Bournemouth University

Prediction and Modelling of Complex Social Networks and Their Evolution

by

Akanda Wahid -Ul- Ashraf

A thesis submitted in partial fulfilment for the degree of
Doctor of Philosophy

in the
Department of Computing and Informatics

June 2020

Declaration of Authorship

I, Akanda Wahid -Ul- Ashraf, declare that this thesis titled, 'Adaptive and robust approach for predictive modelling of dynamics and evolution of Complex Social Networks' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"In my scientific work, I have hunches. I can't explain always why I think a certain path is the right way, but I need to trust it and go ahead. I also have the ability to check these hunches and find out what they are about. That's the science part."

Gerd Gigerenzer

Bournemouth University

Abstract

Department of Computing and Informatics

Doctor of Philosophy

by Akanda Wahid -Ul- Ashraf

This thesis focuses on complex social networks in the context of computational approaches for their prediction and modelling. The increasing popularity and advancement of social networks paired with the availability of social network data enable empirical analysis, inference, prediction and modelling of social patterns. This data-driven approach towards social science is continuously evolving and is crucial for modelling and understanding of human social behaviour including predicting future social interactions for a wide range of applications. The main difference between traditional datasets and network datasets is the presence of the relational components (links) between instances (nodes) of the network. These links and nodes induce intricate local and global patterns, defining the topology of a network. The topology is ever evolving, determining the dynamics of such a networked system. The work presented in this thesis starts with an extensive analysis of three standard network models, in terms of their properties and self-interactions as well as the size and density of the resultant graphs. These crucial analysis and understanding of the main network models are utilised to later develop a comprehensive network simulation framework. A set of novel nature-inspired link prediction approaches are then developed to predict the evolution of networks, based solely on their topologies. Building on top of these approaches, enhanced topological representations of networks are subsequently combined with node characteristics for the purpose of node classification. Finally, the proposed classification methods are extensively evaluated using simulated networks from our network simulation framework as well as two real-world citation networks. The link prediction approaches proposed in this research show that the topology of the network can be further exploited to improve the prediction of future relationships. Moreover, this research demonstrates the potential of blending state-of-the-art Machine Learning techniques with graph theory. To accelerate such advancements in the field of network science, this research also offers an open-source software to provide high-quality synthetic datasets.

Acknowledgements

I would like to give thanks to my parents and my younger brother, Talha, for their help, inspiration, and support for all the achievements in my life.

I would like to express my gratitude towards both of my supervisors, Prof Marcin Budka and Prof Katarzyna Musial for supporting and guiding me throughout my research. It has been an amazing experience to be able to work under their supervisions, which has made me more organised and increased my research skills.

I also would like to thank Bournemouth University (BU) for providing me with a PhD studentship, which has made this work possible.

I would like to thank all the staffs at Bournemouth University for creating such a friendly and excellent research environment. Especially I would like to thank all the Research Administrators and the Doctoral College for their immense help and support towards the graduate students.

I also would like to thank all my colleagues at Bournemouth University, especially in my lab for creating such a friendly environment, which has facilitated an outstanding work environment. I would like to thank my friend and colleague Dr Rashid Bakirov with whom I have played a lot of table-football, which has been a good source of fun in our lab. I would also like to thank Dr Arif Reza Anwary who has been an inspiration for me.

A special thanks towards my friend Gareth Leaney for helping me with academic writing at the beginning of my journey towards this PhD.

A very special thanks to my very dear friend Dr Saad Mohamad, who deserves a separate paragraph, for giving me company during most of my time at BU.

I would like to thank my Grandmother, Fatema Kabir for her tremendous support towards my academic journey. I would also like to thank my cousins, Mehnaz Hoque and Mohammad Moshiul Hoque Riad for their never-ending support and friendship at any given point in my life.

I would like to show my gratitude towards two of my best friends, Aubdullah Munim and Sabbir Ahamed for their friendship and support which has encouraged me towards some of the biggest achievements in my life.

Finally, a special thanks to Sue Burt, Sharon Hartwell, and Tim Peters from the BU chaplaincy team for their immense help and support during my entire journey of this thesis at Bournemouth University.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Networks are everywhere", among all the different types of networks (e.g. social, biological, internet, road networks, etc.) interest in social networks has experienced particularly rapid growth, mainly due to the availability of large real-world network datasets (Liben-Nowell and Kleinberg 2007; Brandes et al. 2013; Barabási and Bonabeau 2003). Research in social networks, leveraging these large-scale data, is mainly focused on patterns and evolution of the social structure. According to Freeman et al. (1987), "social structure refers to a relatively prolonged and stable pattern of interpersonal relations". Although large-scale social networks analysis is ubiquitous nowadays, the groundwork in the shape of small-scale social network analysis began long before the era of MySpace or Facebook. The origin of network science can be traced back to the 18th-century scholar Euler, who introduced graph theory (Euler 1999). However, for social networks, the field which had been previously known as *sociometry* is now transformed into social network analysis and has become a part of much broader research area called network science. The study of social network dynamics could be found as early as 1934 when Moreno (1934) designed a hand-drawn friendship diagram. However, this social network is minimal when compared with today's hundreds of millions or even billions of nodes in online social networks like Facebook and Twitter. From that friendship diagram, Moreno (1934) inferred simple conclusions that there were more friendships or connections between the same gender than opposite genders. Although these findings are relatively straightforward and intuitive, the study had pioneered today's social network analysis. The friendship diagram analysis by Moreno (1934) demonstrated that social behaviour can be understood more easily when the interactions are represented in the form of a diagram, i.e. a network.

A network is typically represented as a graph and consists of a set of connected entities. In the field of network science, following from the graph theory, these entities are referred to as nodes or vertices while the connections between them are known as edges or links (Newman 2010*a*). A network could be defined by its E edges and V vertices. The evolution of societies is

captured, mainly, through the emergence and disappearance of nodes and edges in the network that is a representation of complex system – in here a society. However, changes in the node and edge attributes also reflect the evolution of societies. Thus, analysis of such network could give a broader understanding of the evolution of societies. The field of network science could also be thought of as the study of the collection, management, analysis, interpretation, and presentation of relational data (Brandes et al. 2013). Another large quantity of network science work is also dedicated towards mathematical modelling of the networks (Barabási and Bonabeau 2003; Erdös and Rényi 1959; Watts and Strogatz 1998). Network science is still an emerging field as more and more complex and large-scale network data becomes available. Network science does not belong to a single discipline but is a combination of multiple disciplines, including graph theory, physics (statistical mechanics), data mining and sociology (Brandes et al. 2013; Tiropanis et al. 2015).

Due to the availability of large-scale network data, for the first time in history, we have the possibility to process 'big data' (gathered by computer systems) about the interactions and activities of millions of individuals. It represents an increasingly essential yet under-utilised resource because due to the scale, complexity and dynamics, Complex Social Networks (CSN) extracted from this data are extremely difficult to analyse. A coherent and comprehensive approach to analyse such networks and their dynamics is crucial to advance our understanding of people's continuously changing behaviour. Networked systems and their evolution are usually analysed by building models of interactions using the classic random, scale-free, and small-world models. However, these do not precisely reflect the complex nature of real-world networked systems and their dynamics.

To develop predictive models for a social network, it is essential to recognise its complex topological patterns and changes in those patterns with respect to time. In the majority of complex networks, three aspects affecting the whole network, can typically change. These are (1) changes in links (e.g. emergence and disappearance), (2) changes in nodes (e.g. emergence and disappearance of nodes), and (3) changes of properties/features (of both nodes and links). This project, firstly, focuses on the first type of changes listed above, i.e. changing the number of relationships (emergence of links).

A social network dataset with topological information only, can contain enough information for predicting future relationships with better than random accuracy (Liben-Nowell and Kleinberg 2007). The complex structure or topology of a social network includes information which is not visible to the naked eye but can be used to make predictions for future interactions (e.g. link prediction based on the intrinsic topological patterns). This predictability from topology of the network only, is a very intriguing aspect of the data with relational components. We discuss our approach to the link prediction problem in Section 1.1. Secondly, going further than predicting relationships based on topology only, in this research, the topology of a network is fused

with the node features for the prediction of node labels (e.g. a person's political views in social networks). This type of classification problem is formally known as the node classification problem for networks. Attributes like gender, political preferences or age are a few examples of the standard node features in social networks. In this thesis we also work on the area of node classification in social networks. Elaborated in Section 1.2, our work on the node classification domain also indirectly leverages the link prediction problem for social networks. In the famous study by Liben-Nowell and Kleinberg (2007), it has been discussed that the link prediction problem in social networks is related to the problem of predicting missing links. Although missing link prediction and link prediction may sound similar, there are subtle differences between these two concepts (based on the upper bound of the kind of problems they deal with). In a missing link prediction task, it is assumed that there are hidden interactions between nodes that need to be inferred. Whereas in a link prediction task, the goal could also be predicting missing links but in combination with the main goal of predicting links which are likely to be formed in future. In other words, a link prediction can be thought of as forecasting links, whereas a missing link prediction does not deal with the time dimension. Perhaps, it can also be argued that the missing link prediction problem is a subset of the link prediction problem. An example of link prediction could be future friend prediction in social networks, whereas, an example of the missing link prediction is, predicting suspicious interactions in a mobile network which the users are trying to hide. Hence, although, most of the existing link prediction and missing link prediction techniques can be used interchangeably Liben-Nowell and Kleinberg (2007), the difference in their underlying problem-specific concepts, their precise theoretical benefits become more apparent for a particular application. For the node classification problem, use of the concept of missing link prediction makes more sense (as opposed to the link prediction problem), based on the argument that we are inferring already existing hidden interactions of the network. These hidden interactions can be predicted for a richer representation of the network, which interns, increases the performance of the node classification task (a more detail explanation is given in Section 1.2). In our node classification related contributions, we make use of this very notion of mission link prediction (see Section 1.2). This concept of missing link prediction can be useful for the task of recognising node patterns in networks which is termed as the classification problem. The difference between the classification problem for datasets with the relational components (i.e. networks) and without the relational components is that the relational elements contain intricate and potentially useful topological information. These topological patterns are crucial elements of the datasets for any predictive modelling applied to networked data. If a predictive model for networks does not consider these topological patterns, it can under-utilise the available information within the datasets, leading to suboptimal performance. In this work we show that these relational components can be further enhanced using our proposed approaches in Chapter 7 which are based on the concept of missing link prediction.

1.1 Link Prediction

As discussed earlier, the prediction of social relations, i.e. the link prediction is one of the most essential parts of the evolution of a social network, alongside new node appearance. This link prediction is the first goal of this project. The existing techniques for link prediction are discussed in Chapter 2. Developing new link prediction techniques or improving the existing approaches firstly requires identifying the main contributing factors to the formation of new links. The second step is to determine how to combine or utilise these contributing information for a better link prediction method. In this thesis two main contributing factors to link prediction are identified from the literature, namely popularity and similarity (Papadopoulos et al. 2012; Thwe 2013). The popularity of a node implies how popular or influential a node is in a social network, whereas similarity implies how similar two nodes are. Popularity can be measured using Centralities, such as the Degree Centrality and similarities can be measured using shortest path length between two nodes and other widely used similarity measures such as Katz, Rooted PageRank etc. This work proposes to combine these two contributing factors using a law found not necessarily in the social science but in the natural world, namely Newton's law of universal gravitation.

If we consider a social network, at its local level, how two people make a connection or interact could rely on two factors, (1) how popular, and (2) how similar these people are. These two concepts are well established in the link prediction paradigm (Papadopoulos et al. 2012; Thwe 2013). Intuitively, for social networks, predicting the appearance of links between two people, having both the popularity and similarity factors should entail better prediction accuracy than considering only one of these factors. In social networks, we already have a wide range of measures for calculating the popularity of nodes and similarities between them. Different centrality measures (e.g. degree centrality, closeness centrality or betweenness centrality) could be thought of as notions of popularity. On the other hand, scores from link prediction methods like Katz or AdamicAdar could be thought of as measurements of nodes' similarity (Freeman 1977; Katz 1953; Adamic and Adar 2003). However, the challenge is how to combine these two types of metrics in order to predict links between two particular nodes in the future. This is where we make use of Newton's law of gravity. In Newton's explanation of gravity, the force between two particles is proportional to the product of their masses and inversely proportional to the squared distance between them. We argue that this law of attraction between two point masses could also be applicable in social networks. We measure the popularity or importance of a node using centrality and consider it as mass. We measure dissimilarity by the inverse of similarity (i.e. scores from link prediction methods like Katz, AdamicAdar etc.) or by the path length, and consider them as distance. The detailed theoretical and empirical analysis of this novel Newtonian gravity inspired method is described in Chapter 5.

1.2 Missing Links and Node Classification

A link prediction algorithm typically predicts future links based on the current information available within the network. However, link prediction methods (many of them measure node similarity) can also be categorised as predicting missing links for a network at the current time (i.e. not for a future snapshot) (Liben-Nowell and Kleinberg 2007; Goldberg and Roth 2003; Popescul and Ungar 2003; Taskar et al. 2004). A link prediction method infers links which are not directly visible within the network but have a high likelihood of forming in the future (Liben-Nowell and Kleinberg 2007). This consideration of node similarities as missing links can address some of the limitations of the Graph Convolutional Network (GCN) (Kipf and Welling 2017), a state-of-the-art method for node classification, which we discuss in Chapter 7. GCN has been shown to outperform other state-of-the-art models on citation networks and a knowledge graph for the task of node classification. The GCN can efficiently combine the graph topology with the node features for the task of node classification. However, the mentioned limitation for the GCN is that at a particular l^{th} layer of the neural network model, it can only consider up to the l^{th} order of neighbourhood of nodes as influential, which may not always hold. This strong dependency between the highest layer and the size of node-neighbourhood can limit the node classification accuracy, especially for friendship-based networks. This is because, for a given node, a distant node (i.e. not directly connected) may have higher similarity than the directly connected nodes. These similarities between two distant nodes indicate that they will likely connect in the future, which in turn implies a missing link between them. In the work where GCN has been introduced by Kipf and Welling (2017), it has been applied and benchmarked for citations and knowledge networks. Thus, the evaluation of the full potential of the GCN on a friendship-based social network also requires openly available datasets in larger quantities. However, most available social network datasets are not complete (i.e. they represent a subset of the original networks e.g. ego-networks, not the entire graph or do not include the entire set of node features [1]). On top of that, the majority of the available social network datasets not only do not contain any features but also ground truth labels. Although there are mathematical models available for generating graphs, these do not generate features and labels, thus only the topology of the graph is obtainable. To address the need for good quality synthetic social network data with ground truth labels and features we provide a guideline on how to simulate dynamic social networks, with ground truth labels and features, both coupled with the topology of the network (Section 1.3). We then use three node-similarity measures, our Newtonian gravity inspired method coined as the Graph Gravity (GG), Katz and, rooted PageRank to increase node classification accuracy of the GCN. The models based on the combination of these similarity measures with a unique data reconfiguration technique outperform the original GCN model in 27 out of 30 simulated datasets that we have used. They also outperform or match the original GCN on two real-world

[1] Node attributes and features are used interchangeably in this work

citation network datasets. Additionally, we have proposed another variation of the GCN, which includes a weight matrix to learn the strength of each feature for defining the node labels for all the nodes. Chapter 7 includes the detailed discussion about these four new deep learning models based on the GCN concept.

1.3 Social Network Simulation

As mentioned in Section 1 there is now an unprecedented availability of social network datasets, however, most of the openly available datasets are not complete, i.e. they only represent subsets of bigger networks and/or do not contain node attributes. One major limitation of the neural network-based learning systems is that they require a large amount of data for training. This is one of the most significant differences between human intelligence and artificial non-general intelligence like an artificial neural network. Unlike a deep learning (i.e. deep neural network) model, a human can learn from a minimal number of examples, whereas a deep learning model requires to see a substantially larger number of samples to learn. Thus, it is essential to have access to a large number of training data instances to unlock and evaluate the full potential of the neural network-based model. A straightforward technique to solve this problem of insufficiency of the high quality real-world datasets for neural network-based learning systems is to simulate high quality real-world alike synthetic data and use it to train the model. Additionally, if not for training, simulated datasets are particularly useful to evaluate the models' performance, i.e. during the testing phase. In many cases, it is far more convenient to simulate test cases representing exceptional situations than collecting data for those situations in the real world. In fact, for some real-world scenarios, it might not even be possible to get a dataset describing some exceptional scenarios due to the rarity of the event or ethical constraints.

It is, however, crucial to test the trained model in those exceptional scenarios because the cost of failure for those unlikely situations can be significantly higher than a regular situation. One such area where high quality simulated and augmented data is extensively being used is in the neural network-based learning systems for self-driving cars. Almost all advanced autonomous vehicle technologies use simulated datasets. For example, Nvidia has developed the Nvidia Drive Constellation, a Virtual Reality Autonomous Vehicle Simulator (NVIDIA n.d.). Billions of miles have been driven in the simulated environment by Google's Waymo (Waymo n.d.) etc. Similar to the self-driving cars, in many other applications of deep learning, high quality simulated datasets are now in high demand.

With the advancement of graph specific neural network-based models, the demand for such datasets is growing rapidly. Furthermore, it is becoming more and more difficult to have access to complete (i.e. inclusive of node attributes) datasets representing social networks mainly due to user privacy concerns that we discuss later in this section.

Social network datasets are very complex in nature, thus, they can be difficult to simulate and there is a lack of comprehensive guidelines on how to simulate social network datasets with both the features and ground truth labels.

As mentioned earlier, graph data mining has become an active research area due to the recent advancement and popularity of social networks (Wasserman and Faust 1994; Newman 2018), especially the online ones. Advancements in graph-based predictive modelling or graph community detection algorithms require datasets with ground truth labels for evaluation purposes (Sapountzi and Psannis 2018). However, the majority of the available social network datasets do not contain labels. Moreover, real-world social network datasets contain high dimensional features (Pecli et al. 2018) that represent information about both nodes and relationships. For example, a Facebook user generates a variety of information such as posts he/she likes, photos, status updates, etc. Even in citation networks, there are features such as the domain, authors' affiliations, documents with thousands of words, etc (Popescul and Ungar 2003). In publicly available datasets, such features are rarely included.

This is due to the fact that during the anonymisation process of networked data, in most cases we need to get rid of the majority of features as these could be used to identify individuals (Townsend and Wallace 2016), potentially raising ethical concerns. De-identification of network datasets is particularly tricky because of the unique topological structure a network may have. In a 2011 Kaggle link prediction competition, the winning team successfully de-identified most of the network data by matching the anonymised network topology with the real network, instead of using the actual link prediction algorithm (Narayanan et al. 2011). On top of that, nowadays, even such graph datasets are becoming very difficult to obtain due to the aftermath of the notorious usage of the real-world dataset from social networks for the purpose of political influence (Hand 2018; Cadwalladr and Graham-Harrison 2018).

One of the infamous recent developments in data misuse is that around 50 million Facebook users' profiles have been analysed without their consent by Cambridge Analytica, a British political consulting firm. Moreover, it has been claimed that the data analysis was performed in order to influence the outcome of 2016's US election (Cadwalladr and Graham-Harrison 2018). This incident of the data breach along with many others, has ignited a backlash from social network users and politicians.

To ensure user's data is only used with explicit consent, governments and political unions are increasingly putting pressure on the technology companies on protecting user's data (Quinn 2018). Additionally, new regulations, such as the European General Data Protection Regulation (GDPR) on the usage of personal data, have already come into force in many countries, such as the UK (Bennett 2018). Unquestionably, such regulations are essential to guarantee user privacy. However, as a result, availability of datasets from social media in the public domain is sharply declining. Maintaining the advancement of the research in social networks requires good quality

real-world datasets. One solution is to supplement the real-world social network datasets with synthetic, good quality, real-world alike data.

Typically, a link prediction algorithm is tested based on its predictive power on a future snapshot of the network. A supervised link prediction algorithm should ideally utilise both the topology and available node attributes (Lichtenwalter et al. 2010; Pecli et al. 2018). For example, Scellato et al. (2011) found that including features such as places and other related user activity improves the accuracy of link prediction considerably. Most of the developments in link prediction have been based on a single snapshot of the network, although, incorporating the evolution of the graph may result in better performance in link prediction as shown by Tylenda et al. (2009) and Xu et al. (2018).

To address the need for openly available high-quality social network datasets, we have introduced an open-source Python-based social network simulation library with GPU computation and multiprocessing. In our social network simulation, we argue that the topology of the network is driven by a set of latent variables, termed as the 'social DNA' (*sDNA*), which define the preference of nodes towards the features of other nodes, and which are not necessarily exclusive to a single node, whereas the single node's entire set of features is. We consider the *sDNA* as labels for the nodes, mimicking the real-world social network scenario. We describe our simulation process along with the validation of the simulated datasets in Chapters 6 and 7.

1.4 Original Contributions and Outputs

1.4.1 Research Contribution

The four main research contributions of this thesis are:

1. In depth comparison and analysis of the three classical network models (random, small-world and scale-free networks models), leading to new insights on how different these three network models' properties (many of these properties do not have analytical solutions) are, for a fixed-size network. This has also laid the foundation of our network simulation framework in Chapter 8.

2. Development of a class of new nature inspired and robust link prediction approaches called Graph Gravity (GG). The proposed link prediction algorithms have better predictive power when compared with many of the existing link prediction methods.

3. Combining graph theory based link prediction with modern deep learning models in a coherent approach, resulting in development of four new node classification algorithms, outperforming the existing methods.

4. Comprehensive social network simulation framework, able to jointly model network topology and node features.

This thesis has several research outputs including, high-quality publications and open-source software developed in order to disseminate the research findings. Following are the publications resulting from the research presented in this thesis:

1. Wahid-Ul-Ashraf, A., Budka, M. and Musial-Gabrys, K., 2017, November. Newton's gravitational law for link prediction in social networks. *In International Conference on Complex Networks and their Applications* (pp. 93-104). Springer, Cham.
   DOI: https://doi.org/10.1007/978-3-319-72150-7_8
   Chapter: 5

2. Wahid-Ul-Ashraf, A., Budka, M. and Musial, K., 2018. NetSim – The framework for complex network generator. *Procedia Computer Science*, 126, pp.547-556. DOI: https://doi.org/10.1016/j.procs.2018.07.289
   Chapter: 4

3. Wahid-Ul-Ashraf, A., Budka, M. and Musial, K., 2019. How to predict social relationships – Physics-inspired approach to link prediction. *Physica A: Statistical Mechanics and its Applications*, 523, pp.1110-1129.
   DOI: https://doi.org/10.1016/j.physa.2019.04.246
   Chapter: 5

4. Ashraf, A.W.U., Budka, M. and Musial, K., 2019. Simulation and Augmentation of Social Networks for Building Deep Learning Models. *Preprint*.
   arXiv Preprint: https://arxiv.org/abs/1905.09087
   Chapter: 6, 7

5. Ashraf, A.W.U., Budka, M. and Musial, K., 2020. SocialDNA - Capturing Complex Nature of Human Behaviour in Social Networks. *Submitted, Scientific Reports*.
   Chapter: 6

There are three software libraries that have been developed during this project, two of which have been made open-source:

1. *NetSim* is developed in R to robustly (i.e. comparing many different parameter settings for each of the models) compare properties of three network models with different sizes and density. The library can generate networks with any given set of parameters and relevant reports for analysis. The code has been made open-source. The GitHub repository for the

code:
https://github.com/AkandaAshraf/netsim

2. *LinkPrediction*, an R library to benchmark and evaluate the new link prediction technique along with 12 other widely used link prediction methods. The software library uses multi-threading to optimise the performance. This library is now also being used by the undergraduate students in the University of Technology Sydney as a part of their coursework in Computing Science Studio and Research Projects.

3. *VirtualSoc* is a software developed in Python 3 to generate dynamic synthetic social network datasets with ground truth labels and features. There are two versions of this software, one for the CPU and another one for the GPU. No other network analysis library has been used to develop this software. The source code is available under the MIT license. The GitHub repository for the package can be accessed using the following link:
https://github.com/AkandaAshraf/VirtualSoc

1.5 Thesis Structure

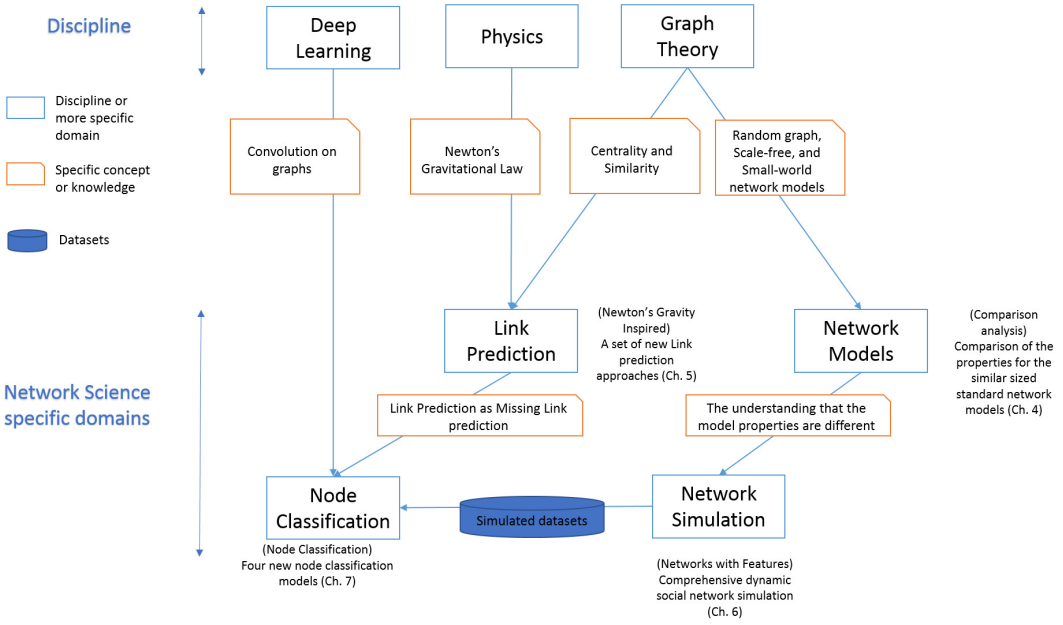


FIGURE 1.1: The relation between research domains, concepts, and Chapters 4-7

This thesis is organised as follows. In Chapter 2 we discuss related work and also propose a new categorisation of the link prediction methods. Chapter 3 is where we describe the methodology of the study, including the research questions, objectives, and tasks that have been undertaken to fulfil those objectives. Chapter 4 includes the analysis of different network models, sizes, densities and properties. We then describe the new link prediction methods with empirical

analysis in Chapter 5. Afterwards, a comprehensive network simulation framework is presented in Chapter 6 which is developed primarily based on our understanding of the three network models from Chapter 4. In Chapter 7, we validate the simulated networks and propose four new variants of the GCN. Finally, in Chapter 8 we conclude this thesis, also outlining future research directions. This thesis also includes an Appendix A with additional figures, statistics and information on the developed software libraries.

Chapter 2

Literature Review

In this chapter, different properties of a network are introduced from literature which is related to Chapter 4 where the behaviour of these properties with respect to different size and types of networks are analysed. Understanding these network properties are essential to any network analysis or modelling. After introducing the network properties, the focus is then shifted to network evolution in the form of link prediction. Also, the importance of link prediction from literature is brought forward in this chapter. In Chapter 5, a new physics inspired link prediction method is proposed. Literature related to this physics inspired link prediction are then pointed out. Additionally, the classification of the link prediction method is proposed in this chapter. A number of link prediction methods are then discussed in this chapter. Afterwards, different network models are presented which is relevant to the network simulation process in Chapter 6.3 and 7. These three network models are essential for the development of our network simulation library in Chapter 4 and 6.3. Finally, the Graph Convolutional Networks (GCNs) are discussed related to the four proposed new variants of the GCNs models that are developed and designed in Chapter 7.

2.1 Properties of Networks

There are different properties and measures of network characteristics, some of them are very easy to calculate precisely while others are must be estimated. They give insights and help infer essential properties of a network (Newman 2010*a*; Costa et al. 2007).

The two most important and straightforward measurements of networks are Degree and Shortest path.

Degree and degree Distribution: One of the most essential and fundamental properties of a network is its degree distribution. The degree of a node is the number of edges connected to it

and histogram of the degree of each node is the degree distribution of that network (Costa et al. 2007).

Shortest Path: The shortest path between a pair of vertices are called shortest path for that pair of vertices. In a whole network, the average shortest path could be calculated by taking the average of all shortest paths between all pairs of vertices. It could be calculated through breadth-first search (Dreyfus 1969). The shortest path is also known as the geodesic path (Newman 2010*a*).

Diameter: The longest shortest path in a network is known as the diameter of a network (Newman 2010*a*).

Transitivity: Transitivity is a critical concept, mainly for social networks. This is because high-level transitivity seems to be a very apparent feature of a social network (Newman and Park 2003). Transitivity was first introduced by (Newman 2001*b*). A straightforward explanation of transitivity could be that friend of my friend is also my friend. Perfect transitivity only occurs in a clique where three vertices form a fully connected subgraph. If A is connected to B, B is connected to C, and C is also connected to A then it could be said that this subgraph of A, B, and C forms a clique which is a perfect transitivity. If vertex C and A are not connected then A, B, and C are partially transitive. If A knows B, B knows C then we have a path of ABC of two edges in the network. If now A knows C then we have the triad. Clustering coefficient [1] in a network is the measurement of triads.

The Clustering Coefficient, C is measured using the following formula:

$$C = \frac{\text{number of closed path of length two}}{\text{number of paths of length two}} \tag{2.1}$$

In Equation 2.1, if C= 1, it implies perfect transitivity and all the components are cliques, c=0 implies no closed triads which indicates the network is a tree (Newman 2010*a*). In social network terms, it could be explained as the fraction of pairs with a common friend which are also friends with themselves.

$$C = \frac{\text{number of triangles *3}}{\text{number of connected triples}} \tag{2.2}$$

Here in Equation 2.2, three is multiplied because the connected triples are counted three times (Newman 2001*b*). Equation 2.2 is also known as the fraction of connected triples. Socials networks usually have higher clustering coefficient when compared with other technological and biological networks (Mislove et al. 2007). The high value of clustering coefficient is thought to be due

[1] this is not the same clustering coefficient where we measure groups or clusters of vertices

to the fact that we do not pick our friends randomly, rather two people have higher chances of being friends if they have a common friend.

Clustering Coefficient for a single vertex: It is a measurement of how strongly the neighbours of a vertex A is connected between pairs (Watts and Strogatz 1998). It is measured via:

$$C = \frac{\text{The number of pairs that are connected}}{\text{number of pairs of the neighbours of A}} \tag{2.3}$$

Equation 2.3 is called local clustering coefficient, and it has a rough dependence on the degree. Vertices with a higher degree have lower local clustering coefficient. This high local clustering coefficient is also an indicator of the structural hole (Butt 1992). The missing links between pairs of vertices are defined as structural holes. Structural holes are especially important in social networks. If we are interested in efficient information flow in a network, then this high clustering coefficient is undesirable. These structural holes reduce the number of alternative routes information can take through the network (Butt 1992). However, structural holes could also be a good thing, as having no connection between the adjacent pair of a vertex A means the adjacent pair have to connect via A which gives A the power to control information flow. In that sense, Structural holes could be a type of centrality measure as it defines the importance of a node in a network regarding controlling information flow. Centrality measures are discussed in later sections.

Mean Local Clustering Coefficient: Watts and Strogatz (1998) proposed a clustering coefficient for the entire network as a mean of local clustering coefficients for each vertex.

2.1.1 Centrality

Vertex centrality measures are fundamental in network analysis. Centrality quantifies how important or central vertices are in a networked system.

Degree Centrality: A simple, but perhaps the most crucial measure of centrality is the degree of vertices in a network. The degree of a vertex in a network is referred to the number of edges attached to it. Degree centrality is a handy measure of centrality in social networks.

Closeness Centrality: which is calculated based on the mean geodesic path from a given vertex to all other vertices in the network (Newman 2010*b*). High closeness centrality of a vertex means the vertex has better access to information or more direct influence on other vertices. Closeness centrality is defined as:

$$CC(v_i) = \frac{1}{\sum_{n \neq i} d(v_i, v_n)} \tag{2.4}$$

Here, d is the geodesic distance between two vertices. As it can be seen in Equation 2.4, if there are a total $n+1$ vertices in a graph, closeness centrality for vertex v_i is calculated using the inverse of the average length of the shortest path from/to all other vertices except itself $v_i \notin \{v_1, v_2, ..., v_n\}$. If the path does not exist between two vertices, then the total number of vertices is used instead of path length (Csardi and Nepusz 2006).

Betweenness Centrality: this centrality gives a score to a vertex v_i based on how many paths connecting any two vertices in the network go through that vertex v_i. If the number of those paths is high, then vertex v_i will have high betweenness centrality.

Vertices that are frequently on the shortest paths between any two vertices of the graph have more control over information flow (Freeman 1977; Anthonisse 1971). Removing a vertex with high betweenness centrality has a negative influence on the overall information flow in a network.

The betweenness centrality is different from other centrality measures as it does not consider how well-connected a vertex is but it measures how much a vertex falls in between others. This way it is possible to have a vertex with a low degree but high betweenness centrality. For example, two groups of vertices can be connected via a single path and then a vertex that connects those groups (a.k.a. bridge node or broker) can have a high betweenness centrality.

If a network has set of vertices V, source vertex $s \in V$ and target vertex $t \in V$, the betweenness centrality of vertex v_i can be defined as (Freeman 1977; Anthonisse 1971; Brandes 2001):

$$BC(v_i) = \sum_{s \neq v_i \neq t} \frac{\sigma_{st}(v_i)}{\sigma_{st}} \tag{2.5}$$

where σ_{st} is number of shortest paths between two vertices s and t and $\sigma_{st}(v_i)$ is the number of shortest paths between two vertices s and t that passes through node v_i.

Eigenvector Centrality: This centrality measure could be thought of as a bit more complicated measure of degree centrality. In degree centrality, a vertex gets one point if it has one neighbour, but in eigenvector centrality, the neighbours also get a rating. This centrality measure was first proposed by (Bonacich 1987). Eigenvector centrality for a vertex v_i is,

$$v_i = K_1^{-1} \sum_j A_{ij} X_{j'} \tag{2.6}$$

Here, M_{ij} is the an element of adjacency matrix M, K_1 is the largest value from eigenvalues K_i. To understand eigenvector centrality from a social network point of view, a person might have few friends or connection but still, have high eigenvector centrality measure because most of his/her connections are very important or central (have higher centrality measure). Usually, the value of eigenvector centrality is not normalised. This centrality works best for undirected

networks for directed networks it arises complicacy. In directed networks, the outward connection from a node does not necessarily give importance to the node from which it is pointed outwards. For example, a web page has an outward connection to thousands of other pages, and some of them could be very important). To get around this problem the eigenvector centrality of an undirected network is measured by the number of inward connections. One big problem with this centrality measure in the directed network is if a vertex has many inward connections and those vertices also have connections pointed towards other vertices. If we encounter one or many vertex or vertices have no inward connection, in this case, we will get zero measure by using eigenvector centrality.

Katz Centrality (Katz 1953): This centrality measure solves the major problem that we face with eigenvalue centrality, the problem of zero centrality discussed earlier. This problem is solved via adding a free weight to every node. Although the Katz Centrality is proposed to deal with the problem with directed network but it can still be applied, and sometimes very useful in undirected networks. It allows a vertex to have high centrality regardless of their neighbour's centrality measure if it has many connections.

$$v_i = \alpha \sum_j M_{ij} X_j + \beta_i \tag{2.7}$$

In Equation 2.7, α and β are positive constants. α is the eigenvector centrality, and it is the sum of centralities connecting to vertex i, and β is the extra constant value that all the vertices receive by default.

PageRank: One drawback of Katz centrality is if a vertex has high centrality value and if it points to another vertex, then the pointed vertex gains more centrality. Let's assume the case that, Facebook is pointed to millions of websites, so it has high centrality. However, if it points to a particular website, that website will gain centrality just because of Facebook has high centrality. Facebook is probably pointed to millions of other websites which are not particularly important. The idea behind solving this problem is by changing the centrality gain by dividing the gain by the vertices' out-degree. This centrality is the core of web ranking technology, and Google gave its trade name PageRank (Brin and Page 2012).

$$v_i = \alpha \sum_j A_{ij} \frac{X_j}{K_j^{out}} + \beta \tag{2.8}$$

In both Equations 2.7 and 2.8, the Katz and PageRank centrality there is a parameter α. This α parameter needs to be tuned. For undirected networks, α is less than one, and in the directed networks it is different, and in practical use usually, it is in the order of one. In PageRank, it is also possible to define different additive constant for different vertices.

2.2 Mesoscale Structures

One important aspect for the topology of social networks are their mesoscale structures, such as communities and core-periphery.

2.2.1 Communities

Community structures represent groups of nodes which have stronger interactions within themselves, i.e. intra-interactions, but weaker interactions with nodes outside that community, i.e. weaker inter node interactions. In social networks, these interactions are formed via links or edges (Fortunato 2010; Lancichinetti et al. 2010). There are several ways to identify communities in a graph. Identifying communities requires algorithms which can infer the best partitions between different subgraphs, representing different communities. Modularity optimisation or more specifically maximisation is one such algorithm which can quantify communities in graphs. A modularity maximisation algorithm typically measures the quality of the communities by finding a division of the network which gives the highest modularity. However, because the number of division of the network can be exponentially high, an exact solution is not practical, thus an approximation of best modularities are inferred for detecting communities in a large graph (Newman 2016). Another pathway to detect or infer communities in a network is by using statistical inference. In statistical inference, one fits a generative network model to the observed network. Initially, a total of n number of nodes is taken without any edges to attach to them, and then they are divided (using some stratigies (Newman 2016)) into q groups. Then edges are independently attached between all the nodes at random. The difference of probability of edges being attached within the nodes in a group and between the groups gives fitness of score of the community structure (Newman 2016). For nodes in a community, the probability of edges being attached within themselves is expected to be higher than nodes which are in a different community.

2.2.2 Core-periphery

A core-periphery structure of network implies that the network consists of two different structures, core and periphery. The core nodes are connected densely and the periphery nodes are sparsely connected (Da Silva et al. 2008; Borgatti and Everett 2000; Rombach et al. 2014; Zhang et al. 2015). A simple way to determine the core-periphery structure in networks is to divide the nodes based on their degrees. However, for more precise detection of the structure other algorithms such as fitting a stochastic block model (which fits the observations using expectation–maximisation algorithms) can be used (Zhang et al. 2015).

In this work, we do not explicitly measure the above mentioned mesoscale structures. However, there is implicit consideration of these structures within this work. For example, the community detection techniques mentioned in Section 2.2.1 are unsupervised, however, there are modern deep learning approaches which can identify communities in a supervised or semi-supervised context, which we talk about later in Section 2.4 and in Chapter 7 (Kipf and Welling 2017). For our link prediction algorithms developed in Chapter 5, we consider methods such ask Katz, rootedPageRank, Average Commute Time (ACT), and they implicitly consider the surrounding of a pair of nodes to predict link formation (Katz 1953; Brin and Page 2012; Lü and Zhou 2011). These surroundings generally contain information of the connectedness of nodes, thus considering the mesoscale structure. For example, if a pair of nodes' surrounding neighbourhood is more densely connected then the total number of paths between those two nodes is expected to be higher as well. We discuss these link prediction methods in more details in the next section.

2.3 Link Prediction

Networks are ubiquitous. Ranging from food webs, to protein, brain or social networks, they underpin many natural phenomena (Cohen et al. 2012; Jeong et al. 2001; Bassett and Bullmore 2016; Krioukov et al. 2012). In the broad landscape of network science, networks which are formed via social interactions, have been increasingly drawing a lot of research attention in recent years, due to the heterogeneity of their components and non-trivial dynamics. Data representing small-scale social networks were available and analysed in the past, for example, the famous Zachary's karate club network has been studied extensively since it was published by Zachary (1977) in 1977. However, Zachary's karate club contains only 34 nodes and 78 vertices, whereas today's social networks (e.g. Facebook, scientific paper citation, Twitter), contain billions of nodes and are far more complex and dynamic (Scott 2017).

2.3.1 Physics-inspired Approaches for Link Prediction in Social Networks

Although these large-scale social networks are formed by social interactions, their topological properties and dynamics are similar to those of networks found in nature. For example, most biological networks exhibits power-law degree distribution, cellular networks have high clustering coefficient, network encoding the large-scale causal structure of spacetime in our accelerating universe exhibits power-law degree distribution and high clustering coefficient (Barabasi and Oltvai 2004; Krioukov et al. 2012). Both these characteristics are also commonly found in social networks.

The similarity between anthropogenic social networks and naturogenic networks gives the opportunity to apply many different prediction and modelling tools developed in the field of naturogenic networks, to social networks. This is due to the fact that large-scale physical and biological networks and social networks exhibit similar topological properties (e.g. degree power-law distribution, high clustering coefficient) (Bassett and Bullmore 2016; Sporns and Zwi 2004; Krioukov et al. 2012). However, the similarities are explored at the global level and this causes some issues with precision of adopted models and methods because the local dynamics are not considered. This raises the question if we could also adopt laws which govern a physical system to predict social network at a local level. Local level of the network implies the level of nodes and links, where the interactions between at most two nodes are considered. Whereas, similarly to the physical world, the analytical solution at the global level are usually governed by theories such as chaotic dynamics (e.g. the three-body problem) in social networks.

Tools which are primarily used in order to analyse, model, or describe physical world have been used in social network analysis on numerous occasions (Borgatti et al. 2009). Some examples include Memetic algorithm for community detection in social networks, reaching of Bose gas state of complex social networks or the molecular model of social network (Gong et al. 2011; Krioukov et al. 2012; Bianconi and Barabási 2001; Juszczyszyn et al. 2009). The field with applications of physical models to social networks has been named as social physics by Urry (2004).

One of the main focuses of this research is the link prediction problem. The proposed model is inspired by the earliest theory of gravity, where Newton described the law of universal gravitation based on the force between two point masses. Authors have already attempted to use models from physics in the context of network structure prediction. In Budka et al. (2013) and Juszczyszyn et al. (2009) they adopted molecular models in the context of evolution of social network. Now, by applying Newton's gravitational law, we can extend the nature-inspired link prediction framework with a new method that allows to take into account more than one characteristic of the network, and not only distance between nodes as it was done in the molecular model.

2.3.2 Newton's Gravity in Social Sciences

Physics-inspired approaches in networked systems have been used in the context of force-directed graph drawing, where node centralities were used as masses Bannister et al. (2012). However, as opposed to our method, Bannister et al. (2012) did not use a measurement of distance or Newton's gravitational equation for predicting future interactions. One of the first applications of gravity in social science dates back to as early as in the mid-19th century, when Simini

et al. (2012); Carey (1867) reasoned that physical laws are also applicable in social phenomena (Griesinger 1979). There are also some approaches using the theory of gravity to solve link prediction problem, however, most of these works are related to modern physics i.e. quantum mechanics (Urry 2004; Budka et al. 2013; Juszczyszyn et al. 2009; Bianconi and Barabási 2001).

In the study by Levy and Goldenberg (2014), Newton's gravitational law is used in link prediction. The authors used spatial distance (i.e. not topological) and substituted friendliness for masses. In fact, inverse square law in terms of geographical distance has been used earlier than in (Levy and Goldenberg 2014). Not specifically in link prediction but in the field of social gravity, Zipf (1949) and Stewart (1948) both have applied the inverse square law. In fact, they have considered the original notion of Newtonian gravitational law where the interaction between two groups of people is proportional to their cardinality, and inversely proportional to their squared geographical distance (Griesinger 1979; Zipf 1949; Stewart 1948). The problem with this approach in online social networks is that in some cases the physical distance is either not available or not indicative of the relationship strength. Therefore, in this study we take the inverse of different similarity measurements from scores of Katz, AdamicAdar, and Rooted PageRank (RPR) as a measurement of distance, and use centrality measures as masses.

Recent developments in the use of gravitational law include the study by Bastami et al. (2019), where the concept of gravity has been used to find optimised subgraphs to reduce the search space for link prediction techniques. In our approach, we have used the concept of Newtonian gravity directly for predicting future links by combining the notions of popularity and similarity, which is different than the approach proposed by Bastami et al. (2019), where the Newtonian gravity is used only to reduce the search space. In another study by Salha et al. (2019), the Newtonian gravity has been used for Graph autoencoders, and variational autoencoders – a class of link prediction approaches targeted towards directed link prediction. The Newtonian gravity has been used to reconstruct the directed graph topology from the node embedding. The authors recognise the fact that Newtonian gravity has limitations and the modern Einstein's theory of relativity is a better candidate to explain gravity as the spacetime curvature instead of a Newtonian force, however, the application of Newtonian gravity is still useful, for example, for precise approximation of the effect of gravity when the gravitational fields are not extreme (Salha et al. 2019). In this section, we have given a review of the uses of Newton's gravity in social sciences, which extends the application of Newtonian gravity beyond physics.

2.3.3 Link Prediction Methods Classifications

There exists different types of link prediction methods. Numerous works have been dedicated to review and classification of link prediction methods (Getoor and Diehl 2005; Liben-Nowell and

Kleinberg 2007; Lü and Zhou 2011; Al Hasan and Zaki 2011; Wang et al. 2015; Martínez et al. 2016). One of the widely used and accepted classifications is by Liben-Nowell and Kleinberg (2007), where the authors grouped link prediction methods as follows:

1. Methods based on node-neighbourhoods (e.g. Common Neighbours (Newman 2001*a*), Jaccard's Coefficient (Salton and McGill 1986), AdamicAdar (Adamic and Adar 2003), Preferential Attachment (Barabâsi et al. 2002))

2. Methods based on the ensemble of paths between a pair of nodes (e.g. Katz (Katz 1953), Hitting time (Liben-Nowell and Kleinberg 2007), PageRank (Brin and Page 2012))

3. Higher-level approaches (Low-rank approximation (Deerwester et al. 1990; Liben-Nowell and Kleinberg 2007), unseen biagrams (Essen and Steinbiss 1992; Lee 1999; Liben-Nowell and Kleinberg 2007), clustering (Liben-Nowell and Kleinberg 2007))

Classifications, like the one above, give us a better understanding of the principles that are used when link prediction methods are proposed, e.g. if a method works at a local or global level of the network or use path or node based similarity, etc. However, they neglect the information about applicability of different methods, i.e. those classifications do not answer a question in what circumstances and for what networks the methods can be used. For example, for some methods (e.g. Katz) an input is a single snapshot of a network, while others (e.g. Triad Transition Matrix (TTM)) require a time series as an input (i.e. a sequence of historical snapshots of the network) (Juszczyszyn, Musial and Budka 2011; Backstrom and Leskovec 2011). As a result, methods like TTM are not applicable to network datasets where only vertices and links are given without historical information (Juszczyszyn, Musial and Budka 2011). Also, there are other methods which may use additional information about node attributes like age, location, etc. (Lichtenwalter et al. 2010; Chen et al. 2009). Based on the type of information exploited by link prediction methods, we categorise link prediction methods into four groups:

1. **Unsupervised – based on topological information**, which are methods that only use structural information such as mutual friend count in social networks, path lengths, triad profiles etc. Some examples include methods like Katz, PageRank, and AdamicAdar (Liben-Nowell and Kleinberg 2007). These methods only require a snapshot of the network topology at any given time t to make predictions for time $t + 1$, and are useful when past and non-topological information is not available. These methods are applicable to any type of network dataset and do not require training.

