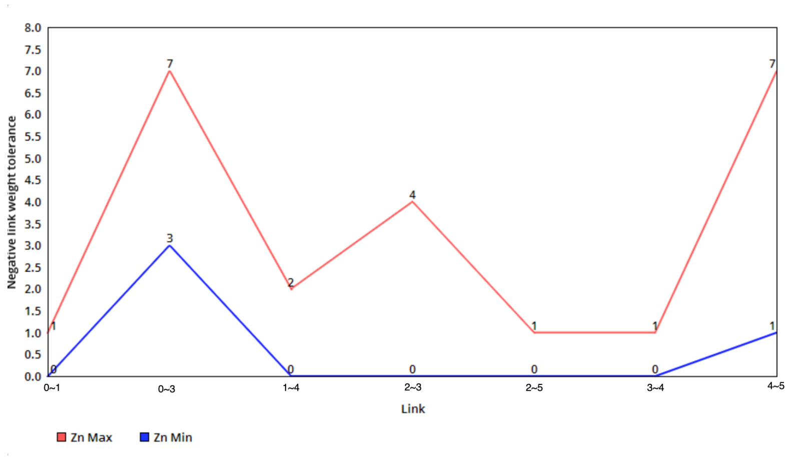


Figure 6.6: Maximum and minimum values of Z_n for figure 6.4

6.3.2. Example 2

Let us consider another example to illustrate the link weight tolerance, in a weighted graph of 5 nodes for a link with the smallest link betweenness. For the weighted graph shown in figure 6.7, the link $(3 \sim 4)$ has the smallest link betweenness as shown in table 6.8.

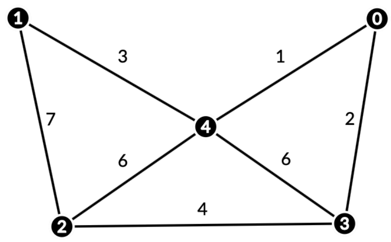


Figure 6.7: Example of an undirected weighted graph

The shortest path matrix for the weighted graph in figure 6.7 is represented as below-

$$X = \begin{bmatrix} 0 & 4 & 6 & 2 & 1 \\ 4 & 0 & 7 & 6 & 3 \\ 6 & 7 & 0 & 4 & 6 \\ 2 & 6 & 4 & 0 & 3 \\ 1 & 3 & 6 & 3 & 0 \end{bmatrix} \quad (6.6)$$

Link	Link betweenness
(0,3)	0.4
(0,4)	0.4
(1,2)	0.1
(1,4)	0.3
(2,3)	0.2
(2,4)	0.1
(3,4)	0.0

Table 6.8: Link betweenness for graph in figure 6.7

Similar procedure as in example 1 is followed to calculate X_p, Y_p and Z_p. Table 6.9 displays the X_p and Y_p values for each node pair.

Node pairs	Does the shortest path traverse link $(3 \sim 4)$	X_p	Y_p
(0,1)	No	4	4
(0,2)	No	6	6
(0,3)	No	2	2
(0,4)	No	1	1
(1,2)	No	7	7
(1,3)	No	6	6
(1,4)	No	3	3
(2,3)	No	4	4
(2,4)	No	6	6
(3,4)	No	3	3

Table 6.9: Positive link weight tolerance calculation

As seen in table 6.9, for all the node pairs, the shortest path does not traverse the link $(3 \sim 4)$ and thus does not contribute to its link betweenness. Therefore, the positive link weight tolerance Z_p for all the node pairs is ∞.

$$P = \begin{bmatrix} 0 & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{bmatrix} \tag{6.7}$$

The procedure as in example 1 is repeated to obtain negative link weight tolerance for link $(3 \sim 4)$ of graph *G*.

Node pairs	Does the shortest path traverse link $(3 \sim 4)$	Y_n	X_n
(0,1)	No	11	4
(0,2)	No	11	6
(0,3)	No	7	2
(0,4)	No	8	1
(1,2)	No	13	7
(1,3)	No	9	6
(1,4)	No	12	3
(2,3)	No	12	4
(2,4)	No	10	6
(3,4)	No	6	3

Table 6.10: Negative link weight tolerance calculation

From table 6.10, since link $(3 \sim 4)$ does not lie in the shortest path for any of the node pairs, all of them are considered to calculate the negative link weight tolerance.

Node pairs	Y_n	X_n	Z_n
(0,1)	11	4	6
(0,2)	11	6	5
(0,3)	7	2	5
(0,4)	8	1	6
(1,2)	13	7	6
(1,3)	9	6	3
(1,4)	12	3	6
(2,3)	12	4	6
(2,4)	10	6	4
(3,4)	6	3	3

Table 6.11: Negative link weight tolerance calculation

Table 6.11 lists the shortest path lengths and negative tolerance for each node pair respective to link $(3 \sim 4)$. The negative link weight tolerance for link $(3 \sim 4)$ in matrix form is given as-

$$N = \begin{bmatrix} 0 & 6 & 5 & 5 & 6 \\ 6 & 0 & 6 & 3 & 6 \\ 5 & 6 & 0 & 6 & 4 \\ 5 & 3 & 6 & 0 & 3 \\ 6 & 6 & 4 & 3 & 0 \end{bmatrix} \tag{6.8}$$

The lowest value in the Z_n is 3, which gives us the negative link weight tolerance respective to link $(3 \sim 4\ )$. Table 6.12 further summarizes maximum and minimum values of positive and negative link weight tolerance for all the links present in the graph displayed in figure 6.7. Matrix ZP and ZN are defined for all the links in the graph.

Link	Link betweenness	Z_p Maximum	Z_p Minimum	Z_n Maximum	Z_n Minimum
(0,3)	0.4	5	1	2	0
(0,4)	0.4	7	3	1	0
(1,2)	0.1	2	0	7	0
(1,4)	0.3	10	5	3	0
(2,3)	0.2	5	1	4	1
(2,4)	0.1	1	0	6	1
(3,4)	0.0	∞	∞	6	3

Table 6.12: Link betweenness, Z_p and Z_n values for all the links for the graph in figure 6.7

$$ZP = \begin{bmatrix} . & . & . & 1 & 3 \\ . & . & 0 & . & 5 \\ . & 0 & . & 1 & 0 \\ 1 & . & 1 & . & \infty \\ 3 & 5 & 0 & \infty & . \end{bmatrix} \tag{6.9}$$

$$ZN = \begin{bmatrix} . & . & . & 0 & 0 \\ . & . & 0 & . & 0 \\ . & 0 & . & 1 & 1 \\ 0 & . & 1 & . & 3 \\ 0 & 0 & 1 & 3 & . \end{bmatrix} \tag{6.10}$$

Figure 6.8 and figure 6.9 display the maximum and minimum values of positive and negative link weight tolerance for the graph in figure 6.7, respectively.