2. **Supervised – based on topological information**, which are methods only applicable to networks where historical information regarding network's topology is available. For example, if snapshots of a network at $t - 1$ and t are given, then $t - 1$ is considered as

historical information. Network characteristics like degree of certain nodes at time $t-1$ can also be considered as historical information. One example of such method is the Triad Transition Matrix (TTM) (Juszczyszyn, Musial and Budka 2011; Backstrom and Leskovec 2011). A wide range of machine learning approaches also fall into this category if the topological information such as mutual nodes, shortest distance etc. is considered as features, and link appearance is considered as class label (Lichtenwalter et al. 2010; Lichtenwalter and Chawla 2011; Fire et al. 2011). Methods in this category do not use non-topological information such as age, location etc.

3. **Unsupervised – based on non-topological and/or topological information**, which are methods that consider non-structural information like age, location, preferences etc. (Tan et al. 2013; Lichtenwalter et al. 2010; Chen et al. 2009). In this category topological information of the network could also be used in combination with the non-structural attributes mentioned above.

4. **Supervised – based on non-topological and/or topological information**, which are methods applicable to the same kind of datasets as in point two above. If non-structural historical information of a network is considered (with or without topological information) any binary classifier could be used to make predictions in this setting (Al Hasan et al. 2006).

There are quite a few methods that fall into the first category (Getoor and Diehl 2005; Liben-Nowell and Kleinberg 2007; Lü and Zhou 2011; Al Hasan and Zaki 2011; Wang et al. 2015; Martínez et al. 2016). These methods are applicable to any kind of network where only one structural snapshot is available. Despite the fact that these methods only exploit network topology without historical information or node attributes, they make more accurate predictions for future links compared with a random predictor, according to Liben-Nowell and Kleinberg (2007). This is because the sole topology of a social network encodes the predictive patterns (e.g. the number of mutual friends, path length etc.) of the network. The link prediction approaches consider these patterns of the social networks from its topology to make predictions.

Definitions of popular linked prediction methods falls into the first category are outlined below:

2.3.4 Link Prediction Methods

1. **Common Neighbours (CN),** which is a similarity metric where the likelihood of two nodes v_i and v_j to develop a link depends on the number of mutual friends (Newman

2001*a*). This method could be quantified via Equation 2.9 (Γ represents the set neighbour of a node v):

$$Score(v_i, v_j) = |\Gamma(v_i) \cap \Gamma(v_j)|, \tag{2.9}$$

2. **Jaccard's Coefficient (JC),** which is a version of Common Neighbours (Salton and McGill 1986) normalised by the total number of neighbours of both nodes:

$$Score(v_i, v_j) = \frac{|\Gamma(v_i) \cap \Gamma(v_j)|}{|\Gamma(v_i) \cup \Gamma(v_j)|} \tag{2.10}$$

3. **AdamicAdar (AA),** which is a similarity metric used in information retrieval (Liben-Nowell and Kleinberg 2007) similar to the Jaccard's Coefficient (JC). In this method the likelihood of two nodes being connected in the future depends on the number of Common Neighbours (e.g. mutual friends in a social network) relative to the nodes' degrees (Adamic and Adar 2003): In Equation 2.11, C denotes rare features (in this case a rare feature is having many mutual friends while the degree of both nodes is low) common between nodes A and B:

$$Score(v_i, v_j) = \sum_{v_k \in \Gamma(v_i) \cap \Gamma(v_j)} \frac{1}{\log |\Gamma(v_k)|} \tag{2.11}$$

As an example, consider nodes A (with 500 friends) and B (with 300 friends) which are not yet connected and have 30 mutual friends. Now, if another two not yet connected nodes X and Y have 100 and 70 friends respectively, and also 30 mutual friends, using this formula the (X, Y) pair will get a higher score although the number of mutual friends between both pairs is the same.

4. **Preferential Attachment (PA),** which is based on the social concept of 'rich get richer' implying that nodes with higher degree are more likely to get new links (Barabâsi et al. 2002):

$$Score(v_i, v_j) = |\Gamma(v_i) \cdot \Gamma(v_j)| \tag{2.12}$$

In our approach product of node's degree has also been used, as it is in here. However, Preferential Attachment does not consider shortest path as we have utilised it in equation 5.2.

5. **Katz,** which considers the number of all the paths from node v_i to v_j (Katz 1953). The shorter paths have bigger weight (i.e. are more important), which is damped exponentially by path length and the β parameter (M denotes the adjacency matrix):

$$Score(v_i, v_j) = \sum_{l=1}^{l=\infty} \beta^l |path^l_{v_i,v_j}| \tag{2.13}$$

$$Score(v_i, v_j) = \beta M + \beta^2 M^2 + \beta^3 M^3 + \cdots \tag{2.14}$$

$|path^l_{v_i,v_j}|$ is the number of paths between v_i and v_j, and the length of this path is l. β is a scalar parameter, which needs to be smaller than the reciprocal of the highest eigenvalue of M (Landherr et al. 2010).

6. **Rooted PageRank (RPR),** which is used by search engines to rank websites. In graph analysis it works by ranking nodes, and the rank is determined by the probability of each node being reached via random walk on the graph (Brin and Page 2012). The $Score(v_i, v_j)$ is calculated using the stationary probability distribution of v_j in a random walk. The random walk returns to v_i with the probability α at each step, moving to a random neighbour with probability $1 - \alpha$. We have calculated RPR for every dataset using two different α parameters and they are $\alpha \in \{0.15, 0.25\}$.

7. **Average Commute Time (ACT),** which is an average number of steps it takes to visit node v_j from node v_i and come back to v_j using random walk (Lü and Zhou 2011):

$$Score(v_i, v_j) = RandWalk(v_i, v_j) + RandWalk(v_j, v_i) \tag{2.15}$$

This could be obtained using pseudoinverse of the laplacian matrix (L), which is L^+, where $L = B - M$ (Kunegis and Lommatzsch 2009; Fouss et al. 2007; Klein and Randić 1993). Here, B is the degree matrix (a diagonal matrix which contains degree of every vertices) and M is the adjacency matrix.

$$Score(v_i, v_j) = \frac{1}{C(l^+_{ii} + l^+_{jj} - 2l^+_{ij})} \tag{2.16}$$

Because we are considering the rank, constant C could be removed. Here l^+ are the elements in matrix L^+.

8. **Average Commute Time Normalised (ACTN),** which is the same as ACT but normalised by stationary distribution, $\pi = \frac{B}{\sum_v B}$ (Lovász 1993; Zhou and Schölkopf 2004).

9. **Pseudoinverse of the Laplacian matrix (PsInLap),** which is simply the pseudoinverse of the graph Laplacian L^+. PsInLap defines kernel of a graph and can be interpreted as a similarity measurement (Fouss et al. 2007).

10. **Local Path Index (LPI),** which is based on the number of paths of different lengths between two vertices. LPI is a generalisation of CN. While CN measures similarity based on mutual friend count, which effectively gives the number of paths with length two between two vertices, LPI also takes into account paths of length three (Zhou et al. 2009; Lü et al. 2009). LPI is hence a more global similarity measure than CN but not as global as Katz:

$$Score(v_i, v_j) = M^2 + \epsilon M^3 \tag{2.17}$$

Here, ϵ is a free parameter. M^2 gives us path counts for distance two and M^3 gives us path counts for distant three. If we choose it to be zero then this would give us common neighbours, and if we consider all higher orders of M (the adjacency matrix) then this would essentially become Katz. In our experiments we have used two values for $\epsilon \in \{0.01, 0.02\}$.

11. **Leicht-Holme-Newman Global Index (LGI),** which is a similarity metric utilising the concept that if two nodes v_i and v_j have neighbours who are themselves similar, then they have higher similarity score (Leicht et al. 2006):

$$Score(v_i, v_j) = B^{-1}\left(I - \frac{\theta}{\lambda}M\right)^{-1}B^{-1} \tag{2.18}$$

Here θ is a free parameter and λ is a the largest eigenvalue of adjacency matrix M. We have used $theta \in \{0.5, 0.7\}$ in our setup.

12. **Matrix Forest Index (MFI),** which is a similarity score between v_i and v_j, defined as ratio of the number of spanning rooted forests, such that vertices v_i and v_j belong to the same tree which is rooted at v_i to all spanning rooted forests of the entire network (Chebotarev and Shamis 2006):

$$Score(v_i, v_j) = (I + L)^{-1} \tag{2.19}$$

Here, I is the identify matrix. A spanning subgraph of a graph contains the same vertices as the main graph, but not all the edges. A forest is a cycleless graph and a tree is a connected forest. A rooted tree is a tree which has only one root (Chebotarev and Shamis 2006).

2.3.5 Network Models

There are many different types of network models. The purpose of studying network models is to generate artificial networks in the large scale which enables us to understand behaviour and properties of networks.

Random-graph Model: In the random graph network model, one creates a network with some properties of interest (specific degree distribution) and otherwise random. Although random graph model was first studied by Solomonoff and Rapoport (1951) this model is mainly associated with Paul Erdős and Alfréd Rényi (Erdös and Rényi 1959; Erdős and Rényi 1961). The random graph model can also be thought of as choosing uniformly a network from the set of all possible networks with exactly n vertices and m edges. This model is often referred to as $G(n, m)$ model for networks where n is the number of nodes and m being the number of edges. A slightly different definition is for $G(n, p)$ model, where p defines the probability of edges appearing between all possible pairs of vertices.

Scale-free Model: The scale-free model shows power law node degree distribution $P(k) \sim k^{-\alpha}$ (where k is the node degree and typically $2 < \alpha < 3$) for a social network. This kind of distribution was first discussed by Price (1976). Price, in turn, was inspired by Herbert Simon, who discusses power law in a variety of non-network economic data (Simon 1955). Simon showed mathematically the fact that 'rich get richer' effect results in power law distribution, calling this mechanism 'cumulative advantage', but it is more often known as 'preferential attachment' as coined by Barabási and Albert (Barabási and Albert 1999). However, a recent study by Broido and Clauset (2019), seems to have refuted the idea that social networks are highly scale-free. The analysis by Broido and Clauset (2019), based on 927 real-world network datasets from five different domains (including social networks), found that a strong scale-free structure in social networks is very rare. They have also found that among all the network types, a handful of technological and biological networks appear as strongly scale-free as opposed to the social networks (Broido and Clauset 2019). However, this study has sparked a debate as the view of scale-free networks being representative social networks has been very prevalent (Holme 2019; Voitalov et al. 2019; Serafino et al. 2019). We discuss this issue in Section 6.7 in more details.

Small-world Model: Transitivity measured by the network clustering coefficient despite being extensively studied, is still one of the least understood properties in network analysis according to Newman (2010*a*). Another important property we observe in real networks is the small-world effect – all nodes are connected with each other by relatively short paths. To model these two properties Watts and Strogatz introduced a small-world network model (Watts and Strogatz 1998). Classical small-world model rewires edges in a simple circle model to random positions. In a slightly modified version, the whole process starts from a circle and no edges are removed,

but new edges are inserted between randomly chosen vertices (Newman and Watts 1999). An important model parameter is p – the probability of rewiring/adding edges.

Forest-fire Model: In this model the new node, i, connects to another existing node j, and then again makes a connection with the adjacent node j_1 of the newly connected node j_0. The node i then carries on making connections with a probability p based on adjacent nodes (Leskovec et al. 2005; Drossel and Schwabl 1992). For example, in citation networks, an author finds a paper and cites it. He or she then cites more papers through that paper recursively (Leskovec et al. 2005). In a social network, a friend j may introduce someone i with his/her mutual friend and then the friend circle grows for the person i (Leskovec et al. 2005). The model is named as forest fire because it imitates self-organising behaviour of a forest fire (Drossel and Schwabl 1992). The spread of forest fire model follows three rules, (1) a burning tree becomes an empty site, (2) a green tree burns if at least one of the adjacent tree is burning, and (3) at an empty site a new tree grows with probability p (Drossel and Schwabl 1992). However, similar to the other models discussed, this model does not account for node features and labels which are present in a social network. Connections made with mutual friends in social networks are not purely random; rather both parties evaluate each others features before making a connection.

These quintessential network models are one of the most important contribution towards understanding and modelling complex networks. However, these mathematical models are solely driven by the topology of a network. For example, the Scale-free model considers the degree of a node and the Small-world model considers mutual friends. Neither features nor labels of nodes and/or connections are mimicked by those models. However, one can generate synthetic social networks with features is to find similarities/correlations between randomly assigned n number of features and let those similarities define connections (Papadimitriou et al. 2011; Symeonidis et al. 2010; Papadimitriou et al. 2012). For obvious reasons, this naïve approach is not ideal due to several limitations. Firstly, correlations between feature vectors do not consider the network topology. Secondly, a common correlation metric would assume every person in a social network views and prefers a potential friend's features equally in a linear fashion. Finally, it is often not obvious what the node labels are, which is an issue we discuss in detail in Section 6.1.

However, there are some recent developments in agent and event-based social network modelling which are discussed below.

Agent-based modelling: Bruch and Atwell (2015) provide a guideline on the agent-based modelling of social networks. In the paper by Bruch and Atwell (2015), it is argued that the interplay between the micro and macro level characteristics is complex, and the macro level characteristics are not emergent solely from the simple aggregation of micro level characteristics or low level entities such as social network users (Granovetter 1978). Instead, micro and macro level behaviour or characteristics form a feedback loop, resulting in a nonlinear interaction. From a

social network point of view, the graph level and node level characteristics could be thought of as macro and micro level characteristics of the network respectively.

In our approach, we consider both the graph and node level characteristics for a comprehensive representation of the real social networks. These characteristics are captured through the well-studied behaviours and mathematical properties of social networks. The node level characterises are the features, and individual preferences. Graph level characteristics are properties such as, considering path length while making connections, common preferences among groups of nodes etc. We discuss these in more details in the Section 6.1. We provide detailed and specific instructions, i.e. equations and algorithms, on how to model these characteristics of social networks resulting from the well-studied social network properties. Our study provides a detailed guideline which can be used straightaway to simulate social networks.

In another work by Kavak et al. (2018), the authors have argued that modelling should be performed by explicitly using available real-world dataset. In their experiment, they have simulated human mobility model based on 826,021,868 twitter messages. Furthermore, they have uncovered the Geolocation of 92,296 users for the purpose of modelling. However, the purpose of our simulation is to produce synthetic good quality graph structured datasets when real-world data is not available, which is increasingly the case.

Event based modelling: One recent interesting development in modelling dynamic event-based graph is the Cognition-driven Social Network (CogSNet) model (Michalski et al. 2018). The CogSNet models social network-based on the human memory model. Authors argue that, similar to the human memory, a social event is strengthened by repeated exposure to a similar event and weakens by deprivation of that event. Although CogSNet proposes a new paradigm in social network modelling, it does not provide an explicit explanation modelling features within the dynamic event based graph. Providing open-source social network datasets with labels, features, and graph or topological characteristics is the primary goal of this study.

To address the issues discussed above, i.e., (1) lack of guidelines on implementing both the well-studied network properties in social networks and features, (2) insufficient research on simulating dynamic social networks with node features, (3) lack of rigorous study providing directions on defining node labels in social networks, we propose a framework for social graph simulation. In our model, the simulated networks have the following characteristics based on understanding of Facebook-type social networks, along with well-studied social network properties such as preferential attachment (Chapter 6.3 and 7).

2.4 GCN - A Neural Network Model for Graphs

Graph Convolutional Network (GCN) is a semi-supervised classification model shown to outperform other state-of-the-art graph classification approaches based on as little as 0.07% of labelled nodes per class (Kipf and Welling 2017). In the paper where GCN is introduced, datasets considered in the experiment were citation networks and knowledge graphs with explicitly defined class labels (Kipf and Welling 2017). However, defining class labels for Facebook, Twitter, LinkedIn like social networks is not trivial. As discussed earlier, the difficulty is mainly associated with obtaining real-world graph datasets with labels and node attributes. The GCN has a limitation of assuming a strong dependency between the layer of the model and the node-neighbourhood, which is also pointed out by Kipf and Welling (2017). This strong dependency results in limiting the receptive field of the GCN up to the higher number of layer used in the model. To solve this issue of the dependency between the highest number of layer and the node-neighbourhood we have proposed three new variants of the GCN model discussed in Chapter 7. These new models are mainly inspired by the analysis of link prediction methods in Chapter 5.

Chapter 3

Methodology

This chapter discusses the methodology of the research project. First, we talk about the Research Questions for this project in Section 3.1. Then the objectives of the thesis are described in Section 3.2. Finally, the tasks that are undertaken to meet the objectives of the project are given in Section 3.3.

3.1 Problem Statement and Research Questions

Availability of large-scale online social network data combined with modern computational power has given the opportunity to infer answers to questions social scientists have been asking for a long time. Understanding how communities form, how information flows through social interaction, the evolution of social interactions etc. are only few examples of useful insights that we can derive from social network analysis. The approach of understanding social science through computational approaches is termed as computational social science (Lazer et al. 2009). One of the most crucial inquiries in the social network analysis is how a social network evolves. Evolution of the social network is termed as dynamics of a social network and has been introduced in Chapter 1. Understanding the pattern of social interactions between people in a social network is essential if we were to capture or understand the dynamics of the network. Understanding the dynamics of the social network is also a requirement for the prediction of the future states of a network, unless a black box classifier model from machine learning approach is used. In a black box model, the model captures the dynamics of the network and uses it for prediction, however, the captured model is largely not human interpretable.

In Chapter 1 three different aspects that affect the evolution of a network have been introduced, i.e.(1) changes in links (e.g. emergence and disappearance), (2) changes in nodes (e.g. emergence and disappearance of nodes), and (3) changes of properties/features (of both nodes and

links), and one of the main ways an online social network evolves is through node interactions, i.e. by forming new links. In this project, link prediction is the main focus.

Given a network at time t, the link prediction problem is to identify new links that will be present in the network at time $t+1$ (Bliss et al. 2014; Hristova et al. 2016). Assuming the network has a set of nodes V and set of edges E at time t expressed as $G(V, E_t)$, and that a link between a pair of vertices v_i and v_j is denoted by $L(v_i, v_j)$, the goal of link prediction is to predict whether $L(v_i, v_j) \in E_{t+1}$, where $L(v_i, v_j) \notin E_t$ (Equation 3.1). The prediction is performed by using topological and/or non-topological information about nodes characteristics and their relationships.

$$\text{Predict } L(v_i, v_j) \in E_{t+1}, \text{ Where } L(v_i, v_j) \notin E_t \tag{3.1}$$

To predict or understand how new links form we also need to understand if there are patterns that the network follows for the formation of new links. These patterns mainly emerge from different rules vertices follow for forming new links. For example, the scale-free is the rule where new links are formed with vertices which already have many links (Barabâsi et al. 2002). Whereas, Common Neighbour based rules prefer forming new links with nodes which have a high number of mutual connections (Newman 2001*a*). However, these rules could also change as the network ages. It is reasonable to think that, if we have multiple patterns or rules for interactions in a social network, combining those multiple rules should entail better prediction compared with when only one rule is considered. Different nodes in a social network are different topologically. Also, in real life, everyone has a different preference towards social interactions. Some people in society have more influence, attention, fame or weight than the others, from a social network analysis point of view. Most of the link prediction methods do not consider these differences in nodes. Treating different nodes according to their characteristic and then combining with rules of link formation may improve the prediction accuracy. One way to distinguish (social importance and influence through the graph topology) between nodes is through measuring their centrality. Centrality reflects a person's importance in society. Another important aspect in link prediction is whether we should consider local topological properties of a pair of nodes or we should consider larger more global characteristic of a pair of vertices for predicting future links between them. Some of the methods such as Common Neighbour consider very local information, whereas some others utilise more global topological properties, such as rooted PageRank.

The link prediction problem can also be formalised as the missing links problem. There are several deep learning models which are specifically designed for graph type datasets, such as the Graph Convolutional Networks (GCNs). These models directly consider the adjacency matrix as the topology of the network for predicting important information such as node labels.

The adjacency matrix is a type of representation of the graph where node-similarities are not considered, whereas the node-similarity matrices (i.e. link prediction methods) measure node similarities, and a high value of the similarity can imply a missing link. Whereas lower value in the similarity may imply the link is not very important or may disappear in the future.

In summary, the research questions naturally emerging from this discussions are as follow:

1. How different are the network properties such as clustering coefficient, average shortest path or node centrality for the three standard network models (random graph, scale-free, and small-world)?
2. What is the principled way to combine both the node importance and similarities between nodes for a better link prediction approach? Additionally, is there a generic approach to fuse both the global measures with local measures for a pair of nodes to predict links between them?
3. How to increase the accuracy of the modern deep learning based models on graphs using the traditional graph theory approaches such as link prediction methods for node classification?
4. Can usage of the node-similarity measures solve one of the limitations (the strong dependency between the layer and node-neighbourhood) associated with the GCN?
5. How to simulate social network datasets beyond the capacity of the basic mathematical network models? How to combine these models in a single generative process?
6. How can we validate the coupling of the features, labels, and the topology in the simulated networks?

3.2 Objectives

Based on the research questions, the following seven objectives are defined:

1. Compare and contrast different properties of the three fundamental network models (i.e. scale-free, random graph, and small-world) with varying size and the density on a network.
2. Design and evaluate a new link prediction technique capable of incorporating individual node's importance and similarity between a pair of nodes.

3. Develop a software library to benchmark the proposed link prediction methods along with the existing ones. Also, based on the developed software, analyse the performance of the newly developed link prediction methods.

4. Adapt the state-of-the-art GCN model to exploit information from link prediction methods in order to increase the predictive power.

5. Develop and validate a comprehensive guideline to simulate social networks such that both the topology and features of the network are coupled together, incorporating dynamic network generation within the simulation process.

6. Develop a simulation software package for the developed simulation algorithms.

7. Disseminate the new findings, methods, analysis, and the developed software packages.

Objective 1 is to answer Research Question 1. Objectives 2 and 3 corresponds to Research Question 2 and Objective 4 is to answer Research Questions 3 and 4. Finally, Objectives 5 and 6 are associated with Research Questions 5 and 6. The evaluation of these objectives is presented in Section 8.1.

3.3 Tasks

In order to fulfil the objectives, the following tasks were undertaken:

1. Random graph, scale-free, and small-world networks are analysed and compared for a wide range of size and density. The detailed analysis are described in Chapter 4. This task is for the partial fulfilment of Objective 1.

2. To understand how a network's property changes as the network's model type changes an open-source software has been developed (link to the software package is given in Chapter 1, Section 1.4). This task is also to fulfil Objective 1.

3. To fulfil Objective 2, based on the existing literature, anthropogenic and naturogenic networks are compared in Chapter 2, Section 2.3.1. In Chapter 5 based on this comparison (i.e. between anthropogenic and naturogenic networks), a Newtonian gravity inspired link prediction method is proposed. This technique can effectively combine both the local and global measures of a network in addition to combining both the popularity and similarity measures.

4. For Objective 3, an optimised (parallelised) software package, implementing 12 widely used link prediction methods along with our proposed ones, has been developed. The library also produces Precision-Recall Curve (AUC), and Receiver Operating Characteristic

(ROC) curve-based ranking of the methods. The usage of the library along with examples of the produced plots and statistics are given in Appendix A, Section A.2.

5. Additionally, for Objective 3, the performance of the proposed, Newtonian gravity inspired link prediction technique are analysed using precision-recall (PR) curve (in Chapter 5) for seven real-world network datasets. Results from the ROC curves are also given in Appendix A, Section A.2.

6. Three new variants of the GCN model has been proposed based on Newton's gravity inspired node-similarity measure along with two other link prediction methods in Chapter 7. This task has been accomplished to address Objective 4.

7. Another variation of the GCN has been developed which can learn the relative importance for each of the features in a network for all the nodes. This has been discussed in Chapter 7 and corresponds to Objective 4.

8. In Chapter 6, a detailed analysis and modelling techniques to simulate social networks have been developed based on our understanding of the three classical network models and today's Facebook-type friendship-based social networks. Furthermore, two algorithms are proposed for the simulation process of dynamic network datasets with features and ground truth labels. This is to address Objective 5.

9. For Objective 5, experiments have been designed to validate the coupling of the topology and features. The simulation process is validated based on the outcome of the experiments (Chapter 7).

10. A highly parallelised (supporting both the Graphics Processing Unit (GPU) and Central Processing Unit (CPU) computation) simulation software package has been developed to simulate social network datasets based on our simulation strategy. Sample scripts of the software to simulate networks are given in the Appendix A, Section A.6. This task addresses Objective 6.

11. The proposed new models for node classification have been tested on the generated simulated (using our simulation framework) and two real-world citation networks. This task addresses Objective 4.

12. Two articles in peer-reviewed conference proceedings and one in a peer-reviewed domain specific high impact factor journal have been published. Another article is currently under review and the preprint of the submitted paper has been made available through arXiv. Additionally, the two developed software packages have been made open-source and are currently on GitHub. This is to fulfil Objective 7. The list of published articles and the open-source software are given in Chapter 1, Section 1.4.

Chapter 4

Overview of Network Models - Simulation Study

In this chapter, we analyse three basic network models which are crucial for understanding real-world social networks and their properties. All these network models represent some partial aspects of real-world social networks. We analyse different network properties for the three networks and how they relate to each other and varying size and density.

Network properties such as closeness centrality, betweenness centrality, degree distribution, clustering coefficient, and average geodesic path vary depending on the network's type and size. These properties contain essential information about the network's structure and dynamics. For instance, a high clustering coefficient with low average geodesic path might indicate a social network, i.e. a specific type of network that is formed through social interactions. Degree distribution could indicate a particular growth mechanism unique to a specific kind of network – power law in the degree distribution indicates that the growth mechanism is likely based on preferential attachment and the network has been formed based on the 'rich get richer' rule. To understand how different properties of a network behave with regards to the type of the network, and how properties change with network's size, we have designed and performed a number of simulations. In these simulations we have compared properties of three network models: (1) Barabási-Albert model for scale-free network (2) Watts-Strogatz model for the small-world network, and (3) Erdős-Rényi model for the random network.

To compare properties across three different network models of varying size and density, all these models for a particular size needs to have the same number of vertices and edges. Due to the different underlying mechanisms of these three network models, this is not very trivial. This difficulty is encountered mainly with scale-free and small-world network models, and we discuss it further in Section 4.2.2. To obtain comparable results we have developed a set of mathematical formulas that allow generating networks following these three models with a set number of

vertices and edges. This, together with a network simulator that can be easily extended to include other network models and properties, is the primary contribution of this work. The developed simulator, together with generated networks have been open-sourced on GitHub as an R package[1]. The entire simulation workflow has been automated, so the user only needs to define input parameters to get a comprehensive set of results in the shape of multiple plots and tables. The second and equally important contribution of this research is the comprehensive analysis of the results of over 30,000 simulations which we have run to generate and investigate networks following different network models.

We have measured three types of network properties: (1) centralities (betweenness and closeness), (2) average shortest path, and (3) global clustering coefficient. For a given network model, we varied the number of vertices and edges, yet keeping them fixed across all three network types to make the results comparable.

In Section 4.1 we present a brief overview of existing network generators. Next, the proposed **NetSim framework** together with implemented network models are introduced in Section 4.2. In Section 4.3, we describe the design of the experiment that enables comparison of properties of implemented models, followed by results analysis in Section 4.4.

4.1 A Brief Overview of Network Generators Landscape

To our knowledge, there does not exist any framework which generates comparable simulated networks in a comprehensive way for the random graph, scale-free, and small-world networks with a wide range of freedom in parameters but keeping the number of edges and vertices equal for all networks. Below we briefly introduce a few widely used packages.

The *igraph* software package (Csardi and Nepusz 2006) is a collection of network analysis tools available in R, Python and C. In our developed framework we have used *igraph* library in R for generating individual models and calculating network properties. However, our NetSim framework allows to generate all three models, with same the same number of edges and vertices for a range of different parameters and to save all the calculated properties as serialised objects which are then used to present the results in a transparent manner with generated plots. The whole process is automated and the user does not need to worry about keeping the number of edges and vertices the same, only vertices are taken as inputs.

Brain Connectivity Toolbox (Rubinov and Sporns 2010) is a MATLAB toolbox for complex network analysis of structural and functional brain connectivity data sets. This toolbox has several models, including random and scale-free networks, and includes tools for network comparison which however focus mainly on different community structures, not network properties.

[1] https://github.com/AkandaAshraf/netsim

MatlabBGL (Gleich 2007) is another MATLAB package for working with graphs. It is based on the Boost Graph Library and enables generation of simple models like random graphs and cycle graph (Siek et al. 2001).

Stanford Network Analysis Platform (SNAP) (Leskovec and Sosič 2016) is a library written in C++ for high-performance large-scale complex network analysis. It allows to generate regular and random graphs.

There exist a plethora of other software packages that allow to generate networks. While those presented above are just a few examples, it is important to emphasize that none of them enables to generate, in an automatic way, networks that are easily comparable. Existing tools enable to generate networks of different models, but each model is used in isolation. If a user wants to create networks that can be compared, i.e. having the same number of nodes and edges regardless of their structure, they have to work out all parameters by themselves which, as presented in the next section, is not a trivial task.

4.2 Proposed NetSim Framework

In this section, we introduce the network models implemented in the NetSim framework as well as our approach to generate networks with a consistent number of nodes/edges regardless of their structure. We have mathematically derived and empirically demonstrated our framework to be able to simulate and support analysis with varying important parameters of three different network models. Our proposed framework, which has been turned into an open-source R package, NetSim, is very user-friendly. We can simulate large-scale networks once, calculate their properties and then save those network properties as serialised objects which can then later be fetched from hard-drive to generate plots and perform comparative analysis. This helps to reduce the complexity of the process as generating a large scale network is computationally expensive. Our R package is based on *igraph* (Csardi and Nepusz 2006) and *ggplot2* (Wickham 2011), where the former is used for the core graph calculations and the latter for generating high resolution plots.

4.2.1 Network Models

In our study, we have simulated three widely used network models: (1) Barabási-Albert model for the scale-free network (Barabási and Albert 1999), (2) Watts-Strogatz small-world model for the small-world network (Watts and Strogatz 1998), and (3) Erdős-Rényi model for the random graph network (Solomonoff and Rapoport 1951; Erdös and Rényi 1959; Erdős and Rényi 1960; Erdős and Rényi 1961). A brief introduction of these models is given below.

4.2.2 Network Generation Approach

To compare different properties of the three different types of networks, two parameters have been chosen to be fixed. These are the number of edges m and the number of vertices n. For the random graph model, it is very simple as we use the $G(n, m)$ model and generate the ensemble directly.

In case of a scale-free network, because this is a growing network model, the final number of edges in the network depends on how many edges are attached in each step S as well as on the total number of vertices n.

The total number of edges in a scale-free network with n nodes and S edges added in each growth step is:

$$m = n \cdot S - \frac{S \cdot (S+1)}{2}. \tag{4.1}$$

For a small-world network, the total number of edges for a network with n vertices and Nei number of neighbourhood within which the vertices of the lattice will be connected are:

$$m = n \cdot Nei. \tag{4.2}$$

To keep the number of edges to m for the small-world network same as the scale-free network (i.e. same as in Equation 4.1) we randomly delete a fixed number of edges x from the small-world network in Equation 4.2. As a result, the total number of edges for a small-world network can be represented using the following equation:

$$m = n \cdot Nei - x. \tag{4.3}$$

If we now solve both Equations 4.1 and 4.3 for x:

$$x = n \cdot (Nei - S) + \frac{S \cdot (S+1)}{2} \tag{4.4}$$

From Equation 4.4, we can see that number of edges that needs to be deleted does not depend on the number of vertices when Nei and S are equal as the first part of the equation on the right-hand side becomes 0. This is why in our comparison, we have considered S and Nei parameters to be equal. When it is 0, $\frac{S(S+1)}{2}$ might not have an impact on the type of the model for a large number of vertices as long as S is comparatively small. In our comparisons, we have considered S up to 16. The value is chosen as power of 2, where the power is from 1 to 4 i.e. $S \in \{2, 4, 8, 16\}$.

The effect on the number of edges deleted from the small-world network is negligible, as the number of edges that we delete are very small compared to the total number of edges. Additionally, we simulate 30 times for each of the models and report the average from those 30 networks. The highest number of edges that need to be deleted (x) in our simulation is 120, when the parameter Nei is set to 16 based on Equation 4.4. In Table 4.2, we show the corresponding number of edges for all the networks when Nei is set to 16. As we can see that the number of edges that needs to be deleted is constant for a particular parameter of Nei, as a result as the network starts to grow the fraction of deletion of edges becomes increasingly smaller. However, if there were any significant effects on the networks generated we would see that effect for networks which are small but with a higher value of the Nei parameter. In Table 4.1, we can see that the effect of the removal of random edges on the average shortest path for the small-world network with $Nei = 16$ and difference in the shortest path in Table 4.1 is very small. Moreover, for a particular value of Nei, the number of edges that are deleted is constant, hence, the trends within the network properties for a particular value of Nei are unaffected.

Nodes	100	200	300	400	500
deleted	1.701063973	1.856758794	1.949889260	2.035537176	2.115439412
original	1.676787879	1.847455611	1.939777778	2.025406850	2.105063727

TABLE 4.1: Average shortest path length of small-world networks with deleted and not deleted edges for $p = 0.3$ and $Nei = 16$. Each of the networks are sampled 30 times.

We have generated networks for scale-free and random graphs with n vertices and m edges. The number of edges is calculated from Equation 4.1, which is the number of edges we get in scale-free networks for a different number of vertices n. For small-world networks, first, a lattice with n vertices and Nei number of neighbourhood within which the vertices of the lattice will be connected is generated. From the generated lattice x edges (Equation 4.4) are randomly removed. After that, the edges are rewired with probability p to obtain small-world network.

Following the process described above, we obtained networks of the three discussed models that are comparable as they have the same number of nodes and edges which, in turn, enables comparative analysis of different networks' properties including clustering coefficient, different types of centralities and average shortest paths.

4.3 Design of the Experiment

In our simulation study, we have varied a number of parameters to generate different networks within three basic mathematical models. For all three network types we have $n \in \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000,$

Network Model	$\mid n \mid$	$\mid \alpha \mid$	$\mid p \mid$	$\mid S \mid$ or $\mid Nei \mid$	Sampling
Scale-free	19	19 × 9 = 171		171 × 4 = 684	684 × 30 = 20520
Small-world	19		19 × 5 = 95	95 × 4 = 380	380 × 30 =11400
Random Graph	19	19		19	19 × 30 = 570
			Total	1083	32490

TABLE 4.3: Number of simulated networks.

$9000, 10000\}$ vertices. For scale-free networks we have used different values of α which are $\alpha \in \{1.5, 1.75, 2, 2.25, 2.5, 2.75, 3, 3.25, 3.5\}$. In the case of small-world networks we used a range of values of $p \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$.

Number of edges for scale-free and small-world networks is calculated using Equations 4.1 and 4.3 from section 4.2.2 respectively. For the random graph, we have used $G(n, m)$ model where the number of edges is directly defined. Note, that number of edges will be the same for all networks for a given set of S and Nei values. Table 4.2 shows number of edges which correspond to each S and Nei parameters for each vertex count defined from set n.

S	Nei	Number of edges in a network
2	2	197, 397, 597, 797, 997, 1197, 1397, 1597, 1797, 1997, 3997, 5997, 7997, 9997, 11997, 13997, 15997, 17997, 19997
4	4	390, 790, 1190, 1590, 1990, 2390, 2790, 3190, 3590, 3990, 7990, 11990, 15990, 19990, 23990, 27990, 31990, 35990, 39990
8	8	764, 1564, 2364, 3164, 3964, 4764, 5564, 6364, 7164, 7964, 15964, 23964, 31964, 39964, 47964, 55964, 63964, 71964, 79964
16	16	1464, 3064, 4664, 6264, 7864, 9464, 11064, 12664, 14264, 15864, 31864, 47864, 63864, 79864, 95864, 111864, 127864, 143864, 159864

TABLE 4.2: Number of edges for different values of S and Nei

In our simulation, we have shown empirically that our method keeps the number of edges and vertices the same for all types of networks and is consistent with Equations 4.1 and 4.3.

For each combination of α and p, both S and Nei are increased as a power of 2. $S, Nei \in \{2, 4, 8, 16\}$. The reason behind keeping S and Nei the same is described in Section 4.2.2. Each of the networks was generated 30 times, and the mean for each analysed property was calculated over those 30 samples. Table 4.3 summarises the total number of generated networks. Without sampling, we have in total 1083 different types of networks with different parameters that we have analysed in our study.

4.4 Results and Analysis

In this section, we present results that were obtained using the NetSim framework. Those include betweenness and closeness centralities, average shortest path, and global clustering coefficient. We compare those properties and analyse them with respect to number of nodes and edges as well as network types.

4.4.1 Number of Edges and Vertices

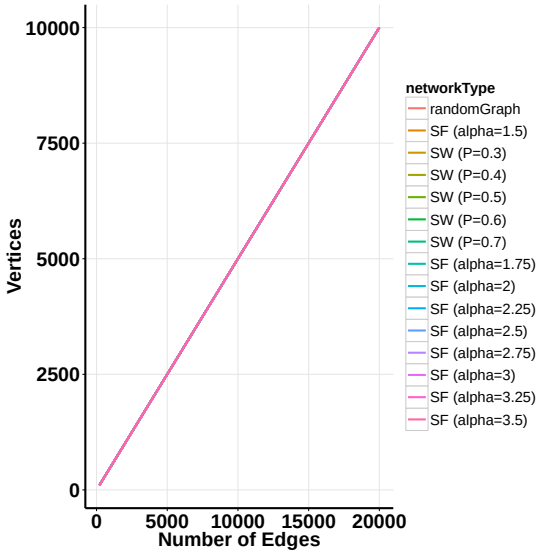
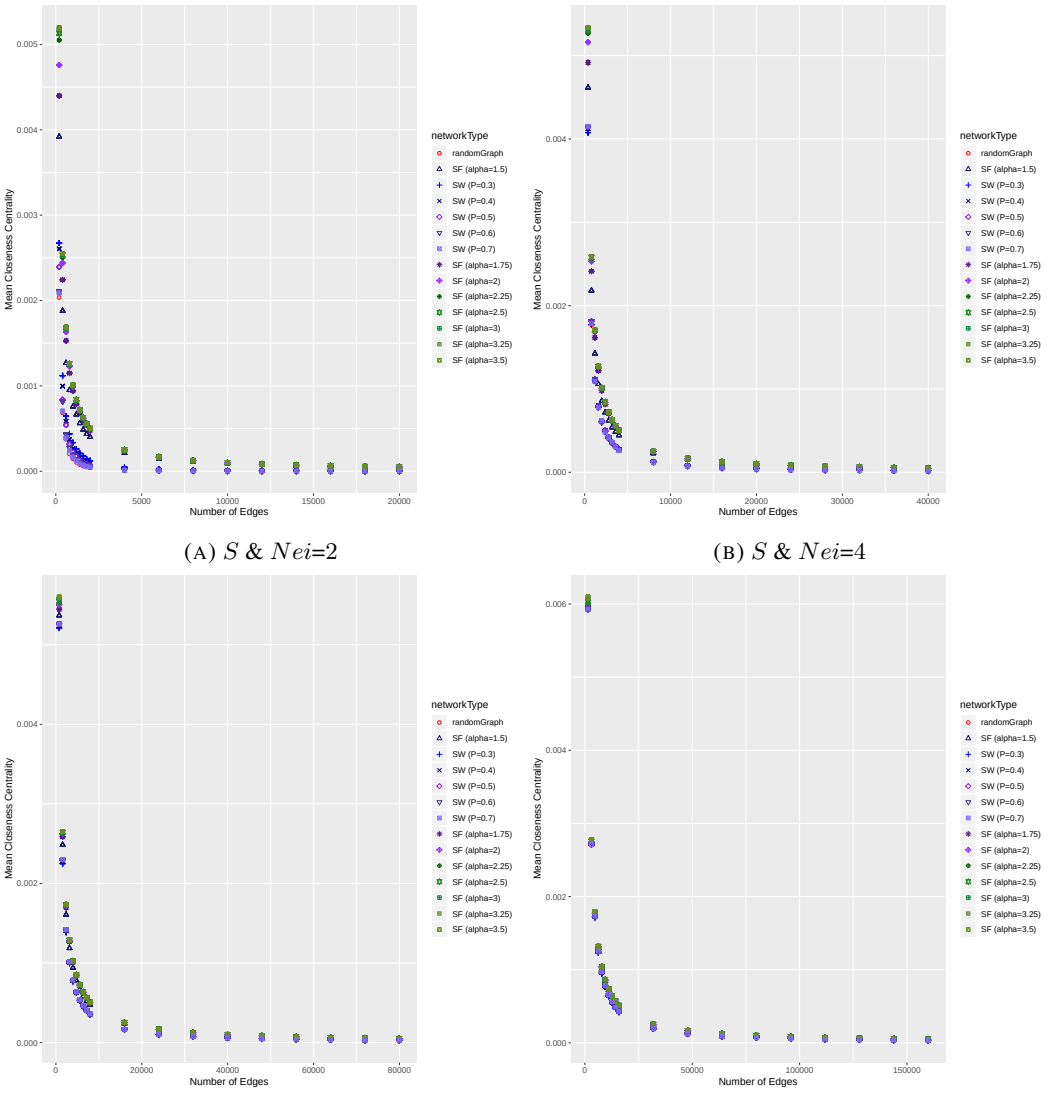


FIGURE 4.1: Edge and Vertices plot for random graph networks, small–world networks, and scale–free networks; $S = 2$ $Nei = 2$.

Figure 4.1 shows the number of edges in relation to the number of vertices for different networks. Plots for all networks overlap and form a straight line. This is because in each of the networks we have the same number of vertices and edges which shows empirically that our proposed method described in section 4.2.2 is viable. In Figure 4.1 the number of edges in relation to the number of vertices is presented for $Nei = 2$ and $S = 2$, but the same trend holds for all other experimental settings from Table 4.2.

4.4.2 Closeness Centrality



(A) S & Nei=2 (B) S & Nei=4

(C) S & Nei=8 (D) S & Nei=16

FIGURE 4.2: Closenesss Centrality in relation to number of edges for random graph, small-world, and scale-free networks with different values of S and Nei. Enlarged plots are available in Appendix A, Section A.1.1

Figure 4.2 shows mean closeness centrality in relation to the number of edges. Closeness centrality measure is calculated as inverse of the average geodesic path. Closeness centrality is expected to have higher values in social networks due to the small-world effect, which implies that every vertex should be connected with other vertices via a short path even if the network is very large.

As network size is increased closeness centrality decreases for all types of networks. This is because all generated networks become sparser as they grow in size (i.e. moving from left to right of a single plot). Although we have increased the number of vertices and edges together,

the number of edges is not increased sufficiently quickly with respect to vertices to maintain a dense network. However, increasing the value of S and Nei implies more edges, thus a denser network. For the value of 16 of both Nei and S, the simulation results in networks with the biggest number of edges among all generated networks. For $Nei = 16$ and $S = 16$ mean closeness centrality is similar for all generated networks. This implies that as the network density grows, different types of networks might exhibit similar behaviour.

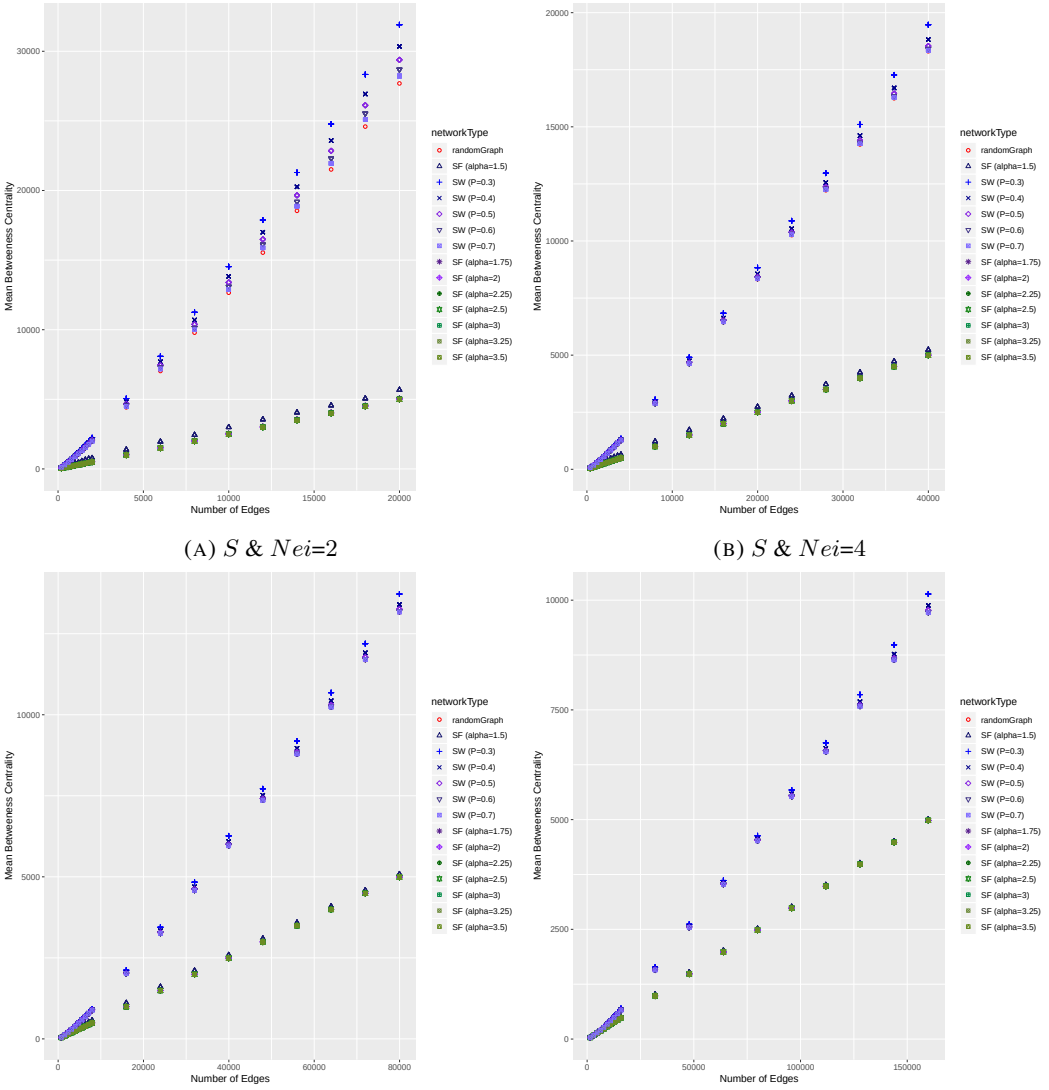
Compare different network types one can see that random graph has the lowest value for mean closeness centrality. This is something expected as a pure randomly grown network does not have the small-world effect. Scale-free networks with higher values of α seem to have a higher value of mean closeness centrality. This might, in turn, implies that the "rich get richer" effect has a positive impact on the mean closeness centrality of the entire network.

4.4.3 Betweeness Centrality

Betweenness centrality is a measure of how much influence a vertex has on the information flow within a network. In our study, we see a linear increase in the mean betweenness centrality values for all the networks with increasing size of the network (Figure 4.3). This is because as the network becomes bigger in terms of the number of vertices, each vertex lies between more vertices in general.

One crucial observation here is that there are apparently two clusters of networks in each of the plots. There is a definite difference between scale-free networks and small-world/random networks. As we make networks denser by increasing values for S and Nei, networks in those two clusters remain separate but tend to behave more like each other within the same cluster. We also see that as the network gets denser, betweenness centrality decreases.

Scale-free networks tend to have lower values of betweenness centrality, and the value decreases with growing values of α. This can imply that the "rich get richer" effect is responsible for the decrease in mean betweenness centrality values. Although for small-world and random networks this centrality measure is far higher than for scale-free networks, randomness tends to play a negative role here. We see that with higher values of p in small-world networks, they tend to have lower mean betweenness centrality although they lie in the upper cluster. Higher values of p in small-world network imply more randomness as the lowest value of all in the upper cluster is a random graph network.



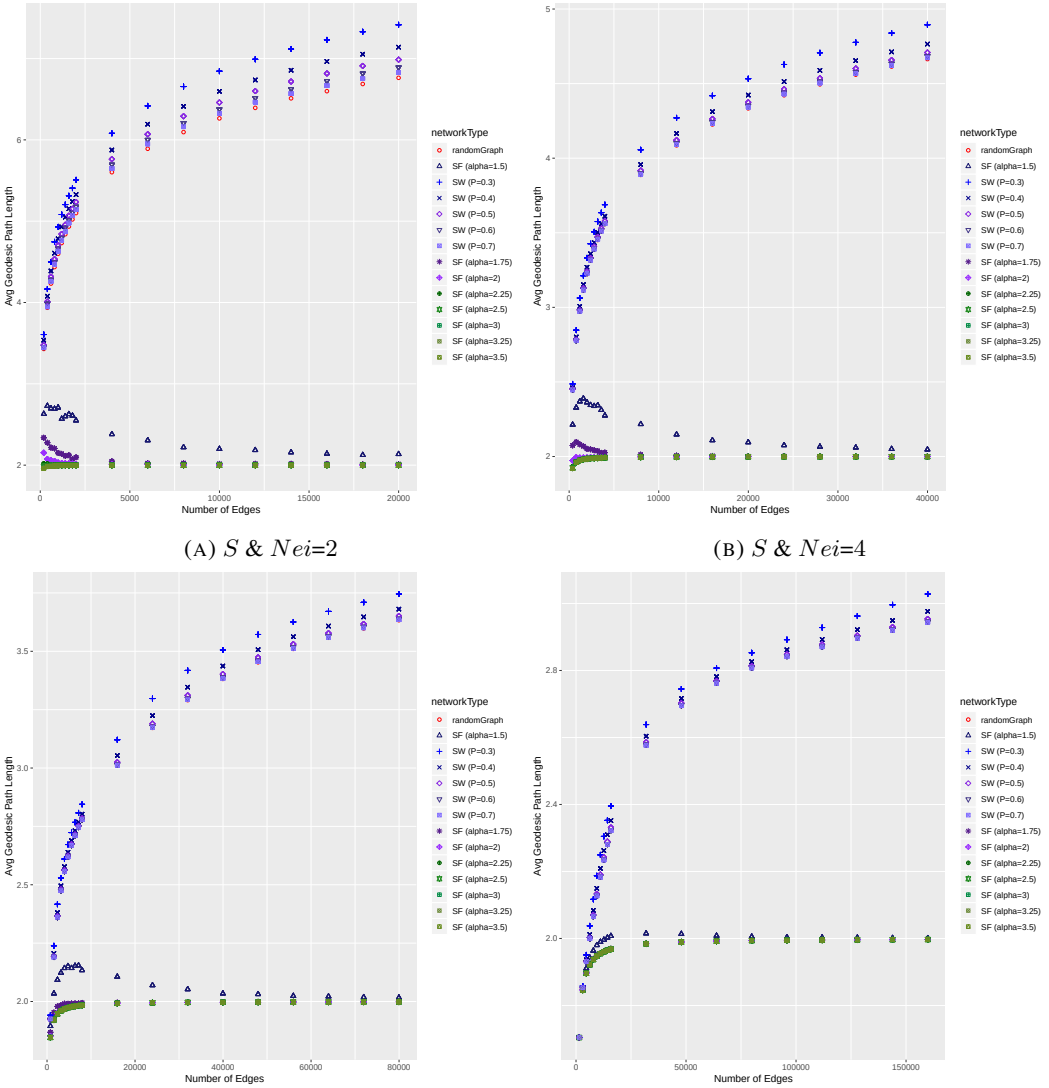
(A) S & Nei=2 (B) S & Nei=4

(C) S & Nei=8 (D) S & Nei=16

FIGURE 4.3: Betweenness Centrality in relation to number of edges for random graph, small-world, and scale-free networks with different values of S and Nei. Enlarged plots are available in Appendix A, Section A.1.2

4.4.4 Average Shortest Path

The average shortest path is calculated as the mean shortest path for all pairs of vertices. In our analysis, we have taken the average of shortest paths between all pairs of vertices and plotted against growing number of edges, see Figure 4.4.



(A) S & Nei=2 (B) S & Nei=4

(C) S & Nei=8 (D) S & Nei=16

FIGURE 4.4: Average Shortest Path in relation to the number of edges for random graph, small-world, and scale-free networks with different values of S and Nei. Enlarged plots are available in Appendix A, Section A.1.3

It is interesting to see that as S and Nei are increased resulting in a denser network, the shortest path is decreased for all three types of networks with different parameters. This implies that the small world phenomena are more apparent in a denser network and this is true for every small-world, random graph, and scale-free network.

Another important observation is, when plotted, every figure shows two clusters with regards to the average shortest path length. Small-world networks with lower values of p tend to fall in the lower cluster which has shorter average shortest paths, whereas networks with higher values of p fall in the upper cluster. This is intuitive as having lower values of p means fewer connections are rewired in our networks from the initial lattice which has less small-world phenomena. When the value of p is increased, it is closer to a random network, which has lower values of average

geodesic path. The purely random network falls into the lower cluster having a very low value of average geodesic path in comparison to the top cluster. All scale-free networks fall in the cluster with lower values of the average geodesic path.

Interestingly size of the network does not seem to influence the value of average geodesic path of scale-free networks, and they stay almost constant, whereas for the cases of small-world and random graph networks the more edges and vertices, the higher the value of average shortest path. In fact, for some of the scale-free networks, increasing the size of the network tends to result in smaller values of the average geodesic paths. This is expected, as new edges tend to be attached to vertices where we already have edges due to the "rich get richer" phenomena. This results in a stable value of average geodesic path with regards to the size of the network.

Scale-free networks with a high value of the α parameter, have an almost constant path length of two. This is due to the network's preference new link attachment towards nodes which already have a higher degree (higher number of links attached to them). This higher rate of preferential attachment creates hubs, where few nodes are connected with all other nodes in the graph, resulting in at most two path length in the graph for most of the nodes (as every pair of nodes will have one high degree node in common as a mutual friend).

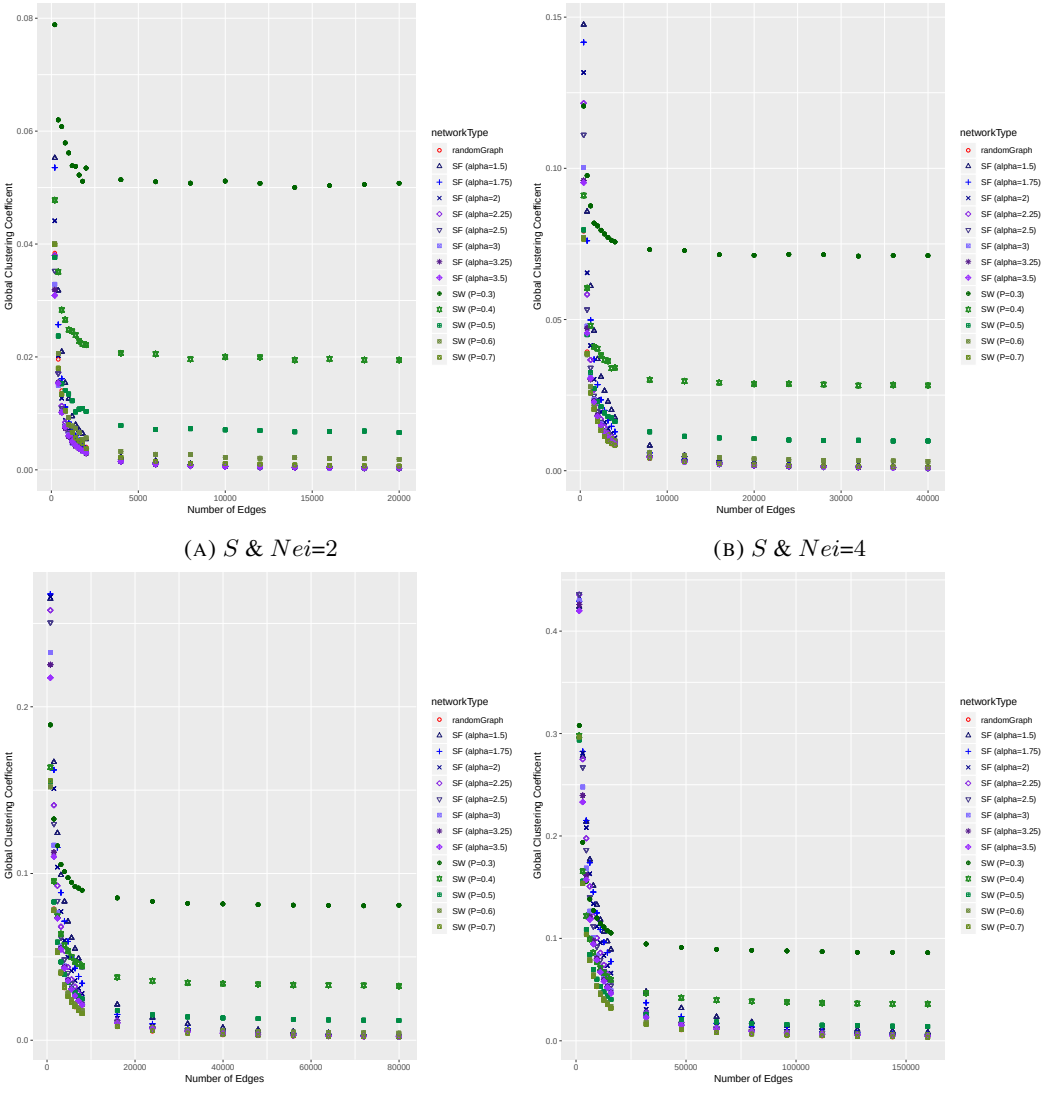
One important note for all the plots in Figure 4.4 is that as we go right on the x-axis, we have more edges and vertices as detailed in Table 4.2. However, with a higher number of vertices, the network tends to become sparser as we are not increasing the number of edges at a rate which will keep the density constant. So, on a single plot, as we move from left to right, we have sparser and larger networks while in four plots with different parameters of S and Nei, increasing the number of S and Nei results in denser networks.

4.4.5 Global Clustering Coefficient

The global clustering coefficient (a.k.a. transitivity) is measured as the ratio of the triangles (for given three vertices it includes three closed triplets) to the connected triples in networks.

We observe larger values for transitivity for small-world networks with lower values of p, which is to be expected (Figure 4.5). The lower the value of p the smaller the number of connections rewired, and the network is closer to the initial lattice which has a higher number of connected triples. Also, scale-free network tends to have lower values of transitivity, and it decreases with increasing α (Figure 4.5). This is also expected as a scale-free network is constructed based on biases towards already existing links, and edges are not evenly spread, resulting in a lower number of connected triples.

Another important observation is that transitivity seems to be constant with respect to increasing number of edges and vertices but not density. With denser networks (increasing S and Nei)



(A) S & Nei=2 (B) S & Nei=4

(C) S & Nei=8 (D) S & Nei=16

FIGURE 4.5: Global Clustering Coefficient in relation to number of edges for random graph, small-world, and scale-free networks with different values of S and Nei. Enlarged plots are available in Appendix A, Section A.1.4

transitivity tends to decrease until it reaches a stable point. This is more visible with small-world networks that have a smaller number of edges rewired. Other networks seem not to be significantly influenced by the changes in density.

Our NetSim software allowed us to conduct a simulation study that resulted in comparative analysis of three main network models: random, small-world and scale-free. For all generated networks, we analysed closeness and betweenness centrality as well as average shortest path and global clustering coefficient.

Our experiments allowed to compare selected properties of different network models. The conducted study revealed some interesting insights into how different network structures influence the properties in question.

Results for clustering coefficient and average shortest path confirmed the analytical results that are known for the three considered models (Newman 2010*a*). Additionally, it is apparent that for scale-free networks the average shortest path does not depend on network density, whereas for small-world and random networks the average shortest path decreases as the density increases. The clustering coefficient achieves the biggest value, regardless of the density, for the small-world network with the smallest probability of rewiring $p = 0.3$. This confirms the theoretical consideration as this network in its structure, is closest to regular lattice which has very high clustering.

For closeness centrality, especially for denser networks, there is not much difference in its values across the network models. For sparse networks, closeness centrality tends to be smaller for small-world and random networks than in the case of scale-free networks. For scale-free networks, density does not influence closeness centrality, and it becomes stable whereas for small-world and random networks the closeness centrality is bigger as the network gets denser. For betweenness centrality, there is a clear difference between scale-free networks and other models. Betweenness for scale-free networks is consistently smaller than for other models. We attribute that to the 'rich get richer' phenomenon that causes a concentration of edges in hub nodes.

4.4.6 Discussion

In our approach we have not normalised the metrics as we are interested in comparing them against each other and analysing their trends with respect the the size and density. In order to compare the properties of graphs generated based on different models, with different model specific parameters, we need to calculate their actual properties as opposed to simply normalising them based on their size and density. This is due to the complex nonlinearity existing in all the graph models. For example, the trend we see in a graph for different properties does not linearly scale with respect to its size and density. In a complex system the behaviour is governed by chaotic dynamics, and such complex system's future behaviour may even result in drastic changes which are impossible to predetermine even though they are fully deterministic (e.g. a system's drastic change in characteristics after the tipping point (Bellamy and Hulme 2011; Lindsay and Zhang 2005)). In Section 6.7.1, we have discussed this deterministic nonlinear behaviour of social networks in more detail.

4.5 Chapter Summary

This study offers a generic network simulator and a set of analyses which have revealed some interesting, and previously unknown characteristics of the networks. Our results indicate that only

by looking at a wide spectrum of generated networks we can identify extraordinary phenomena (e.g. mean betweenness centrality differentiates a scale-free network from a random graph or small-world network) that can otherwise be overlooked. Such information could be useful not only to identify the type of real-world networks but also to calculate properties that cannot be derived analytically.

Based on the analysis in this chapter we can see that all three network models represent some aspects of the real-world social networks, but their properties and characteristics are vastly different from each other for any given size or density. For a comprehensive, real-world alike social network simulation, all these network model's characteristics should ideally be considered and combined within the simulation process.

Chapter 5

Newton's Law of Universal Gravitation in Link Prediction

In this chapter, we develop a novel, nature-inspired link prediction approach, which can combine both the local and global characteristic of a network. This advanced link prediction approach can also combine both the popularity of nodes and similarities between them to predict interactions between nodes, i.e. links. The developed link prediction approaches are evaluated on seven real-world datasets.

To predict the formation of new links, we consider measures which originate from network science and use them in the place of mass and distance within the formalism of Newton's Gravitational Law. The attraction force calculated in this way is treated as a proxy for the likelihood of link formation. In particular, we use three different measures of vertex centrality as mass, and 12 dissimilarity measures including shortest path and inverse Katz score in place of distance, leading to over 50 combinations that we evaluate empirically. The combination through gravitational law allows us to couple popularity with similarity, two important characteristics for link prediction in social networks. Performance of our predictors is evaluated using Area Under the Precision-Recall Curve (AUC) for seven different real-world network datasets. The experiments demonstrate that this approach tends to outperform the setting in which the vertex similarity measures like Katz are used on their own. This approach also gives us the opportunity to combine network's global and local properties for predicting future or missing links. Our study shows that use of the physical law which combines node importance with measures quantifying how distant the nodes are, is a promising research direction in social link prediction.

Our approach to link prediction in social networks is inspired by Newton's law of universal gravitation, which states that the force exerted between two masses is proportional to the product of

those masses, and inversely proportional to the squared distance between their centres (Newton 1987):

$$F = G\frac{m_1 \cdot m_2}{r^2}, \tag{5.1}$$

where F is the force between masses m_1 and m_2, G is the gravitational constant, and r is the distance between m_1 and m_2. Newton derived this equation by empirical observation and inductive reasoning (Crombie 1957), which is an approach that we have also taken.

As discussed earlier, we use importance or popularity of a node to express mass. We argue that different centrality measures are direct measurements of how important, central or popular a node is in a given network. Dissimilarity or distance is measured via path distances (e.g. shortest path) or inverse of various similarity measures (e.g. AdamicAdar, Jaccard's Coefficient). It is also possible to define distance in terms of dissimilarity in non-topological node properties, like age, physical distance etc. A weighted sum of these factors can be incorporated into the distance, allowing to naturally exploit non-topological information. This however is not the focus of our study.

The above analogy leads to the following formula for calculating the score of two nodes forming a link in the future:

$$\boldsymbol{Score(v_i, v_j) = Score(v_j, v_i)} \propto \frac{P(v_i) \cdot P(v_j)}{D(v_i, v_j)^2}, \tag{5.2}$$

where P denotes popularity or centrality and D is dissimilarity or distance for an undirected graph.

The formula in Equation 5.2 can be interpreted as a modification of the Preferential Attachment method (i.e. product of centralities), where the resultant scores are weighted by the inverse of squared distance between the two nodes in question. This arguably gives our method more expressive power by taking proximity into account, which as demonstrated in Table 5.1 and Section 5.3.1.12, not only makes sense intuitively, but also tends to produce more accurate predictions in practice.

As for the gravitational constant G, without loss of generality we have assumed $G = 1$, since in order to make a prediction, a ranked list of scores is required with their absolute values being irrelevant. Note, that if the score was to be interpreted as probability, for a particular network this could be achieved by setting the value of G as follows:

$$G = \frac{\min \forall_{(i,j),i\neq j} D(v_i, v_j)^2}{\max \forall_i P(v_i) \cdot \max \forall_{j\neq i} P(v_j)}, \tag{5.3}$$

where the numerator is equal to 1, which reflects the obvious existence of a direct link between at least one pair of nodes.

Figure 5.1 depicts a simple social network to illustrate the intuition behind the proposed method using degree centrality and shortest path for Equation 5.2 (Wahid-Ul-Ashraf, Budka and Musial-Gabrys 2018). Application of Equation 5.2 produces a ranked list of scores for all pairs of nodes which are not already directly connected, given in Table 5.1. The following observations can be made here:

- If two people have many friends but are themselves distant (e.g. nodes 1 and 10) then the distance will make it improbable for these nodes to connect. On the other hand, someone who joins a network and have fewer friends might connect with the nearest friend who has many connections. This is why the score for nodes 10 and 16 is the same as for nodes 1 and 10, although node 16 just have one link in the network, whereas 1 and 10 have the highest degrees. This phenomenon is intuitive in social networks as people who have just made their first connection (we assume that node 16 has joined the network relatively recently), tend to connect with people at shorter distance, who are popular (i.e. have a high degree).
- If two nodes have many connections and are close to each other yet not connected, it is very likely they have many mutual friends and will eventually connect. This is the case for nodes 2 and 10, which have the highest score of connecting according to the proposed algorithm.

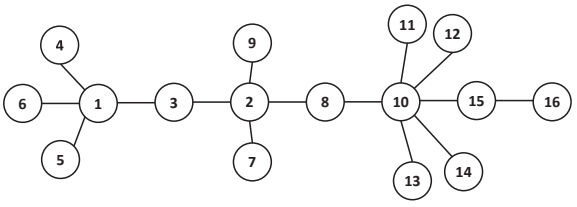


FIGURE 5.1: Example for link prediction with a simple graph

Rank	1	2	3		4	5		6		7		8								
Score	6.0	4.0	1.5	1.5	1.33	1.0	1.0	0.89	0.89	0.67	0.67	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
V_i	2	1	1	10	3	3	8	1	2	7	9	3	3	3	3	3	7	8	8	8
V_j	10	2	10	16	10	8	15	8	15	10	10	4	5	6	7	9	8	9	11	12

TABLE 5.1: Prediction value for a simple graph in Figure 5.1

5.1 Experimental Setup

In order to empirically evaluate our approach proposed in Equation 5.2 we use three different centrality measures along with 12 similarity measures.

Centrality measures which used as a measurement of popularity are:

1. **Degree Centrality (DC),** which is the degree of a vertex in a network.
2. **Closeness Centrality (CC),** High closeness centrality of a vertex means the vertex has better access to information or more direct influence on other vertices.
3. **Betweenness Centrality (BC),** this centrality gives a score to a vertex v_i based on how many paths connecting any two vertices in the network go through that vertex v_i. If the number of those paths is high, then vertex v_i will have high betweenness centrality.

All these three centralitites are discussed in details in Chapter 2 Section 2.1.1.

12 link prediction methods described in Chapter 2 Section 2.3.4 give different similarity scores that denote how likely two nodes will be connected in the future. For Equation 5.2, inverse of these similarity measures are used as dissimilarity scores, D. Table 5.2 shows different parameters for the 12 similarity measures. In Table 5.2, only Katz has different sets of parameters for different datasets. This difference is because β parameter in Katz needs to be smaller than the reciprocal of the highest eigenvalue of M (Landherr et al. 2010). -

Methods	Parameter	collegeMsg	contact	hep-th	hep-ph	hypertext	infectiousContact	MITContact
CN	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
JC	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
AA	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
PA	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Katz	β	0.001, 0.0005, 0.00005	0.001, 0.0005, 0.00005	0.001, 0.0005, 0.00005	0.001, 0.0005, 0.00005	0.001, 0.0005, 0.00005	0.005, 0.0005, 0.00005	0.1, 0.05, 0.005
RPR	α	0.15, 0.25	0.15, 0.25	0.15 ,0.25	0.15, 0.25	0.15, 0.25	0.15, 0.25	0.15,0.25
ACT	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
ACTN	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
PsInLap	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
LPI	ϵ	0.01, 0.02	0.01, 0.02	0.01, 0.02	0.01, 0.02	0.01, 0.02	0.01, 0.02	0.01, 0.02
LGI	θ	0.5, 0.7	0.5, 0.7	0.5, 0.7	0.5, 0.7	0.5, 0.7	0.5, 0.7	0.5, 0.7
MFI	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TABLE 5.2: Similarity measures with parameters, Common Neighbours (CN), AdamicAdar (AA), Preferential Attachment (PA), Rooted PageRank (RPR), Average Commute Time (ACT), Average Commute Time Normalised (ACTN), Pseudoinverse of the Laplacian matrix (PsInLap), Local Path Index (LPI), Leicht-Holme-Newman Global Index (LGI), and Matrix Forest Index (MFI)

In Table 5.2, the given parameters are selected based on the study conducted by Kleinberg (1999). In this work we are comparing different approaches, thus extensive parameter search is not performed. For all the approaches, the set of parameters are kept the same.

In our method we use the inverse of these scores to denote the dissimilarity/distance[1], plugging them into the formula of Equation 5.2.

5.1.1 Datasets

For the experimental comparative evaluation of the proposed method we selected seven datasets from various domains and of different sizes, frequently used in the literature, all representing undirected graphs and arrival of new nodes for the future snapshots are not considered:

1. **hep-th:** Collaboration graph of authors of scientific papers from High Energy Physics – Theory (hep-th) Section, where edges between two nodes represent a common publication. This dataset is acquired from the KONECT database (Kunegis 2017*b*, 2013*b*; Leskovec et al. 2007) and has been used in the experiment of Liben-Nowell, which is a very important research work in the area of link prediction (Liben-Nowell and Kleinberg 2007).

2. **hep-ph:** Collaboration graph of authors of scientific papers from High Energy Physics – Phenomenology (hep-ph) Section, where edges between two nodes represent a common publication. This dataset is acquired from the KONECT database (Kunegis 2017*a*; Leskovec et al. 2005).

3. **contact:** Dataset representing a network where edges are human contacts using small portable wireless devices distributed among different groups of people (Chaintreau et al. 2007; Kunegis 2013*a*).

4. **hypertext:** Face-to-face contacts of ACM Hypertext 2009 conference attendees, where edges represent interactions between people lasting at least 20 seconds (Isella et al. 2011; Kunegis 2017*c*).

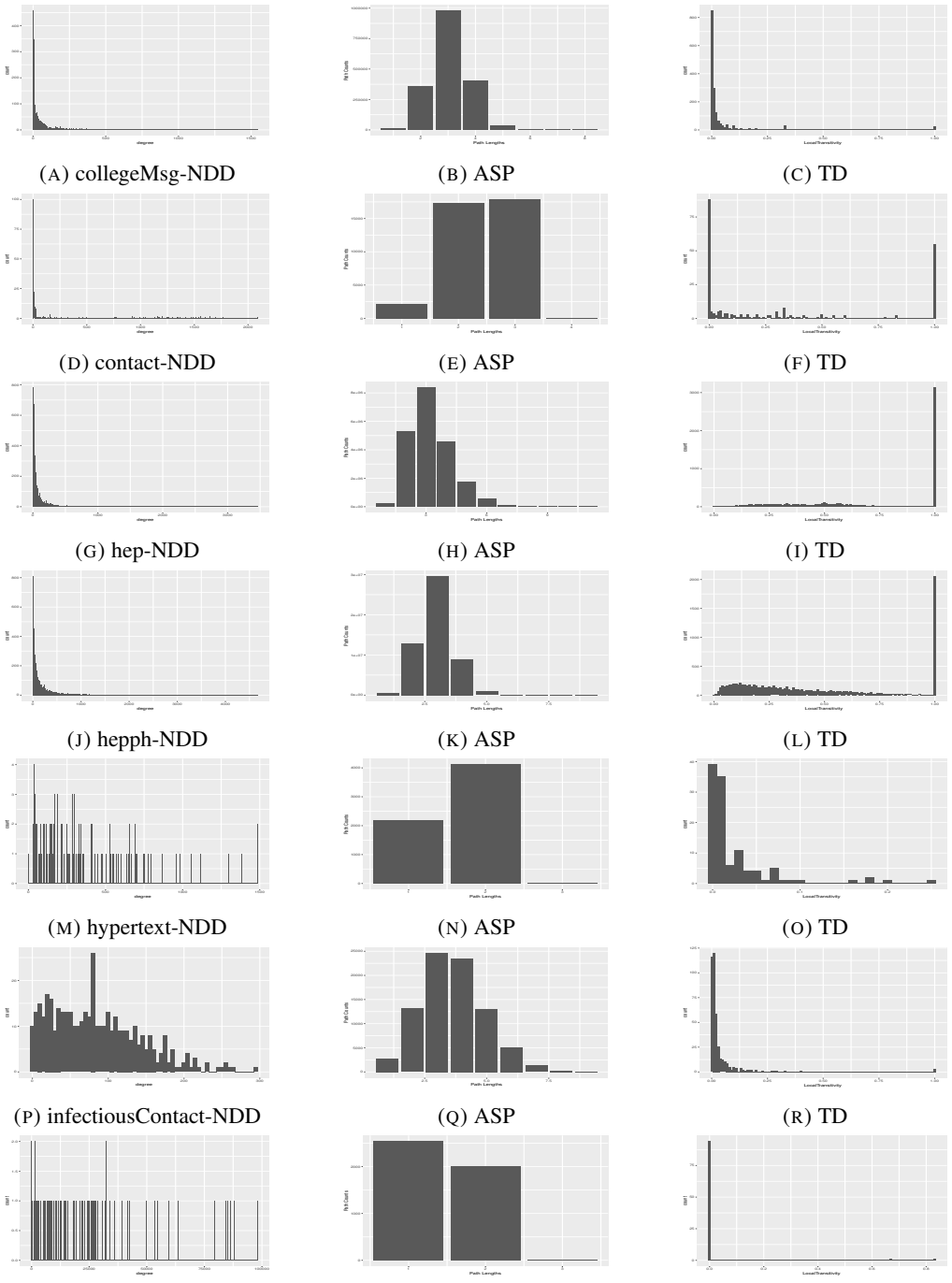
5. **collegeMsg:** Private messages sent via an online social network at the University of California, Irvine for over 193 days (Panzarasa et al. 2009).

6. **infectiousContact:** This dataset represents network of the face-to-face interactions of people during an exhibition INFECTIOUS: STAY AWAY in 2009 at the Science Gallery in Dublin. Each node is a person and edges between two nodes represent face-to-face contacts that lasted at least for 20 seconds. This network contains data about the interactions gathered on the day of the exhibition when highest number of contacts took place. This dataset is also acquired from KONECT database (Kunegis 2017*d*; Isella et al. 2011)

[1]We are considering dissimilarity as distance, noting that in some cases the symmetry and triangle inequality may not hold. For an unweighted and undirected graph $Score(v_i, v_j) = Score(v_i, v_j)$ (symmetry) but other than shortest path, triangle inequality may or may not hold for every dissimilarity score.

7. **MITContact:** This dataset is based on human contact and it is a part of Reality Mining experiment preformed in 2004. In this network, vertices represent physical contact between a group of students from Massachusetts Institute of Technology (MIT) (Kunegis 2017*e*; Eagle and Pentland 2006). This dataset is also acquired from KONECT. Data has been collected over a period of nine months.

As it can be seen from Figure 5.2 and Table 5.3, the selected datasets differ greatly in size and most of them represent typical social networks with power law node degree distribution, normal distribution of shortest path and small mean shortest path length as well as high global clustering coefficient. There are of course some exceptions to this profile, e.g. *collegeMsg* has very low global clustering coefficient, making the network more similar to random rather than social network. For a fully connected graph the highest density of a network is one. However, networks with multiple edges, density can be higher than one, as multiple links between two vertices is possible. We can see this higher than one density for, *hypertext* and *MITContact* contact datasets. The density is higher than one for both the datasets and both of these networks have multiple edges. However, in the training portion (i.e. the part of the data which is used for making prediction. Discussed in more details in the next Section 5.1.2) of those two networks we still have many nodes where no edges exist. In Section 5.3 we make predictions for these missing edges or links.



(A) collegeMsg-NDD (B) ASP (C) TD

(D) contact-NDD (E) ASP (F) TD

(G) hep-NDD (H) ASP (I) TD

(J) hepph-NDD (K) ASP (L) TD

(M) hypertext-NDD (N) ASP (O) TD

(P) infectiousContact-NDD (Q) ASP (R) TD

(S) MITContact-NDD (T) ASP (U) TD

FIGURE 5.2: Network properties (distribution). NDD: node degree distribution, ASP: average shortest path, TD: local transitivity (clustering coefficient) distribution

dataset	no. vertices	no. edges	density	node degree dist.	avg. shortest path dist.	avg. shortest path	transitivity dist.	global clustering coeff.
collegeMsg	1899	59835	0.033	power law	normal	3.055	power law	0.057
contact	274	28244	0.755	power law	normal	2.424	power law	0.566
hep-th	6776	290484	0.013	power law	normal	3.224	normal	0.333
hep-ph	10324	955423	0.018	power law	normal	2.946	normal	0.351
hypertext	113	20818	3.290	power law	normal	1.656	power law	0.495
infectiousContact	410	17298	0.206	power law	normal	3.631	power law	0.436
MITContact	96	1086405	238.247	power law	normal	1.445	power law	0.725

TABLE 5.3: Basic statistics of the datasets selected for the experiment

5.1.2 Data Partition

All networks considered in this study are with timestamps that indicate when a given relationship was created. This allows us to test prediction results against actual links that appeared in the future. We have divided each of the datasets into two parts based on the timestamps available. A similar setup has been used by Liben-Nowell and Kleinberg (2007) for benchmarking several link prediction methods, and in particular:

1. The **hep-th** dataset has been divided into two parts. Part one consisted of publications from years 1992-1994 and part two consisted of publications from years 1995-1997. Part one is where the link prediction is performed and part two is used as a ground truth in order to evaluate the method.

2. The **hep-ph** is also divided into two parts, part one containing publications between year 1994 and 1996, and part two with publications between year 1997 and year 1999. Similar to the previous dataset, part one is where the link prediction is performed and part two is used as ground truth.

3. Datasets **contact, hypertext, collegeMsg, infectiousContact, and MITContact** have also been divided into two parts with respect to time. However, the timespans within each part are not equal. Each part contains approximately[2] equal number of edges.

5.2 Experimental Setup

In our experiment, we apply each of the link prediction algorithms (12 existing and the proposed ones) on the first snapshot of the network. We then evaluate the prediction of new links

[2]For *collegeMsg* and *MITContact* datasets have an odd number of edges, thus there there is one less than half number of edges in the first part of the datasets

from each of the link prediction methods against the second snapshot of the network. Arrival of new nodes in the networks are not considered in this experiment. We evaluate the performance of each of the predictors on each of the datasets based on the area under the Precision Recall Curve. Additionally we also measure the area under the curve of Receiver Operating Characteristic curve, given in Appendix A, Section A.4. We discuss why Precision Recall Curve is a superior choice in next section. In order to have an effective comparison of our approaches to link prediction against the existing ones, we have developed an extensive comparison technique which is described in Section 5.3.1.

5.3 Results

We are using Area Under the Precision-Recall Curve (AUC) to evaluate performance of each of the predictors. In total, we have calculated AUC for combinations of 74 different predictors and seven datasets. These 74 predictors involve (1) similarity measures from Chapter 2 Section 2.3.4, (2) combinations of these similarity measures with centrality measures from Chapter 2 Section 2.1.1 and, (3) combinations of shortest path with the centrality measures mentioned above.

The summary of results is given in Figures 5.3 and 5.4. In Figure 5.3 AUC values are sorted in descending order. Each of the bars is the sum of all the AUCs over all datasets for a given approach (i.e. a given predictor from the three categories listed above) to link prediction. For example, the bottom-most bar in Figure 5.3 represents AUCs for combination of closeness centrality and MFI using Equation 5.2. This predictor has the best overall performance if we sum AUCs for this method for all seven datasets. On the other hand, Figure 5.4 depicts individual performance for all the predictors for individual datasets. From Figures 5.3 and 5.4 we see that for some of the datasets, overall AUCs are very small. However, later in Sections 5.3.1.1–5.3.1.12 we have compared each of methods with a random predictor. The results show that overall low values of AUC for a certain dataset do not necessarily mean that particular dataset has low predictability. This is because all networks are different in size. For a larger (in terms of vertices) or less dense network, the total number of predictions made is higher. This is because, we make predictions for a total of $\frac{|V|(|V|-1)}{2} - |E|$ links. As a network gets denser, the term $|E|$ also becomes larger. As a result, the total number of predictions gets lower. Because our AUC is from Precision-Recall curve, when we make predictions for a higher number of links there is a higher chance of having more false positives. This is because of the number of new links that a network forms may not increase at the same rate as the growth of the network. The Precision is calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{5.4}$$

From Equation 5.4, where TP is true positives and FP is false positives, we could see that, if we have larger values for false positives (FP) the value for Precision gets lower.

In Figure 5.3 it can be seen that the first three overall best performing methods are the ones with our Newton's gravitational law inspired combination approach. On the other hand, ACTN used as a standalone method makes worst prediction among all the 74 predictors. Interestingly, when ACTN is combined with DC using Equation 5.2 its performance jumps to rank 32 from 74. In addition, this combination of ACTN with DC performs better than DC with shortest path. This improvement reveals that the increment in predictability is not because of DC, or ACTN's independent predictability but because of the combination that we use. More on this improvement due to the combination is discussed later in Section 5.3.1. We also see a similar improvement with CN, where CN combined with CC ranks as the fourth overall best method. Improvements due to the combination approach we take could also be seen in several other combinations of predictors with MFI, Katz, RPR, etc. These improvements suggest that our combination approach has a great potential in the area of link prediction.

We further analyse the results in two ways: (i) we group methods based on the similarity measure used and then we compare the results within the groups (Sections 5.3.1.1–5.3.1.12) and (ii) we discuss the results in the context of each dataset separately and try to interpret why certain methods work on some datasets and not on others (Section 5.3.3).

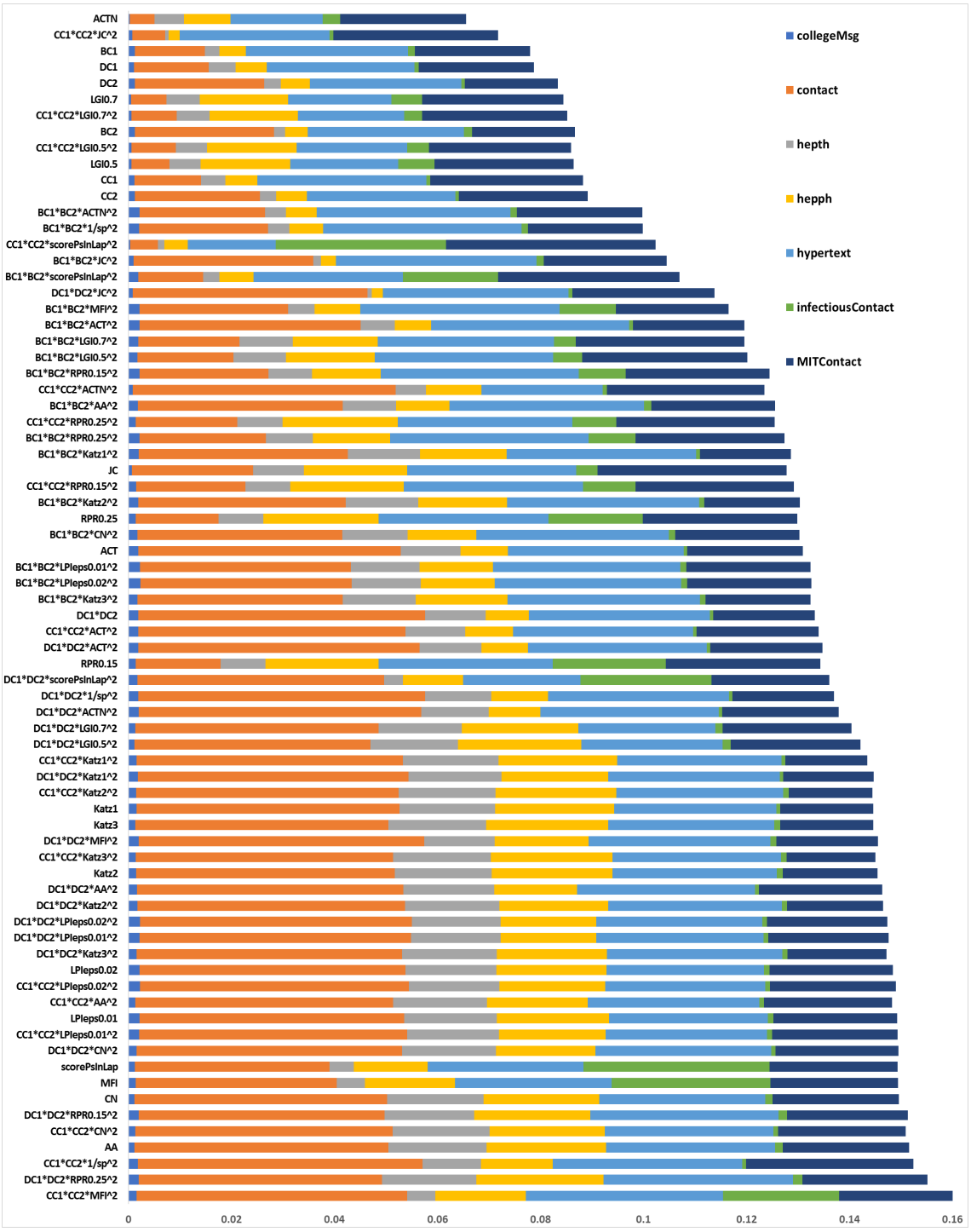


FIGURE 5.3: Combined Average (AUC)

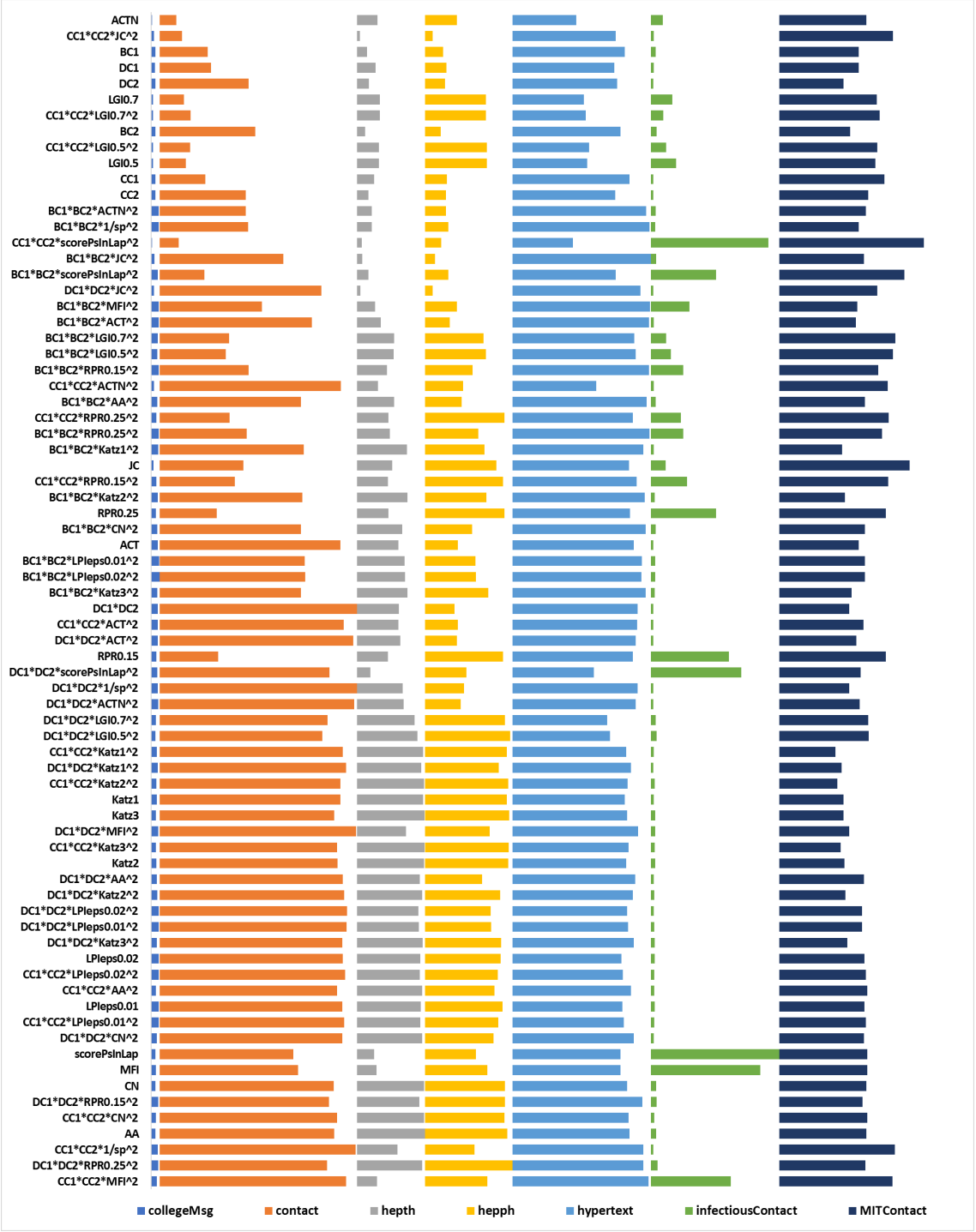


FIGURE 5.4: Individual Method's Performance (AUC)

5.3.1 Overall performance using AUC

For any pair of vertices v_i and v_j, we can consider all the similarity methods from Chapter 2 Section 2.3.4 as a set of predictors $S = \{Katz1, Katz2, Katz3, AA, .., CN\}$. Similarly, all centrality measures from Chapter 2 Section 2.1.1, could be expressed as a set $P = \{DC, BC, CC\}$

where $DC = DC_i \cdot DC_j, BC = BC_i \cdot BC_j, CC = CC_i \cdot CC_j$. As we use dissimilarity or distance by taking the inverse of each similarity measure for Equation 5.2, our proposed combination approach could be expressed as:

$$W = \{P \times S\}, \tag{5.5}$$

where each of the elements $w \in W$ is a particular predictor which gives prediction for any two vertices v_i and v_j. For any predictor $w \in W$, it is a combination of one particular similarity measure $s \in S$ and one particular centrality measure $p \in P$. For such a combined predictor w, with similarity measure s and centrality p we check if:

$$\big(AUC(w) > AUC(s)\big) \wedge \Big(AUC(w) > AUC(p/d^2)\Big) \tag{5.6}$$

Here in Inequality 5.6, d is the shortest path. If for a particular combination approach w, Inequality 5.6 holds, those AUC values are highlighted using dark grey boxes in Tables 5.4–5.14. The dark grey boxes indicate if a particular well-established similarity measure $s \in S$, when combined with centralities using Equation 5.2 performs better than the similarity measure on its own. The improvement could also be due the product of centralities in p which we have in the combination method w. In fact, product of degree centrality of v_i and v_j is a similarity measure, Preferential Attachment (PA) from Chapter 2 Section 2.3.4. Similarly, it is possible to use a product of another centrality measure as a standalone predictor. Due to this we also check if AUC of a particular combination $w \in W$ is greater than the AUC of $\frac{p}{d^2}$. The denominator of d^2 in most cases increases the performance with respect to only considering the centrality measures (see Section 5.3.1.12) (Wahid-Ul-Ashraf, Budka and Musial-Gabrys 2018), where dividing by shortest path squared mostly improves (where it does not, the difference is very small) the score as compared with the standalone product of centralities. The analysis in Table 5.3.1.12 confirms this improvement. As a result, if Inequality 5.6 holds, the inverse of similarity measure improves the predictor when used for Equation 5.2. It also shows that the improvement is due to the combination approach we take using Equation 5.2 but not due to the independent predictability of the similarity measure or product of centralities divided by squared shortest path. In Sections 5.3.1.1–5.3.1.11, when the performance of a combination method is said to be better or improved, it entails that Inequality 5.6 holds.