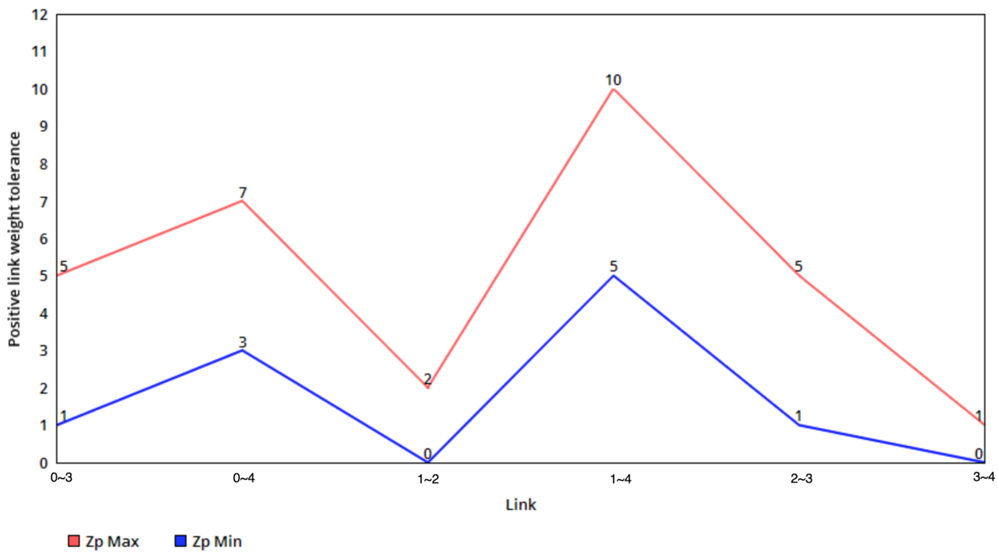


Figure 6.8: Maximum and minimum values of Z_p for figure 6.7

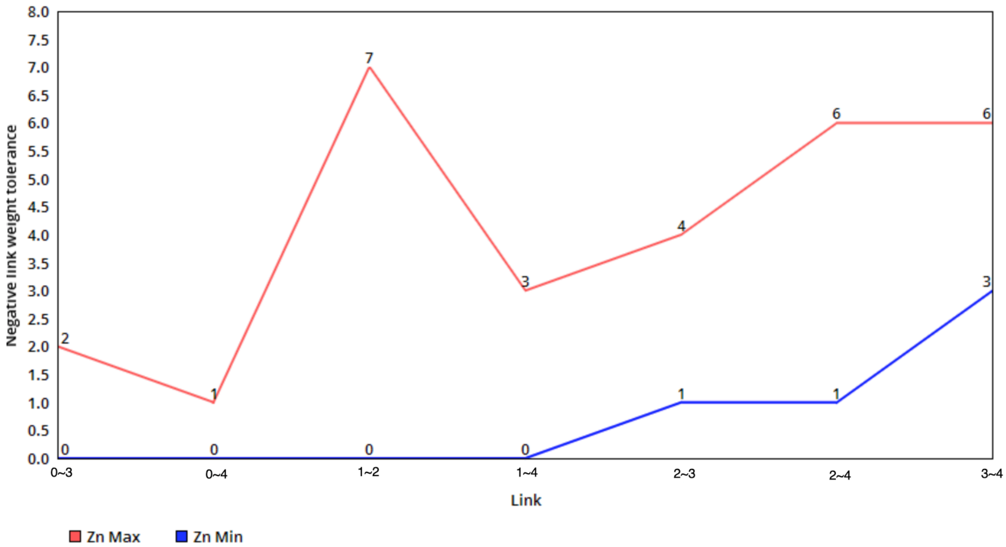


Figure 6.9: Maximum and minimum values of Z_n for figure 6.7

6.4. Observations

From the above two examples, we can observe that the links with high link betweenness have a high positive link weight tolerance but a low negative link weight tolerance. While a link with zero link betweenness cannot have a positive tolerance, i.e. Z_p for such a link is ∞ since that link does not contribute to the shortest path between the node pairs in a network. But such links have a high negative link weight tolerance since its link weight has to be decreased by Z_n for the shortest path to traverse the link. To summarize,

- When Z_p of a link is ∞, it implies that no shortest path traverses the link i.e. when a link *l* with one or more shortest paths running through it, and for each node pair that has its shortest path running through that link, there are no other shortest paths when we would remove that link, then Z_p for that link is ∞.
- When Z_p of a link is zero, it implies that when the weight of the link is increased even by one unit, the shortest path between some pairs that initially traversed the link, will stop traversing it. Z_p=0 is also observed for a link when for one or more of the shortest paths for a node pair (a,b) running through that link, there are other shortest paths for that node pair (a,b) not traversing that link.
- When Z_p of a link is any other positive value, it is the minimum value that can be added to the weight of the link, to retain all the shortest path traversing the link i.e. it maintains the same link betweenness of the link.
- When Z_n of a link is zero, it means that the shortest paths between all the node pairs already traverses the link. This is observed for links with high link betweenness. Z_n=0 is also observed for a link when for one or more of the shortest paths for a node pair (a,b) running through that link, there are other shortest paths for that node pair (a,b) not traversing that link.
- When Z_n of a link is the value of its link weight, it implies that a shortest path will never traverse the link irrespective of how much the weight of the link is decreased.
- When Z_n of a link is any other positive value, it means that it is the minimum value that has to be decreased from the link weight, such that a shortest path begins to traverse the link.

Positive and negative link weight tolerance therefore tells us the extent to which the weight of the link can be increased or decreased without changing the link betweenness of a link and also tells us the link weight at which a shortest path can begin or cease to traverse the link.

7

Link Tension in Weighted Graphs

This chapter introduces a new measure for weighted graphs, *Link tension*. Section 7.1 provides a definition for link tension and section 7.2 provides link tension distribution graphs for three classes of weighted random graph models. Section 7.3 contains a generic evaluation for link tension.

7.1. Definition

In a graph, the ability of a node to communicate with various other nodes is inferred from its degree. The greater the degree of a node, the higher is its connectability with other nodes as the number of links emerging from the node is also greater. The connectability of the nodes can also be expressed through the strength of the nodes in a weighted graph, which can be interpreted in two different manners. Strength of a node s_i was given a conventional definition by De Montis in [20], where s_i is defined as the sum of the weights of the links incident to the node. Since the strength of a node quantifies the node connectability, there can be another interpretation for the weight of a link which in turn defines the strength of a node. In the case of weighted graphs where the weight of a link represents its distance or cost, we can point out that the higher the weight of a link, the smaller are the chances for the link to add to the connectability of a node on an end of the link. The probability for the link to add to the amount of connectability of a node is inversely proportional to the link weight incident to that node. Therefore, the second interpretation of the strength of a node is the sum of the inverse of the respective link weights and not the sum of the weights of the incident links [43]. In a weighted network, link weight is not a sufficient attribute to assess the significance of the link. There is a need for quantization of link significance to generate efficient and robust networks. Usually, Link betweenness, which is a metric based on shortest paths, is the measure predominantly used to characterize the importance of a link.

The measurement of the importance of a link can be extended from link betweenness to link tension, T_l. Link tension takes into account both link weight and weighted link betweenness to rank the importance of a link i.e. to know the most significant link and the least significant link in a network. The metric link tension is defined as,

$$T_l = C_B^l * w_l \tag{7.1}$$

where C_B^l is the weighted betweenness of the link *l* and w_l is the weight of the link *l* as shown in figure 7.1.

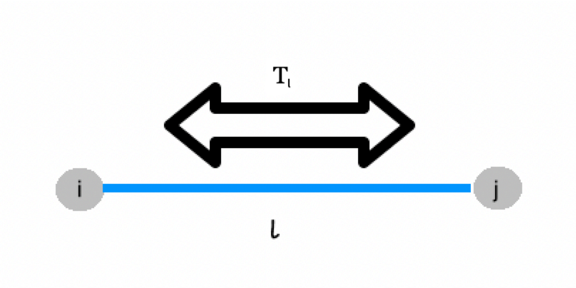


Figure 7.1: Link tension for link *l*

To comprehend link tension, let us consider an electrical circuit. An electrical circuit is a network with current flowing in a closed path in wires, where the wires and components in a circuit offer some hindrance to the current flow, this hindrance is known as the resistance. From ohm's law, voltage can be defined as $V = I * R$ where *V* is the voltage, *I* is the current and *R* is the resistance. Now, let us consider link betweenness to be "link current" and link weight to be "link resistance". The product of these two measures yields "link tension" (or "link voltage"). In this manner, we can calculate link tension for all the links in the graph. Link tension shows the capability of a link to carry information from one node to another; the greater the link tension of a link, the greater is the capability of the link to allow flow of data packets through it. Measuring link tension can help us to understand and recognize the weak links and strong links in a network during an attack on a network since it can help us identify the link with low or high weights and their ability to transmit information. Let us consider the weighted undirected graph in figure 7.2.

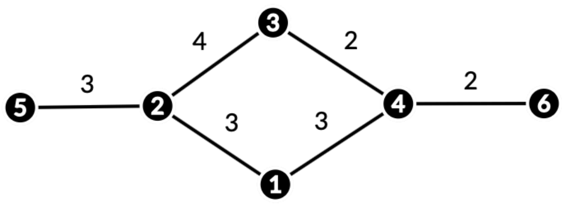


Figure 7.2: Example of a weighted graph

In figure 7.2, the information from node 2 to node 4 can take the route through $(2 \sim 3) + (3 \sim 4)$ or $(2 \sim 1) + (1 \sim 4)$. To know which path has better capability to transmit information, we sum the link tension of the links involved in the route. When we calculate link tension values for both the paths, we obtain

- Path 1: = $T_{2\sim3}$ + $T_{3\sim4}$ = 1.752
- Path 2: = $T_{2\sim1}$ + $T_{1\sim4}$ = 1.8

Though both the paths have the same shortest path value 6, the link tension values are different. This tells us that the links in the latter path, $(2 \sim 1) + (1 \sim 4)$ have greater link importance and could have better capability to handle transmission of data from node 2 to node 4. Figure 7.3 shows the values of link tension for all the links in the weighted graph displayed in figure 7.2.

Link	Link Betweenness	Link Weight	Link Tension
(1,2)	0.267	3	0.801
(1,4)	0.333	3	0.999
(2,3)	0.267	4	1.068
(2,5)	0.333	3	0.999
(3,4)	0.333	2	0.666
(4,6)	0.333	2	0.666

Table 7.1: Link betweenness, link weight and link tension values for figure 7.2

Table 7.1 lists the weighted link betweenness, link weight and links tension for all the links in figure 7.2.

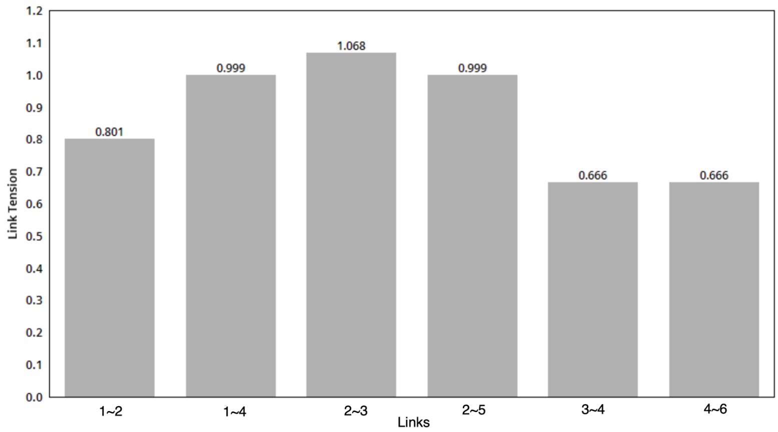


Figure 7.3: Link tension for all links in the graph displayed in figure 7.2

Let us now consider the weighted graph in figure 7.4 where few links have same weight but different link betweenness and link tension.