In addition to validating Inequality 5.6, for each of the datasets, we also identify if AUC of a predictor is smaller than the AUC of a random predictor. A random predictor predict occurrence of a link randomly without considering any information of the graph. For each predictor, AUC is calculated using R package called PRROC (Keilwagen et al. 2014; Grau et al. 2015). The AUC of a random predictor is also generated from the same package. For each dataset AUC of a random predictor is calculated from an ensemble of 1000 random predictors (Keilwagen

et al. 2014). In Tables 5.4–5.15, values of AUC which are not higher than the AUC of a random predictor for a particular dataset, have been highlighted as light grey.

5.3.1.1 Combinations with Katz

	college Msg	rnk	contact	rnk	hep-th	rnk	hep-ph	rnk	hyper text	rnk	infectious Contact	rnk	MIT Contact	rnk
$Katz1$	0.01132	39	0.35702	22	0.13032	8	0.16138	8	0.22064	52	0.00532	60	0.12643	68
$Katz2$	0.01061	43	0.35138	25	0.13167	5	0.16412	5	0.22377	49	0.00815	38	0.12842	66
$Katz3$	0.00969	50	0.34395	30	0.13258	2	0.16578	3	0.22576	48	0.00826	37	0.1265	67
$DC1 * DC2 * Katz1^2$	0.01286	29	0.36789	9	0.12653	15	0.14505	22	0.23282	37	0.00499	66	0.12262	71
$DC1 * DC2 * Katz2^2$	0.01229	35	0.36401	13	0.12775	12	0.1479	21	0.23663	35	0.00673	48	0.13064	64
$DC1 * DC2 * Katz3^2$	0.01121	40	0.36078	18	0.12869	10	0.14986	19	0.23854	32	0.00703	46	0.13417	63
$BC1 * BC2 * Katz1^2$	0.01405	15	0.28444	41	0.09813	28	0.11762	40	0.25738	15	0.00534	59	0.12364	70
$BC1 * BC2 * Katz2^2$	0.01351	25	0.28187	42	0.09871	27	0.1207	36	0.26073	12	0.00743	44	0.12947	65
$BC1 * BC2 * Katz3^2$	0.01242	32	0.27896	43	0.09922	26	0.12444	31	0.2619	10	0.00766	41	0.14257	58
$CC1 * CC2 * Katz1^2$	0.0114	37	0.36158	16	0.13031	9	0.16125	9	0.2234	50	0.00518	62	0.11112	74
$CC1 * CC2 * Katz2^2$	0.01069	42	0.3567	24	0.13166	6	0.16398	6	0.22683	45	0.00753	43	0.11415	73
$CC1 * CC2 * Katz3^2$	0.00974	49	0.35033	26	0.13257	3	0.16557	4	0.22879	43	0.00767	40	0.12089	72

TABLE 5.4: AUC for Katz with different centralities. Highlights in dark grey represent that Inequality 5.6 holds (no such case exist in this table), and light grey represents AUC values lower than the AUC of a random predictor.

Katz similarity performs poorly for *infectiousContact* and *MITContact* datasets – we can see from Table 5.4, most of the AUC values are lower than random predictors. Also, we do not see any combination of Katz performing better than both the standalone Katz and the product of centralities divided by distance (Table 5.15), which means the combination does not satisfy Inequality 5.6. As a result, we do not have any empirical evidence suggesting that using inverse of Katz as distance in our proposed approach of Equation 5.2, could entail improved performance.

5.3.1.2 Combinations with AdamicAdar (AA)

	college Msg	rnk	contact	rnk	hep-th	rnk	hep-ph	rnk	hyper text	rnk	infectious Contact	rnk	MIT Contact	rnk
AA	0.00845	61	0.34479	29	0.13344	1	0.16241	7	0.22985	40	0.01069	26	0.17158	29
$DC1 * DC2 * AA^2$	0.01183	36	0.36166	15	0.12377	19	0.11257	42	0.24188	28	0.00541	57	0.16746	42
$BC1 * BC2 * AA^2$	0.01263	30	0.2785	45	0.07275	40	0.07279	53	0.26434	8	0.00978	30	0.16848	38
$CC1 * CC2 * AA^2$	0.00947	52	0.35018	27	0.12764	13	0.1371	26	0.23314	36	0.00658	50	0.17386	26

TABLE 5.5: AUC for AdamicAdar (AA) with different centralities. Highlights in dark grey represent that Inequality 5.6 holds (no such case exist in this table), and light grey represents AUC values lower than the AUC of a random predictor

In Table 5.5 we also see similar pattern to Katz that, inverse of AdamicAdar (AA) similarity measure as a measurement of distance for Equation 5.2 does not entail improved[3] performance (i.e. it does not satisfy Inequality 5.6).

5.3.1.3 Combinations with Common Neighbours (CN)

	college Msg	rnk	contact	rnk	hep-th	rnk	hep-ph	rnk	hyper text	rnk	infectious Contact	rnk	MIT Contact	rnk
CN	0.00825	62	0.3433	31	0.13139	7	0.1572	12	0.22578	47	0.00984	28	0.17138	30
$DC1 * DC2 * CN^2$	0.0114	38	0.36073	19	0.12757	14	0.13528	27	0.23876	30	0.00563	55	0.1673	43
$BC1 * BC2 * CN^2$	0.01236	34	0.27863	44	0.0884	34	0.09332	49	0.26171	11	0.00898	32	0.16842	39
$CC1 * CC2 * CN^2$	0.00965	51	0.34967	28	0.13173	4	0.15674	13	0.22885	42	0.0064	51	0.17366	28

TABLE 5.6: AUC for Common Neighbours (CN) with different centralities. Highlights in dark grey represent that Inequality 5.6 holds, and light grey represents AUC values lower than the AUC of a random predictor

We can see in Table 5.6, that combining inverse of Common Neighbour (CN) with centrality (as a measurement of popularity or mass for Equation 5.2) improves performance of link prediction for one dataset. This is expressed by the fact that one of the values of AUC satisfies Inequality 5.6. There is one such case which is highlighted using dark grey box in Table 5.6. This improvement is seen when the combination of CN is with closeness centrality for *hep-th* dataset. However, except for combination of CN with CC in the *hep-th* dataset, there is no other evidence that any other combination of CN satisfies Inequality 5.6.

5.3.1.4 Combinations with Jaccard's Coefficient (JC)

	college Msg	rnk	contact	rnk	hep-th	rnk	hep-ph	rnk	hyper text	rnk	infectious Contact	rnk	MIT Contact	rnk
JC	0.00476	68	0.16494	57	0.06883	43	0.14048	25	0.22959	41	0.02935	19	0.25703	2
$DC1 * DC2 * JC^2$	0.00615	65	0.31865	37	0.00574	73	0.01561	73	0.25224	19	0.00508	64	0.19332	18
$BC1 * BC2 * JC^2$	0.00721	64	0.24436	48	0.0103	71	0.02022	72	0.2725	1	0.01009	27	0.16706	44
$CC1 * CC2 * JC^2$	0.00541	67	0.04442	72	0.00489	74	0.0151	74	0.20351	61	0.00524	61	0.2237	7

TABLE 5.7: AUC for Jaccard's Coefficient (JC) with different centralities. Highlights in dark grey represent that Inequality 5.6 holds, and light grey represents AUC values lower than the AUC of a random predictor

In quite a few cases, as presented in Table 5.7, Jaccard's Coefficient (JC) combined with betweenness centrality gives improved performance (i.e. satisfies Inequality 5.6). These improvements are seen for *contact*, *hep-ph*, and *hypertext* datasets. In fact, for *hypertext* dataset, JC

[3]Throughout this section, whenever we say a combination approach performs better or has improved performance, we imply it satisfies Inequality 5.6. Please see Section 5.3.1 for more details.

combined with betweenness centrality entails the best result (i.e. AUC value ranked one). These improvements support that, JC combined with betweenness centrality using 5.2 is a better link prediction method than using JC alone. Also, there is one case where for *hypertext* dataset, JC performs better when combined with degree centrality. However, closeness centrality combined with Jaccard's Coefficient (JC) does not satisfy Inequality 5.6.

5.3.1.5 Combinations with Average Commute Time (ACT)

	college Msg	rnk	contact	rnk	hep-th	rnk	hep-ph	rnk	hyper text	rnk	infectious Contact	rnk	MIT Contact	rnk
ACT	0.0134	26	0.35688	23	0.08106	38	0.06478	56	0.23875	31	0.00481	68	0.157	51
$DC1 * DC2 * ACT^2$	0.01371	22	0.38183	6	0.08451	35	0.06308	58	0.24294	25	0.00468	71	0.15241	56
$BC1 * BC2 * ACT^2$	0.01508	8	0.30064	38	0.04642	50	0.0492	61	0.26911	5	0.00535	58	0.1515	57
$CC1 * CC2 * ACT^2$	0.01351	24	0.3632	14	0.08108	37	0.06486	55	0.24524	23	0.00466	73	0.16568	45

TABLE 5.8: AUC for Average Commute Time (ACT) with different centralities. Highlights in dark grey represent that Inequality 5.6 holds, and light grey represents AUC values lower than the AUC of a random predictor

In Table 5.8 there are several cases when ACT combined with any of the three centrality measures gives better performance than using ACT alone or only centralities divided by the squared shortest path. However, such improvements are mainly observed for the *collegeMsg* dataset. Other than the *collegeMsg* dataset, combination of ACT with closeness centrality gives better prediction for *hep-th*. From this analysis we can see that, ACT combined with closeness centrality has more predictive power in link prediction than ACT combined with degree or betweenness centrality. This is because the first combination, ACT with closeness centrality, performs better (i.e. satisfies Inequality 5.6) in two (*collegeMsg* and *hep-th*) datasets and the other best performing combination, ACT with closeness centrality performs better in only one (*hep-th*) dataset. However, the number of datasets for which the combination with ACT satisfies Inequality 5.6 is lower than what we have seen for JC, MFI, and RPR. Combination of JC performs better i.e. satisfies Inequality 5.6 in two datasets whereas JC, MFI, and RPR performs better in three, four, and five datasets respectively.

5.3.1.6 Combinations with Average Commute Time Normalised (ACTN)

	college Msg	rnk	contact	rnk	hep-th	rnk	hep-ph	rnk	hyper text	rnk	infectious Contact	rnk	MIT Contact	rnk
$ACTN$	0.00216	74	0.03339	74	0.03996	56	0.0634	57	0.12492	73	0.02379	21	0.17125	31
$DC1 * DC2 * ACTN^2$	0.01398	18	0.38394	5	0.09163	32	0.07025	54	0.24261	27	0.00475	69	0.15853	50
$BC1 * BC2 * ACTN^2$	0.01516	7	0.17047	55	0.02844	64	0.04175	66	0.26346	9	0.00896	33	0.17062	34
$CC1 * CC2 * ACTN^2$	0.00581	66	0.35745	21	0.04116	55	0.07523	52	0.16509	67	0.00583	54	0.21415	11

TABLE 5.9: AUC for Average Commute Time Normalised (ACTN) with different centralities. Highlights in dark grey represent that Inequality 5.6 holds, and light grey represents AUC values lower than the AUC of a random predictor

Table 5.9 shows two cases of ACTN, where the predictability is improved when combined with degree centrality for *collegeMsg* and *hep-th* datasets. There is also one similar improvement with betweenness centrality for the *collegeMsg* dataset. However, there is no combination with closeness centrality which satisfies Inequality 5.6. Based on the number of datasets where combination with ACTN perform well, we could argue there is weak evidence that the two different combinations of ACTN with degree and closeness centrality may have good potential for predicting future links.

5.3.1.7 Combinations with Rooted PageRank (RPR)

	college Msg	rnk	contact	rnk	hep-th	rnk	hep-ph	rnk	hyper text	rnk	infectious Contact	rnk	MIT Contact	rnk
$RPR0.15$	0.01025	45	0.11534	62	0.06051	48	0.15376	17	0.23686	34	0.15386	6	0.20978	13
$RPR0.25$	0.00991	47	0.11244	63	0.06116	46	0.15653	15	0.23106	38	0.12798	8	0.21024	12
$DC1 * DC2 * RPR0.15^2$	0.01403	17	0.33405	33	0.12221*	21	0.15744	11	0.25588	16	0.01163	23	0.16394	46
$DC1 * DC2 * RPR0.25^2$	0.01404	16	0.33078	35	0.12806	11	0.17288	1	0.2575	14	0.01265	22	0.16996	35
$BC1 * BC2 * RPR0.15^2$	0.01499	11	0.17568	51	0.0588	49	0.09366	48	0.26893	6	0.06414	11	0.19521	17
$BC1 * BC2 * RPR0.25^2$	0.01504	10	0.17169	54	0.06406	44	0.10471	43	0.26978	4	0.06413	12	0.20235	15
$CC1 * CC2 * RPR0.15^2$	0.01058	44	0.1488	58	0.0607	47	0.15398	16	0.24394	24	0.07155	10	0.21519	10
$CC1 * CC2 * RPR0.25^2$	0.01018	46	0.138	59	0.06137	45	0.15673	14	0.23727	33	0.05944	13	0.21563	9

TABLE 5.10: AUC for Rooted PageRank (RPR) with different centralities. Highlights in dark grey represent that Inequality 5.6 holds, and light grey represents AUC values lower than the AUC of a random predictor

Inverse of Rooted PageRank (RPR) is one of the best measures for distance (according to Equation 5.2) from Chapter 2 Section 2.3.4. Table 5.10 shows that for *hep-th*, *collegeMsg*, *hypertext* and, *hep-ph* datasets, when RPR is combined with degree centrality, the combination outperforms individual performance of RPR or degree centrality divided by shortest path (i.e. satisfies Inequality 5.6). Also, for *collegeMsg*, *hep-th* and, *Contact* datasets similar improvement is observed when RPR is combined with betweenness centrality. Only in one case (with two different

values for α parameter of RPR) we see that combination of RPR with closeness centrality satisfies Inequality 5.6. From this analysis it is apparent that, RPR combined with degree centrality could be a better choice for link prediction than only using RPR.

5.3.1.8 Combinations with Pseudoinverse of the Laplacian matrix (PsInLap)

	college Msg	rnk	contact	rnk	hep-th	rnk	hep-ph	rnk	hyper text	rnk	infectious Contact	rnk	MIT Contact	rnk
$PsInLap$	0.00909	54	0.2641	47	0.03336	61	0.10005	45	0.21214	59	0.25286	1	0.17385	27
$DC1 * DC2 * PsInLap^2$	0.01239	33	0.33506	32	0.02588	65	0.08148	50	0.15964	68	0.17834	4	0.16009	49
$BC1 * BC2 * PsInLap^2$	0.01374	21	0.08809	67	0.02189	68	0.04634	62	0.20339	62	0.1288	7	0.24667	3
$CC1 * CC2 * PsInLap^2$	0.00245	73	0.03747	73	0.00873	72	0.03238	70	0.11912	74	0.23169	2	0.28475	1

TABLE 5.11: AUC for Pseudoinverse of the Laplacian matrix (PsInLap) with different centralities. Highlights in dark grey represent that Inequality 5.6 holds, and light grey represents AUC values lower than the AUC of a random predictor

In Table 5.11, there are two combinations (with betweenness centrality and closeness centrality) with Pseudoinverse of the Laplacian matrix (PsInLap) which perform better than PsInLap or product of these centralities divided by shortest path. Because these improvements are only seen for one dataset, we do not have strong evidence to support the use of the combination of PsInLap using Equation 5.2 for link prediction.

5.3.1.9 Combinations with Local Path Index (LPI)

	college Msg	rnk	contact	rnk	hep-th	rnk	hep-ph	rnk	hyper text	rnk	infectious Contact	rnk	MIT Contact	rnk
$LPIeps0.01$	0.01495	12	0.36019	20	0.12541	16	0.15286	18	0.21593	55	0.00762	42	0.16774	40
$LPIeps0.02$	0.01547	5	0.3609	17	0.12387	18	0.14961	20	0.21409	56	0.0073	45	0.16773	41
$DC1 * DC2 * LPIeps0.01^2$	0.01506	9	0.36898	8	0.12182	22	0.1301	28	0.22758	44	0.00627	52	0.16357	47
$DC1 * DC2 * LPIeps0.02^2$	0.01557	3	0.36979	7	0.12083	23	0.12967	29	0.22596	46	0.00603	53	0.16355	48
$BC1 * BC2 * LPIeps0.01^2$	0.0161	2	0.28604	40	0.09365	31	0.09973	46	0.25459	17	0.00839	36	0.16875	37
$BC1 * BC2 * LPIeps0.02^2$	0.01663	1	0.28689	39	0.09398	30	0.10043	44	0.25346	18	0.00796	39	0.16882	36
$CC1 * CC2 * LPIeps0.01^2$	0.01491	14	0.36414	12	0.12466	17	0.14473	23	0.21932	53	0.007	47	0.17081	33
$CC1 * CC2 * LPIeps0.02^2$	0.01556	4	0.36552	11	0.1234	20	0.14378	24	0.21743	54	0.00665	49	0.17082	32

TABLE 5.12: AUC for Local Path Index (LPI) with different centralities. Highlights in dark grey represent that Inequality 5.6 holds, and light grey represents AUC values lower than the AUC of a random predictor

From Table 5.12 we could see that Local Path Index (LPI) performs better when combined with betweenness centrality than on its own. This improvement can be observed for *collegeMsg* and *MITContact* datasets. In addition, for *collegeMsg* dataset, LPI improves when it is combined with degree centrality and closeness centrality. These improvements are not due to the product of centralities or LPI itself but due to the applied combination. This is because these combinations satisfy Inequality 5.6. However, there is more prevalent evidence that, LPI combined

with betweenness centrality is a better predictor of future links than LPI combined with degree centrality.

5.3.1.10 Combinations with Leicht-Holme-Newman Global Index (LGI)

	college Msg	rnk	contact	rnk	hep-th	rnk	hep-ph	rnk	hyper text	rnk	infectious Contact	rnk	MIT Contact	rnk
$LGI0.5$	0.00418	70	0.05162	70	0.04235	54	0.12159	35	0.14685	70	0.04951	14	0.18933	21
$LGI0.7$	0.00385	72	0.04821	71	0.04477	52	0.12031	39	0.14028	72	0.04178	15	0.1923	20
$DC1 * DC2 * LGI0.5^2$	0.00851	59	0.32063	36	0.11859	24	0.16784	2	0.19186	65	0.01133	25	0.17625	22
$DC1 * DC2 * LGI0.7^2$	0.0094	53	0.33098	34	0.11305	25	0.15801	10	0.18625	66	0.00982	29	0.17523	24
$BC1 * BC2 * LGI0.5^2$	0.01244	31	0.13035	61	0.07136	42	0.12059	37	0.24265	26	0.03963	16	0.22435	6
$BC1 * BC2 * LGI0.7^2$	0.01333	27	0.13741	60	0.07271	41	0.11554	41	0.23936	29	0.02988	17	0.22917	4
$CC1 * CC2 * LGI0.5^2$	0.0043	69	0.06027	69	0.04242	53	0.12161	34	0.15047	69	0.02967	18	0.19328	19
$CC1 * CC2 * LGI0.7^2$	0.00398	71	0.06144	68	0.04486	51	0.12031	38	0.1441	71	0.02439	20	0.19758	16

TABLE 5.13: AUC for Leicht-Holme-Newman Global Index (LGI) with different centralities. Highlights in dark grey represent that Inequality 5.6 holds, and light grey represents AUC values lower than the AUC of a random predictor

In Table 5.13 we can see that Leicht-Holme-Newman Global Index (LGI) when combined with degree centrality always exhibits improved performance for *hep-th* and *hep-ph* datasets. These improvements might indicate that, this combination performs well for collaboration networks. Because *hep-th* and *hep-ph* both are the only collaboration networks we have. These improvements could suggest that for collaboration networks, combining LGI with degree centrality using Equation 5.2 could be a good approach for predicting future collaborations. However, this claim for collaboration network needs to be corroborated by evaluating this combination for more network datasets of collaboration networks. Performance for combination of LGI with betweenness centrality for the *hep-th* and *MITContact* datasets, and closeness centrality for *hep-ph* dataset, are also improved. Here, we have weak evidence of degree and betweenness centrality to perform better when combined with LGI, thus a better predictor than LGI itself.

5.3.1.11 Combinations with Matrix Forest Index (MFI)

	college Msg	rnk	contact	rnk	hep-th	rnk	hep-ph	rnk	hyper text	rnk	infectious Contact	rnk	MIT Contact	rnk
MFI	0.00978	48	0.27332	46	0.03846	58	0.12256	33	0.21244	57	0.21582	3	0.17394	25
$DC1 * DC2 * MFI^2$	0.01397	19	0.38795	3	0.09617	29	0.12733	30	0.24704	20	0.00846	35	0.13802	62
$BC1 * BC2 * MFI^2$	0.01535	6	0.20197	49	0.03556	60	0.06253	59	0.27084	2	0.07619	9	0.15345	55
$CC1 * CC2 * MFI^2$	0.01114	41	0.36762	10	0.03858	57	0.12314	32	0.26806	7	0.15773	5	0.22291	8

TABLE 5.14: AUC for Matrix Forest Index (MFI) with different centralities. Highlights in dark grey represent that Inequality 5.6 holds, and light grey represents AUC values lower than the AUC of a random predictor

Table 5.14 shows that Matrix Forest Index (MFI) when combined with degree centrality using Equation 5.2 outperforms the predictability of (1) MFI when used on its own and (2) product of degree centrality divided by shortest path. This can be observed for four out of seven datasets: *collegeMsg*, *hep-th*, *hep-ph*, and *hypertext*. Also, in two datasets, similar improvement is seen when combined with closeness (*hep-ph* and *hypertext*) and betweenness (*collegeMsg* and *hypertext*) centrality. We hence argue that MFI combined with degree centrality is a strong method for link prediction.

5.3.1.12 Combinations with Shortest Path

	college Msg	rnk	contact	rnk	hep-th	rnk	hep-ph	rnk	hyper text	rnk	infectious Contact	rnk	MIT Contact	rnk
$DC1$	0.00778	63	0.10156	64	0.03643	59	0.04261	65	0.20073	64	0.00543	56	0.15655	52
$DC2$	0.00893	56	0.17552	52	0.02298	66	0.03956	68	0.20564	60	0.00472	70	0.12636	69
$BC1$	0.00886	57	0.09526	65	0.01959	69	0.0358	69	0.22064	51	0.00947	31	0.15653	53
$BC2$	0.00907	55	0.18862	50	0.01516	70	0.03071	71	0.21233	58	0.01139	24	0.13938	59
$CC1$	0.00847	60	0.0902	66	0.03301	62	0.04356	64	0.23007	39	0.00507	65	0.20749	14
$CC2$	0.00865	58	0.16996	56	0.02245	67	0.04154	67	0.20218	63	0.00449	74	0.17528	23
$DC1 * DC2$	0.01377	20	0.38969	2	0.08237	36	0.05854	60	0.24605	21	0.00466	72	0.13807	61
$DC1 * DC2 * 1/sp^2$	0.0136	23	0.38976	1	0.08983	33	0.07696	51	0.24601	22	0.00483	67	0.13812	60
$BC1 * BC2 * 1/sp^2$	0.01492	13	0.17511	53	0.0288	63	0.04596	63	0.26983	3	0.0085	34	0.15626	54
$CC1 * CC2 * 1/sp^2$	0.01314	28	0.38662	4	0.07964	39	0.09746	47	0.25756	13	0.00511	63	0.22749	5

TABLE 5.15: AUC for Shortest path with different centralities. Highlights in dark grey represent that a combination method performs better than PA, and light grey represents AUC values lower than the AUC of a random predictor

From Table 5.15 we could see that for the case where we use the shortest path in combination with degree centrality, even with a slight variation of shortest path length (due to the small-world phenomena the range of shortest path tend to be small) gives better performance than only using the product of degree centrality i.e. the Preferential Attachment (PA) similarity measurement. These improvements are seen in five out of seven datasets. This finding is consistent with findings by Wahid-Ul-Ashraf, Budka and Musial-Gabrys (2018). Here we have compared degree centrality combined with the shortest path against PA because PA is the product of degree centrality. The baseline method here is PA instead of Inequality 5.6 as the combination of centrality and the shortest path itself served as baselines for other results of combination methods discussed so far. PA is a well-established link prediction method that we have discussed further in Chapter 2 Section 2.3.4 (Barabâsi et al. 2002). Other than the DC with the shortest path, BC and CC combined with the shortest path also perform better than PA. BC with the shortest path performs better in four datasets and CC with the shortest path performs better in three datasets (although it performs better than PA for the *infectiousContact* dataset the predictability is not better than a random predictor).

5.3.2 Best Methods

Methods which satisfy Inequality 5.6 are the only ones which we analyse here. The reason for this selection is discussed in Section 5.3.1. From the selected combination methods, we use three different evaluation techniques to calculate scores in Table 5.16. The 'Dataset variability score' is the number of datasets for which a combination approach satisfies Inequality 5.6. We also calculate a score based on ranks. In our analysis the lowest rank of a method is 74, as we have 74 methods in total including the standalone methods from Tables 5.4-5.15. We subtract 73 (so that the worst method with rank 74 will have a score 1) from the rank of a method in a dataset to get a score instead of rank. Afterwards, we sum the scores up to get the final score which is represented as 'Score (73-Rank)' in the table. This score not only tells us for how many datasets a method performs well but also that method's relative performance among all the other methods. Finally, we normalise 'Score (73-Rank)' by the number of datasets for which a method satisfies Inequality 5.6. This normalised version of the rank-based score is represented as 'Normalised Score (73-Rank)' and considers a combination method's rank based on the average performance on all datasets.

Method	college Msg	contact	hep-th	hep-ph	hypertext	infectious Contact	MIT Contact	Dataset Variablity Score	Score (73-Rank)	Normalised Score (73-Rank)
RPR0.25+DC	Y(16)		Y(11)	Y(1)	Y(14)			4***	254***	63.5
MFI+BC	Y(6)				Y(2)			2*	140	70*
MFI+DC	Y(19)		Y(29)	Y(30)	Y(20)			4***	198**	49.5
RPR0.15+DC	Y(17)			Y(11)	Y(16)			3**	178*	59.3
DC+SP		Y(1)	Y(33)	Y(51)			Y(60)	3**	151	50.3
LGI0.5+DC			Y(24)	Y(2)				2*	122	61
LGI0.7+DC			Y(25)	Y(10)				2*	113	56.5
LPIeps0.02+BC	Y(1)						Y(36)	2*	111	55.5
LPIeps0.01+BC	Y(2)						Y(37)	2*	109	54.5
MFI+CC				Y(32)	Y(7)			2*	109	54.5
LGI0.7+BC			Y(41)				Y(4)	2*	103	51.5
JC+BC		Y(48)		Y(72)	Y(1)			3**	101	33.67
LGI0.5+BC			Y(42)				Y(6)	2*	100	50
ACTN+DC	Y(18)		Y(32)					2*	98	49
RPR0.25+BC	Y(10)		Y(44)					2*	94	47
ACT+CC	Y(24)		Y(37)					2*	87	43.5
RPR0.15+BC	Y(11)	Y(51)						2*	86	43
PsInLap+CC							Y(1)	1	73	73***
PsInLap+BC							Y(3)	1	71	71**
LPIeps0.02+DC	Y(3)							1	71	71**
CN+CC			Y(4)					1	70	70*
LPIeps0.02+CC	Y(4)							1	70	70*
ACTN+BC	Y(7)							1	67	67
ACT+BC	Y(8)							1	66	66
LPIeps0.01+DC	Y(9)							1	65	65
RPR0.25+CC				Y(14)				1	60	60
RPR0.15+CC				Y(16)				1	58	58
JC+DC					Y(19)			1	55	55
ACT+DC	Y(22)							1	52	52
LGI0.5+CC				Y(34)				1	40	40
LGI0.7+CC				Y(38)				1	36	36

TABLE 5.16: Methods which satisfy Inequality 5.6. The dataset(s), in which a method satisfied Inequality 5.6 is marked as Y, and the rank of that method mentioned in the parenthesis, i.e. Y(rank). First best score is marked with ***, second best with** and third best with *. For all the scores, higher is better. The '+' operator entails a combination based on the Equation 5.2.

5.3.3 Results Analysis for each Dataset

Based on the methods we use and the combination of them we conclude that some datasets can be more predictable than others. By comparing AUC of the PR curves, it seems that *hep-th* and *collegeMsg* datasets are the most predictable, as only two methods perform worse than a random predictor. Overall the collaboration networks *hep-th* and *hep-ph* have good predictability. Only two methods for the *hep-th* and three methods for the *hep-ph* collaboration network

perform worse than a random predictor. On the other hand, *infectiousContact* dataset has the lowest predictability – there are 37 out of 74 (including combinations) methods whose performance is worse than a random predictor. The second worst dataset in terms of predictability is *MITContact* where 11 methods perform lower than a random predictor. For *hypertext* we have six methods performing worse than a random predictor. Overall, except *contact* dataset, where we have only three methods with AUC lower than a random predictor, all the networks representing human contact seem to have low predictability. We discuss below the results from the perspective of individual datasets and interpret those outcomes in the context of characteristics of each social network tested:

1. **collegeMsg:** Overall, performance of methods on *collegeMsg* does not appear to be very good when compared to the remaining datasets. However, when we compare with a random predictor, many of the predictors seem to perform better. The best performing methods for *collegeMsg* are those based on LPI in combination with BC. As LPI in its nature is similar to CN it is surprising that the highest rank for CN–based method for *collegeMsg* dataset is ranked 34. It means that consideration of friend-of-friend-of-friend (path of length three) in LPI rather than friend-of-a-friend (CN) makes a (positive) difference for prediction.

2. **contact:** For contact network the best performing methods are the ones based on DC and the top ranked is DC coupled with the shortest path. Also, DC on its own (rank two), DC+MFI (rank three), CC+shortest path (rank four), DC+ACTN (rank five) and DC+ACT (rank six) perform well. All these methods are path-based but they must be combined with information about node degree to achieve good performance, e.g. DC+ACTN has rank five and ACTN on its own is last in the ranking (rank 74). However, this improved performance when combined with DC might be due to the fact that Preferential Attachment (product of degrees) is the second best predictor. Thus, although dividing DC by ACTN still makes it a good predictor, its performance is worse than when only degree product is used.

3. **hep-th:** Although the best method for *hep-th* is AA, the best performing set of methods are those based on Katz and combined with CC. Katz2 and Katz3 also performed very well with ranks five and two respectively. Also, methods combing CC with Katz3, CN, and Katz2 were performing very well (rank three, four, and six respectively). However, standalone Katz performs better than in a combination. On the other hand, note that again, we need to have a proper combination of metrics because CC combined with JC gives the poorest performance. It shows that taking into account the greater network (Katz enables that) not only the immediate neighbourhood of a node (JC) may result in better performance. It is surprising that although AA is very similar to JC, their performance differs so much with AA being ranked one and JC - 43 (0.06 accuracy for JC and 0.13

for AA). The interpretation may be that AA gives importance to the degree of common neighbour and if common neighbour degree is lower then there is a bigger chance that he/she will introduce two of his/her neighbours to each other. JC on the other hand focuses only on overall number of common friends. This indicates that when developing new prediction methods, we should also focus on other factors and capacity of other nodes rather than just the nodes in question.

4. **hep-ph:** Overall, for *hep-ph* dataset methods based on Katz and Katz combined with CC and DC perform best. However, the top two results are those that combine DC with RPR and LGI. Methods based on JC combined with different centralities give the worst results. It seems that merging local information (DC) with knowledge about paths throughout the network and appropriately weighting them (Katz, RPR, LGI) gives the best results. Similarity RPR and LGI combined with degree centrality outperform DC, RPR or LGI used as a standalone predictor. Similarly, for this dataset, LGI performs better (compared with using it independently) when combined with betweenness and closeness centrality.

5. **hypertext:** For the *hypertext* dataset the best set of methods are those that use BC as the centrality measure which is the most overreaching centrality out of those we analysed. BC is present in 11 out of 13 top ranked methods for this dataset. This improvement could be explained by looking at Table 5.15. We can see that BC combined with shortest path is the third best predictor for this dataset. In addition, Table 5.7 shows that JC works well for a measurement of distance for *hypertext* dataset when JC is combined with BC, it has the best predictability.

6. **infectiousContact:** Most of the predictors perform poorly for the *infectiouseContact* dataset. This low predictability may be indicative of the dataset containing many random interactions between people. Each of the edges represents interaction between two people at the INFECTIOUS: STAY AWAY exhibition at the Science Gallery in Dublin, Ireland, from April 17th to July 17th, 2009 (Isella et al. 2011). This dataset captured interactions between members of general public at the exhibition (Isella et al. 2011). Other contact networks however, such as the *hypertext* network, capture interaction between the attendees (Kunegis 2017*c*). It would be more likely that in the conference people would have interacted less randomly than the exhibition. This is because in the conference, people would speak to other people who might have similar research interests. Also, in a conference one person who might have a very interesting research contribution might get more interaction with other people. Methods based on PsInLap work best for *infectiousContact* network. It is very interesting as PsInLap can be interpreted using the concept of conductance and it can be very much connected with the fact that the network is a set of face-to-face interactions that took place in one location.

7. **MITContact:** This dataset is interesting as methods that include Katz are the ones whose performance is the poorest and this is very uncommon that Katz performance capability is so low. 11 out of 12 worst performing methods include Katz element. However, Katz seems to perform better for collaboration networks as it has been seen in the study by Liben-Nowell and Kleinberg (2007). We also see similar result in Table 5.4 that for both of the collaboration networks *hep-th* and *hep-ph*, performance of Katz is good. It is interesting to see that when PsInLap is combined with closeness centrality and betweenness centrality, it outperforms PsInLap used as a standalone predictor. Also, using inverse of PsInLap instead of geodesic path as a measurement of distance gives better performance for this dataset only. In addition, LPI combined with BC satisfies Inequality 5.6.

5.3.4 Computational Complexity

In terms of computational complexity, we have discussed in Section 5.3 that we need to make predictions for $\frac{|V|(|V|-1)}{2} - |E|$ links in total. Thus the time complexity is $O(|V|^2)$, if we wish to predict all possible non-existing links based on Equation 5.2. However, based on different algorithms, each of the methods (i.e. CN, Katz, rooted PageRank etc.) we have used in our combination approach may have different time and space complexities. For example, for CN, JC and AA, where traversal of node neighbourhood is required, the computational complexity is at least $O(|V|b^2)$, where b is the average degree of the graph (Papadimitriou et al. 2012; Lü et al. 2009). Among all the methods, PA has the lowest computational complexity of $O(2|V|)$, as we only need to multiply the predicted pair of nodes' degree. RPR could be calculated using different algorithms and the complexities vary from $O(|V|)$ to $O(|V|^2)$ (in case of a sparse network) (Haveliwala et al. 2003; Berkhin 2005). The computational complexity of calculating an inverse or pseudoinverse of a matrix is usually $O(|V|^3)$ (Courrieu 2005) which is required for MFI, PsInLap, ACT, ACTN, Katz, and LGI. However, there is a faster alternative algorithm proposed especially for Katz, reducing the computational complexity from $O(|V|^3)$ to $O(|V| + |E|)$ (Foster et al. 2001). LPI has a computational complexity of $O(|V|b^3)$ (Lü et al. 2009).

As for centralities, DC from adjacency matrix has a time complexity of $O(|V|)$ (if calculated from a list of edges then complexity of the search algorithm will become the complexity of calculating DC). BC has $O(|V||E|)$ (Brandes 2001) and CC also has the same time complexity of $O(|V||E|)$ (Brandes 2001; Okamoto et al. 2008; Landherr et al. 2010). However, the complexity may vary depending on the algorithm used as pointed out in (Landherr et al. 2010).

For shortest path calculation, there is a range of algorithms available and time complexity depends on the used algorithm. Algorithm selection for shortest path calculation of a graph is based on several factors, such as available computational power and memory, graph type (weighted, directed etc.), graph size, and graph density. Additionally, calculating a selective set of pairs'

shortest path or calculating an all pair shortest path could require different algorithms, resulting in different computational and space complexities. For example, all pair shortest path calculation using the Floyd–Warshall algorithm has a time complexity of $O(|V|^3)$ (Floyd 1962) and the Seidel's algorithm has complexity of $O(H(|V|)log|V|)$ (where $H(|V|)$ is the time complexity of multiplying two $|V| \times |V|$ matrices of small integers) (Seidel 1992). The time complexity of the Johnson's all pair shortest path is $O(min(|V|^{2+\frac{1}{k}}+|V||E|, |V|^2log|V|+|V||E|log|V|))$ (Johnson 1977).

The space complexity of CN, AA, JC is $O(|V|b^2)$ (Lü et al. 2009) and for a matrix inversion it is $O(|V|^2)$ (Lü et al. 2009). Floyd–Warshall algorithm has a space complexity of $O(|V|^2)$.

All the time complexities discussed here are based on a serial processor. However, with the advancement of GPU and distributed computing, parallel and distributed graph algorithms are emerging and can be found in the literature very often. For example, You et al. (2017) proposed an algorithms to calculate degree, closeness, and betweenness centrality measures in directed graphs. In terms of GPU computation, Gunrock is an excellent library which can calculate different centrality measures and shortest path (Wang et al. 2016). In his paper Wang et al. (2016) used very large graphs with millions of vertices and edges and shown the performance of their GPU computation from their graph analysis library Gunrock, which is much better than the performance of a serial processor. There is also another graph processing library with GPU computation available, which comes free with CUDA (NVIDIA's parallel computing framework) named nvGraphs, which shows a very fast PageRank calculation on a very large 1.5 billion edge dataset (Nvidia 2019). The library currently supports PageRank, single-source shortest path, and single-source widest path calculation (Nvidia 2019). The recent revolution of the GPU computation is not only benefiting deep learning but also graph computation (Aher and Walunj 2019; Merrill et al. 2012; Harish and Narayanan 2007; Zhong and He 2014; Shi et al. 2018).

In this research, we proposed a new approach to link prediction in social networks, inspired by Newton's law of universal gravitation, which states that the force exerted between two masses is proportional to the product of those masses, and inversely proportional to the squared distance between their centres (Newton 1987). We have performed extensive empirical analysis to investigate the potential of our link prediction method.

5.4 Results Conclusion

Our experiments indicate that in many cases a combination method, using Equation 5.2 improves performance in terms Precision-Recall Curve, with respect to either standalone similarity measure used in that combination or the product of centralities divided by distance squared

(Inequality 5.6). In cases where we see these improvements (i.e. for all the datasets except *infectiousContact*), we have also seen that AUC values are higher than that of a random predictor. The selection of a method for a particular dataset seems to be related with the type of network the dataset represents, for example, the significant improvements of RPR, LGI,and MFI in terms of the AUC on average, demonstrate that our combination approach has great potential as a link prediction method. Combinations of LGI, shortest path, and MFI with DC work well for both of the collaboration networks, *hep-th* and *hep-ph*. ACT, ACTN with DC, LPI with DC, BC, and CC, MFI and RPR with DC and BC, work best for *collegeMsg* dataset. JC with BC and shortest path with DC work best for *contact* dataset. As for *hypertext* dataset JC with BC and DC, RPR with DC, MFI with DC, BC, and CC, work best. In *MITContact* dataset, PSInLap with BC and CC, LPI with BC, LGI with BC perform best. As for *infectiousContact* none of the combinations works well. In fact, most of the standalone similarity measures perform worse than a random predictor. The exception is PsInLap which works best for *infectiousContact* dataset.

From our empirical analysis, we have concluded that there are a number of combinations which perform better than others. The combination of RPR with degree centrality in Table 5.3.1.7 can be used as a better predictor than using RPR on its own. In addition to RPR, LGI with DC for collaboration networks, MFI with DC, and DC with shortest paths are the best overall combinations that we found in our study.

One powerful property of our approach also allows us to combine local and global measures (e.g. DC with RPR, which considers the larger structure of the surrounding vertex or vertices such) for link prediction. For a pair of vertices, it might happen that the global structure may not indicate link formation probability strongly enough, but the local structure indicates otherwise or vice versa. Due to the combination of local and global measures, in such cases, the final score of link formation would still be higher compared with considering only a local or global measure. Thus, a combination of global and local may improve link formation predictability for pairs of nodes which are likely to be ignored (i.e. false negatives) by a predictor which considers only single local or global measure.

We have discussed similarities between physical networks and social networks in Chapter 2 Section 2.3.1. Our Newtonian gravity inspired link prediction method shows that even at a local level the dynamics of a social network can be interpreted through physical law. The similarity between physical and social world are often encountered. Perhaps one of the most well-known examples is the similarity between complex weather models and social dynamics (Helbing 2012), which supports the idea of benefiting from this kind of similarities between social and physical world. The benefits would come from cross-applying modelling and analytical tools from these domains. However, most of these similarities are emergent phenomena due to the characteristics of a complex system, at a global level. For example, we have discussed how physical and social networks exhibit similar global properties like high clustering coefficient,

degree centrality etc. However, our study shows that we may also benefit from applying laws from physical world to a social network even at the local level.

In terms of computational and space complexity, we have discussed in Section 5.3.4 that we need to make a prediction of $\frac{|V|(|V|-1)}{2} - |E|$ links in total. Thus the worst case time complexity is at least $O(|V|^2)$, if we wish to predict all possible non-existing links. However, each of the methods (i.e. Katz, rooted PageRank etc.) we have used in our combination approach may have different time and space complexity. For example computational complexity of different algorithms to calculate Katz could range from $O(|V|^3)$ to $O(|V| + |E|)$ (Foster et al. 2001).

5.5 Chapter Summary

The link prediction approaches developed in this chapter can also be used as missing link prediction techniques (Liben-Nowell and Kleinberg 2007). As will be shown in further parts of this thesis, the consideration of link prediction as missing link prediction turns out to be a powerful property to overcome one of the limitations and increase performance of node classification algorithms in the context of deep learning. In Chapter 7, we propose a set of deep learning based node classification algorithms inspired by the link prediction methods analysed and developed in this chapter.

Chapter 6

Dynamic Social Network Simulation

In this chapter, we develop a guideline for simulating real-world dynamic social network datasets with ground truth labels and features (i.e. node attributes). Our understanding from the intensive analyses of the three network models in Chapter 4 has contributed to this network simulation development framework. In Chapter 4, we have found that, although all the main three network models are primarily focused on modelling real-world social networks, their simulated networks properties and characteristics are vastly different from each other. Thus a comprehensive network simulation process should blend all the three main network models together (as they all model some aspect of the real-world social networks which we discuss later in Section 6.2) to achieve better modelling of the real-world social networks. In this chapter, the developed simulation process does precisely that and additionally, a novel approach to integrate features and ground truth labels of social networks are also developed.

Graph Convolutional Networks (GCNs), a neural network-based classification model on graphs, have been shown to outperform (in terms of accuracy of node classification) other state-of-the-art models. However, one major limitation of the GCN is that it assumes at a particular l^{th} layer of the neural network model only the l^{th} order neighbourhood nodes of a social network are influential. Furthermore, the GCN has been evaluated on citation and knowledge graphs, but not especially on friendship-based social graphs. Moreover, the drawback associated with the dependencies between layers and the order of node-neighbourhood for the GCN can be more prevalent for friendship-based social graphs. The evaluation of the full potential of the GCN on friendship-based social network requires openly available datasets in larger quantities. However, most available social network datasets are not complete (i.e. represent a subset of the original networks, not the entire graph or do not include the entire set of node features). On top of that, the majority of the available social network datasets not only do not contain any features but also ground truth labels. To address the need for good quality synthetic social network data with ground truth labels and features we firstly provide a guideline on how to simulate dynamic social

networks, with ground truth labels and features, both coupled with the topology of the network. Secondly, we introduce an open-source Python-based social network simulation library with GPU computation and multiprocessing [1]. In our social network simulation, we argue that the topology of the network is driven by a set of latent variables, termed as the social DNA (*sDNA*). We consider the *sDNA* as labels for the nodes, mimicking the real-world social network scenario. Finally, by evaluating on our simulated datasets, we propose four new variants of the GCN, mainly to overcome the limitation of dependency between the order of node-neighbourhood and a particular layer of the model. We then evaluate the performance of all the models and our results show that on 27 out of the 30 simulated datasets our proposed GCN variants outperform the original model.

To address the issues discussed above, i.e., (1) lack of guidelines on implementing both the well-studied network properties in social networks and features, (2) insufficient research on simulating dynamic social networks with node features, (3) lack of rigorous study providing directions on defining node labels in social networks, we propose a framework for social graph simulation. In our model, the simulated networks have the following characteristics based on understanding of Facebook-type social networks, along with well-studied social network properties such as preferential attachment.

- Node features are evaluated by other nodes before connecting. If two nodes are forming a connection, the decision of forming a link is taken by both of the nodes, thus both parties should evaluate each other's features.
- The decision of forming a connection is based on the preferences of nodes, which are consist of a set of latent variables. These preferences are not directly linked with users' features. For example, two people could live in any state or county, but the preference towards a particular political party could be same, thus resulting in different features but common preferences.
- People have common preferences. For example, a group of people in social network may prefer a common ideology or political view.
- The node and graph level characteristics should both be taken into account while modelling a network. Node level characteristics consist of features (i.e. node attributes such as age, gender, etc.), individual preferences (latent variables such as preference towards a particular type of people, discussed in more detail in the Section 6.1), node degree (i.e. preferential attachment). Whereas, graph level characteristics is e.g. smaller path length preference, i.e. connecting with friends who are nearer in terms of the graph topology.

[1] https://github.com/AkandaAshraf/VirtualSoc

6.1 Proposed Approach

The proposed simulation is based on preferences (i.e. a set of latent variables) of nodes, which can be interpreted as social rules. Node preference represents the preference of a person in a social network, and at the same time the network-based projection of personality and behaviour. This in turn translates into the network topology. For example, one of network-based behaviours might be to only connect with people with many mutual friends, shaping the topology of the ego network.

We identify the types of preference a node can have based on their topological and non-topological characteristics. The preferences/behaviours emerge from the following phenomena:

- Feature-based (non-topological): From node/user point of view, a combination of variable preferences towards the features of other nodes/users acts as a deciding factor for who they wish to connect with. For example, someone may prefer to connect with people who live near in terms geographic location (e.g. city, town), thus similar location feature is preferred. Whereas, for some other features such as gender, being opposite or same, could be preferred. Thus, for this particular node, preference towards geographic location in combination with gender is considered while connecting.
- Topology-based: Besides node features, the local topological characteristics may also play important role, e.g. someone may prefer to connect with other people with whom he/she has mutual friends, whereas others may be more open. Secondly, some people may still prefer to connect with someone who has many friends, i.e. popular nodes. Both of these preferences are solely based on the graph topology and could be mostly identified via the topological properties of the social graph.
- Hybrid feature and topology based (combination of topological and non-topological features): People in social networks may prefer someone who is nearer to them in terms of geographic location and has similar age, education level and has many mutual friends. In this scenario, we have a combination of both the feature-based and topology-based preferences. Someone may also only connect with a politician who has many friends, only if, he or she has similar political views.

All three types of human preferences are reflected in both non-topological features of a node and the topology of a graph. Although the first, feature-based preference, solely emerges from the non-topological component of a node, once the connections are made, these preferences are also reflected or encoded within the topology of the graph. Without the consideration of the graph or relations between nodes, the predictive model will not be able to capture these complex patterns,

which may negatively influence model performance. As a result, by including node attributes in graphs we can achieve higher predictive power.

We propose that the labels of social networks in supervised or semi-supervised classification will capture patterns resulting from the preferences discussed above. We name these preferences or behaviours for a particular node as their social-DNA (*sDNA*). Although most people in a social graph have different features, many people have a similar *sDNA* (i.e. they share preferences). As a result, the *sDNA* is the most valuable and meaningful candidate for class labels for grouping nodes in a graph. However, these labels may not be explicitly defined for a given node classification problem. For example, a classification task may require identifying a group of nodes who may prefer to buy a certain type of product, for marketing purposes. The label for the class who have bought the products should capture a certain group of people with similar preferences in the social network. In semi-supervised classification, if we have a dataset for only a few people who may have bought the product, the classification model would associate a certain type of *sDNA* in the social network as the most likely group to buy that product. In addition, if we do not have any historical information which tells us who have bought the product, it may be possible that a group of people with similar *sDNA* may prefer the product more often than others. However, a person or node itself in a social network, may not entirely know his or her preferences or *sDNA*. As a result, finding these preferences (i.e. labels) in terms of *sDNA* is a nontrivial task. One solution to this problem could be to define a few randomly selected nodes with different labels. These labels could also be selected based on a strategy of focusing on the features of nodes. For example, selecting a few nodes with very different features from each other. After labelling, the semi-supervised classification algorithm, such as GCN will infer other nodes with similar preferences/*sDNA*. Even if no prior knowledge is available, randomly selected nodes with different class labels could be used with only one label per class. GCN is shown to be powerful enough to accurately classify nodes with only one label per class (Kipf and Welling 2017).

6.2 Graph Formation

Let's assume that we have $|N| = n$ number of nodes with $\mathbf{f}$ features each. Each element or feature in the $\mathbf{f}$ could be unbounded or bounded. Each and every node subscribes to an *sDNA*. There is a total of y different *sDNA*s, such that $y <= n$. Nodes which subscribe to the same *sDNA* have the same preference, thus the same label. Each *sDNA* consists of two vectors of length $\mathbf{f}$ (i.e. the same as the number of node features). These two vectors are, (1) $\tilde{\mathbf{w}}$, which defines the strength or weight of a particular feature's preference and ranges between 0 and 1, and (2) $\tilde{\mathbf{l}}$, which defines whether similar or dissimilar features are preferred with a binary attribute 1 or -1. Although $\tilde{\mathbf{l}}$ could be incorporated into $\tilde{\mathbf{w}}$ as its sign, to make the

preference standout separately for user readability and its contribution in the *sDNA* mutation process discussed in Section 6.5 for dynamic graphs, the vector $\tilde{\mathbf{l}}$ is used. This also allows to have a separate label for the preferences within the *sDNA*, which could be learned using machine learning algorithms from the graph enabling a more in-depth predictive analysis. The feature-based scores between two nodes i and j are calculated as (where $\odot$ is the Hadamard product):

$$\Phi_{i \to j} = |\mathbf{f}_i - \mathbf{f}_j|^\top (\tilde{\mathbf{w}}_\mathbf{i} \odot \tilde{\mathbf{l}}_\mathbf{i}) \tag{6.1}$$

Equation 6.1 gives feature-based score which entails if node j is a potential friend for node i. In this case, node i evaluates if it wants to connect with node j or not, as we consider only i's *sDNA*. In many social networks, mainly for the undirected ones, the final connection or friendship decision is made by both of the nodes. We can introduce two-way evaluation simply by adding node j's *sDNA* based score in Equation 6.1.

$$\Phi_{i \leftrightarrow j} = \Phi_{i \to j} + \Phi_{j \to i} \tag{6.2}$$

If both the i and j subscribe to the same *sDNA* then $\Phi_{i \leftrightarrow j} = 2\Phi_{i \to j}$. However, Equation 6.2 does not prefer similar *sDNA* over different *sDNA* or vice versa. In a social network, the preference or *sDNA* is a set of latent variables. It may well be that two people have a similar preference and this results in a lower score. For example, if two social network users prefer to connect with the opposite gender more often, then if they have the same gender then they are less likely to connect.

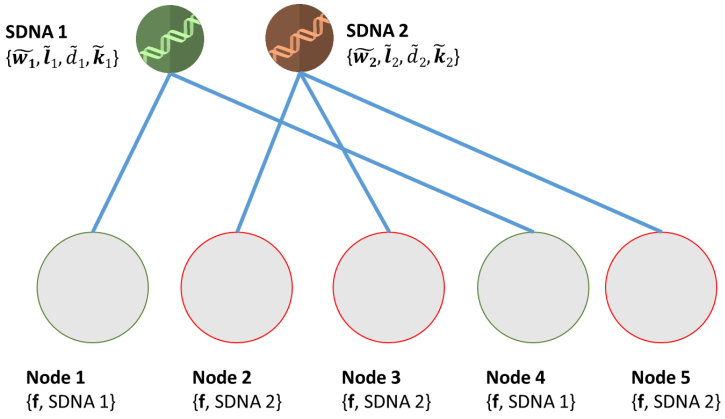


FIGURE 6.1: Two types of *sDNA* subscribed by 5 nodes (The lines do not represent edges in the graph and *sDNA*s are not nodes. The arrows define subscription or common preferences of different nodes as *sDNA*s)

Equation 6.2 does not consider topological (i.e. graph based geometric features) of the nodes while calculating the score. In social networks, popularity, i.e. the degree of the nodes is a common topological feature with a significant effect on the growth of the network. Typically,

people tend to prefer other people who have large number of connections. This is why famous people tend to get more connections. This phenomenon is well studied and known as preferential attachment (Barabási and Albert 1999). To add the preferential attachment effect, we can simply add the degree of the connected nodes to the score. If node j has m_j degree or connections, then from i's perspective, the popularity-based score can be calculated as follows:

$$\delta_{i \to j} = m_j \tag{6.3}$$

However, the preference for nodes with higher degrees varies from person to person. We can incorporate this variability by including *sDNA*'s preferential attachment parameter while calculating the score, resulting in:

$$\Delta_{i \to j} = m_j \tilde{d}_i \tag{6.4}$$

Equation 6.4 only considers score of i and j from i's perspective (i.e. i's *sDNA*), where $\tilde{d}_i$ is an $sDNA$ variable from i which determines the preference of j's popularity (i.e. degree). For an undirected graph, we can use the following Equation 6.5 to calculate preferential-attachment-based score from both i and j's perspective by adding both of their scores from each other's perspective:

$$\Delta_{i \leftrightarrow j} = \Delta_{i \to j} + \Delta_{j \to i} \tag{6.5}$$

A social network user tends to prefer people who are nearer to them in terms of the graph topological distance (Wahid-Ul-Ashraf et al. 2017). Creating a connection with somebody who is friend-of-our-friend is usually more likely than starting a relationship with someone who is further away from us in the structure. However, this preference is also subjective and varies among social network users. As a result, we add this variability of path length preferences subjective to a user by using *sDNA*'s $\tilde{\mathbf{k}} = \tilde{k}_2, \tilde{k}_3, ...\tilde{k}_q$ variables, where q is the longest path-length in the graph that the model is considering, and $\tilde{k}_2 > \tilde{k}_3 > .... > \tilde{k}_q$. The *sDNA*'s $\tilde{\mathbf{k}}$ vector has a length of $(q-1)$.

$$[\mathbf{A}^x[i,j]] = \begin{cases} 0 & \mathbf{A}^x[i,j] = 0 \\ 1 & \mathbf{A}^x[i,j] > 0 \end{cases} \tag{6.6}$$

$$\begin{aligned} \mathbf{\Pi}_{i \to j} = [\mathbf{A}^2[i,j]]\tilde{k}_(i,2), [\mathbf{A}^3[i,j]]\tilde{k}_(i,3), \\ ...., [\mathbf{A}^n[i,j]]\tilde{k}_(i,q) \end{aligned} \tag{6.7}$$

In Equation 6.7, $[\mathbf{A}^x[i,j]]$ is a generalised Kronecker delta function in the Iverson bracket where $\mathbf{A}$ is the adjacency matrix of the graph. The value of $[\mathbf{A}^x[i,j]]$ is one, if a path of length x between i and j exists, and zero otherwise. This function of path length introduces non-linearity in the score. Equation 6.7 gives the score of j when i is evaluating j's potential to be able to connect or become friends with i. This is done by using i's *sDNA* parameter to calculate the score of J. This imitates the behaviour, firstly, i may find someone j interesting to send him a friend request on Facebook. Secondly, the final connection will be made if j also finds i interesting. Equation 6.7 accounts for the first case and the following Equation 6.8, similar to score based on features in Equation 6.2 accounts for the score from j's point of view for i. The final score function based on path topology is:

$$\mathbf{\Pi}_{i \leftrightarrow j} = \mathbf{\Pi}_{i \rightarrow j} + \mathbf{\Pi}_{j \rightarrow i} \tag{6.8}$$

Equation 6.8 is required if we were to simulate a directed graph. For an undirected graph, we only consider Equation 6.7.

Finally, we add all three scores, i) feature based from Equation 6.2, ii) popularity based from Equation 6.5, and iii) shortest path length based from Equation 6.8 (directed) or 6.7 (undirected) to calculate the final score s. We consider an undirected graph where both i and j make mutual decision to connect with each other. In case of an undirected graph, for i connecting with j, we can simply consider Equations 6.1, 6.4, and 6.7. For simplicity we are not including the subscript $i \leftrightarrow j$ in the final score function s.

$$s(\Phi, \Delta, \mathbf{\Pi}) = \Phi + \Delta + \mathbf{\Pi} \tag{6.9}$$

Equation 6.9 gives the score between any two nodes. The scores are weighted or modified according to the *sDNA* a node belongs to. To further enforce some global graph level control in the effects of feature-based, popularity-based, and shortest-path scores we introduce two hyperparameters. This global control is useful in many situations, for example, one may wish to generate networks where strong preferential attachment phenomena exist. To be able to control this global weighting, we introduce r and $\mathbf{c}$ global weighting factors in Equation 6.9. $\mathbf{c}$ is a vector of length $q-1$, where $q-1$ is the number of shortest path length considered starting from length two. We discuss on the selection of these global parameters in Section 6.7.

$$s(\Phi, \Delta, \mathbf{\Pi}) = \Phi + r\Delta + \mathbf{c}^\top \mathbf{\Pi} \tag{6.10}$$

Equation 6.10 contains $sDNA = \{\tilde{\mathbf{w}}, \tilde{\mathbf{l}}, \tilde{d}, \tilde{\mathbf{k}}\}$ variables from Φ, Δ, and $\mathbf{\Pi}$. These do not come from the nodes directly but from their *sDNA*s, which in turn expresses their behaviour in the

network. Equation 6.10 is one possible linear combination of Φ, Δ, and $\mathbf{\Pi}$, however, other possible nonlinear combination functions may be used depending on the target domain.

6.3 Simulation Process

The link formation process for a graph with n nodes is given in Algorithm 1. Each node subscribes to exactly one of y different types of *sDNA* (Figure 6.1) and contains f features. In Line 2 the algorithm generates all pairs of nodes. In case of an undirected graph, the pairwise permutation (without repetition) is considered. Furthermore, if self-connection is desired then self pairwise combination is also included. Social network users do not necessarily explore all potential friends whom they might connect with. For example, a Facebook user does not explore all existing Facebook users to connect with. As a result, the simulation process selects a pair of nodes to calculate scores with the exploration probability p (Line 6), much like how connections are made in a random graph. $p = 1$ will result in calculation of scores for all possible pairs, while for $p = 0$, no score will be calculated between any pair of nodes. The exploration probability p incorporates controlled stochasticity. In order to determine the minimum score a pair of nodes should have to connect, we define a cut-off point t. The top scoring t fraction of nodes are connected.

To sum up, first, we calculate scores between pairs selected based on the exploration probability p and then we sort these scores in descending order. After that, we connect t fraction of pairs of nodes in the entire graph. Smaller values of t will result in a social network where the users are very particular about with whom they connect. On the other hand, very high values of t will result in a network where users do not care about features or topological properties while connecting. Thus the latter will be close to a random graph model with probability of edge occurring being equal to p, i.e. $t = 1$ will result in a pure random graph model with p probability of edges formation. In Line 4 and 5, from all sets of pairs we select a pair for score calculation with probability, p. In Line 5, $r(0, 1)$ is a random number generator function which returns a random number from 0 to 1 from uniform distribution. In Line 6, we use Equation 6.10 to calculate a score between the selected pairs of nodes. Afterwards, the stopping length based on the suggested fraction of node pairs to be connected is calculated (Line 11). In Line 12 we sort the selected pairwise nodes' scores in descending order. Afterwards, in Line 14, we connect the first t fraction pairs of nodes' for which we have calculated scores in $Scors$, in Line 12, thus, pairs with higher scores will have a higher likelihood of forming connections.

Algorithm 1 Socialise algorithm

```
1:  procedure SOCIALISE(N, p, t)
2:      Pairs ← PairCombination(N)
3:      i ← 0
4:      for all pair in Pairs do
5:          if r[0, 1] ≤ p then
6:              Scores[i] ← s(pairs)
7:              i + +
8:          end if
9:      end for
10:     i ← 0
11:     StoppingLen ← ⌊t × Length(Pairs)⌋
12:     Scores ← sort(Scores, descending = True)
13:     for all score in Scores do
14:         Connect(pairs)
15:         if i ≥ StoppingLen then break
16:         end if
17:         i + +
18:     end for
19: end procedure
```

6.4 Curse of Dimensionality in Networks

Real-world social networks contain high dimensional features. If we consider a Facebook user's posts, likes, photos, comments, etc. as features, then we have thousands of features for each of the node. One problem with nodes with high dimensional features is the linear increase of computational complexity for the simulation process discussed in Section 6.3. To overcome this problem, GPU computation can be used to calculate Scores in Equation 6.10. In our simulation library, we have enabled GPU computation and Figure 6.2 shows computation time with 300 nodes with increasing number of features.

6.5 Dynamic Graphs

Real social networks evolve over time and are dynamic in their nature. However, dynamic graph datasets are very rare to find, especially with ground-truth labels and node attributes. These datasets are crucial in the field of dynamic graph research, but also essential for the evaluation of a link prediction, which usually deals only with static graphs or a snapshot of a graph at time t. The link prediction problem is to identify new links that will be present in the network at time $t + 1$ (Bliss et al. 2014; Hristova et al. 2016). Assuming the network has a set N of nodes and set E of edges at time t expressed as $G(N, E_t)$, and that a link between a pair of vertices i and j is denoted by $L(i, j)$, the goal of link prediction is to predict whether $L(i, j) \in E_{t+1}$, where $L(i, j) \notin E_t$. The prediction is performed by using topological

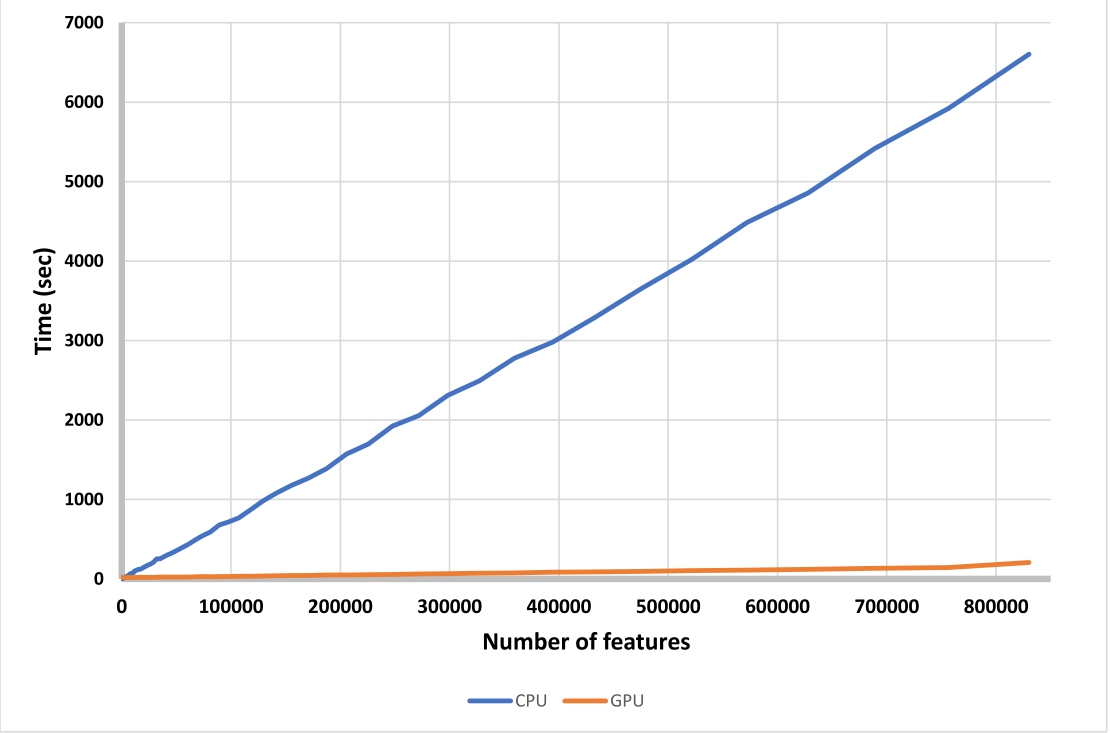


FIGURE 6.2: CPU vs GPU computation time with varying number of features. (CPU: Intel(R) Xeon(R) W3680 @ 3.33GHz 6 cores and 12 threads, system memory: DIMM DDR3- 20 GB, GPU: NVIDIA GeForce GTX 1080Ti)

and/or non-topological information about node characteristics and their relationships. Thus, to evaluate or test the performance of a link prediction method, a future snapshot at $t+1$ time is required. Additionally, machine learning based link prediction algorithms require a future snapshot of the network at time $t+1$ as ground truth for training purposes. Interestingly, by using multiple runs of Algorithm 1 we can already get dynamic graphs, i.e. a future snapshot of the network. Every time we socialise the graph using Algorithm 1 containing pairs of nodes which are not yet connected, we will get new connections occurring within the graph. However, this is perhaps not the best simulation of the dynamic nature of real social networks. The reason is that by running Algorithm 1 multiple times we are forcing each of the social network users to make consideration and connect with people which they did not find interesting enough in the previous run(s)[2]. What we really want is not to force the users in the graph to make new connections but allow the users' interest and preference to change and then run the Socialise Algorithm 1. This will result in a concept drift in the user preferences, which can be achieved via changing values of the variables in the *sDNA*'s of the nodes. This changes in the *sDNA* reflect the phenomenon that, the rules that govern social networks can and do change over time. This change of preference can be achieved via the *sDNA* Mutation given in Algorithm 2. The intensity of mutation can be controlled by mutation intensity parameter z, which results in changing values of the variables in $sDNA = \{\tilde{\mathbf{w}}, \tilde{\mathbf{l}}, \tilde{d}, \tilde{\mathbf{k}}\}$. A lower value of z would change only few

[2]Here, we are assuming no arrival of new nodes or a constant number of nodes. In case of new nodes, we can easily run Algorithm 1 with the new nodes and include them in the graph.

of the $\tilde{\mathbf{w}}$. As a result, the user's preference towards a potential friend's feature $\mathbf{f}$ would change. In case the value of the mutation intensity parameter z is defined larger, this would result in changes to the entire preference vector $\tilde{\mathbf{l}}$. In Algorithm 2, in Line 1, the procedure takes all y existing $sDNAs$ from the graph, and $mutatePreference$, a Boolean parameter to determine if l should also be changed. In Line 2 we iterate through each of the $sDNAs$, one at a time. For the given *sDNA*, we then iterate through each of the elements in $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{l}}$ in Line 3. We then reassign the value of $\tilde{w}$ with the probability z (Line 5). In Line 7 we check if the $mutatePreference$ is set True. If so, then we also reassign the value of $\tilde{l}$, 1 or -1, with a probability of z (Line 9). The selection between 1 or -1 selected randomly from uniform random distribution. In Line 14 we reassign the $\tilde{d}$ parameter of *sDNA*, which is for preferential attachment strength. Afterwards, in Line 16, q number of random numbers are generated, for each path length preference in Equation 6.7. The intervals are selected such that it satisfies, $\tilde{k}_2 > \tilde{k}_3 > .... > \tilde{k}_q$. Afterwards, in Line 20, we again iterate through each elements of $\tilde{\mathbf{k}}$ and reassign from the already generated random numbers in Line 16.

Algorithm 2 Mutation algorithm

```
1:  procedure MUTATE(z, sDNA, mutatePreference)
2:      for all sDNA in sDNA do
3:          for all w̃, l̃ in w̃,l̃ do
4:              if r(1) ≤ z then
5:                  w̃ ← r(1)
6:              end if
7:              if mutatePreference then
8:                  if r(1) ≤ z then
9:                      l̃ ← rand(−1|1)
10:                 end if
11:             end if
12:         end for
13:         if r(1) ≤ z then
14:             d̃ ← r(1)
15:         end if
16:         K_r ← r[1, (q−1)/q), r[(q−1)/q, (q−2)/q), ..., r[1/q, (q−q)/q)
17:         i ← 0
18:         for all k̃ in k̃ do
19:             if r(1) ≤ z then
20:                 k̃ ← K_rand[i]
21:                 i++
22:             end if
23:         end for
24:     end for
25: end procedure
```

An interesting observation is, generally people's behaviours or preference changes are correlated with time. This change in behaviour, for social network users, contributes to change in the

topology of the social network. In our simulation strategy, a snapshot to snapshot time difference then should also be correlated with the change of the users' behaviour or preferences, i.e. *sDNA*s. The parameter z in Algorithm 2 defines this intensity of mutation in *sDNA* or intensity of social network users change in behaviour. As a result, the value of z is proportional to the time between two snapshots of the network. For example, if one wishes to run the Socialise algorithm (Algorithm 1), it will produce a social network with the first snapshot, $snapshot - 1$. Then running the Mutation algorithm (Algorithm 2) will result in change in preferences with a particular value of the parameter z, and then rerunning the Socialise algorithm will result in another snapshot of the network in a forward time dimension, $snapshot - 2$. A high value of z will result in higher time difference between these two snapshots, $snapshot - 1$ and $snapshot - 2$.

One may wish to generate event based dynamic networks, i.e. time-stamped link formation. This can also be achieved by setting the 'fraction of nodes to be connected' parameter t from the Socialise Algorithm 1 such that only one link is formed. A repeated run of Algorithm 1 will result in edge stream with timestamp for each of the edges. As we have discussed earlier in the section, the time between each of the edge appearing could also be manipulated by changing the value of parameter z.

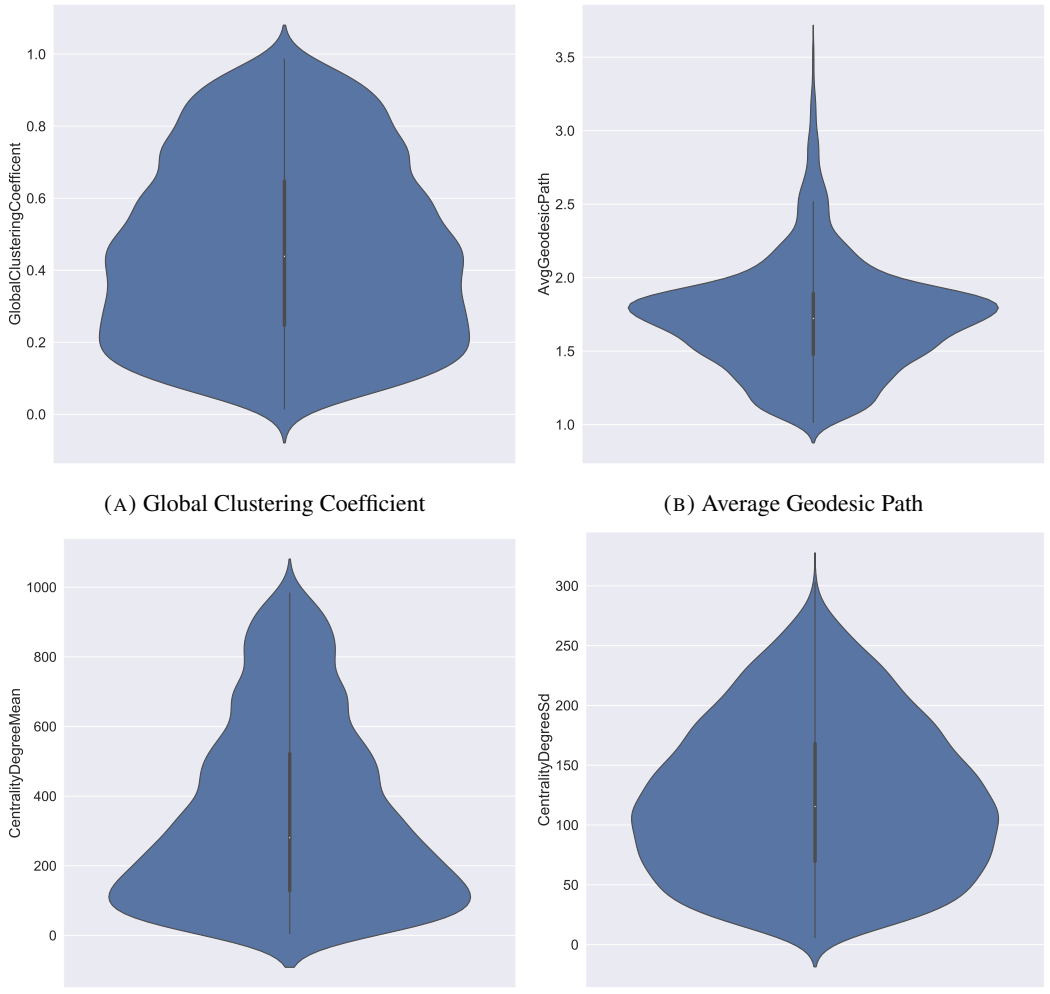
6.6 Generated Networks

We have generated a total of $1400 \times 3 = 4200$ (three snapshots for each of the 1400 networks) networks for the purpose of inspecting their properties. Three snapshots of all the 1400 networks are generated with 50 node features. The features are integer type and the range is $[1, 500)$. All the features are generated from a discrete uniform distribution.

The wide combinations of parameters are selected based on standard sampling method for global sensitivity analysis. Each of the model inputs are sampled using Saltelli's extension of the Sobol sequence (Sobol 2001; Saltelli 2002; Saltelli et al. 2010). These input parameters are generated using the open-source library *SALib - Sensitivity Analysis Library in Python* (Herman et al. 2014). The range of the parameters for sampling are given in Table 6.1

exploration probability p	popularity Preference Intensity, r	node-pair fraction connection, t	Path (len:2) Preference Intensity c_1	Path (len:3) Preference Intensity c_2	Path (len:4) Preference Intensity c_3
[0.1, 1.0]	[0.1, 10]	[0.1, 0.80]	[0.7, 0.9]	[0.3, 0.6]	[0.1, 0.2]

TABLE 6.1: Range of input parameters for the simulated networks. Generated from the sampling approach developed by Saltelli (2002) (Saltelli et al. 2010; Herman et al. 2014) for global sensitivity analysis.



(A) Global Clustering Coefficient

(B) Average Geodesic Path

(C) Degree Centrality (mean)

(D) Degree Centrality (standard deviation)

FIGURE 6.3: Kernel density estimation of the underlying distribution for the properties of the generated 4200 networks

In Figure 6.3a we see that most of the networks exhibit a healthy range (see Figure 7.1 in Chapter 7 for examples of real-world online social network's global clustering coefficient) of global clustering coefficient for social networks (Watts and Strogatz 1998; Lee et al. 2014; Mislove et al. 2007). The high clustering coefficient is indicative of community structure that is expected in a real-world social network (Newman 2003; Girvan and Newman 2001; Ravasz and Barabási 2003; Davidsen et al. 2002). This emergence of high global clustering coefficient is mainly due to the path length preference in a social network (Jost and Joy 2002), which can be controlled by the $\mathbf{c}$ parameter in Equation 6.10. Based on Figure 6.3b, we can see that the average path length of most of the networks is low, indicating a strong small-world effect. The real degree of separation for social networks is decreasing significantly according to the study by Edunov et al. (2016*a*). However, this study has been published in 2016, as more and more people are joining, the interconnectivity within Facebook is expected to increase more and the degree of separation may reduce even further (Edunov et al. 2016*b*). As for degree centrality, we have

calculated mean and standard deviation of the degree of all nodes, which are presented in Figure 6.3c and 6.3d. The high range for standard deviation for the degree is consistent with the high level of variation in degree that is found in real-world social networks (Barabási and Bonabeau 2003). The scale-free network model is built-in within our network topology simulation and can be controlled with the parameter r in Equation 6.10.

Rather than limiting ourselves with a few specific types of network models, our simulation framework allows us to model a wide range networks based on any specified input parameters. In this generated networks, we do not model any specific network type, rather analyse a wide range of possible types of networks. Our example of simulated networks is limited by the ranges shown in Table 6.1, which can also be changed depending on the user's preference.

6.7 Individual Properties and Simulation Validation

It is impossible for any simulator to replicate all the complex human behaviour which are the main source of real-social networks as it would require us to simulate a human brain and the nature. Thus, our network generator's goal is not to produce real-world networks, but to imitate some of their aspects and dynamics to provide us with high quality synthetic datasets. We have discussed how simulated datasets are still beneficial for the advancement of deep learning in Section 1.3. In this section, we show some properties of five individual simulated networks to demonstrate that our simulation framework can imitate real-world social networks.

6.7.1 Nonlinear Dynamics of Social Networks

The evolution of any social network with thousands or millions of nodes are governed by chaotic dynamics (Gregersen and Sailer 1993; Smith 1998; Thietart and Forgues 1995; Lorenz 1963). The Newtonian physics view of the universe as 'clockwork', where the complexity of the (similar to a clock) universe can be reduced to its basic components. Additionally, the Newtonian physics argues that, in this 'clockwork' universe, by understating and modelling its basic components we can make a very precise prediction of the future. This Newtonian epistemology has been successful in predicting a linear system, but for a richer system which is complex and consists of two or more components and their nonlinear interaction, the Newtonian view has been proven to be unsuccessful (e.g. the three-body problem) (Nielsen et al. 2001; Mazzocchi 2008; Lorenz 1963). In fact, for modelling a complex system, the gradual inability for long term prediction is the sign of a good predictive model (Sugihara and May 1990; Ashraf 2016). For a social network, even if we assume that the social interactions are fully deterministic, i.e. fully dependent on their previous states, due to its complexity and nonlinearity, the network should become unpredictable in long run. As a result, similar to a real-world social network, a social

network simulator can be fully deterministic but due to its nonlinearity and chaotic nature, accurate prediction of the long-term properties is not possible with our current understanding of the physics of nonlinear dynamical systems (Kellert 1993; Firth 1991; Lorenz 1963). On top of that, in our simulation framework, we incorporate some randomness using the parameter p in Algorithm 1. Additionally, also in the generation of features and $sDNA$ are random. This randomness is included in order to capture the chaotic nature of social networks. However, one may argue that the real-world social networks are fully deterministic (the arguments regarding determinism of human behaviour largely fall in the realm of philosophy). However, even if one assumes that the social network is fully deterministic, as discussed earlier in this section, due to the sheer complexity of a social network, it becomes almost impossible to capture its deterministic complexity in a model. The nonlinarity is introduced in our simulation model mainly in two ways, 1) the $sDNA$ variable $\tilde{\mathrm{I}}$ in Equation 6.1 and 6.2 2) the parameter t as a cut-off point for the score in Algorithm 1, which results in a nonlinear complex system. As a result, long term states of the networks from our simulation have a sensitive dependence on all the initial features, the parameter p, and $sDNA$ variables. However, one may wish to generate networks with some desired properties and in order to achieve that, in our simulation software we provide a fast multi-threaded approach to simulate a large number of social networks with an automatic selection of a variety of parameters. One can then select networks based on the desired properties (see Appendix A, Section A.6). This type of generative approach to model network has already been used in network modelling (Newman 2016). Although a short term prediction of the behaviour of the simulated networks can be achieved by the input parameters of the simulation, as the networks grow (i.e. more snapshots into the future), their properties will start to become unpredictable although they are still governed by all the input parameters and random states of the simulator.

6.7.2 Properties of Simulated Networks

In this section we show few of the network's properties vs properties of real-world networks. In Table 6.3 we show the properties of the simulated networks and in Table 6.2 the input parameters used to simulate the networks. However, our network's dynamics are also governed by the $sDNA$ variables and the feature sets, which are provided online on Kaggle [3]. The parameters reported here provide some global control, but much like real-world networks, each node has its own preferences and features (see Section 6.2). These preferences ($sDNA$ in our simulator) and features also dictate the dynamics and topology of the network.

[3] https://www.kaggle.com/akandaashraf/virtualsoc1

Networks	p	r	t	c_1	c_2	c_3
23	0.591308594	6.007128906	1.077148438	0.813867187	0.475488281	0.175488281
24	0.591308594	4.054199219	2.465820313	0.813867187	0.475488281	0.175488281
25	0.591308594	4.054199219	1.077148438	0.735351563	0.475488281	0.175488281
26	0.591308594	4.054199219	1.077148438	0.813867187	0.534082031	0.175488281
27	0.591308594	4.054199219	1.077148438	0.813867187	0.475488281	0.140722656
101	0.185253906	2.816699219	71.59082031	0.810351562	0.421582031	0.153222656
1247	0.572412109	5.296533203	11.06787109	0.733300781	0.497607422	0.116455078
1250	0.572412109	5.296533203	44.31884766	0.733300781	0.497607422	0.116455078

TABLE 6.2: Input parameters of the simulated networks. The parameters are: exploration probability p, popularity preference intensity r, node-pair fraction connection t, path 2 preference intensity c_1, path 3 preference Intensity c_2, path 4 preference intensity c_3

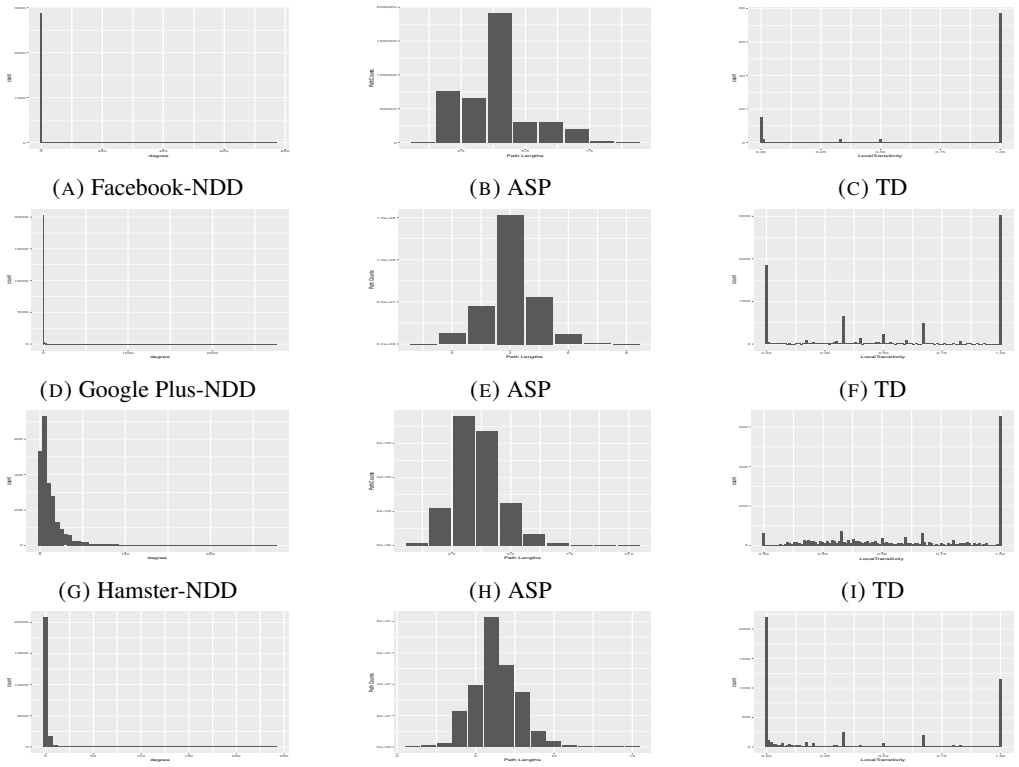
In Table 6.2, we can see that the node-pair fraction connection t and exploration probability p controls the density of the networks. Networks 23-27 have a lower value of t but higher value of p, thus these networks' nodes tends to explore more potential nodes to connect with (see Algorithm 1). The parameter p controls the keenness of nodes to explore potential connections and the parameter t controls the fraction of the potential nodes that a node is likely to connect with. For a higher value of t, nodes will be more open in terms of who they connect with. In network 101, the nodes tend not to explore many potential friends but they are more likely to connect with nodes which they do explore (low p and high t). We see that for network 1250, both the p and t are high, thus resulting in a very dense network. We also see that network 101 has the lowest value of r, thus the degree distribution of the network in the final snapshot starts becoming less scale-free. This parameter is the global weight of for the scale-free preference of a node (see Equation 6.5). As for the input parameters c_1, c_2, and c_3, they indicate if a person would prefer to connect with someone who is two, three, or four nodes away (i.e. path length distance). As mentioned in Section 6.2, for the description of Equation 6.8, these parameters are set such that $c_1 > c_2 > c_3$. Due to these two parameters, we can see that in Table 6.3, the simulated networks have a high range of global clustering coefficient. Most of the real-world networks shown here, many of them are only a subset of the original networks, thus their properties may not represent the entire network. Additionally, the real-world networks are vastly different from each other (also see the real-world network properties shown in Chapter 5, Figure 5.2) and it is very difficult to pinpoint a single precise property or indicator range which may indicate if a network is indeed a social network (Broido and Clauset 2019; Holme 2019; Voitalov et al. 2019). However, we see that the transitivity increases and the average geodesic path decreases over time for the simulated networks as we move from the first snapshot to the third one. These indicate that the simulation imitates the aspect of a real social network, where people tend to have more friends and become well connected (Edunov et al. 2016*a*). As for the degree distribution, we have shown the network's degree distribution in terms of the fraction of

nodes' degree. There seems to be some debate if scale-free network's are unique or common for social networks (Broido and Clauset 2019; Holme 2019; Voitalov et al. 2019). There are also debates on the procedure to determine if a network is scale-free (Holme 2019; Voitalov et al. 2019). Voitalov et al. (2019) argued that the assumption that real-world social networks should be very precisely scale-free may not be true and in that case observation of scale-free networks are not rare as Broido and Clauset (2019) argued in their study. The finding from the study of Voitalov et al. (2019) states that "According to their estimates, real-world scale-free networks are definitely not as rare as one would conclude based on the popular but unrealistic assumption that real-world data come from power laws of pristine purity, void of noise, and deviations.". In our study we take a neutral view on this issue of whether social networks are indeed scale-free or not, thus, we do not validate our simulated datasets based on their scale-freeness. However, we focus on the global clustering coefficient of the simulated networks to validate if our generated networks are sufficiently similar to real-world networks.

In terms of parameter selection, if one wishes to increase the scale-free behaviour the parameter r can be defined to have a larger value, in case one wishes to generate a denser network then p and t parameters need to have a higher value. As we discuss in Section 6.3, defining the parameter t high (close to 100%) would result in nodes ignoring their preferences dictated by the $sDNA$ and features, but only dependent on the probability p, similar to a random graph. Additionally, the c_1, c_2, and c_3 parameters can be defined with higher value to ensure higher level of clustering coefficient.