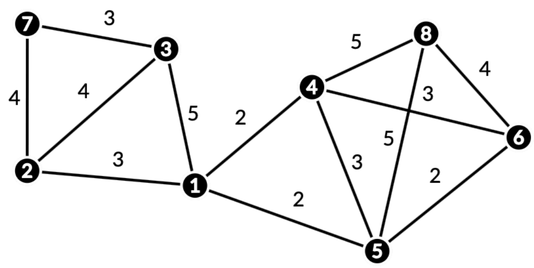


Figure 7.4: Example of a weighted graph

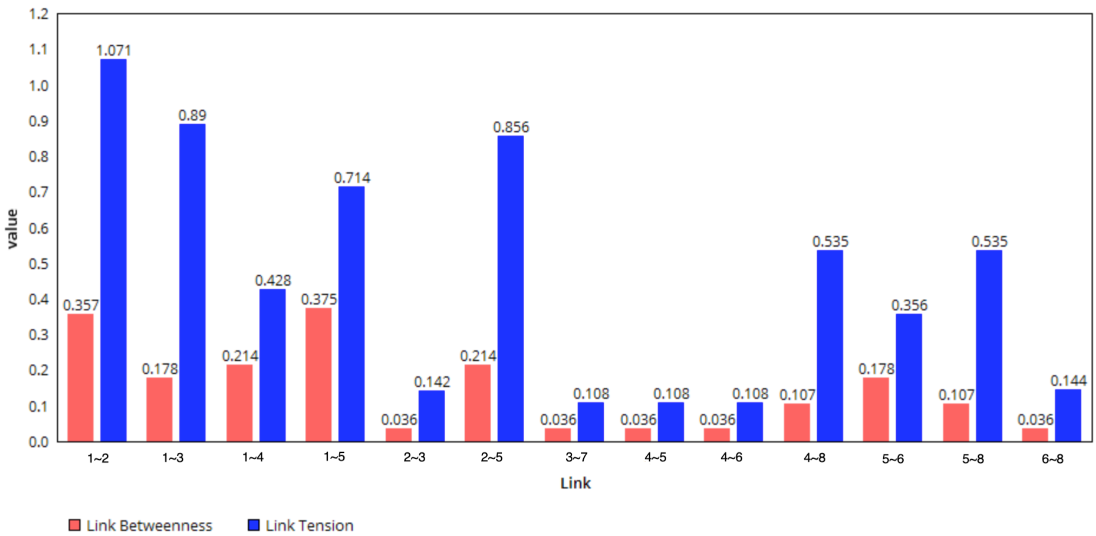


Figure 7.5: Illustration of link tension and link betweenness for all the links in figure 7.4

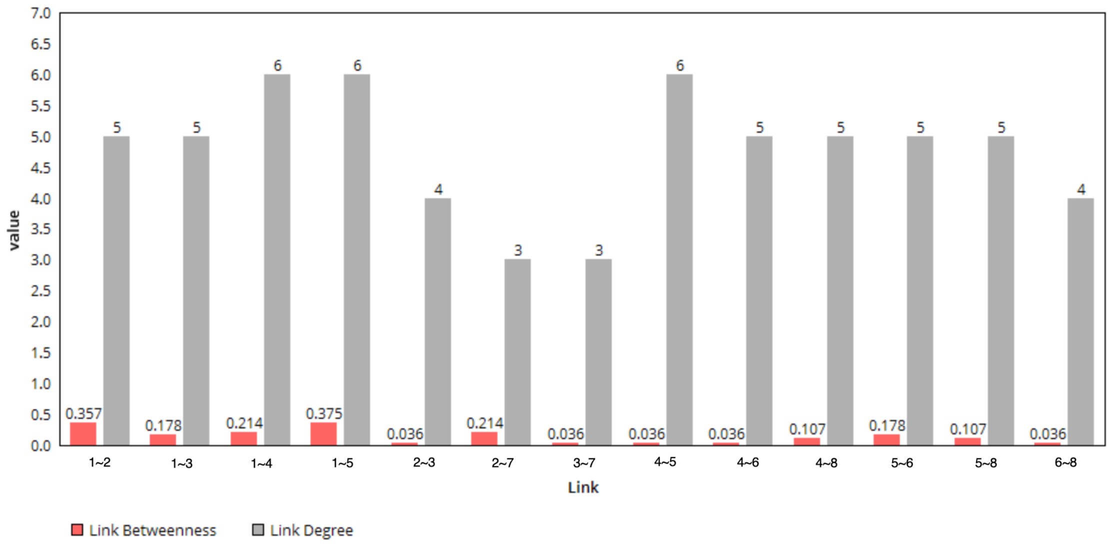


Figure 7.6: Illustration of link tension and link degree for all the links in figure 7.4

Link	Link Betweenness	Link Weight	Link Tension
(1,2)	0.357	3	1.071
(1,3)	0.178	5	0.89
(1,4)	0.214	2	0.428
(1,5)	0.357	2	0.714
(2,3)	0.035	4	0.14
(2,7)	0.214	4	0.856
(3,7)	0.036	3	0.108
(4,5)	0.036	3	0.108
(4,6)	0.036	3	0.108
(4,8)	0.107	5	0.535
(5,6)	0.178	2	0.356
(5,8)	0.107	5	0.535
(6,8)	0.036	4	0.144

Table 7.2: Weighted Link betweenness, link weight and link tension values for figure 7.4

Table 7.2 lists the weighted link betweenness, link weight and links tension for all the links in figure 7.4. Links with same betweenness can have different link tension, as shown in table 7.4. For the weighted graph in figure 7.4, figure 7.5 shows link tension and link betweenness comparison and figure 7.6 shows a comparison between link tension and link degree. From figure 7.5 and 7.6, we can observe that the links with high link betweenness have high link tension but not necessarily a high link degree.

Two links can have the same link betweenness, like links $(1 \sim 3)$ and $(5 \sim 6)$ in figure 7.4, with weight of link $(1 \sim 3)$ greater than the weight of link $(5 \sim 6)$. Link $(1 \sim 3)$ is therefore more important than link $(5 \sim 6)$ since despite the higher weight of the link $(1 \sim 3)$, it has the same number of shortest paths running through it as link $(5 \sim 6)$. Link tension aids in ranking the importance of the link since the link tension values can be different for links with same betweenness, indicating that the link with higher link tension is crucial and is better capable of handling flow of information through it.

7.2. Link tension distribution for different classes of graph

Figure 7.7 shows the link tension distribution for an Erdos Renyi weighted random graph instance with n=1000 and p=0.5. The link tension distribution as observed in figure 7.7 is a binomial distribution evolving into a Poisson distribution.

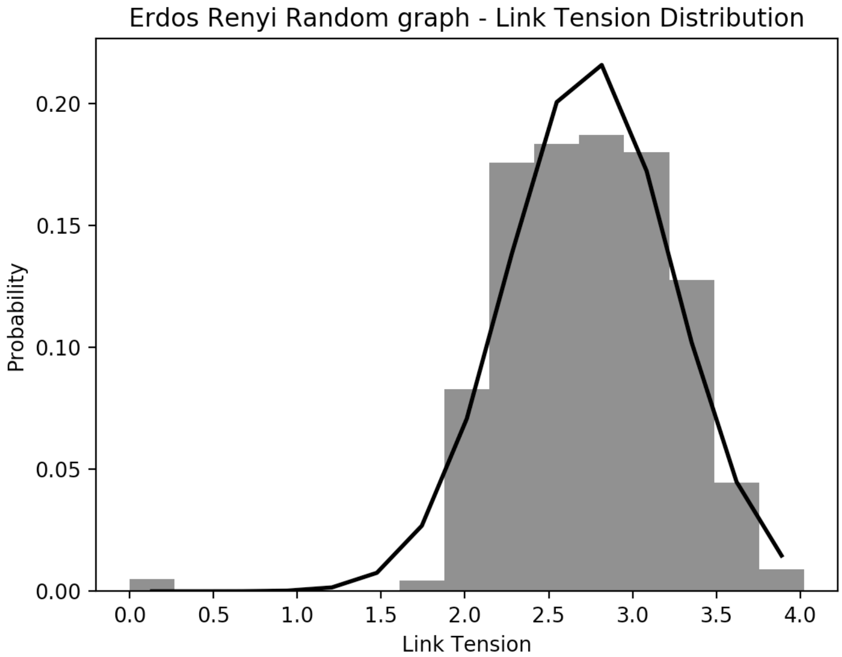


Figure 7.7: Link tension distribution for weighted ER graph

Figure 7.8 shows the link tension distribution for a Watts Strogatz weighted small world graph instance with *n*=1000 and *k*=4. The link tension distribution obtained for weighted Watts Strogatz weighted small world model is a poisson distribution. This tells us that most of the links in the graph have the same value of link tension.

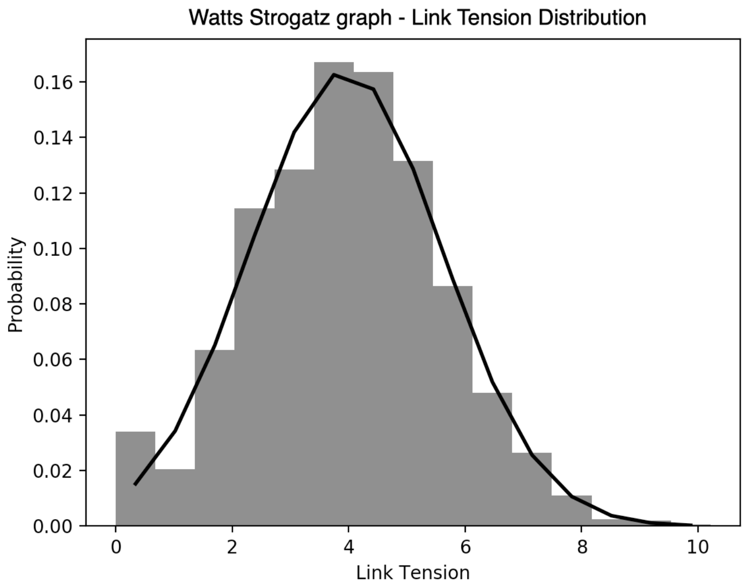


Figure 7.8: Link tension distribution for weighted Watts Strogatz graph

Figure 7.9 shows the link tension distribution for a Barabasi Albert weighted scale free graph instance with 1000 final nodes and initial nodes=10. The link tension distribution as observed in figure 7.9 is a power law distribution. This distribution tells us that many links have a low value of link tension and few links have higher value of link tension.

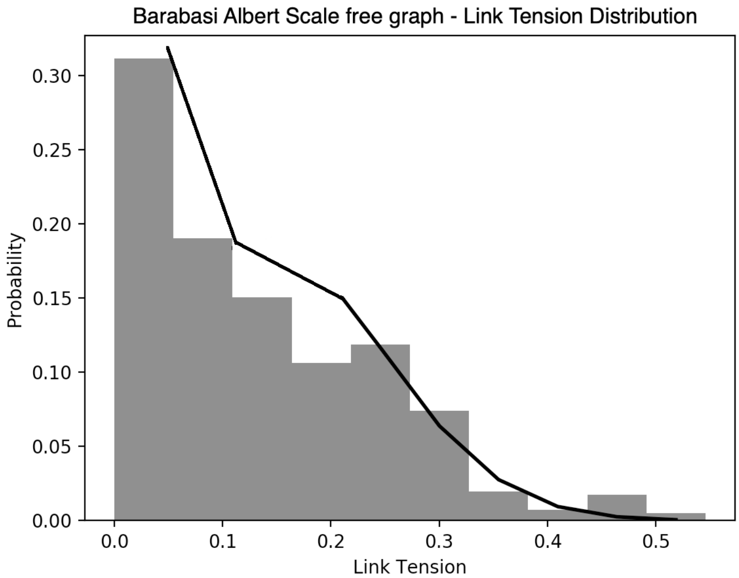


Figure 7.9: Link tension distribution for weighted Barabasi Albert graph

7.3. Observations

Link tension as a metric, is an extension from link betweenness to rank the significance of a link in a network. Through link tension, we are provided further information about the path taken between a node pair and the capability of the path to handle transmission of data.

In the case of multiple shortest path between two node pairs, link tension can help us decide the potential path which possesses the better capability to transfer information. Since this metric considers weight of a link along with link betweenness, it is helpful in cases when two links have the same betweenness and suggests that a link with higher weight has more importance.

8
Conclusion

This chapter provides conclusions for this thesis. Section 8.1 describes the contributions made through this thesis. Section 8.2 briefly discusses the future work that can be done related to this thesis.

8.1. Contributions of this thesis

Analyzing the links in weighted networks is a significant step towards understanding their dynamic behaviour. The goal of our research was to develop a method to obtain weight tolerance of a link in a network. The primary research question of the thesis was to propose an algorithm to calculate positive and negative weight tolerance of a link. The secondary research question was to understand the ability of the link to handle transmission of data.

We started with reviewing few important graph measures of complex network theory: clustering coefficient, assortativity, modularity, betweenness centrality for both unweighted and weighted graphs. We continued our research by studying the basic concepts of graph theory which includes types of graphs, regular graphs, directed and undirected graphs, weighted and unweighted graphs, degree of a node, random graph models and graph metrics. We then studied extensively about betweenness centrality for both nodes and links. Then, we generate unweighted and weighted random graph models (Erdos Renyi graphs, Small world graphs, Scale free graphs) and obtain few important distributions for the generated graphs. Distribution of link strength which was obtained for three classes of weighted random graph models, is a novel addition to chapter 5.

In the final parts of the thesis, we have proposed an algorithm to obtain the weight tolerance of a link, both positive and negative tolerance. The positive tolerance obtained for a single link tells us the extent to which the weight of the link can be increased, so as to maintain the same shortest path between all node pairs. Similarly, the negative tolerance tells us the extent up to which weight of the link has to be decreased, so that the shortest path between a node pair which did not traverse the link previously, begins to traverse the link. We also discuss how the tolerance is for links with high and low link betweenness.

In addition, the other concept introduced for the weighted networks is link tension. This measure helps us to understand the capability of a link to handle data transmission. We also discuss how link tension can be used to rank the importance of a link when two links have the same link betweenness. Both the concepts that have been introduced gives us a new type of information related to the links in a weighted network which can be further explored.