Network	Vertices	Edges	GCC	CCS	CDM	CCM	CDS	CCS	AGPL
23-0	1000	7283	0.056	0.0704	14.57	7.79E-06	15.0	2.42E-06	2.922
23-1	1000	14460	0.122	0.1056	28.92	2.21E-05	31.5	4.37E-06	2.549
23-2	1000	21533	0.199	0.1637	43.07	5.38E-05	50.7	6.80E-06	2.329
25-0	1000	3182	0.059	0.0666	6.36	1.50E-06	9.7	4.99E-07	2.845
25-1	1000	6344	0.118	0.1040	12.69	1.67E-06	18.9	6.02E-07	2.486
25-2	1000	9486	0.173	0.1597	18.97	2.10E-06	28.1	8.35E-07	2.410
26-0	1000	3182	0.058	0.0955	6.36	1.63E-06	10.0	5.81E-07	2.951
26-1	1000	6344	0.101	0.1062	12.69	2.12E-06	18.9	8.41E-07	2.631
26-2	1000	9486	0.128	0.1828	18.97	2.79E-06	27.2	1.13E-06	2.476
27-0	1000	3182	0.079	0.1210	6.36	1.91E-06	10.5	7.39E-07	3.386
27-1	1000	6344	0.139	0.1156	12.69	2.62E-06	20.2	1.06E-06	2.802
27-2	1000	9486	0.173	0.1579	18.97	3.03E-06	29.5	1.22E-06	2.493
101-0	1000	10135	0.095	0.1193	20.27	4.11E-06	24.0	1.57E-06	2.492
101-1	1000	20064	0.185	0.1330	40.13	5.00E-06	47.5	1.81E-06	2.245
101-2	1000	29792	0.261	0.1572	59.58	6.46E-06	69.8	2.16E-06	2.142
1247-0	1000	10169	0.040	0.0480	20.34	3.53E-05	14.7	5.52E-06	2.677
1247-1	1000	20131	0.088	0.0615	40.26	6.92E-05	31.1	7.65E-06	2.315
1247-2	1000	29890	0.141	0.0777	59.78	9.78E-05	49.1	9.04E-06	2.174
1250-0	1000	31646	0.203	0.1116	63.29	5.73E-05	59.8	7.02E-06	2.228
1250-1	1000	61287	0.361	0.1377	122.57	5.31E-04	120.2	4.23E-05	1.898
1250-2	1000	89050	0.405	0.1855	178.10	5.55E-04	170.8	6.25E-05	1.822
Facebook	2888	2981	0.000	0.3871	2.06	9.19E-05	22.9	1.29E-05	3.867
Google Plus	23628	39194	0.004	0.4196	3.32	2.22E-06	35.2	7.50E-08	4.033
Hamster	2426	16631	0.231	0.3288	13.71	8.22E-07	19.9	3.01E-07	3.588
Twitter	23370	32831	0.021	0.4177	2.81	3.89E-08	10.0	8.02E-09	6.305

TABLE 6.3: Properties of simulated and real-world networks. Simulated networks are represented as numbers (https://www.kaggle.com/akandaashraf/virtualsoc1 to download the datasets). GCC - Global Clustering Coefficent , CCS - Clustering Coefficent Standard Deviation, CDM - Centrality Degree Mean, CCM - Centrality Closeness Mean, CDS - Centrality Degree Standard Deviation, CCS - Centrality Closeness Standard Deviation, AGPL - Avg Geodesic Path Length. The real-world networks Facebook and Google Plus are friendship based ego networks (Leskovec and Mcauley 2012; *Facebook (nips) network dataset KONECT* 2017; *Google+ network dataset – KONECT* 2017). Twitter (De Choudhury et al. 2010; *Twitter lists network dataset KONECT,* 2017) containing information on who follows whom on Twitter. The Hamster network is a full network containing friendships and family links between users of the website hamsterster.com (*Hamsterster full network dataset KONECT* 2017)



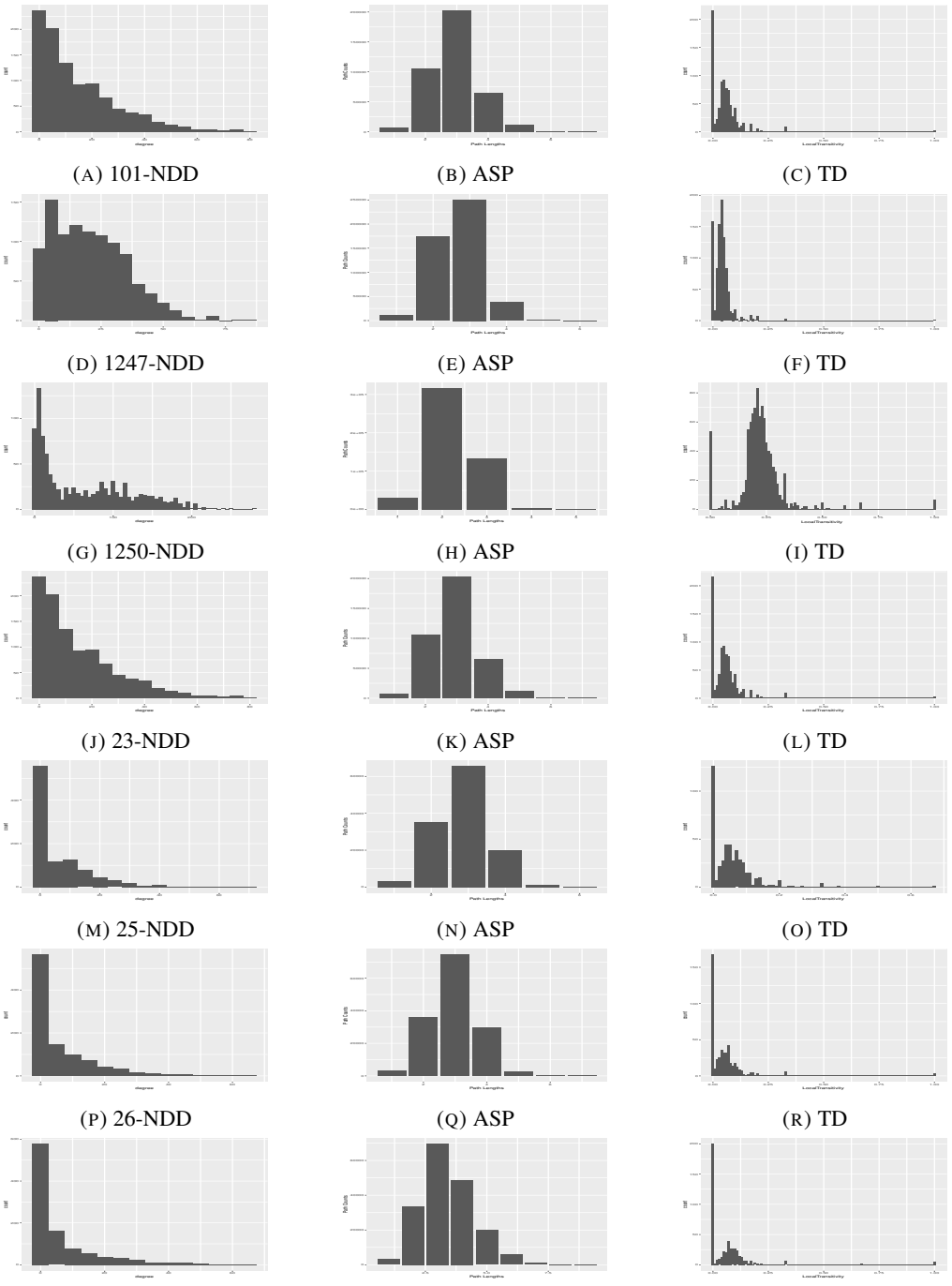
(A) Facebook-NDD (B) ASP (C) TD

(D) Google Plus-NDD (E) ASP (F) TD

(G) Hamster-NDD (H) ASP (I) TD

(J) Twitter-NDD (K) ASP (L) TD

FIGURE 6.4: Real-world network properties (distribution). NDD: node degree distribution, ASP: average shortest path, TD: local transitivity (clustering coefficient) distribution



(A) 101-NDD (B) ASP (C) TD

(D) 1247-NDD (E) ASP (F) TD

(G) 1250-NDD (H) ASP (I) TD

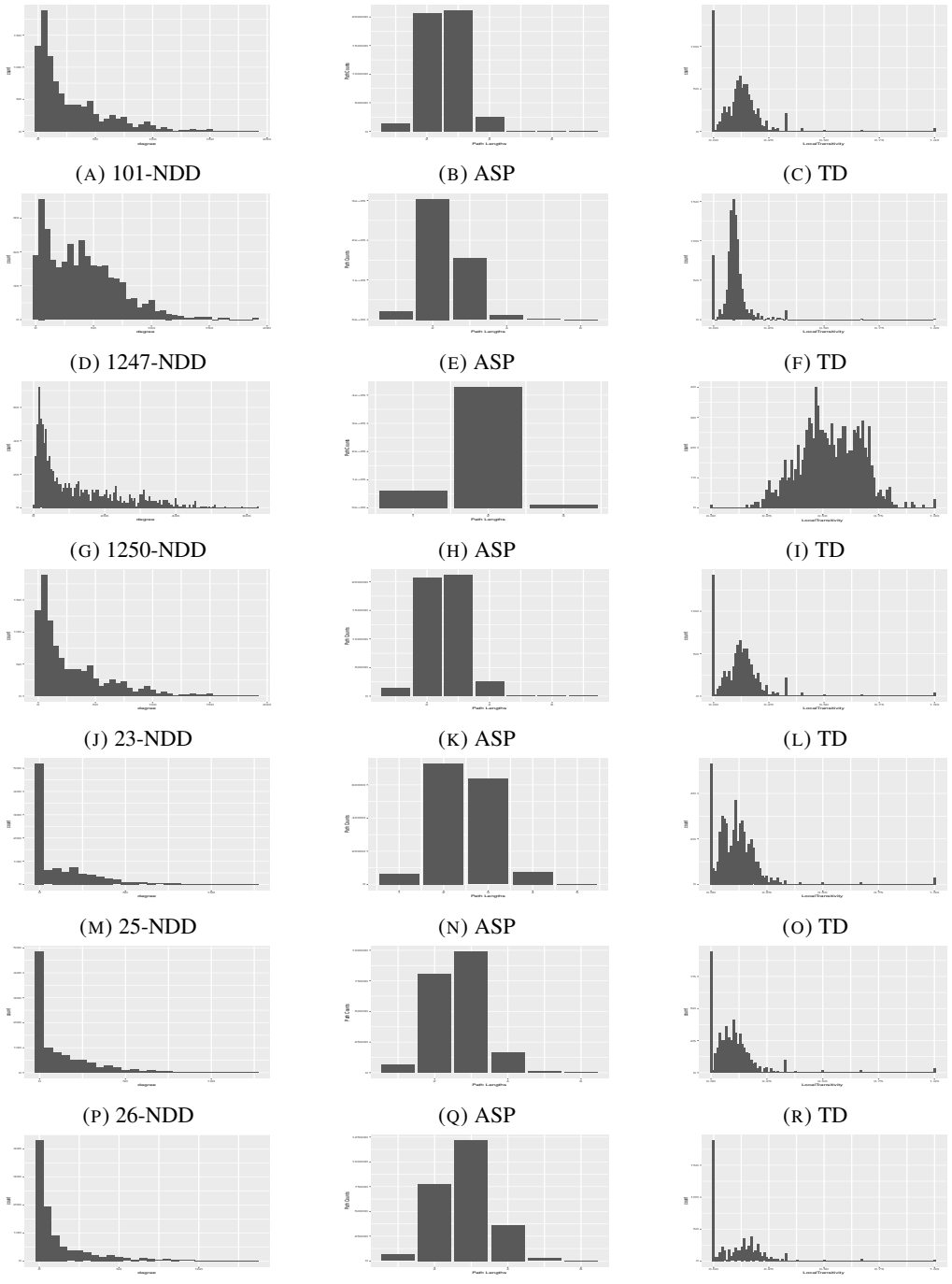
(J) 23-NDD (K) ASP (L) TD

(M) 25-NDD (N) ASP (O) TD

(P) 26-NDD (Q) ASP (R) TD

(S) 27-NDD (T) ASP (U) TD

FIGURE 6.5: Snapshot-0 of the selected simulated networks. Network properties (distribution). NDD: node degree distribution, ASP: average shortest path, TD: local transitivity (clustering coefficient) distribution



(A) 101-NDD (B) ASP (C) TD

(D) 1247-NDD (E) ASP (F) TD

(G) 1250-NDD (H) ASP (I) TD

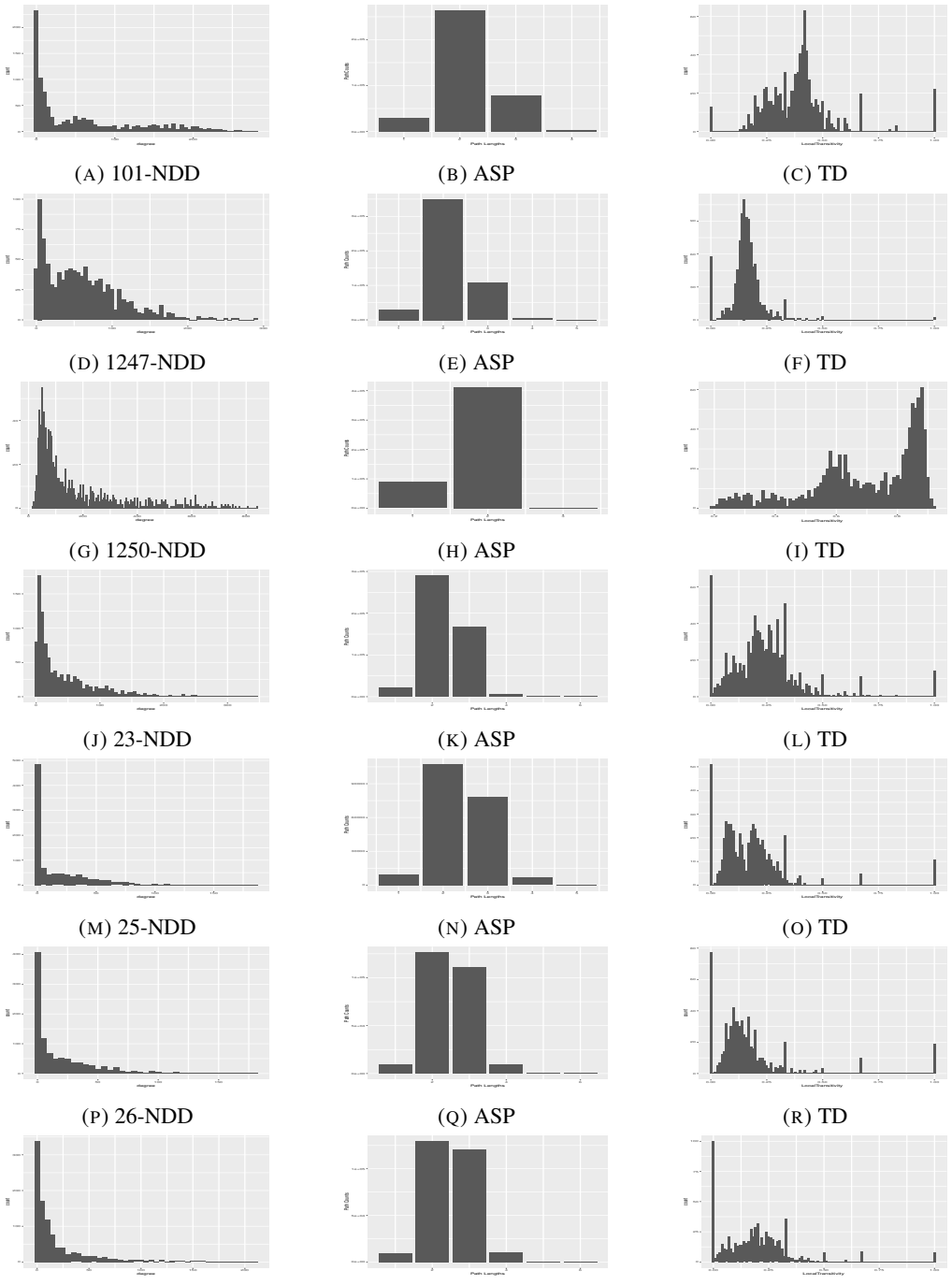
(J) 23-NDD (K) ASP (L) TD

(M) 25-NDD (N) ASP (O) TD

(P) 26-NDD (Q) ASP (R) TD

(S) 27-NDD (T) ASP (U) TD

FIGURE 6.6: Snapshot-1 of the selected simulated networks. Network properties (distribution). NDD: node degree distribution, ASP: average shortest path, TD: local transitivity (clustering coefficient) distribution



(A) 101-NDD (B) ASP (C) TD

(D) 1247-NDD (E) ASP (F) TD

(G) 1250-NDD (H) ASP (I) TD

(J) 23-NDD (K) ASP (L) TD

(M) 25-NDD (N) ASP (O) TD

(P) 26-NDD (Q) ASP (R) TD

(S) 27-NDD (T) ASP (U) TD

FIGURE 6.7: Snapshot-2 of the selected simulated networks. Network properties (distribution). NDD: node degree distribution, ASP: average shortest path, TD: local transitivity (clustering coefficient) distribution

6.7.3 Validation of the Feature and Topology Integration

In our simulation framework we provide a novel technique to simulate node features. The modelling of the node features needs to be validated separately from the topology of the network.

As we already use network models within the topology generation, the topology generation is supported by theoretical understanding of social networks and the integration of the features and the topology should be valid regardless of the topology of the network. Thus to validate the integration of the features and topology we select the first 10 networks, generated from our 1400 simulated networks and use a deep learning approach to validate this integration of the network topology and features. More specifically, we compare the predictive power of the graph convolutional networks (GCNs) with and without the inclusion of the node features. The higher predictive power of the GCNs models with the node features compared with the ones without the node features indicates a better integration of the node features within the simulated networks' topology. Similar approaches to validate causal relations between different components in a system are prevalent for a range of well established statistical causality detection techniques, namely the Granger Causality tests (Granger 1988; Bressler and Seth 2011; Ashraf 2016). Additionally, in the next chapter, we also provide four new deep learning models for node classification and evaluate their accuracy on two real-world social networks.

6.8 Chapter Summary

In this chapter, we proposed a comprehensive guide for simulating social networks. The simulation is facilitated by a set of equations which are supported by our current understanding of the three main network models. The algorithms developed in this research provide opportunities to generate a wide range of networks. Due to their complex nature, no two networks are exactly the same. Furthermore, these complex patterns also change over time. Our Mutation Algorithm (Algorithm 2) provides a novel approach to render such changes in complexity. Thus, the simulation approach is capable of modelling the evolution of networks, which are defined by social interactions. In the next Chapter 7 we validate the integration of the features and topology of the simulated networks.

Chapter 7

Deep Learning on Graphs

In Chapter 5 we have analysed a variety of widely used existing link prediction approaches along with the ones we our proposed. These link prediction algorithms can also be considered as missing link prediction algorithms (Liben-Nowell and Kleinberg 2007). In this chapter, this concept is utilised in the context of node classification. More specifically, the link prediction algorithms are used to improve the performance of a deep learning based model, Graph Convolutional Network (GCN), for classifying nodes in a network. Additionally, we have also developed and applied a novel strategy to validate our simulation framework presented in Chapter 7. This validation of the simulated network is also constructed using GCN.

In order to assess if the desired integration of features, labels, and topology is achieved, we measure and compare different trained model's predictability of the labels of the nodes. This comparison is done by designing different setups of the models such that, the models are able to perform predictions with the entire set of information (features, labels, and topology) as well as with partial information.

We validate the integration of features, labels, and topology of the simulated graphs through the measurement and comparison of the predictability of labels.

Furthermore, we propose four new variations of the GCN. Three of these variations are inspired from the node-similarity analysis that has been performed in Chapter 5, and the fourth one is inspired from the social network specific scenario for networks where not all the features play the same role. All the models have been tested on our simulated networks and compared with the original GCN.

7.1 How to validate simulation

Here we discuss the validation setup. In a network, the prediction of the label of a node, i.e. *sDNA*, can be performed via the following configurations of an ideal (what we mean by ideal is discussed later in this section) machine learning model:

1. Predictability of nodes' *sDNA*s with features combined with the graph topology
2. Predictability of nodes' *sDNA*s using features only
3. Predictability of node's *sDNA* using topology only

We can expect for an ideal machine learning model to fully capture and learn patterns both from the topological and feature based information from the network without over-fitting or being susceptive to the noise or stochasticity in the network. Needless to say, such an ideal model is not currently available in the real-world. However, we should at least use a machine learning model which can directly utilise both the topological and non-topological information, i.e. features.

In our case we use the GCN (Kipf and Welling 2017) to analyse *sDNA* predictability of the simulated networks, which can be regarded as one of the best models to directly combine both the topological and non-topological information of the graph (Li et al. 2018).

7.1.1 Graph Convolutional Networks (GCNs)

GCN is a multi-layer graph based neural network. In each layer, the features are multiplied with the topology of a graph in the spectral domain (i.e. symmetric normalised Laplacian matrix (Kipf and Welling 2017)). Weights of connections (edges/links by which the features of a node are passed, considered or summed) are learned using backpropagation. However, as most of the real-world social networks are not regular graphs, one single weight is learned for all links of a particular node.

The layer-wise propagation rule for the l^{th} layer is:

$$H^{(l+1)} = \sigma(GH^{(l)}W^{(l)}) \tag{7.1}$$

In Equation 7.1, $W^{(l)}$ are the trainable weight matrices for each layer. $H^{(0)} = X$ (the feature matrix) and G is a graph representative matrix that we will discuss in more detail in Section 7.1.2. G is fed in every layer of the model until the output layer. Finally, σ denotes a nonlinear activation function.

For this model, the receptive field grows with the depth of the network (Kipf and Welling 2017). In the first layer, only friends' features are considered, and in the second layer friend of friends' features are also considered, i.e. summed before passing through a non-linearity. This is because the summarised friends' information is already gathered in the first layer.

The direct translation from a graph to the structure of the neural network[1] is achieved via the graph representative matrix G. Symmetric normalised Laplacian matrix of the adjacency matrix A has been used in the original formulation of GCN, i.e. $G = L^{\tilde{sym}}$ (Kipf and Welling 2017).

$$\tilde{L}^{sym} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} \tag{7.2}$$

$$\tilde{A} = A + I_N \tag{7.3}$$

In Equation 7.3, the identity matrix (same dimensions as the adjacency matrix A) I_N adds self-connections for each of the nodes in A, $\tilde{D}$ is the degree matrix (a degree matrix is a matrix with the same dimensionality of A where the diagonal elements represent the degree of a graph and all other elements are set to zero) and $\tilde{A}$ is the adjacency matrix with added self-connections. The addition of self-connections facilitates incorporation of self-features of the nodes for better predictability. For example, a social network user's friends may give away his or her preference or class label (i.e. predictability based on the labels of the connected nodes), but additionally, his or her own features (i.e. self-connections in the graph) are also important to be considered to predict his or her preference.

In Equation 7.1, the main transformation to the neural network from a graph is performed through $G = L^{\tilde{sym}}$. If the adjacency matrix A in $L^{\tilde{sym}}$ (Equation 7.2) is replaced with a different representative function of the graph, the structure of the neural network itself will change. However, this does not change the input feature matrix X. As a result, this is not exactly data preprocessing technique but rather a change in the architecture of the neural network. We discuss this usage of different graph representatives later in Section 7.1.2.

Using the GCN we calculate the three mentioned setups for node label predictions in Section 7.1 by changing the propagation rule in Equation 7.1 as follows:

[1] In this paper when we talk about a graph (i.e. a social network) we mention it as a 'graph' or a 'network' but when we talk about a neural network it is written in its full form.

1. Prediction of nodes' *sDNA* with both the features and graph topology using propagation rule of Equation 7.1 for the first layer, where $G = \tilde{L}^{sym}$, and $H^0 = X$, where X is the feature matrix:

$$H^{(1)}_{(\Phi,\Delta,\mathbf{\Pi})} = \sigma(\tilde{L}^{sym} X W^{(0)}) \tag{7.4}$$

   This is the straightforward GCN model proposed by Kipf and Welling (2017). Here, the graph representative $G = \tilde{L}^{sym}$ is fed in every layer of the model, but the feature matrix X is fed only in the first layer.

2. Prediction of nodes' *sDNA* with features excluding graph topology with the following propagation rule:

$$H^{(1)}_{(\Phi)} = \sigma(I_A X W^{(0)}) \tag{7.5}$$

   In this first layer propagation rule in Equation 7.5, I_A is the identity matrix with the dimension of the adjacency matrix A. I_A is fed into the model until the output layer [2]. Thus, only features of each node are considered and the graph topology does not play any role for label or *sDNA* predictions.

3. Prediction of node's *sDNA* excluding the features but solely with the graph topology:

$$H^{(1)}_{(\Delta,\mathbf{\Pi})} = \sigma(\tilde{L}^{sym} I_X W^{(0)}) \tag{7.6}$$

   In this propagation rule in Equation 7.6, I_X is the identity matrix with the dimensions of the feature matrix X [2]. As a result, only the graph topology is considered, and features do not play any role in the model. Here, the graph representative $G = \tilde{L}^{sym}$ is fed in each layer of the model until the output layer of the model, however $H^0 = I_X$ is fed only in the first layer of the model.

We assume that during the simulations, the first setup will produce more accurate results than the remaining two. This hypothesis is represented through the following inequality:

$$\begin{aligned}\Big(Acc(H_{(\Phi,\Delta,\mathbf{\Pi})}) > Acc(H_{(\Phi)})\Big) \wedge \Big(Acc(H_{(\Phi,\Delta,\mathbf{\Pi})}) > \\ \big(Acc(H_{(\Delta,\mathbf{\Pi})}) \vee Acc(H_{\overline{(\Delta,\mathbf{\Pi})}})\big)\Big)\end{aligned} \tag{7.7}$$

where Acc is the test accuracy of the trained neural network model for four different setups, based on four different propagation rules in Equations 7.4, 7.5, 7.6, and 7.17 (Equation 7.17 is

[2] the identify matrix provided in here just for the purpose of comparison and how the implementation of the model works but for obvious reason this identify matrix doesn't have any mathematical significance

discussed later in Section 7.2). However, the assumption of GCN is that for a l^{th} layer of the model, only the l^{th} order neighbourhood nodes are influential (Kipf and Welling 2017; Li et al. 2018). To work around this problem, we develop a strategy of replacing the adjacency matrix A in the Laplacian transformation in Equation 7.2 graph representative function G, with three different existing node-similarity measures. In social networks, not all connected nodes have the same influence and in fact, some non-directly connected nodes in the graph may have greater influence over a node in question than the directly connected ones. As a result, usage of the adjacency matrix A as a graph representative G may not always entail the best performance of the neural network.

7.1.2 Node-similarities as Graph Representatives for GCN

In social networks, the adjacency matrix represents direct links between nodes. In GCN the features propagate through those links. Thus, a node's label is predicted by utilising patterns on the surrounding connected nodes' features and labels. However, in social networks, not all the connections of a given node have same or even similar effect on this node. It can be assumed e.g. that the influence that one node has on its neighbour will increase with the number of their mutual friends. In a similar way, it may happen that a friend of a friend of node i can influence node i more than a directly connected node (a not influential node, e.g. does not have any common friend with the node i). As a result, this effectively changes the representation of the network so one can incorporate these relationship characteristics as a form of social node-similarity-based matrix for the GCN. One way to extract and represent these types of social relationship (not necessarily direct ones) strengths and other information between nodes is to use a matrix which describes the similarity between nodes instead of an adjacency matrix. For example, the Katz similarity measurement considers the number of all direct paths from node i to j (Katz 1953). Thus, more mutual friends would result in a higher number of paths, resulting in a higher value of the Katz score. In this study, we replace the adjacency matrix A with the three different types node-similarity matrices, $\hat{A}$ as they encompass richer information about underlying structure than traditional adjacency matrix. Following are the three node-similarity measures we have considered:

- **Katz,** which considers the number of all the paths from node i to j (Katz 1953). The shorter paths have bigger weight (i.e. are more important), which is damped exponentially with the increase of the path length and the β parameter (where A is the adjacency matrix and see Chapter 2, Section 2.3.4 for more details on the β parameter):

$$Similarity(i, j) = \beta A + \beta^2 A^2 + \beta^3 A^3 + \cdots \tag{7.8}$$

The above similarity in Equation 7.8 will result in the following graph representative $G_{katz} = \tilde{L}^{sym}_{katz}$

$$\tilde{L}^{sym}_{katz} = \tilde{D}^{-\frac{1}{2}} \tilde{A}_{katz} \tilde{D}^{-\frac{1}{2}} \tag{7.9}$$

$$\tilde{A}_{katz} = \hat{A}_{katz} + I_N \tag{7.10}$$

In Equation 7.9 and 7.10, $\hat{A}_{katz}$ represents the nodes-similarity matrix (for all possible links) based on Katz similarity measure from Equation 7.8

- **Rooted PageRank (RPR)** is used by search engines to rank websites. In graph analysis it ranks nodes by the probability of each node being reached via random walk on the graph (Brin and Page 2012). The $Similarity(i, j)$ is calculated using the stationary probability distribution of the degree matrix D in a random walk. The random walk returns to i with probability α at each step, moving to a random neighbour with probability $1 - \alpha$. This results in the following graph representative $G_{RPR} = \tilde{L}^{sym}_{RPR}$:

$$\tilde{L}^{sym}_{RPR} = \tilde{D}^{-\frac{1}{2}} \tilde{A}_{RPR} \tilde{D}^{-\frac{1}{2}} \tag{7.11}$$

$$\tilde{A}_{RPR} = \hat{A}_{RPR} + I_N \tag{7.12}$$

In Equation 7.11 and 7.12, $\hat{A}_{RPR}$ represents the nodes-similarity matrix (for all possible links) based on RPR.

- **Graph Gravity (GG),** Inspired by the Newton's law of universal gravitation, this node-similarity measure uses degree centrality as the mass of the nodes, while the lengths of shortest paths between them act as distances (Wahid-Ul-Ashraf et al. 2017, 2019). The above analogy leads to the following formula for calculating the score between two nodes:

$$Similarity(i, j) = \frac{C_D(i) \times C_D(j)}{SP(i, j)^2}, \tag{7.13}$$

where C_D denotes the degree centrality, SP is the shortest path. Node-similarity in Equation 7.13 will result in the following graph representative $G_{GG} = \tilde{L}^{sym}_{GG}$:

$$\tilde{L}^{sym}_{GG} = \tilde{D}^{-\frac{1}{2}} \tilde{A}_{GG} \tilde{D}^{-\frac{1}{2}} \tag{7.14}$$

$$\tilde{A}_{GG} = \hat{A}_{GG} + I_N \tag{7.15}$$

In Equation 7.14 and 7.15, $\hat{A}_{GG}$ represents the nodes-similarity matrix (for all possible links) based on GG in Equation 7.13.

For all the above three node-similarity measures, each $\hat{A}$, represents the nodes-similarity matrix (only for all possible links) which has been preprocessed and reconfigured further which is discussed in Section 7.2.2.

7.1.3 Weighted Feature Matrix

GCN is a powerful model for node classification, and it has been shown to perform well even only with the graph topology, i.e. without the feature matrix Kipf and Welling (2017). The reason for such a good predictability without the features could be due to two reasons. Firstly, when our focus is on node classification for graph-structured datasets, the preferred features of the nodes should be reflected in the topology of the graph as we have discussed in our simulation process in Section 6.2. The fact that these features are encoded in the topology may result in a good predictability even when the features are not directly considered in the model. Additionally, this better predictability based on feature only or topology only may vary from node to node. For some nodes, the topology only may have better predictability when compared with the node's feature. This could be due to the fact that topological position of a node overshadows the importance of the features.

Secondly, the good performances solely based on topology could be because, similar to real-social network users, we have defined our *sDNA* for nodes such that it results in some of the features of other nodes being preferred and some others not (Section 6.2). In other words, not all the features play similar roles when it comes to the predictability of the *sDNA*. As a result, in the entire graph, some of the features may be disliked or not preferred by the majority of the nodes when forming graph connections. This is why an additional learnable common weight for a particular feature for all the nodes may result in better predictability. In our analysis, we have found that adding this additional weight, which defines the weight for each of the features for all the nodes, seems to perform best, and this is what we present in Section 7.3. To introduce this relative importance of features we use one additional weight vector in the GCN model. We use a common weight for a particular feature for all the nodes. If we have a network with $1,000$ nodes and 50 different features each, for each feature of all the $1,000$ nodes a common (i.e. across all the nodes) weight is used to learn the strength of each feature. This additional feature weight matrix is the size of the number of features and is used only in the first layer of the model. Hence all the input features, X are weighted before passing to the hidden layers.

This additional weight vector results in the following first layer propagation rule based on the Equation 7.1:

$$H^{(2)} = \sigma(G((\mathbf{1}S) \odot X)W^{(1)}) \tag{7.16}$$

where $S^{1\times|f|}$ is the matrix containing the unbounded learnable parameters to define strength of the feature matrix $X^{|N|\times|f|}$. $1^{1\times|f|}$ is an all one matrix. The operator $\odot$ defines the Hadamard product (i.e. element-wise product) between the feature matrix X and the dot product of 1 and S.

7.2 Experimental Setup

In the experiments, we simulate 30 social networks, with $1,000$ nodes each. Each of the networks has four different types of *sDNA*s with 250 nodes subscribing to a single type. We take three different snapshots of the same network, resulting from an initial 10 networks to a total of 30 networks (i.e. three snapshots of the same network). Each of the nodes has 50 features, each set of features of a node is generated from a uniform distribution. All the variables for the *sDNA* (described in Figure 6.1 and Section 6.2) are also sampled from uniform distributions. The input parameters used to simulate these 30 networks are given in Appendix A, Table A.3. Additionally, we also report the properties of the 30 simulated networks in Appendix A, Table A.4. Social networks tend to have high clustering coefficient, and in our networks we observe the phenomenon (Barabasi and Oltvai 2004; Watts and Strogatz 1998; Lee et al. 2014; Mislove et al. 2007). In our simulation, we have considered to replicate the dynamics of the modern online, friendship-based social networks. Figure 7.1 shows the approximated clustering coefficient of a number of modern online social networks as reported in (Lee et al. 2014). Based on this, the global clustering coefficients of our simulated networks, are well within the range of the real-world online social networks. Additionally, we present the degree, path length, and transitivity distribution of the simulated networks. In Figure 7.2 the first snapshot, Figure 7.3 the second and, in Figure 7.4 the third snapshot of the simulated network's distributions are presented. It is apparent that as we move from the first to the third snapshot the networks' average geodesic path length becomes smaller, i.e. the small-world phenomena. In the final snapshot many of the networks become so small that all nodes can be accessed via one mutual friend which we see in the real-world networks we have used earlier in Chapter 5, Figure 5.2.

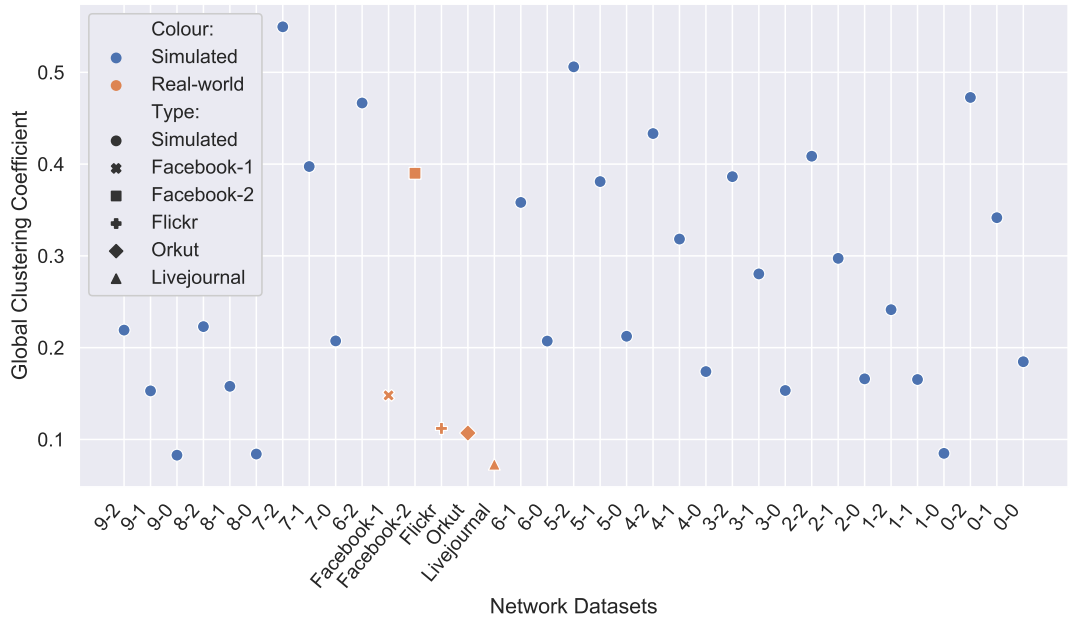
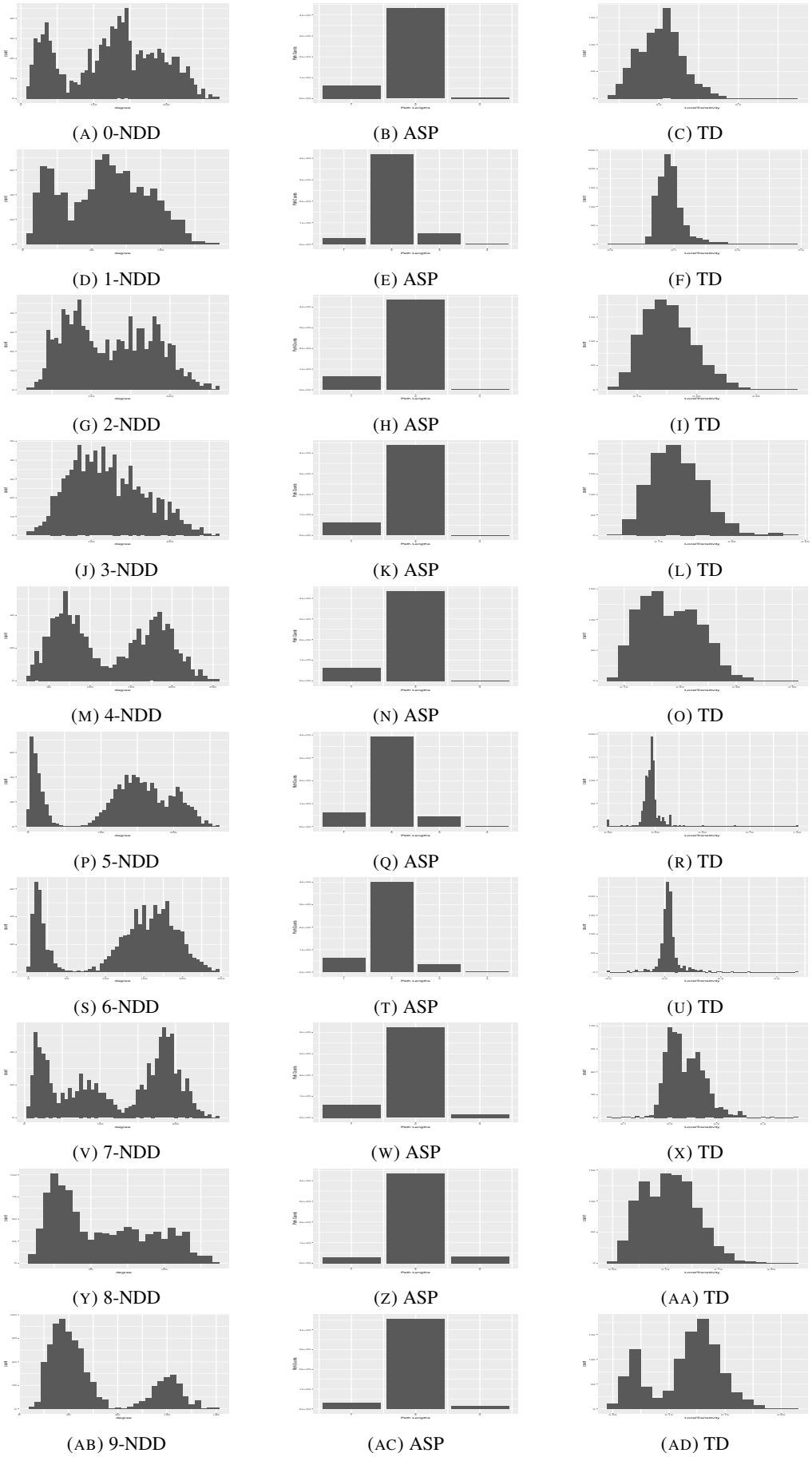


FIGURE 7.1: Global clustering coefficient for the simulated network datasets and real-world datasets (the real-world network's global clustering coefficient measures are measured by Lee et al. (2014))

Another classical understanding of social networks, is the power-law degree distribution (Barabási and Albert 1999). However, we have discussed in Chapter 6, Section 6.7, that we do not compare the simulated networks in terms of their scale-freeness to deem if they are close to real-world networks.



(A) 0-NDD (B) ASP (C) TD

(D) 1-NDD (E) ASP (F) TD

(G) 2-NDD (H) ASP (I) TD

(J) 3-NDD (K) ASP (L) TD

(M) 4-NDD (N) ASP (O) TD

(P) 5-NDD (Q) ASP (R) TD

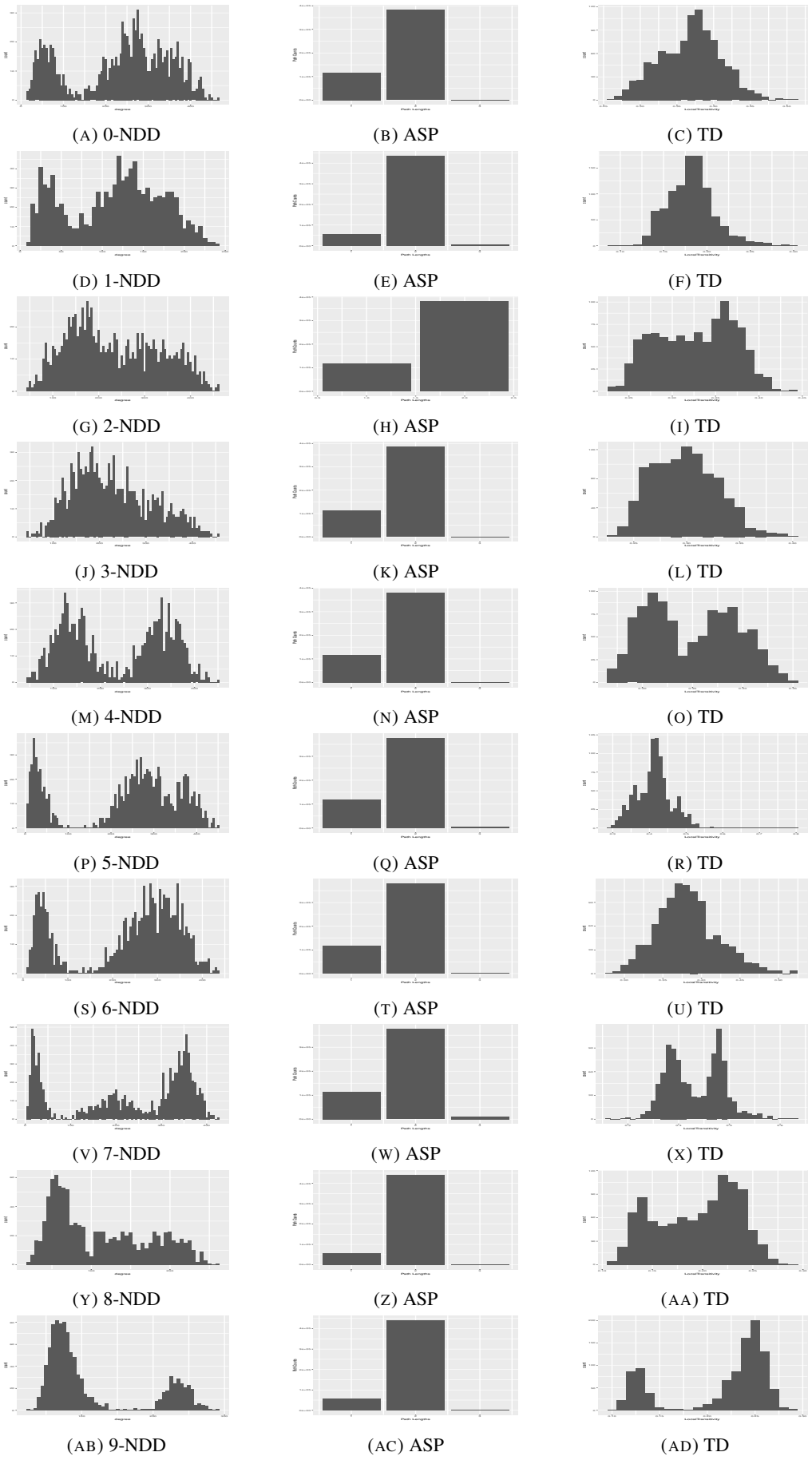
(S) 6-NDD (T) ASP (U) TD

(V) 7-NDD (W) ASP (X) TD

(Y) 8-NDD (Z) ASP (AA) TD

(AB) 9-NDD (AC) ASP (AD) TD

FIGURE 7.2: Snapshot-0 of the selected simulated networks. Network properties (distribution). NDD: node degree distribution, ASP: average shortest path, TD: local transitivity (clustering coefficient) distribution



(A) 0-NDD (B) ASP (C) TD

(D) 1-NDD (E) ASP (F) TD

(G) 2-NDD (H) ASP (I) TD

(J) 3-NDD (K) ASP (L) TD

(M) 4-NDD (N) ASP (O) TD

(P) 5-NDD (Q) ASP (R) TD

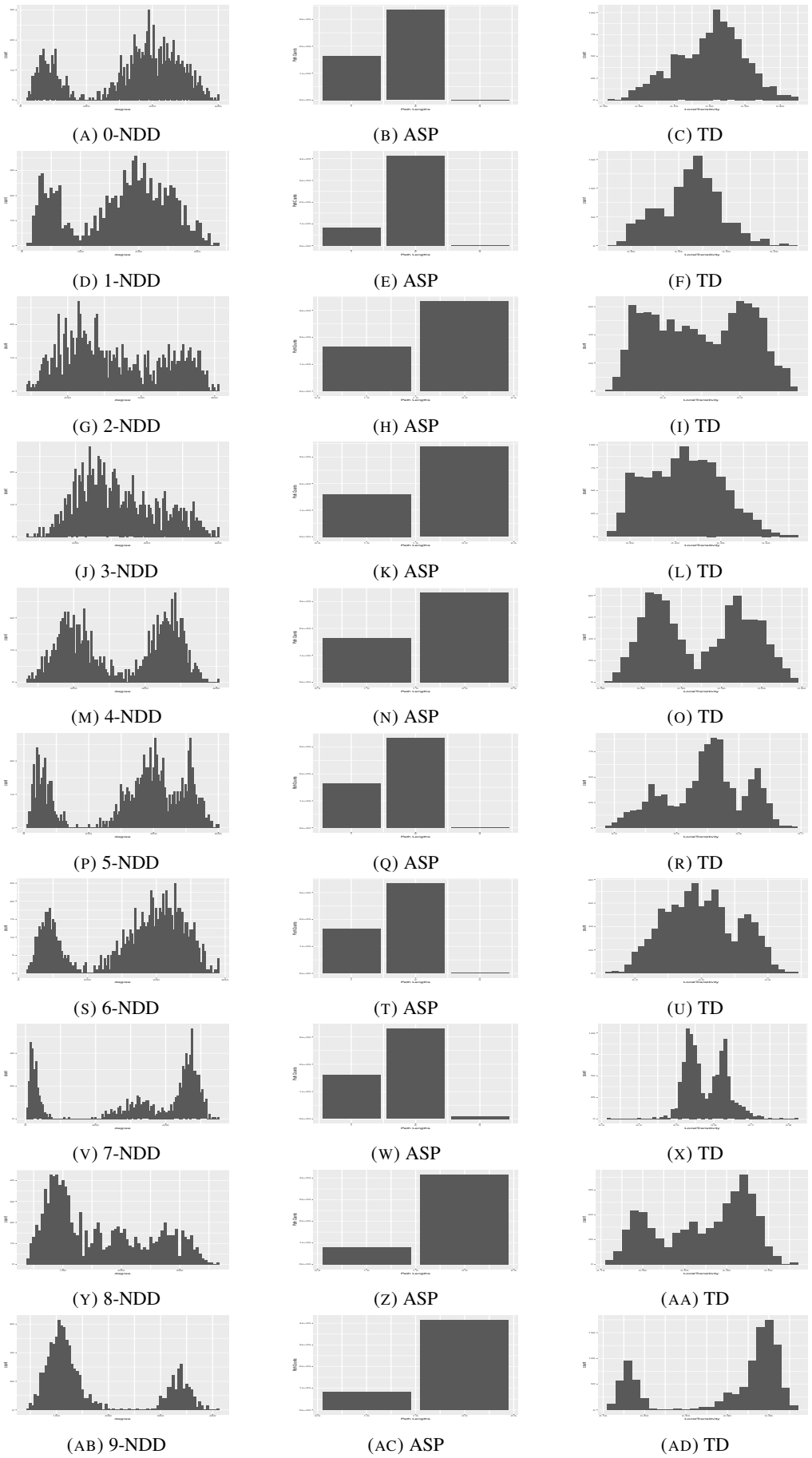
(S) 6-NDD (T) ASP (U) TD

(V) 7-NDD (W) ASP (X) TD

(Y) 8-NDD (Z) ASP (AA) TD

(AB) 9-NDD (AC) ASP (AD) TD

FIGURE 7.3: Snapshot-1 of the selected simulated networks. Network properties (distribution). NDD: node degree distribution, ASP: average shortest path, TD: local transitivity (clustering coefficient) distribution



(A) 0-NDD (B) ASP (C) TD

(D) 1-NDD (E) ASP (F) TD

(G) 2-NDD (H) ASP (I) TD

(J) 3-NDD (K) ASP (L) TD

(M) 4-NDD (N) ASP (O) TD

(P) 5-NDD (Q) ASP (R) TD

(S) 6-NDD (T) ASP (U) TD

(V) 7-NDD (W) ASP (X) TD

(Y) 8-NDD (Z) ASP (AA) TD

(AB) 9-NDD (AC) ASP (AD) TD

FIGURE 7.4: Snapshot-2 of the selected simulated networks. Network properties (distribution). NDD: node degree distribution, ASP: average shortest path, TD: local transitivity (clustering coefficient) distribution

For all the models, we have four graph convolutional layers. All layers except the output layer use rectified linear units (ReLU) as nonlinear activation functions. The output graph convolutional layer contains softmax activation and categorical cross entropy loss is calculated for the four types of *sDNA*s or node labels. Each of the layers, except for the output layer contains 32 units of neurons, and the output layer has $1,000$ units, the same as the number of nodes that need to be classified. Finally, we used Adam optimiser, a first-order gradient-based algorithm for our differentiable neural network model to learn the weights (i.e. to optimise the loss function). Each and every model is evaluated with the same setup. On every network, 10-fold cross-validation is performed and the average accuracy is reported. Additionally, the standard deviation of the accuracy is reported for the best accuracy and the original GCN in Table 7.3. All the hyperparameters are kept fixed for all the models. We have used the learning rate of 0.01, L2 kernel regularisation (i.e. weight decay) for all the hidden layers with the decay rate of 0.0005, and a dropout layer after each hidden layer with $p = 0.5$, i.e 50% of the randomly selected neurons are trained in each training iteration.