8.2. Future research

The concept of link weight tolerance and link tension which has been introduced in this thesis is for undirected weighted graphs. The logical next step would be to extend both these concepts for directed weighted graphs. This would include research on shortest paths in directed weighted graphs, link betweenness centrality in directed weighted graphs. Tolerance of a link in weighted directed graphs can help us understand the nature of links in directed graphs. Similarly, link tension in directed graphs can help us determine the ability of a link to transmit information in a directed network. Additionally, the weighted network in this thesis includes only positive link weights. Link weight tolerance can be developed for weighted networks with negative link weights. Besides this, research on more large-scale networks can also be a part of future work.

Bibliography

[1] Erdös, P. Rényi, A. (1959). On Random Graphs I. Publicationes Mathematicae Debrecen, 6, 290.

[2] Gilbert, E. N. Random Graphs. Ann. Math. Statist. 30 (1959), no. 4

[3] Newman, Mark. E. J.; Strogatz, S. H.; Watts, D. J. (2001). Random graphs with arbitrary degree distributions and their applications

[4] Mark Newman. 2010. Networks: An Introduction. Oxford University Press, Inc., New York, NY, USA

[5] Noga Alon , Joel H. Spencer The Probabilistic methods (2000)

[6] D. J. Watts and S. H. Strogatz. (1998). Collective dynamics of 'small-world' networks. Nature, 393:440–442.

[7] Telesford, Qawi Joyce, Karen Hayasaka, Satoru Burdette, Jonathan Laurienti, Paul. (2011). The Ubiquity of Small-World Networks. Brain connectivity. 1. 367-75. 10.1089/brain.2011.0038.

[8] D. Watts. Networks, dynamics and the small-world phenomenon. American Journal of Sociology, 105 (2):493–527, 1999.

[9] M. E. J. Newman and D. J. Watts. Renormalization group analysis of the small-world network model. Physics Letters A, 263:341–346.

[10] R. K. Department and R. Kasturirangan. (1999). Multiple scales in small-world networks. In Brain and Cognitive Science Department, MIT.

[11] J. M. Kleinberg. (2000). Navigation in a small world. Nature, 406(6798):845.

[12] A. L. Barabási and R. Albert. 1999) Emergence of scaling in random networks. Science, 286(5439):509–5129.

[13] D. J. Watts and S. H. Strogatz.(1998) Collective dynamics of "small-world" networks. Nature, 393:440-442.

[14] Barrat, Alain, and Martin Weigt. (2000). On the properties of small-world network models. The European Physical Journal B-Condensed Matter and Complex Systems 13.3: 547-560.

[15] Strogatz, S. H., 2001. Exploring complex networks. Nature 410, 268–276.

[16] P. L. Krapivsky, S. Redner, and F. Leyvraz. (2000). Connectivity of growing random networks, Physical Review Letters, vol. 85, no.21, pp. 4629–4632.

[17] S. N. Dorogovtsev, J. F. F. Mendes, and A. N. Samukhin. (2000). Structure of growing networks with preferential linking, Physical Review Letters, vol. 85, no. 21, pp. 4633–4636.

[18] S. N. Dorogovtsev, J. F. F. Mendes, and A. N. Samukhin. (2001). Size-dependent degree distribution of a scale-free growing network, Physical Review E, vol. 63, no. 6 I, Article ID 062101, 4 pages.

[19] S. H. Yook, H. Jeong, A. L. Barabási, and Y. Tu. Weighted evolving networks," Physical Review Letters, vol. 86, pp. 5835–5838, 2001

[20] A. Barrat, M. Barthélemy, and A. Vespignani, "Weighted evolving networks: coupling topology and weight dynamics," Physical Review Letters, vol. 92, no. 22, Article ID 228701, 4 pages, 2004.

[21] T. Tanaka and T. Aoyagi. (2008). Weighted scale-free networks with variable power-law exponents, Physica D, vol. 237, no. 7, pp.898–907,

[22] A. L. Barabási, R. Albert, and H. Jeong.(1999). Mean-field theory for scale-free random networks, Physica A, vol. 272, pp. 173–187,

[23] Soffer, Sara Nadiv, and Alexei Vazquez. Network clustering coefficient without degree-correlation biases. Physical Review E 71.5 (2005): 057101.

[24] B. Bollobás, O. Riordan, J. Spencer, and G. Tusnády, "The degree sequence of a scale-free random graph process," Random Structures and Algorithms, vol. 18, no. 3, pp. 279–290, 2001

[25] Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D. U. (2006). Complex networks: Structure and dynamics. Physics reports, 424(4-5), 175-308.

[26] Newman, Mark EJ, and Michelle Girvan. (2004). Finding and evaluating community structure in networks. Physical review E 69.2 : 026113.

[27] P. X.-K. Song.(2007). Correlated Data Analysis: Modeling, Analytics, and Applications, Springer Series in Statistics (Springer, New York,).

[28] R. Guimerà and L. A. N. Amaral, Nature London 433, 895, 2005.

[29] A. Arenas, A. Fernández, and S. Gómez, New J. Phys. 10, 053039, 2008.

[30] E. A. Leicht and M. E. J. Newman, Phys. Rev. Lett. 100, 118703 ,2008.

[31] S. Fortunato and M. Barthelemy, Proc. Natl. Acad. Sci. U.S.A. 104, 36 ,2007.

[32] J. Kumpula, J. Saramäki, K. Kaski, and J. Kertész, Eur. Phys. J. B 56, 41 2007.

[33] Freeman, L.: A set of measures of centrality based on betweenness. Sociometry 40(1), 35–41 (1977)

[34] Brandes, U.: A faster algorithm for betweenness centrality. J. Mathematical Sociology 25(2), 163–177 (2001)

[35] Freeman, Linton C., Stephen P. Borgatti, and Douglas R. White. Centrality in valued graphs: A measure of betweenness based on network flow. Social networks 13.2 (1991): 141-154.

[36] Freeman, Linton C. Centrality in social networks conceptual clarification. Social networks 1.3 (1978): 215-239.

[37] A. Bavelas, A mathematical model for group structure. Human organization," Applied Anthropology, vol. 7, no. 3, pp. 16–30, 1948.

[38] Travers, Jeffrey, and Stanley Milgram. An experimental study of the small world problem. Social Networks. Academic Press, 1977. 179-197.

[39] Dodds, Peter Sheridan, Roby Muhamad, and Duncan J. Watts. An experimental study of search in global social networks.science 301.5634 (2003): 827-829.

[40] Wang, Huijuan, Javier Martin Hernandez, and Piet Van Mieghem. Betweenness centrality in a weighted network. Physical Review E 77.4 (2008): 046105.

[41] Park, Kwangho, Ying-Cheng Lai, and Nong Ye. Characterization of weighted complex networks. Physical Review E 70.2 (2004): 026109.

[42] M. E. J. Newman, D. J. Watts, and S. H. Strogatz. Random graph models of social networks. Proc. Natl. Acad. Sci. U.S.A., 99(Suppl 1):25662572, 2002.