Model	Description	Graph representative G	Eq.
FTVanilla	Original GCN, feature + Toplogy	$G = L^{\tilde{sym}}$	7.1
T	Original GCN, Topology only	$G = L^{\tilde{sym}}$	7.6
TLR	Original GCN, Topology only	$G = L^{\tilde{sym}}$	7.17
F	Original GCN, feature only	N/A	7.6
FTKatz	Feature + Katz based Toplogy	$G_{katz} = \tilde{L}^{sym}_{katz}$	7.9
FTRPR	Feature + RPR based Toplogy	$G_{RPR} = \tilde{L}^{sym}_{RPR}$	7.11
FTGG	Feature + GG based Toplogy	$G_{GG} = \tilde{L}^{sym}_{GG}$	7.13

TABLE 7.1: Models used along with the original GCN. All of the models with features are trained twice, once with the weighted feature matrix in Equation 7.16 and once without

In Table 7.1, for the topology only model, *T* (Equation 7.6), the weight matrix contains more trainable parameters compared with the the model in *FTVanilla* (Equation 7.4). This is because we have 1000 nodes per network with 50 features each. As a result for the model with both the topology and feature matrix model, *FTVanilla* the dimension of the first layer weight matrix $w^{(0)}$ needs to be 50×32, where 32 is the hyperparameter for the number of units we consider in all the models, i.e. $\tilde{L}^{sym,1000\times1000}X^{1000\times50}W^{(0),50\times32}$, and the resulting matrix has a dimension of 1000×32, while the output from the first layer has a dimension of 1000×32. Whereas for the topology only *T*, in Equation 7.6, where the feature matrix is only an identity matrix, I_X, the weight matrix $w^{(0)}$ is directly multiplied with the graph representative, i.e. the graph topology, $\tilde{L}^{sym,1000\times1000}$. As a result the dimension of the weight matrix is a lot higher (1000×32), i.e. $\tilde{L}^{sym,1000\times1000}W^{(0),1000\times32}$, and the resulting output from the first layer has a dimension of

1000×32. As we can see there are more trainable parameters in the *T* model compared with *FTVanilla*, i.e. 50×32 vs 1000×32. If we were to compare both of the models' performance, *T* vs *FTVanilla*, to test if the Inequality 7.7 holds as a validation of the feature and topology integration process, the total number of trainable parameters for both models should be as close as possible. To make both the models comparable, we introduce another setup for the topology only model to keep the number of parameters at the similar level to the model using both the features and topology.

$$H^{(1)}_{(\Delta,\Pi)} = \sigma(\tilde{L}^{sym} I_X W^{(a0)} W^{(b0)}) \tag{7.17}$$

In Equation 7.17, weight matrix $W^{(0),1000\times32}$ is split into two matrices $W^{(a0)_{1000\times1}}$ $W^{(b0)_{1\times32}}$ to keep the number of trainable parameters roughly in line with the *FTVanilla* model.

7.2.1 Citation Networks

In addition to testing the models on the simulated network datasets, we have also tested our models on two of the real-world citation network datasets, *Cora* and *Citeseer* (Kipf and Welling 2017; Yang et al. 2016). It's worth re-iterating (detailed discussion in Chapter 6) that node attributes in the citation networks do not have the same semantics as in the friendship-based social networks. However, high-quality real-world datasets, with both features and node attributes are very rare to obtain. We have thus used our simulated network datasets, for which our proposed variations of the GCN models are designed. However, we test our proposed models on the citation networks to compare the accuracy with the original GCN (i.e. $FTVanilla$) model.

We closely imitate the experimental setup of Kipf and Welling (2017). The citation network contain sparse bag-of-words as features and each of the papers are considered as nodes and the edges are citation links (Sen et al. 2008). The citation links are considered as undirected links between nodes in this experiment, similar to the experiment of (Kipf and Welling 2017). The number of classes for the *Cora* dataset is seven and for the *Citeseer* it is six.

Network	Nodes	Edges	Classes	Features	Label rate
Citeseer	3,327	4,732	6	3,703	0.036
Cora	2,708	5,429	7	1,433	0.052

TABLE 7.2: Two citation networks' statistics (Kipf and Welling 2017)

In Table 7.2, the label rate is the number of labelled nodes that are used in training divided by the total number of nodes in the network. In the case of citation networks, we imitate the similar

setup by (Kipf and Welling 2017), where not all the node labels are used during the training steps (i.e. semi-supervised classification). A two layer GCN and our versions of the GCN from Table 7.1 are used for all the experiments. Each of the models is trained for a maximum of 200 epochs with an early stopping threshold of 10 (i.e. if the loss does not decrease after 10 iteration the training is stopped). The learning rate used for these models is 0.01 and 16 units within the hidden layers are used. Additionally, a dropout rate of 0.5 and L2 regularisation set to 0.0005 are used during the training steps. For each of the models, weights are randomly initialised 46 times (the random seeds are available in Section A.9.1, Appendix A) and we report the mean accuracy and standard deviation in Table 7.4. Additionally, in Table 7.5, we randomly split each of the networks into test and training sets a total of 100 times (with the same label rate in Table 7.2 and by Kipf and Welling (2017)).

7.2.2 Augmented Node-similarity Matrix

For $\hat{A}_{GG}$ (Equation 7.15), the similarity scores for all the non-existing links are calculated and then all the scores are normalised between zero and one. Afterwards, the adjacency matrix A is summed with the calculated scores for all the non-existing links. As a result, all the existing links for $\hat{A}_{GG}$ has a value of one and for the non-existing links, the value ranges from zero to one. For $\hat{A}_{katz}$ and $\hat{A}_{RPR}$, the path-based similarity scores are calculated for all possible links. For all the networks, to calculate Katz score, with the highest exponent of five for the adjacency matrix A (i.e. A^5 in Equation 7.8) and the $\beta = 0.005$ is used. As for the RPR, the α parameter is set to 0.85. For each of the calculated similarity matrices ($\hat{A}_{GG}$, $\hat{A}_{katz}$ and $\hat{A}_{RPR}$), the row is normalised for each of the non-zero elements using the $L2$ norm. Moreover, on the similarity-based adjacency matrices (i.e. $\hat{A}_{katz}$, $\hat{A}_{RPR}$, and $\hat{A}_{GG}$), several thresholds are used. The thresholds are applied on the $L2$ row normalised matrices. The thresholds are set in a way that, if the value in the similarity-based matrix is less than or equal to the first threshold then it is set to zero. Whereas for the second threshold point, if the value is greater than the threshold, it is set to one. If the thresholds are set as zero and one respectively, then none of the values is changed in the matrix. Also, for some set of thresholds, if they are not the same, the elements in the matrix which are in between the two thresholds, are unaltered in the matrix. The sets of thresholds are selected based on empirical analysis, i.e. cross-validation accuracy of the model. However, we also select a threshold based on the mean value of the elements of the matrix. The mean value threshold hold is applied such that, if a non-zero element in the matrix is less than or equal to the mean value then it is set to zero and one otherwise.

In GCN, for the l^{th} layer, only the l^{th} path length neighbouring nodes are considered (Kipf and Welling 2017). Thus, it limits the scope of the receptive field of the node in each layer and also the maximum receptive field is limited by the maximum number of layers used in the model. This limitation has also been pointed out in the paper where GCN was first introduced (Kipf and

Welling 2017). However, using node-similarity measures along with the augmentation process we describe here allows the model to consider a three-path distant node j even in the first layer (i.e. a direct connection) for the classification of the node i, assuming that they have a high node-similarity score. As a result, this augmented node-similarity measure solves the limitation of layer-wise node-neighbourhood dependencies for the GCN. Similar augmentation of the network is also used by (Bahulkar et al. 2018) for better community detection algorithm.

7.3 Results and Discussion

In Figure 7.5, we show accuracy for all the models that we have tested on 30 simulated networks. All the results are 10-folds cross-validated and average accuracy is reported. In Figure 7.5 and Table 7.3, we observe that according to the hypothesis of Equation 7.7, the accuracy of the model which uses node features only, i.e. F, is very low. In fact, the predictability is not better than random chance (the accuracy is around 0.25 and we have four equally represented labels or *sDNA* types to predict). Additionally, from Figure 7.5 and Table 7.3 we can see that for the majority of the datasets (except only three networks) models which utilise both the topology and features of the graph perform better than the two other setups where topology and features are considered independently. When only topology is used (i.e. T), the model T performs the best in three networks. Two of them are a third snapshot(i.e. the 3rd run of the Algorithm 1) of a network (networks 1, 2 and 6-2), and the third one is the second snapshot of the network (2-1). This can be due to the fact that as we run Algorithm 1 multiple times, the patterns of preferences get encoded within the network topology so that the topology only model performs better. This is something we also expect in real world networks i.e. as people make more connections, their tendency towards who they choose to connect with become eminent.

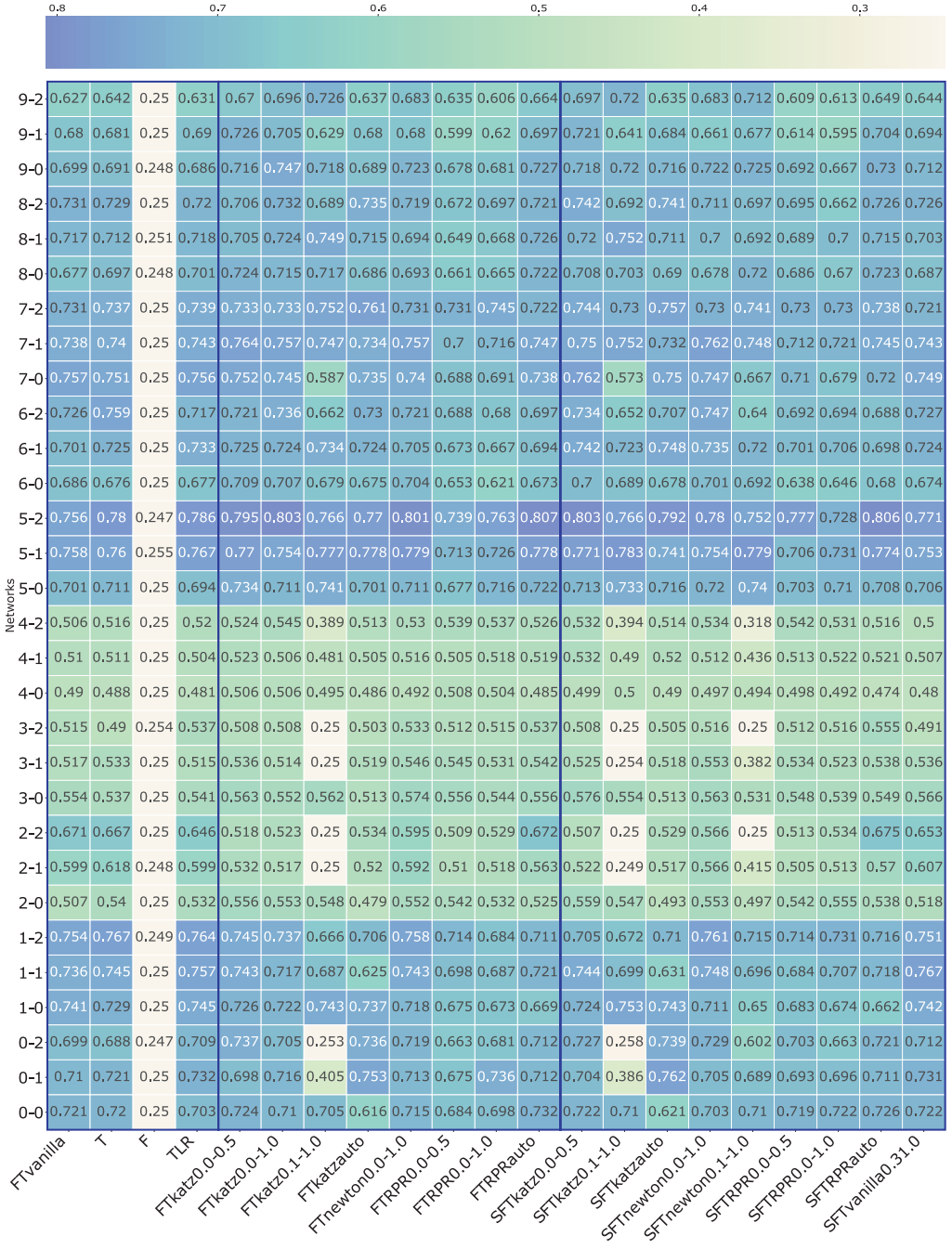


FIGURE 7.5: Node label prediction accuracy (in fractions i.e. 0.7 implies 70% accuracy and the top horizontal bar show colour map for accuracy) from different Models (average from 10-fold cross-validation), networks are in X axis and models in Y axis. Models written as, F-feature only, T - topology only, FT - both the feature and topology, $FTvanilla$ - the original GCN. Models with S in the right box (the blue outlined boxes) represents if an additional feature weight matrix in the first layer (Equation 7.16). The left box shows results for models that use graph representative $G = L^{\tilde{sym}}$, Equation 7.1 (i.e. adjacency matrix). The middle box uses $G_{NS} = \tilde{L}^{sym}_{NS}$, where NS is a node-similarity ($Katz$, RPR, and GG) measure with different thresholds (Equation 7.9, 7.11 and , 7.13). Similarity-based G is preprocessed based on Section 7.2.2. The preprocessing threshold $auto$ implies automatic selection of a threshold based on the mean value of the $\hat{A}$ (Section 7.2.2). All the networks are represented in terms of snapshots. For example, 0-0, is the first network's first snapshot, 0-1 is the first network's second snapshot.

Amongst methods using node-similarity matrices instead of the node adjacency matrix, we see that the choice of threshold seems to have a significant effect on the model-performance. However, we can also see that using the mean value of the L2 normalised node-similarity matrix as a threshold (described in Section 7.2.2) performs quite well. In fact, on seven networks with the setup of using mean value as a threshold on the node-similarity matrix (discussed in Section 7.2.2) outperforms all the other models (Table 7.3). The models with the mean value as thresholds are written as 'auto' in Table 7.3.

If we do not consider differences between thresholds and usage of the vector S on the node-similarity measures, $Katz$ significantly outperforms the original GCN (i.e. $FTvanilla$) and two other node-similarity measures (RPR and GG) based on the results in Table 7.3. On five networks, RPR performs the best and GG on one network. However, on the basis of average best-performing models on all the datasets from all the 20 different models, following models performs best: (1) $FTGG0.0 - 1.0$, (2) $SFTkatz0.0 - 0.5$, and (3) $FTkatz0.0 - 0.5$. Thus, on average, GG performs best across all the datasets.

The results also show that, the usage of a trainable parameter S based on Equation 7.16 gives us a better model for many datasets than when not using it. In 15 out of 30 datasets, using S on the first layer of the model outperforms the other models (Table 7.3). Furthermore, models with S which perform best are mainly not the original GCN but the node-similarity-based models, except for one dataset. However, this may not imply that the use of additional weights in the first layer based on Equation 7.16 only performs well on node-similarity-based models. This is because the usage of node-similarity may have better predictability in general than the adjacency matrix.

From Figure 7.5 and Table 7.1 we can see that the performance of a node-similarity-based model varies depending on the network the model is trained on. This is because all the networks are simulated with different rules, and no two networks are exactly the same. We can expect to see the same in real-world networks as well. Thus, the choice of a node-similarity method could be based on empirical analysis. However, one may also use the mean value of the normalised node-similarity matrix, especially with GG as we have discussed earlier in this section.

The results in Figure 7.5 show that we can achieve high accuracy (in fact higher than using only the adjacency matrix) on node classification when a node-similarity-based graph topology is used. This is particularly useful for very dense networks. The training time that is required for a very dense network is extremely high for GCN. Many real-world datasets, such as face-to-face interaction networks, tend to be very dense. Thus, the similarity-based matrices can be used (with a suitable threshold to reduce the number of connections as per Section 7.2.2) in such scenario to reduce training time.

Net.	FTvanilla((Kipf and Welling 2017)) Acc (SD)	Max Acc (SD)	Max Acc Model	Net.	FTvanilla((Kipf and Welling 2017)) Acc (SD)	Max Acc (SD)	Max Acc Model
0-0	0.721 (0.011)	0.732 (0.007)	FTRPRauto	5-0	0.701 (0.028)	0.741 (0.005)	FTkatz0.1-1.0
0-1	0.71 (0.022)	0.762 (0.011)	SFTkatzAuto	5-1	0.758 (0.010)	0.783 (0.005)	SFTkatz0.1-1.0
0-2	0.699 (0.032)	0.739 (0.012)	SFTkatzAuto	5-2	0.756 (0.044)	0.807 (0.004)	FTRPRauto
1-0	0.741 (0.013)	0.753 (0.012)	SFTkatz0.1-1.0	6-0	0.686 (0.009)	0.709 (0.010)	FTkatz0.0-0.5
1-1	0.736 (0.034)	0.767 (0.007)	SFTvanilla	6-1	0.701 (0.030)	0.748 (0.018)	SFTkatzAuto
1-2	0.754 (0.027)	0.767 (0.033)	T	6-2	0.726 (0.035)	0.759 (0.015)	T
2-0	0.507 (0.045)	0.559 (0.009)	SFTkatz0.0-0.5	7-0	0.757 (0.010)	0.762 (0.008)	SFTkatz0.0-0.5
2-1	0.599 (0.044)	0.618 (0.074)	T	7-1	0.738 (0.013)	0.764 (0.010)	FTkatz0.0-0.5
2-2	0.671 (0.017)	0.675 (0.014)	SFTRPRauto	7-2	0.731 (0.015)	0.761 (0.013)	FTkatzAuto
3-0	0.554 (0.018)	0.576 (0.009)	SFTkatz0.0-0.5	8-0	0.677 (0.016)	0.724 (0.007)	FTkatz0.0-0.5
3-1	0.517 (0.027)	0.553 (0.013)	SFTGG0.0-1.0	8-1	0.717 (0.028)	0.752 (0.007)	SFTkatz0.1-1.0
3-2	0.515 (0.045)	0.555 (0.016)	SFTRPRauto	8-2	0.731 (0.016)	0.742 (0.013)	SFTkatz0.0-0.5
4-0	0.49 (0.011)	0.508 (0.006)	FTRPR0.0-0.5	9-0	0.699 (0.014)	0.747 (0.011)	FTkatz0.0-1.0
4-1	0.51 (0.010)	0.532 (0.013)	SFTkatz0.0-0.5	9-1	0.68 (0.012)	0.726 (0.019)	FTkatz0.0-0.5
4-2	0.506 (0.010)	0.545 (0.009)	FTkatz0.0-1.0	9-2	0.627 (0.012)	0.726 (0.006)	FTkatz0.1-1.0

TABLE 7.3: Accuracy of correctly predicting node labels (ACC) and standard deviation (SD) of the best vs original GCN model. Models written as: F- features only, T- topology only, FT- both features and topology, $FTvanilla$ - the original GCN. S in the right column denotes usage of an additional feature weight matrix in the first layer (Equation 7.16). The models that use $G_{NS} = \tilde{L}^{sym}_{NS}$, where NS is a node-similarity ($Katz$, RPR, and GG) measure with different thresholds (Equation 7.9, 7.11 and, 7.13) are represented in the last column with the corresponding node-similarity matrix (e.g. katz for the model FTkatz0.0-0.5). All the similarity-based G are preprocessed and reconfigured based on Section 7.2.2. The preprocessing threshold $auto$ implies automatic selection of a threshold based on the mean value of the normalised node-similarity matrix (as per Section 7.2.2). Networks are represented in terms of snapshots, e.g. 0-0: first network's first snapshot, 0-1: first network's second snapshot etc.

7.3.1 Citation Networks

In Table 7.4, we report the average accuracy of the models, each of which are trained with 46 random initialisation of weights on the two real-world citation networks. Similar setup with 100 randomly changed weights was reported by Kipf and Welling (2017). In our analysis with the two citation networks *Citeseer* and *Cora*, we see that all three best performing models for the *Citeseer* dataset are the ones with our proposed model $FTRPR$ with different thresholds. Furthermore, when these three models are compared with the original GCN model (i.e. $FTvanilla$), it is apparent that for the best two models with $FTRPR$ (i.e. $FTRPR0.1 - 1.0$ and $FTRPR0.0 - 0.5$) exhibits a lower standard deviation (SD) ($FTvanilla$ has SD of 0.047 whereas the best two models with $FTRPR$ has an SD of 0.006 and 0.007) for the *Citeseer* dataset. This lower range of SD indicates higher robustness in modelling of the underlying patterns which defines the labels for the two citation networks. However, as for the models starts with an S (i.e. with an additional feature weight matrix in the first layer using Equation 7.16, e.g. $SFTRPR0.0 - 0.5$, $SFTkatz$) does not perform well for the *Citeseer* network. We have pointed out a few reasons why learning this additional feature weights (Equation 7.16) can be beneficial from the context of friendship-based social networks in Section 7.1.3. Perhaps, this type of friendship-type social network scenarios may not apply to the citation networks for

the reason that the relation between features, class labels and topology may not be similar to the friendship-type social networks. As for the network *Cora*, we see that the best performing model is the $FTVvanilla$, which is the original GCN, and $FTkatz$. We see similar situation of worst performance with the additional feature weights (models with S, Equation 7.16) matrix for the *Cora* network as well. Here we also speculate the same reason of not being a friendship-type social network, for which the model in Equation 7.16) is specifically designed (Section 7.1.3).

In Table 7.5, we split test set and training sets randomly 100 times and train our models and report their mean accuracy and standard deviation on the test sets. Here, we see similar results as in Table 7.4, where all three best models for the *Citeseer* dataset are the ones with our proposed model $FTRPR$ with different thresholds. Similarly, for *Cora* dataset, the original GCN model performs best.

Model	*Citeseer* Acc (SD)	*Cora* Acc (SD)	Model	*Citeseer* Acc (SD)	*Cora* Acc (SD)
FTRPR0.0-0.5	0.712** (0.006)	0.808(0.007)	SFTRPR0.0-0.5	0.696 (0.005)	0.809(0.058)
FTRPR0.0-1.0	0.711* (0.007)	0.809(0.006)	SFTRPR0.0-1.0	0.694 (0.012)	0.811*(0.041)
FTRPR0.1-1.0	0.713*** (0.046)	0.809(0.006)	SFTRPR0.1-1.0	0.698 (0.007)	0.808(0.006)
FTkatz0.0-0.5	0.683 (0.115)	0.815**(0.007)	SFTkatz0.0-0.5	0.682 (0.017)	0.810(0.007)
FTkatz0.0-1.0	0.702 (0.049)	0.816***(0.006)	SFTkatz0.0-1.0	0.683 (0.007)	0.810(0.008)
FTkatz0.1-1.0	0.682 (0.102)	0.811*(0.006)	SFTkatz0.1-1.0	0.684 (0.010)	0.806(0.007)
FTnewton0.0-0.5	0.584 (0.210)	0.810(0.007)	SFTnewton0.0-0.5	0.682 (0.017)	0.789(0.116)
FTnewton0.0-1.0	0.613 (0.184)	0.811(0.006)	SFTnewton0.0-1.0	0.686 (0.008)	0.795(0.086)
FTnewton0.1-1.0	0.687 (0.093)	0.810(0.006)	SFTnewton0.1-1.0	0.682 (0.009)	0.807(0.007)
FTvanilla	0.702 (0.047)	0.816***(0.006)	SFTvanilla	0.686 (0.017)	0.810(0.007)

TABLE 7.4: Accuracy (ACC) and standard deviation (SD) of the models with 46 random weight initialisation on two citation networks, *Cora* and *Citeseer*. Models written as: F- features only, T- topology only, FT- both features and topology, $FTvanilla$ - the original GCN. S in the right column denotes usage of an additional feature weight matrix in the first layer (Equation 7.16). The models that use $G_{NS} = \tilde{L}^{sym}_{NS}$, where NS is a node-similarity ($Katz$, RPR, and GG) measure with different thresholds (Equation 7.9, 7.11 and, 7.13) are represented in the last column with the corresponding node-similarity matrix (e.g. atz for the model FTkatz0.0-0.5). All the similarity-based G are preprocessed and reconfigured based on Section 7.2.2. Models with S represents if an additional feature weight matrix in the first layer (Equation 7.16). First best accuracy is marked with ***, second best with** and third best with *.

Model	*Citeseer* Acc(SD)	*Cora* Acc(SD)	Model	*Citeser* Acc(SD)	*Cora* Acc(SD)
FTRPR0.0-0.5	0.638***(0.100)	0.533(0.229)	SFTRPR0.0-0.5	0.637**(0.124)	0.593(0.188)
FTRPR0.0-1.0	0.638***(0.101)	0.536(0.228)	SFTRPR0.0-1.0	0.635*(0.126)	0.574(0.179)
FTRPR0.1-1.0	0.635*(0.094)	0.524(0.201)	SFTRPR0.1-1.0	0.631(0.105)	0.585(0.188)
FTkatz0.0-0.5	0.629(0.102)	0.471(0.190)	SFTkatz0.0-0.5	0.622(0.117)	0.546(0.186)
FTkatz0.0-1.0	0.630(0.097)	0.495(0.190)	SFTkatz0.0-1.0	0.625(0.113)	0.553(0.186)
FTkatz0.1-1.0	0.626(0.101)	0.497(0.191)	SFTkatz0.1-1.0	0.622(0.113)	0.581(0.202)
FTnewton0.0-0.5	0.597(0.138)	0.420(0.190)	SFTnewton0.0-0.5	0.590(0.157)	0.594(0.196)
FTnewton0.0-1.0	0.592(0.143)	0.419(0.193)	SFTnewton0.0-1.0	0.591(0.159)	0.599*(0.194)
FTnewton0.1-1.0	0.626(0.094)	0.498(0.191)	SFTnewton0.1-1.0	0.621(0.108)	0.609**(0.180)
FTvanilla	0.632(0.098)	0.670***(0.159)	SFTvanilla	0.635*(0.106)	0.585(0.188)

TABLE 7.5: Accuracy (ACC) and standard deviation (SD) of the models with 100 random splits on two citation networks, *Cora* and *Citeseer*. Models written as: F- features only, T- topology only, FT- both features and topology, $FTvanilla$ - the original GCN. S in the right column denotes usage of an additional feature weight matrix in the first layer (Equation 7.16). The models that use $G_{NS} = \tilde{L}^{sym}_{NS}$, where NS is a node-similarity ($Katz$, RPR, and GG) measure with different thresholds (Equation 7.9, 7.11 and, 7.13) are represented in the last column with the corresponding node-similarity matrix (e.g. katz for the model FTkatz0.0-0.5). All the similarity-based G are preprocessed and reconfigured based on Section 7.2.2. Models with S represents if an additional feature weight matrix in the first layer (Equation 7.16). First best accuracy is marked with ***, second best with** and third best with *.

7.4 Chapter Summary

In this chapter, we have evaluated the performance of GCN on simulated friendship-based social network datasets and on two real-world citation networks. One constraint of the GCN is that it is limited to a specific order of neighbourhood by the number of layers used in the model. We argued that using the node-similarity matrix as a graph representative allows us to solve this dependency between the l^{th} layers and the l^{th} order of the neighbourhood nodes. Additionally, our approach with the node-similarity measures may perform well enough with only a few layers compared with the original GCN due to the less dependency between the highest number of layers used in the model and the highest order of node-neighbourhood considered. The GCN or any deep learning model is prone to overfitting when a large number of layers are used (Kipf and Welling 2017), and our approach may get around this problem and achieve higher accuracy only with a few layers. It has also been empirically shown that most of the models with the augmented node-similarity measures outperform the original GCN.

In total we have proposed four new variations of the GCN model. Three of them are primarily based on the Katz, RPR, and GG scores as a form of the graph topology encoding. The fourth model is where we add learnable parameters for each of the features independent of the nodes for the entire graph, allowing the model to ignore the input features, if it so chooses. This variation of the model can be used with the adjacency matrix as well as with the Katz, RPR or GG scores,

and its primary motivation was the observation that for some datasets using the topology only, gives superior results. The results show that these new variations outperform the original GCN model in terms of accuracy.

Chapter 8

Conclusions and Future Work

The primary contributions of this thesis are in the area of prediction of the dynamics of a social network from the point of view of link appearance (Chapter 5). For better link predictability, we have combined two of the major contributing factors when it comes to link formations between nodes in a social network. These two factors are, (1) popularity of a node and (2) similarity between nodes. While these two factors are recognised based on the understanding of our social behaviour, we have also considered graph theory based factors (i.e. global and local measures on the graph) on top of popularity and similarity for an advanced link prediction approach. Majority of the link prediction methods operates on one of the two granularity levels of a graph: (1) on the global (graph level) or (2) local one (node level). An ideal link prediction algorithm may exploit both the global and local patterns in the graph for better prediction. The journey to find a technique, which can combine the popularity, similarity, graph level, and node level characteristics has directed us to a physical law, namely, the Newton's gravitational law. We have shown empirically, on a wide variety of the datasets, that this new class of gravity inspired link prediction methods outperforms most of other methods (Wahid-Ul-Ashraf, Budka and Musial-Gabrys 2018; Wahid-Ul-Ashraf et al. 2019).

It may be possible that the disappearance of links can also be predicted based on the same methods which are used to predict the appearance of links. The scores which are diminishing (obtained from the link prediction techniques between pairs of nodes) can be considered as indications that the links may cease to exist in the future. The prediction of a disappearing link can be achieved by assuming that the already existing link does not exist between a node pair and then calculating a score based on the link prediction technique. Afterwards, based on how weak the score is, one can determine the likelihood of that link being broken in future. However, the link disappearance predictions based on the link prediction approaches are not empirically evaluated in this work due to being out of scope for this thesis (as discussed in the beginning of Chapter 1).

The proposed new class of link prediction methods, namely Graph Gravity (GG), are not only useful for predicting links between nodes but also can be applied for classifying nodes in a social network. In this thesi (Chapter 7), we have shown that the combination of the classical graph theory based link prediction and the modern deep learning models has superior predictive power in the area of node classification. The reason is that the combined force of classical graph theory and modern deep learning approaches may circumvent some limitations that can arise when these approaches are used independently. In this thesis, state-of-the-art Graph Convolutional Networks (GCNs) is the family of deep learning techniques that are combined with the traditional graph theory based link prediction techniques (including our own Newton's gravity inspired approach). The blending of these two concepts for social network analysis produced a variety of new, more powerful predictive models for node classification. In our case, the graph theory based link prediction approach helped to reduce the limitation of the layer-wise dependency for the GCN models. Additionally, the graph theory provides a generic approach to augment the graph datasets for deep learning models. This technique of the graph data augmentation may have other potential benefits which we discuss later in this chapter in Section 8.2.

The final contribution is in the domain of social network simulation. While designing and testing different algorithms for social networks analysis, we have understood that there is a short supply of useful good quality social network datasets. To address this need, in this thesis, we have designed, developed, and validated a comprehensive network simulation framework in Chapter 6 and 7. For the simulation framework, we have analysed and compared different network properties for the three standard network models (random graph, scale-free, and small-world) in Chapter 4 (Wahid-Ul-Ashraf, Budka and Musial 2018). The analyses showed that although these three standard network models can represent some aspects of the real social networks, their properties are very different from each other. This has been seen even for the generated networks with the same size and density but for different network models. In this thesis, we have considered the fact that the three main network models have different properties, and proposed a network simulation framework which can fuse these models together for a more comprehensive social network modelling.

A potential future application of the GCN combined with the simulation framework is transfer learning for graphs, i.e. training a GCN on the simulated networks to be used in real-world networks. The GCN can be trained on a simulated network which may closely resemble (i.e. in terms of number of nodes, number of features, and other network properties such as clustering coefficient) a targeted social network and then be used on the real-networks for preference (i.e. $sDNA$) classification. On top of that, our graph augmentation technique using node similarity measures can allow us to have multiple graph representative matrices for a single graph. As a result, using several graph representative matrices can reduce graph level overfitting and allow

even better transfer learning. This type of transfer learning for graphs can have a potential application on social network marketing and advertisement as a certain type of $sDNA$ or preference may correlate with a preference towards certain products.

The proposed network simulation framework is capable of simulating high quality synthetic social networks with all the major required qualities of a real-world social network dataset. This type of network simulation framework, which is capable of not only generating nodes and links but also the features, node labels, the dynamic aspects, and even changes in the dynamics (i.e. dynamics of the dynamics) in one comprehensive formulation, to the best of our knowledge has not been accomplished to this date. The demand for such a comprehensive social network generator is further on the rise due to the introduction of user privacy enforcing laws such as the GDPR. Due to these types of privacy enforcing laws obtaining Facebook-type friendship-based social network datasets are becoming increasing difficult. We hope that the developed open-source software as an implementation of our proposed simulation algorithms will help to accelerate developments in the area of network science by manifolds.

8.1 Evaluation of the Objectives

We have fulfilled all the objectives from Section 3.2. For Objective 1 we have compared and analysed the three standard network models in Chapter 4. We have developed an approach to compare these three networks to each other, such that their size, and density stays the same while their network types are relatively stable (e.g. the scale-free models do not become similar to the random graph models). In Chapter 5 we have proposed the Newton's gravity inspired link prediction technique which fulfils Objective 2. Additionally, we have developed a software package in R (Appendix A, Section A.2) to fulfil Objective 3. In Chapter 7 we have combined the GCN with link prediction approach to fulfil Objective 4, and proposed four new variants of the GCN with better classification accuracy. In Chapter 6 we have developed a novel and comprehensive social network simulation framework, which is to complete Objective 5. Finally, for Objective 6 we have developed an open-source software for our proposed simulation framework. Examples on how to use the software is given in Section A.6. The GitHub repository for this software is given in Section 1.4.

8.2 Future Work

For the Newton inspired link prediction methods (in Chapter 5), methods which give better predictions when combined with centrality measures could be used for force directed graph

embedding to derive useful insights about the network other than link predictions (Mutton 2004; Ahn et al. 2014; Dawson et al. 2010).

Furthermore, in our case of social networks, we are currently assuming that any network operates in three-dimensional dissimilarity space for a particular dissimilarity measurement (or inverse of a similarity measurement); although there might be more dimensions, their impact on the inverse square law at large is negligible. The quadratic form of inverse squared distance that we observe for several cases of intensity or quantity in nature is due to three spatial dimensions, which characterise our physical world (Adelberger et al. 2003). The inverse square relation between physical quantity (or intensity) and distance is widely found in nature, and is known as the Inverse-Square law. Some examples include sound transmission (Marten and Marler 1977), force between two electrostatic charges (de Coulomb 1785), intensity of radiation (Gutiérrez and Sabra 2014) and more. The quadratic form of inverse distance that we observe for several cases of intensity or quantity in nature, is due to three spatial dimensions, which characterise our physical world (Adelberger et al. 2003). In our case of social networks, we are directly using the same Inverse-Square Law found in nature. For example, in the combination method of RPR with DC, the inverse of RPR is the path length analogous to the distance in Newton's gravitational law in Equation 5.1. The squared distance in Newton's law is a result of three spatial dimensions. But for our approach in Equation 5.2, other than the quadratic order, it might be possible to obtain better performance by using an order of one, three, four etc of the RPR. Optimal order of the dissimilarity measure could be learnt from the ground truth of the data such that the dimension for which using Equation 5.2 gives the best prediction result.

The optimal dimension could be learnt (i.e. using machine learning) from ground truth of the data, such that the dimension for which using Equation 5.2 in Chapter 5 gives the best prediction result. Additionally, all the link prediction approaches can be tested to predict the disappearance of links that we have discussed earlier in Section 8. Predictive analysis of disappearance of links is another potential future research direction of this thesis.

In this thesis, we have considered the likelihood of two different nodes connecting in future by using Newton's gravitational law. The measurement of mass based on some of the centralities (e.g. betweenness centrality) and the distance based on node some of the node similarities (e.g. Katz) considers not only the two nodes in question but also their surroundings. However, a more accurate application of the gravitational force requires considering a more precises effects of other surrounding nodes' effect. This can be achieved by mapping the topology and properties of the network in a gravitational field similar to the Metric Tensor in the theory of General Relativity. A gravitational field mapping may not only result in better link predictability but also it may reveal some interesting patterns within the gravitational field by defining special characteristics of a social network.

As discussed earlier, the equation of universal gravitation by Newton is not the only area where we observe this kind of relationship between particles or bodies. Coulomb's law for attraction or repulsion between particles with static charges (de Coulomb 1785) works based on similar principle. If we have two charged particles with q_1 and q_2 and r is the distance between them then the attraction or repulsion force is defined as:

$$F = k_e \frac{q_1 q_2}{r^2} \tag{8.1}$$

In Equation 8.1, k_e is the Coulomb's constant. This force between two nodes could be attraction or repulsion force depending on the sign of the charges (different attraction and same repulsion). Similar to the Newton's gravitational law, the Coulomb's law can be applied not only to predict appearance of links but also to predict disappearance of links. The repulsion of links can be measured by using filters on the similarity measures. For example, a similarity score more than some threshold t can be defined as repulsion.

As for our work in the area of node classification, so far we have used Katz, RPR, and GG node-similarity measures (in Chapter 7). Other node similarity measures such as SimRank, Triad Transition Matrix (TTM), AdamicAdar can also be used as graph representatives for the GCN (Jeh and Widom 2002; Juszczyszyn, Budka and Musial 2011; Adamic and Adar 2003). Additionally, we have used a few empirically selected thresholds for the augmented node-similarity matrix (Chapter 7). A more effective way to select optimal thresholds is another future direction to explore.

In Chapter 7, for the node-similarity-based matrices, we have proposed a reconfiguration technique. This reconfiguration results in augmentation of the graph represented by the node-similarity matrix. This is particularly important as for node classification task with GCN-like models, we only have one graph sample to train the model. The augmentation technique can be used to better train the model on the same graphs with several different augmented node-similarity matrices (with different thresholds and similarity measurements). Several representations of the same graph topology can also work as a regularisation technique to prevent overfitting of the model. Furthermore, the augmentation technique on the node-similarity matrix with thresholds can reduce the number of connections without losing important information about the graph structure. This reduction can come in handy when we have a large and dense graph. A dense graph requiring more arithmetic operations to train the model can become computationally expensive. The reduction of the connections in the augmented node-similarity matrix will then speed up the process significantly.

Also, for the node classification approaches in Chapter 7, the three cases where topology only model performs better could be due to significantly more learnable parameters the model has compared with the feature and topology model that we have discussed. To solve this problem

of an unequal number of learnable parameters, we have introduced another variation for the topology only model, where the number of learnable parameters is reduced by using a low-rank approximation of the weight matrix. The reduced parameter model for the topology only model also performs well compared with the model with more parameters. A further inspection of those datasets may reveal the underlying reason why the topology only models perform well for them. However, it could be possible that for those three datasets, the features are reflected within the topology so well that the topology only model becomes more powerful and adding features simply results in redundancy.

In Chapter 6, we have introduced a comprehensive social network simulation guideline and introduced an open-source software. The software is highly optimised and uses GPU computation, however there is room for further optimisation. Additionally, integrating more features such as graph analytic with the software can be another future direction towards upgrading the software package.

In this thesis, we have developed and tested a set of nature-inspired link prediction approaches, reinforcing the value of guidance provided by well-established physical models. Furthermore, we have designed deep learning models for networks with the combined power of graph theory based link prediction and state-of-the-art GCNs for node classification, demonstrating how inspirations coming from different branches of science can complement each other, leading to development of a new breed of techniques.