[43] Newman, Mark EJ. Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. Physical review E 64.1 (2001): 016132.

[44] R. D. Luce and A. D. Perry (1949). A method of matrix analysis of group structure. Psychometrika. 14 (1): 95–116. doi:10.1007/BF02289146. PMID 18152948

[45] Scott, J., 2000. Social Network Analysis: A Handbook Sage Publications, London, UK.

[46] Stanley Wasserman, Katherine Faust, 1994. Social Network Analysis: Methods and Applications Cambridge: Cambridge University Press.

[47] R. Albert and A.-L. Baraba´si: Statistical mechanics of complex networks(2002)

[48] Huberman, B. A., Adamic, L. A. (1999). Growth dynamics of the world-wide web. Nature, 401(6749), 131-131.

[49] Goh, K. I., Kahng, B., Kim, D. (2001). Universal behavior of load distribution in scale-free networks. Physical review letters, 87(27), 278701.

[50] Bianconi, G., Barabási, A. L. (2001). Competition and multiscaling in evolving networks. EPL (Europhysics Letters), 54(4), 436.

[51] Solé, R. V., Pastor-Satorras, R., Smith, E., Kepler, T. B. (2002). A model of large-scale proteome evolution. Advances in Complex Systems, 5(01), 43-54. Chicago

[52] M. Newman. Assortative mixing in networks, Phys. Rev. Lett., Vol. 89(20), 208701 (2002).

[53] U. Brandes and C. Pich. Centrality estimation in large networks. International Journal Of Bifurcation And Chaos, 17(7):2303, 2007

[54] R. Geisberger, P. Sanders, and D. Schultes. Better approximation of betweenness centrality. In J. I. Munro and D. Wagner, editors, ALENEX, pages 90–100. SIAM, 2008

[55] Y. Makarychev. Simple linear time approximation algorithm for betweenness. Technical report, 2009

[56] D. A. Bader and K. Madduri. Parallel algorithms for evaluating centrality indices in real-world networks. In Proceedings of the 2006 International Conference on Parallel Processing, ICPP ’06, pages 539–550, Washington, DC, USA, 2006. IEEE Computer Society

[57] P. Holme. Congestion and centrality in traffic flow on complex networks. Advances in Complex Systems, 6(2):163–176, Jan. 2003

[58] S. Jin, Z. Huang, Y. Chen, D. G. Chavarria-Miranda, J. Feo, and P. C. Wong. A novel application of parallel betweenness centrality to power grid contingency analysis. In IPDPS, pages 1–7. IEEE, 2010.

[59] L. Leydesdorff. Betweenness centrality as an indicator of the interdisciplinarity of scientific journals. Journal of the American Society for Information Science and Technology, 58(9):1303–1309, 2009.

[60] J. W. Pinney and D. R. Westhead. Betweenness-based decomposition methods for social and biological networks. In Interdisciplinary Statistics and Bioinformatics, pages 87–90. Leeds University Press, 2006.

[61] Euler, Leonhard (1736).Solutio problematis ad geometriam situs pertinentis. Comment. Acad. Sci. U. Petrop 8, 128–40.

[62] Borgatti, S. P., Everett, M. G., 2006. A graph-theoretic perspective on centrality. Social Networks 28, 466–484.

[63] Everett, M. G., Borgatti, S. P., 1999. The centrality of groups and classes. Journal of Mathematical Sociology 23 (3), 181–201.

[64] Anthonisse, J. M. (1971). The rush in a directed graph. Stichting Mathematisch Centrum. Mathematische Besliskunde, (BN 9/71).

[65] S.H. Yook, H. Jeong, and A.-L. Baraba´si, and Y. Tu, Phys. Rev. Lett. 86, 5835, (2001)

[66] D. Zhang, S. Trimper, B. Zheng, and P.M. Hui, Phys. Rev. E 67, 040102, (2003)

[67] Zhang, B., Horvath, S. (2005). A general framework for weighted gene co-expression network analysis, Statistical Applications in Genetics and Molecular Biology,4

[68] Kalna, G., Higham, D.J. (2006). Clustering coefficients for weighted networks, University of Strathclyde Mathematics Research report 3

[69] Lopez-Fernandez, L., Robles, G., Gonzalez-Barahona, J.M. (2004). Applying social network analysis to the information in CVS repositories.

[70] Onnela, J.-P ., Saramäki, J., Kertész, J., Kaski, K. (2005). Intensity and coherence of motifs in weighted complex networks, Physical Review E, 71 (6), 065103.

[71] Serrano M. A., Boguñá M., Pastor-Satorras R. (2006). Correlations in weighted networks, Physical Review E 74, 055101 (R)

[72] Holme P., Park S.M., Kim B.J., Edling C.R. (2004). Korean university life in a network perspective: Dynamics of a large affiliation network, cond- mat/0411634v1.

[73] Li, M., Fan, Y., Chen, J., Gao, L., Di, Z., Wu J. (2005). Weighted networks of scientific communication: the measurement and topological role of weight, Physica A 350, 643–656.

[74] Barrat, A., Barthelemy, M., Pastor-Satorras, R., Vespignani, A. (2004). The architecture of complex weighted networks, Proceedings of the National Academy of Sciences (USA) 101, 3747-3752

[75] Ravasz, E. B.; Barabási, A. L. S. (2003). Hierarchical organization in complex networks. Physical Review E. 67 (2).

[76] Caldarelli, G.; et al. (2002). Scale-free networks from varying vertex intrinsic fitness

[77] Zaidi, Faraz. (2012). Small world networks and clustered small world networks with random connectivity. Social Network Analysis and Mining. 3. 10.1007/s13278-012-0052-1.

[78] H. Chang, B. Su, Y. Zhou, D. He. Assortativity and act degree distribution of some collaboration networks, Physica A: Statistical Mechanics and its Applications, Vol. 383(2), pp. 687–702

[79] C. Leung, H. Chau. (2007). Weighted Assortative and Disassortative Networks Model, Physica A: Statistical Mechanics and its Applications, Vol. 378, pp. 591-602

[80] H. Wang, L. Douw, J. Hernández, J. Reijneveld, C. Stam, P. Van Mieghem. (2010). Effect of tumor resection on the characteristics of functional brain networks, Physical Review E, Vol. 82, 021924

[81] Leung, C. C., and H. F. Chau. Weighted assortative and disassortative networks model. Physica A: Statistical Mechanics and its Applications 378, no. 2 (2007)

[82] Newman, Mark EJ. Analysis of weighted networks. Physical review E 70.5 (2004): 056131.

[83] Girvan, Michelle, and Mark EJ Newman. Community structure in social and biological networks. Proceedings of the national academy of sciences 99.12 (2002): 7821-7826.

[84] Girvan M. and Newman M. E. J., Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA 99, 7821–7826 (2002)

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

[86] U. Brandes. On variants of shortest-path betweenness centrality and their generic computation. Social Networks, 30(2):136–145, 2008.

[87] Lee, Min-Joong Lee, Jungmin Park, Jaimie Choi, Ryan Chung, Chin-Wan. (2012). QUBE: A quick algorithm for updating betweenness centrality. WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web

[88] Chen, Guanrong, Xiaofan Wang, and Xiang Li. Fundamentals of complex networks: models, structures and dynamics. John Wiley Sons, 2014.

[89] Kleinberg, Jon.(2000). The small-world phenomenon: An algorithmic perspective. Proceedings of the thirty-second annual ACM symposium on Theory of computing.

[90] Bollobás, Béla, et al. Directed scale-free graphs. Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2003.

[91] Dijkstra, E.W. A note on two problems in connexion with graphs. Numer. Math. 1, 269–271 (1959)

[92] E. D. Kolaczyk, D. B. Chua, and M. Barthélemy. (2009).Group betweenness and co-betweenness: inter-related notions of coalition centrality, Social Networks, vol. 31, no. 3, pp. 190–203,

A

Appendix

This appendix contains the Python code used in the thesis for the calculation of link weight tolerance and link tension.

```
import networkx as nx
import numpy as np
from itertools import islice
from itertools import count
import itertools


#creates a graph object
G= nx.Graph()
# creating a sample graph by adding nodes and edges
G.add_edge(1,0,weight=1)
G.add_edge(1,4,weight=2)
G.add_edge(5,2,weight=1)
G.add_edge(3,0,weight=7)
G.add_edge(3,4,weight=1)
G.add_edge(5,4,weight=7)
G.add_edge(3,2,weight=4)

pairs= list ( itertools .combinations(range(6), 2))
print pairs
edgeweight= list(G.edges.data('weight'))
maxw= max(edgeweight)
# list of X_p
for (u,v) in pairs:
    path=nx.dijkstra_path(G,u,v, weight='weight')
    xp=nx.dijkstra_path_length(G,u,v, weight='weight')
    print path
    print u,v,xp
lb=nx.edge_betweenness_centrality(G, normalized=True, weight='weight')

#link tolerance for 3~4
i=3
j=4

#Code to generate positive link weight tolerance calculation

result_x = np.array ([])
for (u,v) in pairs:
    if (nx.has_path(G,u,v)==True):
        sp_count=[p for p in nx.all_shortest_paths(G,source=u,target=v, weight='weight')]
        count=len(sp_count)
        if count>1:
            x1=maxw
        else:
            x1=nx.dijkstra_path_length(G,u,v, weight='weight')
        result_x = np.append(result_x, x1)
G.remove_edge(i,j)
result_y = np.array ([])
```

```
for (u,v) in pairs:
    y1=nx.dijkstra_path_length(G,u,v, weight='weight')
    result_y = np.append(result_y, y1)
result_z=np.array ([])
result_z=np.subtract(result_y,result_x)
plt= result_z. tolist ()
plt_list =[int(i) for i in plt]
temp_list= zip(pairs, plt_list )

pt_list =[(u,v,i) for ((u,v),i) in temp_list]
p_list =[(u,v,np.nan) if i==0 else (u,v,i) for (u,v,i) in pt_list ]
plt_flist = p_list =[(u,v,0) if i < 0 else (u,v,i) for (u,v,i) in p_list ]


#Matrix representation for Zp

matrix=[[0 for x in range(n)] for y in range(m)]
matrix={}
for (u,v) in pairs:
    matrix[u,v]= plt_flist [j][2]
    matrix[v,u]= plt_flist [j][2]
    j+=1
idx = np.array(matrix.keys())
vals = np.array(matrix.values())
# Get dimensions of output array based on max extents of indices
dims = idx.max(0)+2
# Setup output array and assign values into it indexed by those indices
out = np.zeros(dims,dtype=vals.dtype)
out[idx [:,0], idx [:,1]] = vals
print out[:-1,:-1]

# Code to generate negative link weight tolerance
l=(i,j)
ij_w= H.get_edge_data(*l)
ij_weight=ij_w.get('weight')
result_x2 = np.array ([])

for (u,v) in pairs:
    x2_1=nx.dijkstra_path_length(H,u,i)+ ij_weight + nx.dijkstra_path_length(H,j,v, weight='weight')
    x2_2=nx.dijkstra_path_length(H,u,j)+ ij_weight + nx.dijkstra_path_length(H,i,v, weight='weight')
    x2= min(x2_1, x2_2)
    result_x2 = np.append(result_x2, x2)

result_y2 = np.array ([])
for (u,v) in pairs:
    y2=nx.dijkstra_path_length(H,u,v, weight='weight')
    result_y2 = np.append(result_y2, y2)
result_z2 = np.array ([])
result_z2=np.subtract(result_x2,result_y2)
pltnt= result_z2. tolist ()
pltnt_list =[int(i) for i in pltnt]
tempnt_list= zip(pairs, pltnt_list )

ptnt_list =[(u,v,i) for ((u,v),i) in tempnt_list]
ptnt_flist =[(u,v,ij_weight) if i > ij_weight else (u,v,i) for (u,v,i) in ptnt_list ]

# Matrix representation for Zn
matrixnt=[[0 for x in range(n)] for y in range(m)]
matrixnt={}
for (u,v) in pairs:
    matrixnt[u,v]= ptnt_flist [j][2]
    matrixnt[v,u]= ptnt_flist [j][2]
    j+=1
idx2 = np.array(matrixnt.keys())
vals2 = np.array(matrixnt.values())
# Get dimensions of output array based on max extents of indices
dims2 = idx2.max(0)+2
# Setup output array and assign values into it indexed by those indices
outnt = np.zeros(dims2,dtype=vals2.dtype)
outnt[idx2 [:,0], idx2 [:,1]] = vals2
print outnt[:-1,:-1]

# Link tension calculation
```

```
# Weighted link betweenness for all the links in the graph
gblc=nx.edge_betweenness_centrality(g,normalized=True, weight='weight')
print (gblc)
l_b = gblc.values()
l_b1 = [round(num, 3) for num in l_b]
print (l_b1)
edgewidth = [ d['weight'] for (u,v,d) in g.edges(data=True)]
edgeweight = [round(num) for num in edgewidth]
result_lt = np.array ([])

for (u,v) in g.edges:
        l=(u,v)
        uv_w= g.get_edge_data(*l)
        uv_weight=uv_w.get('weight')
        lb= gblc[u,v]
        lt = uv_weight* lb
        print (u,v, lt )
        result_lt =np.append(result_lt, lt )
lt_list = result_lt . tolist ()
```