Appendix A

Appendix

A.1 Enlarged plots for the Network Models - Simulation Study, Chapter 4

A.1.1 Closeness Centrality

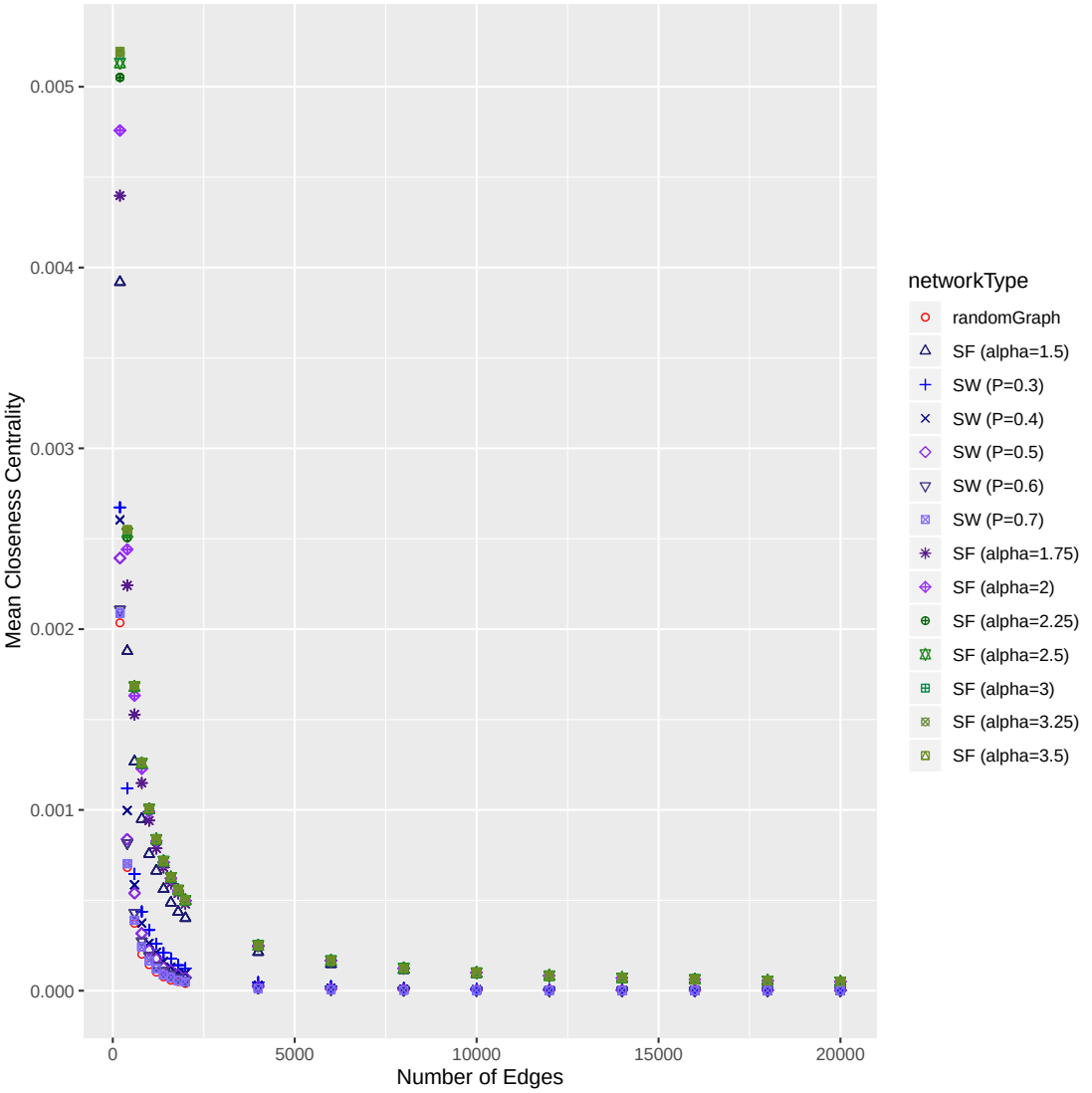


FIGURE A.1: S & Nei=2

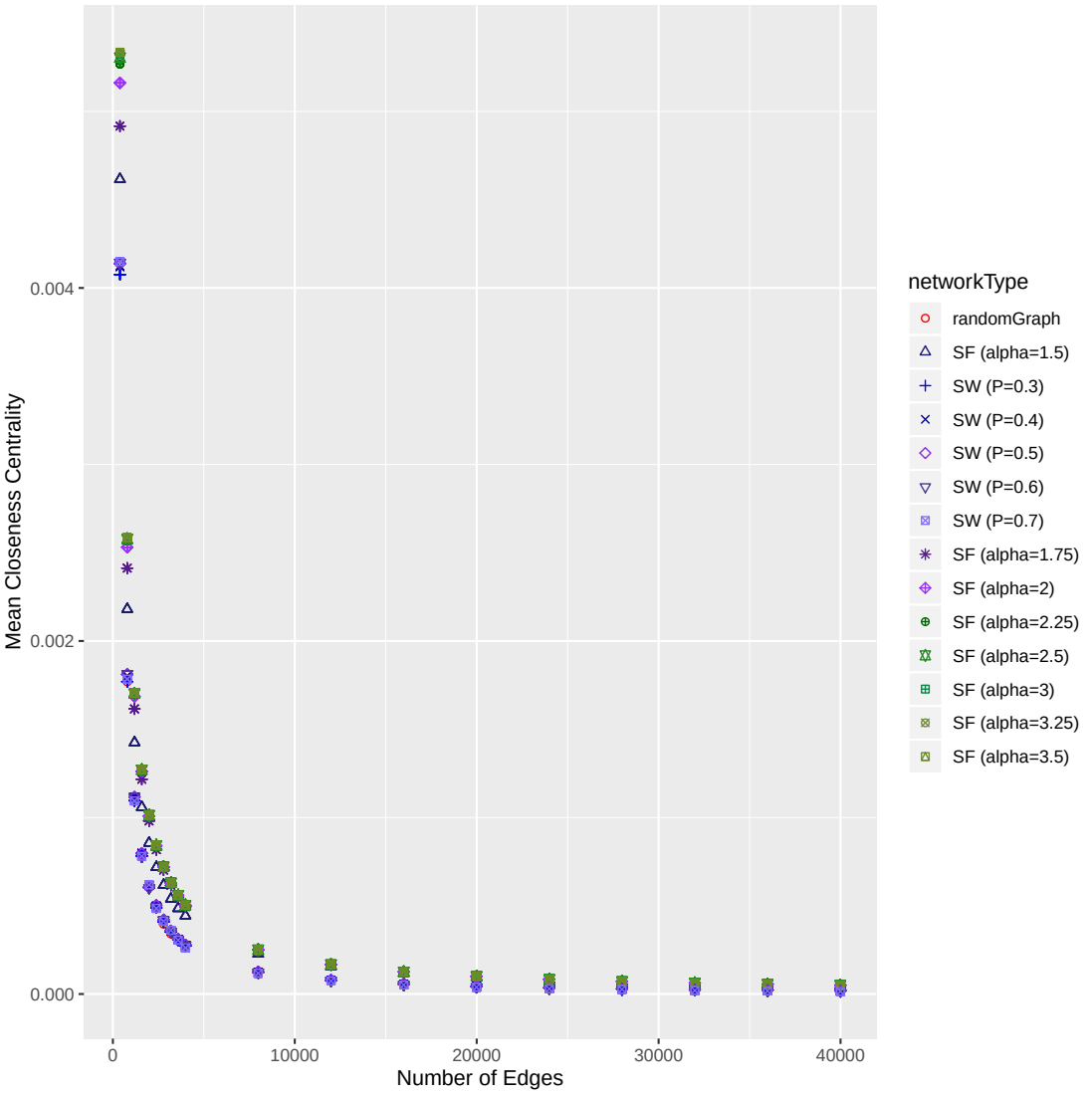


FIGURE A.2: S & Nei=4

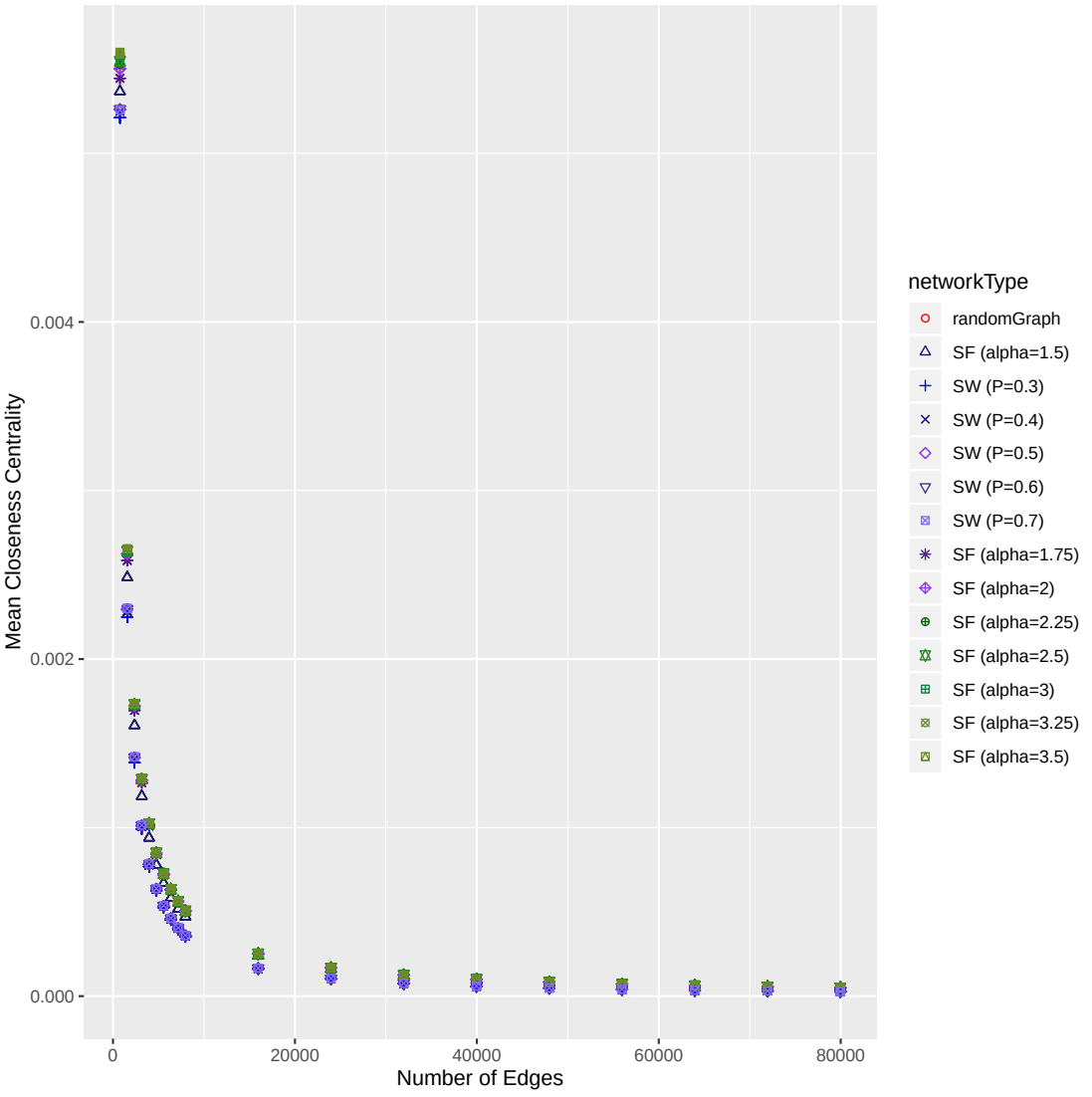


FIGURE A.3: S & Nei=8

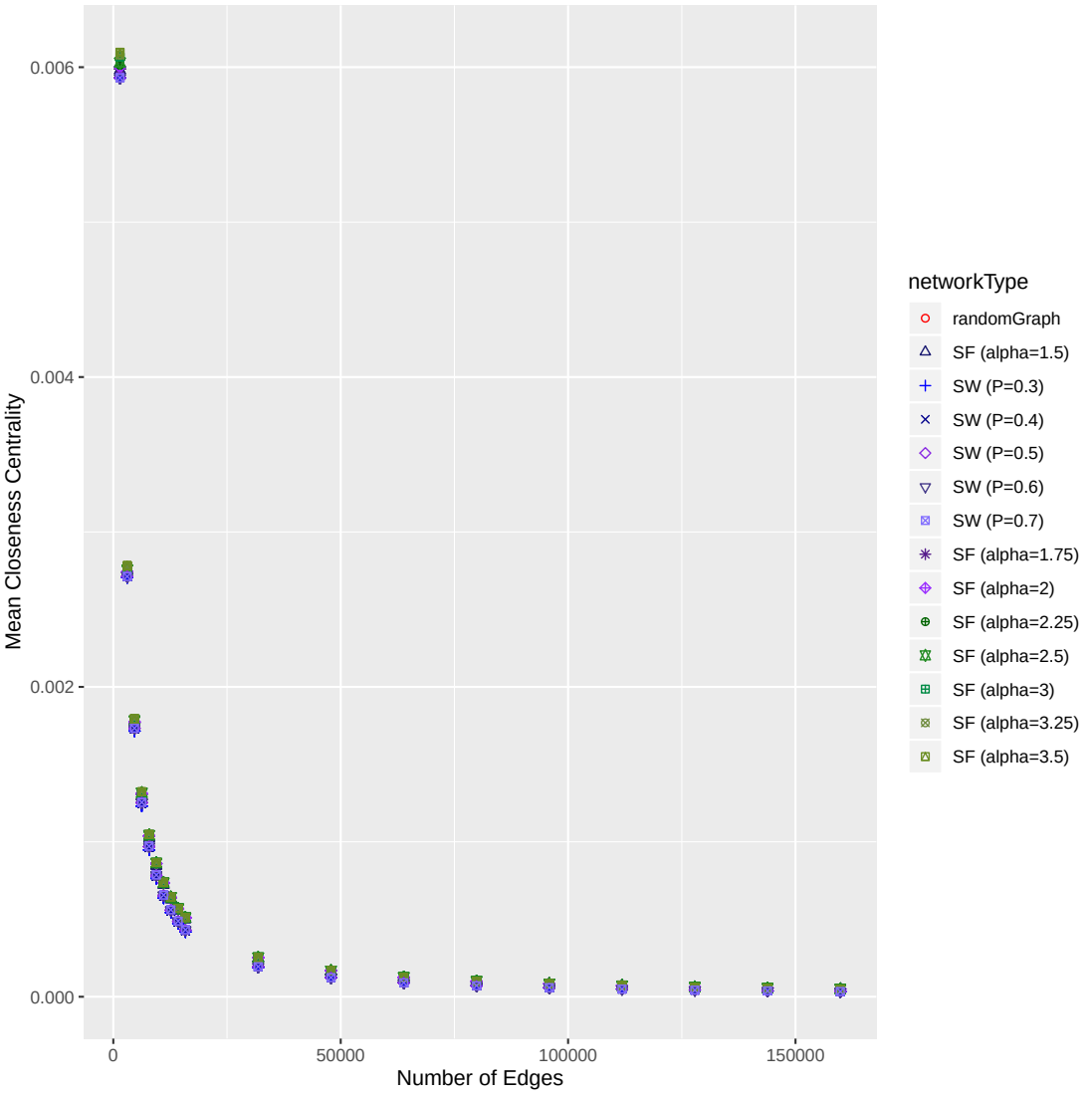


FIGURE A.4: S & Nei=16

A.1.2 Betweeness Centrality

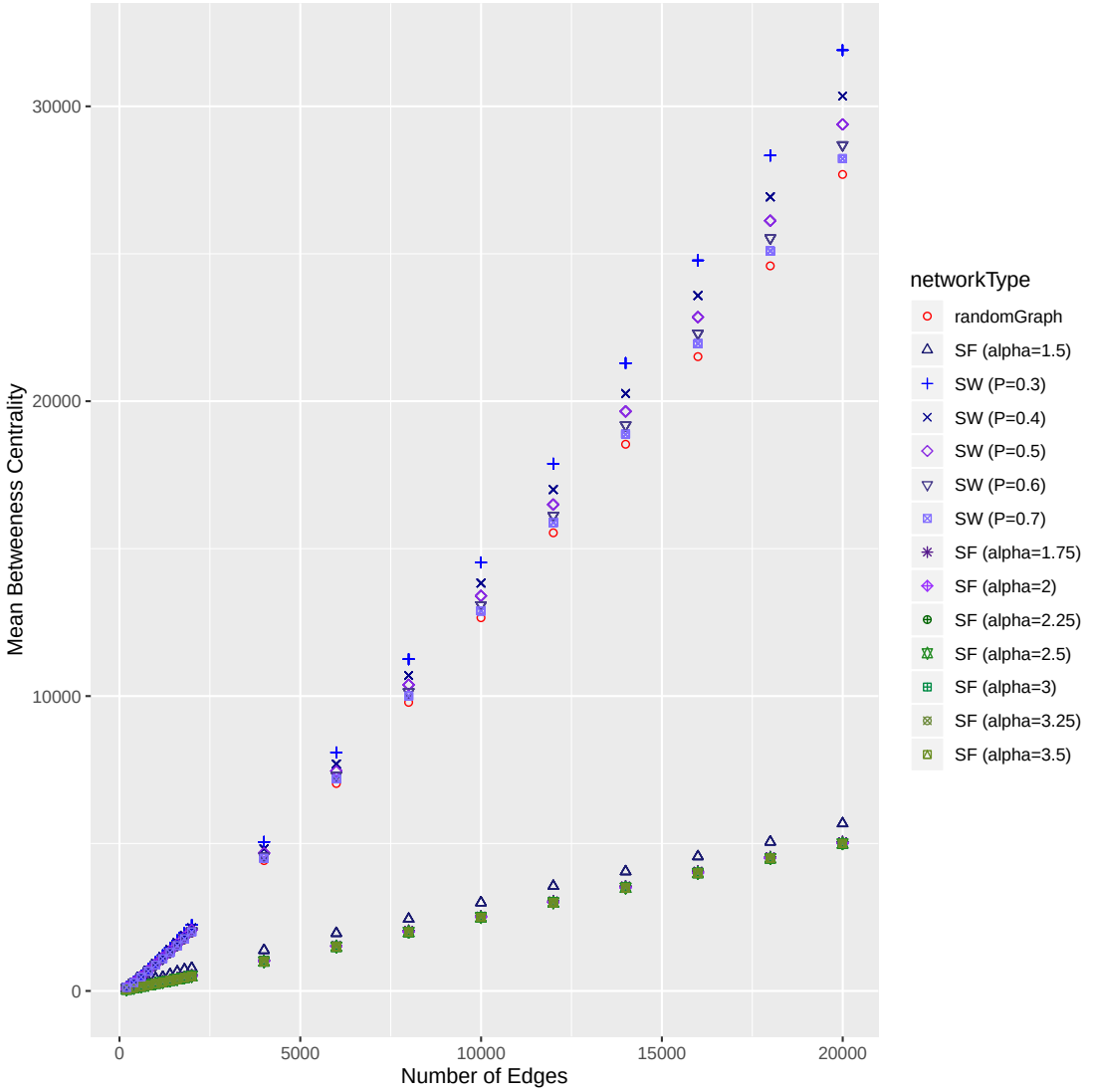


FIGURE A.5: S & Nei=2

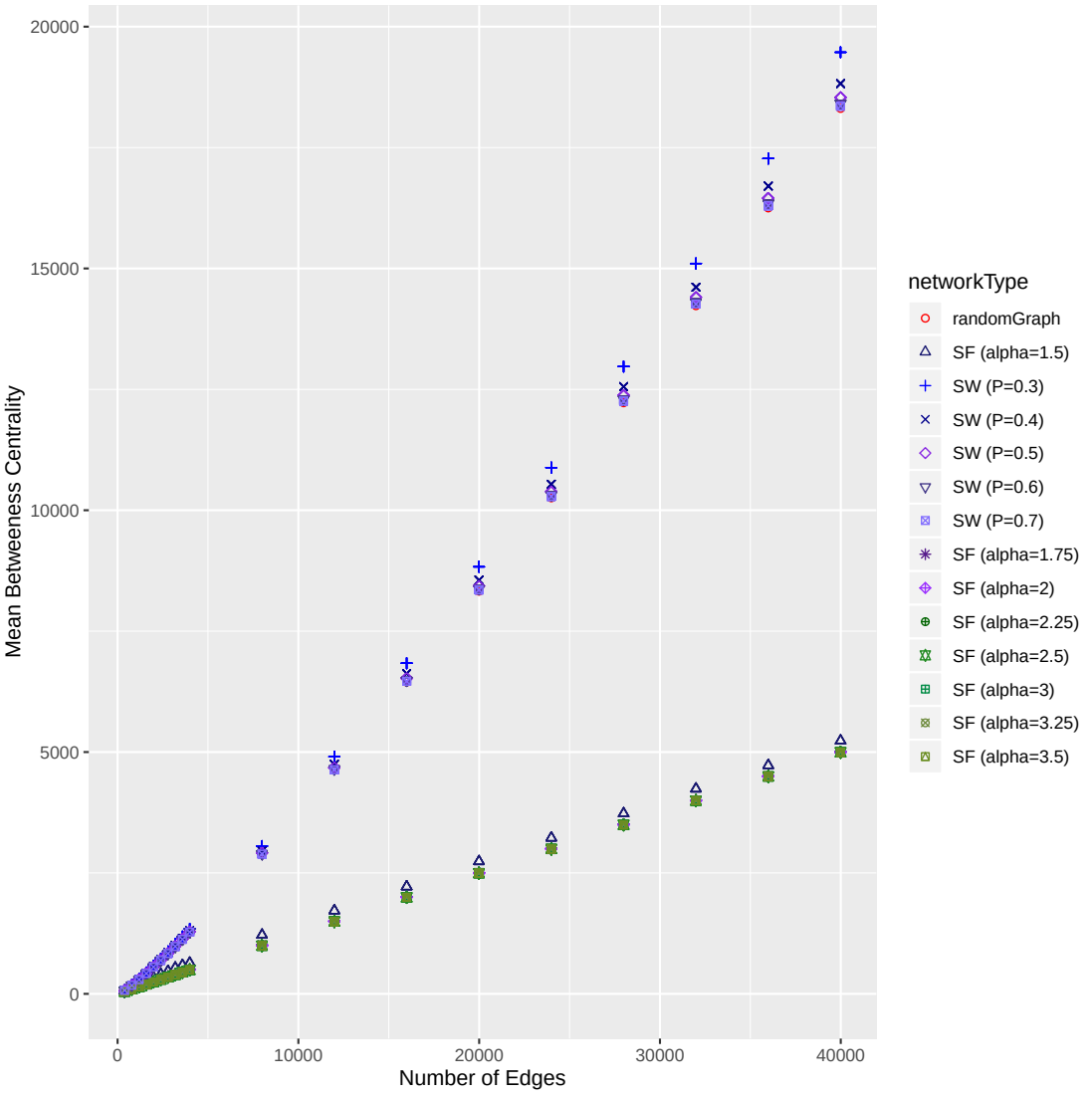


FIGURE A.6: S & Nei=4

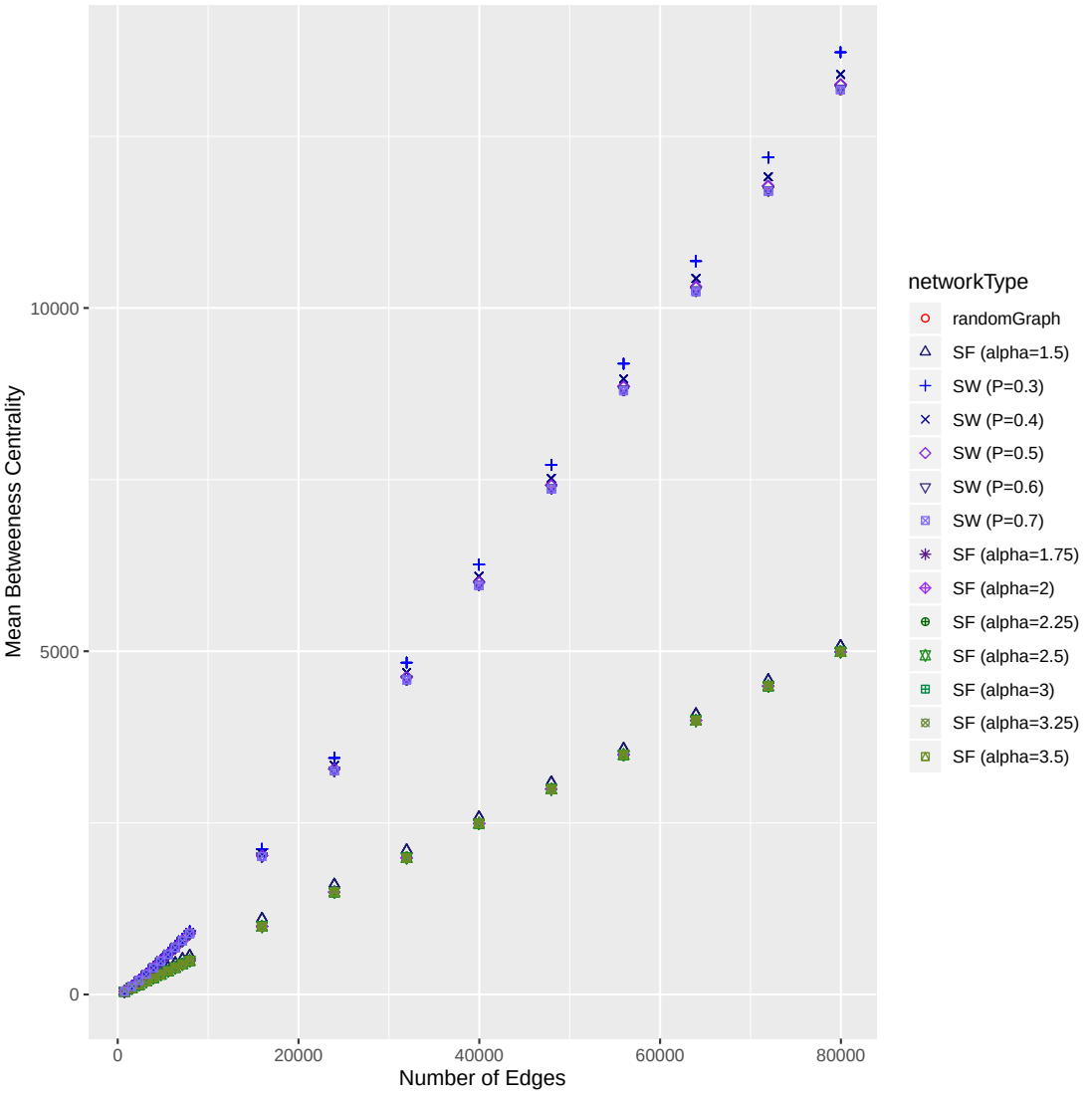


FIGURE A.7: S & Nei=8

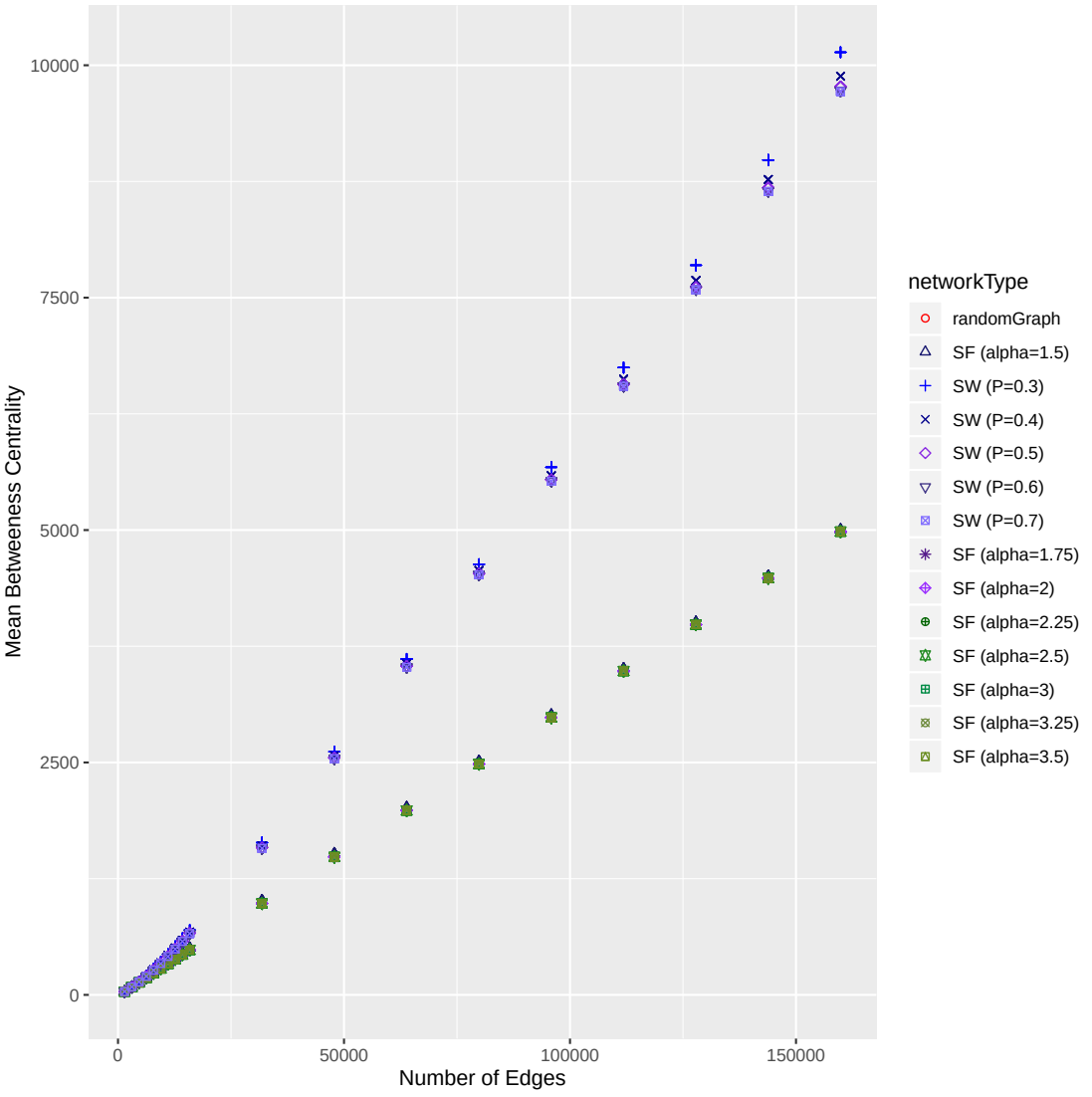


FIGURE A.8: S & Nei=16

A.1.3 Avg Geodesic Path Length

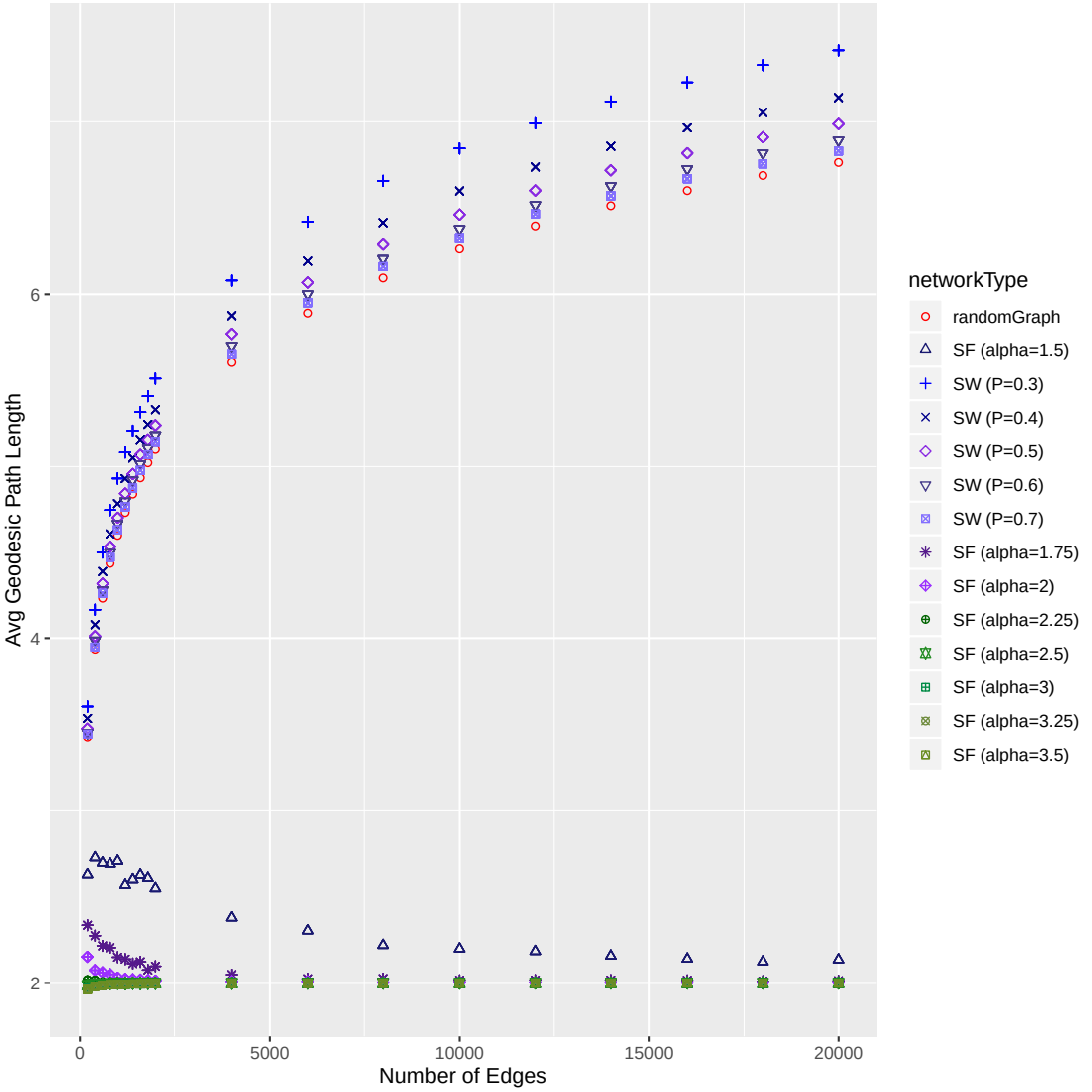


FIGURE A.9: S & Nei=2

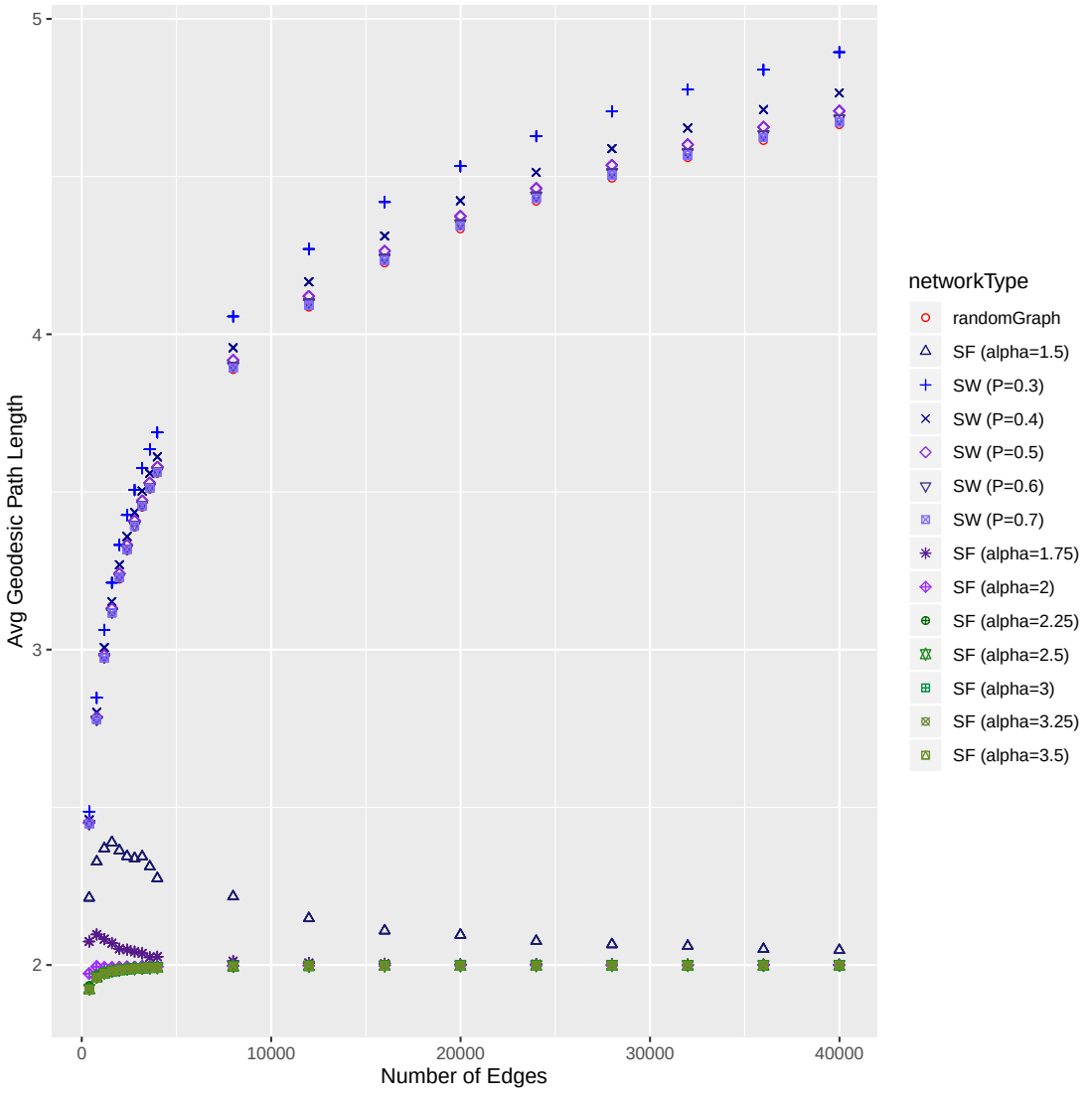


FIGURE A.10: S & Nei=4

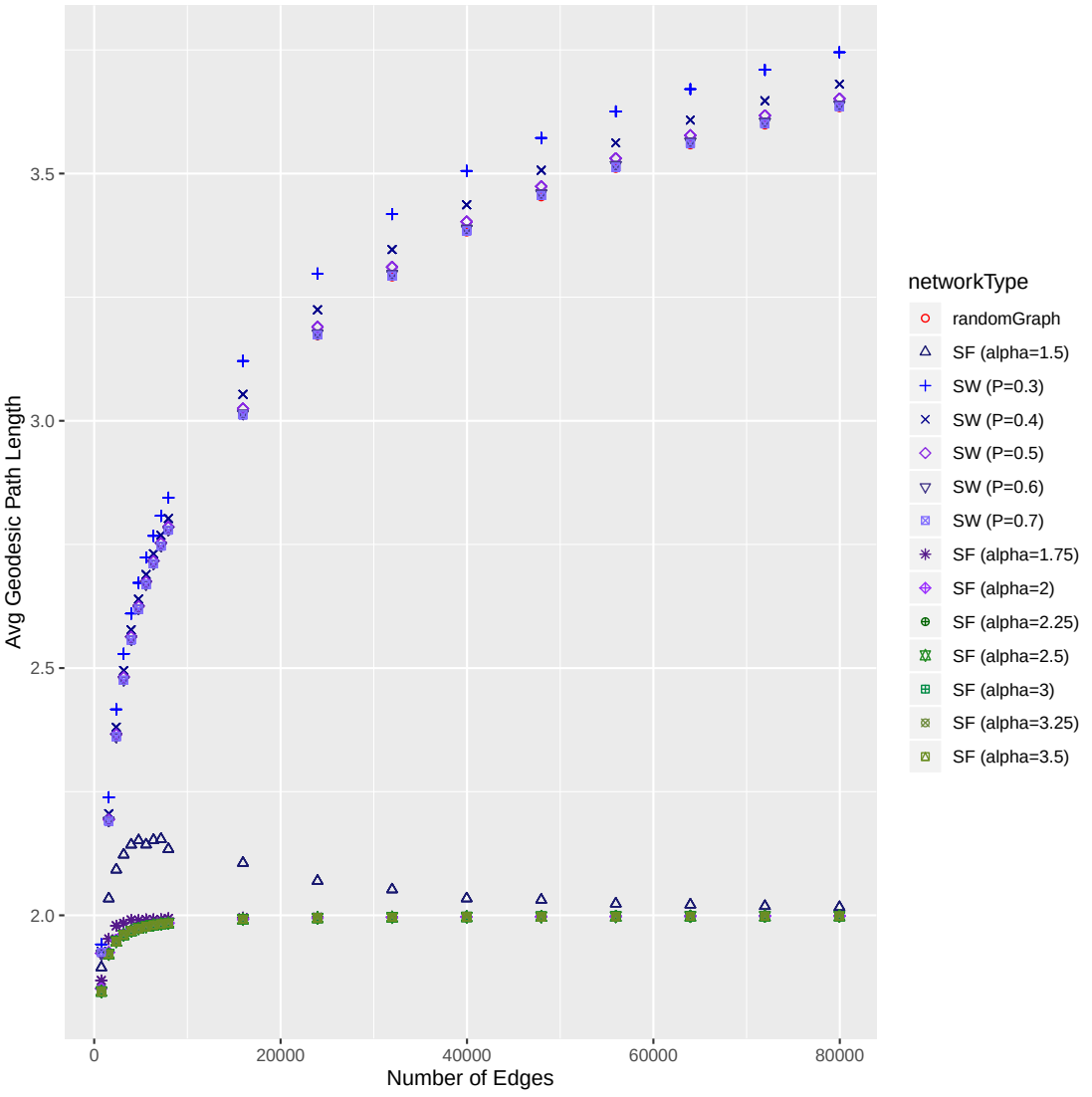


FIGURE A.11: S & Nei=8

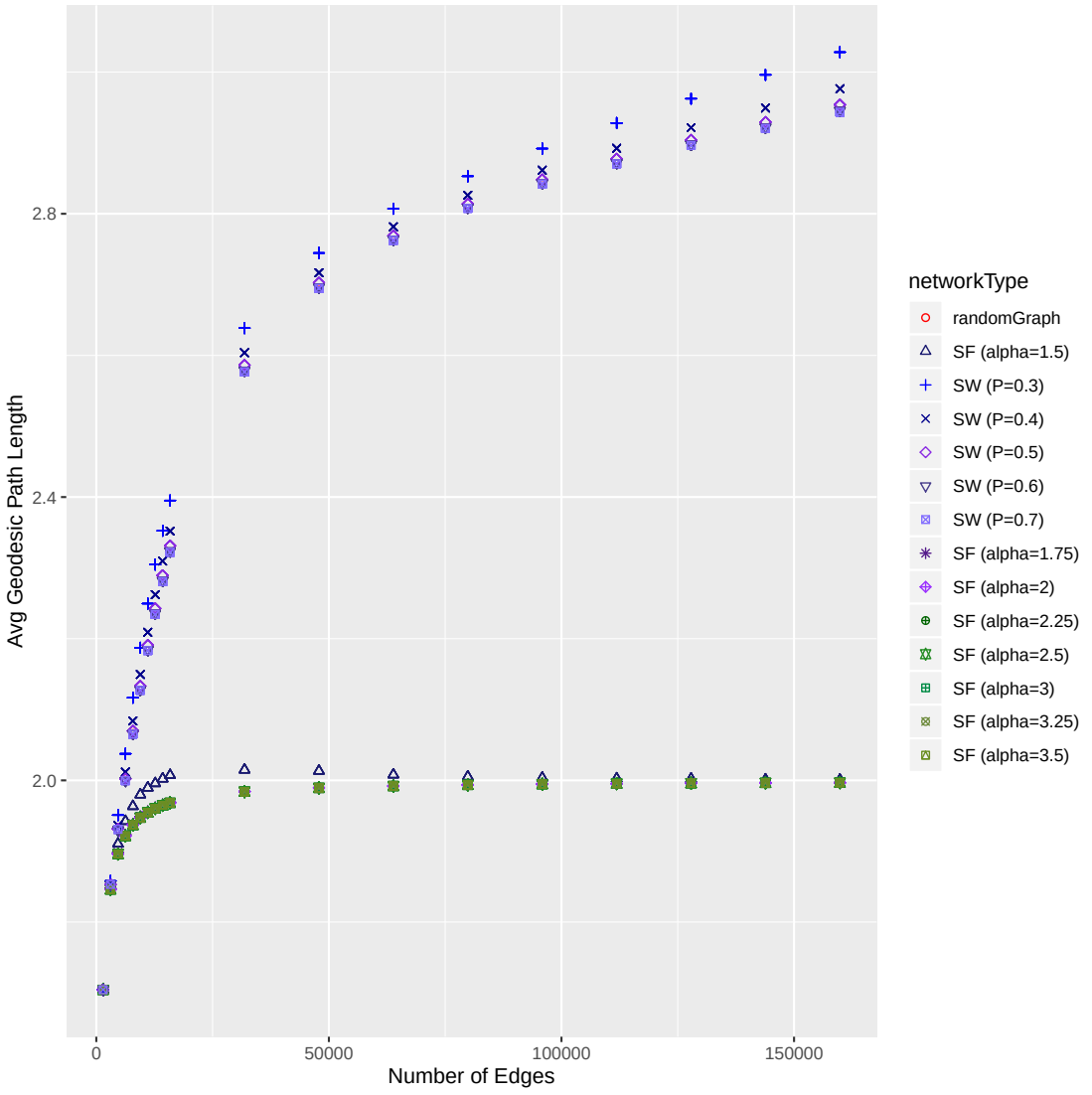


FIGURE A.12: S & Nei=16

A.1.4 Global Clustering Coefficient

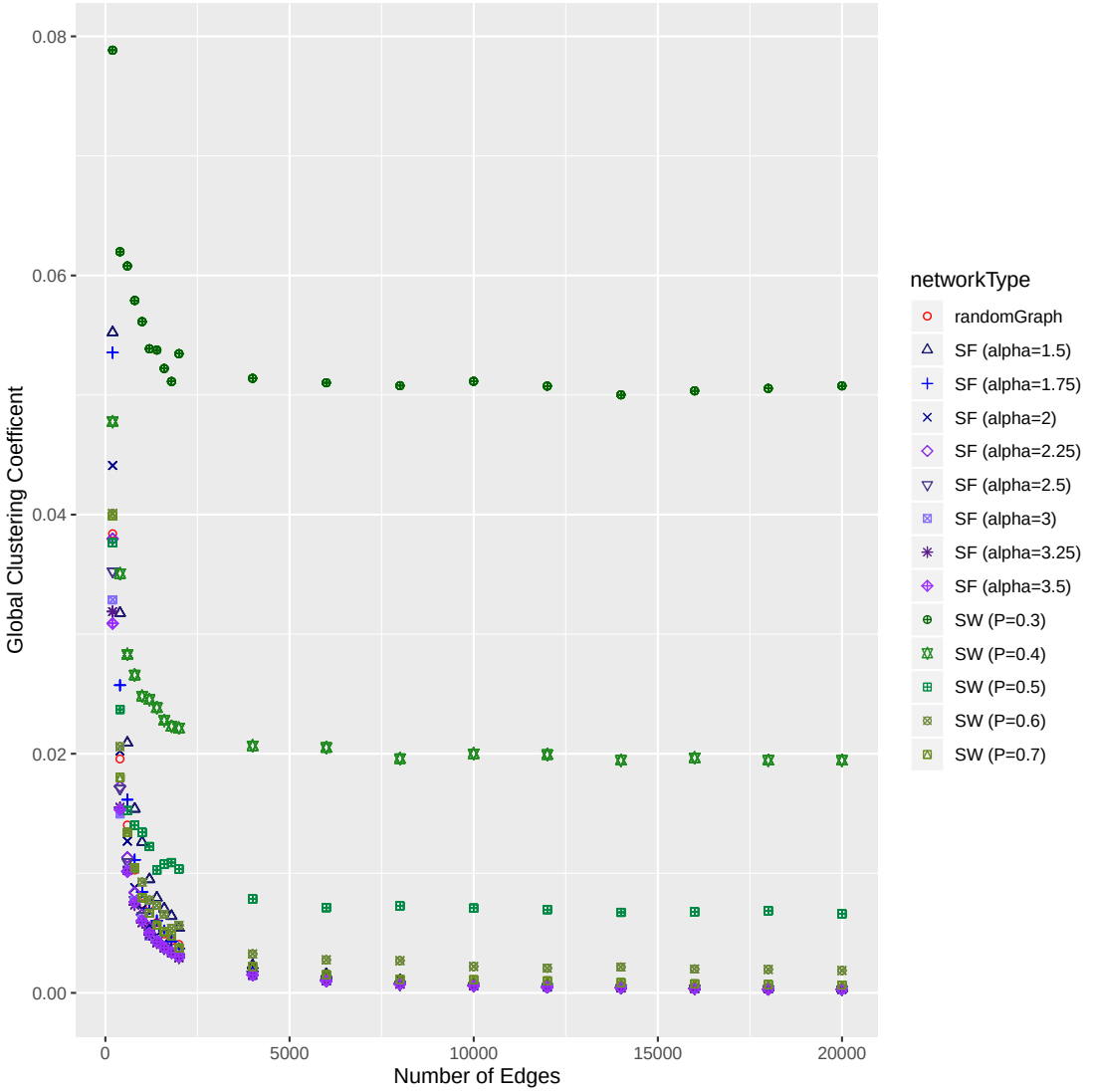


FIGURE A.13: S & Nei=2

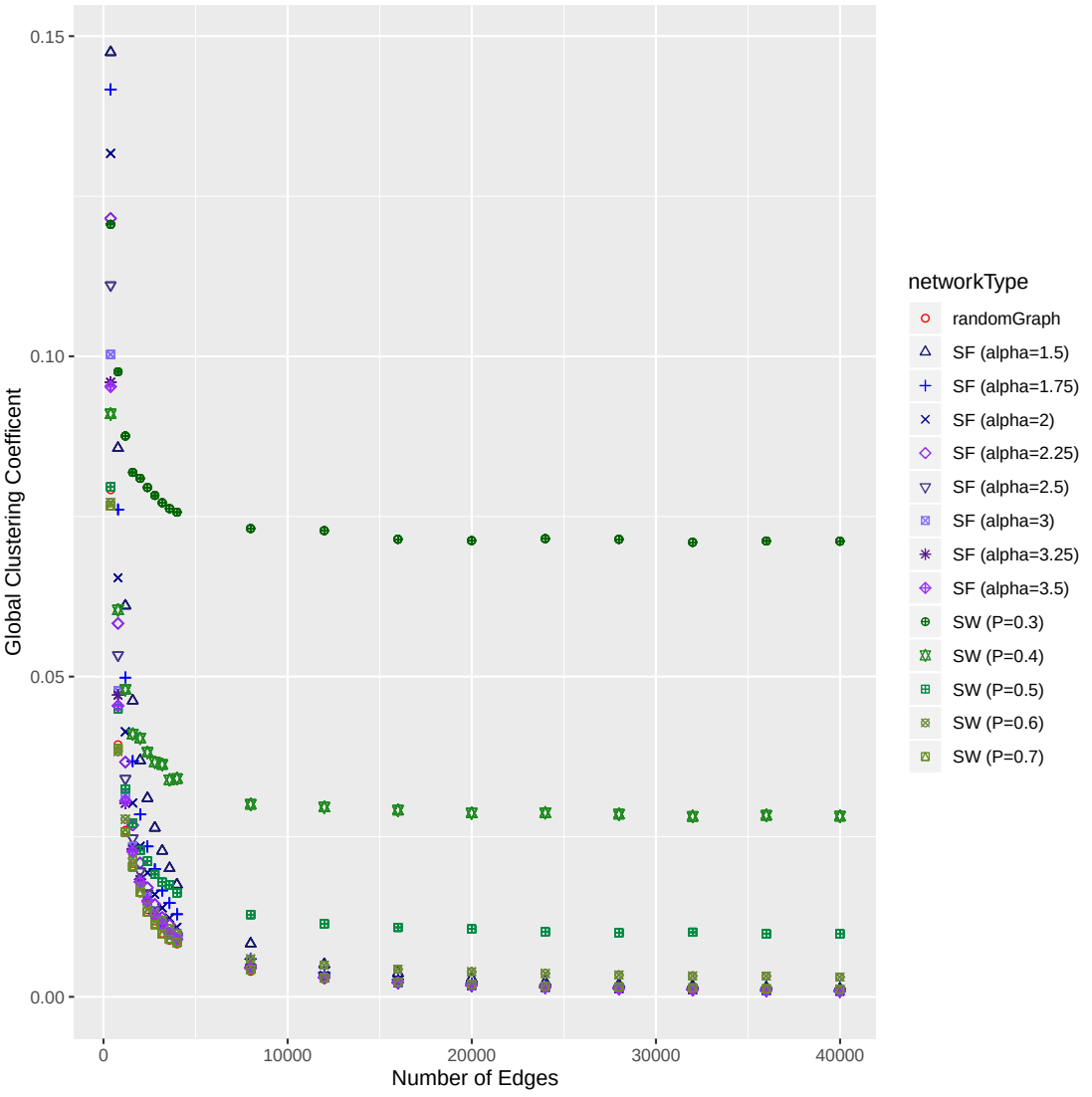


FIGURE A.14: S & Nei=4

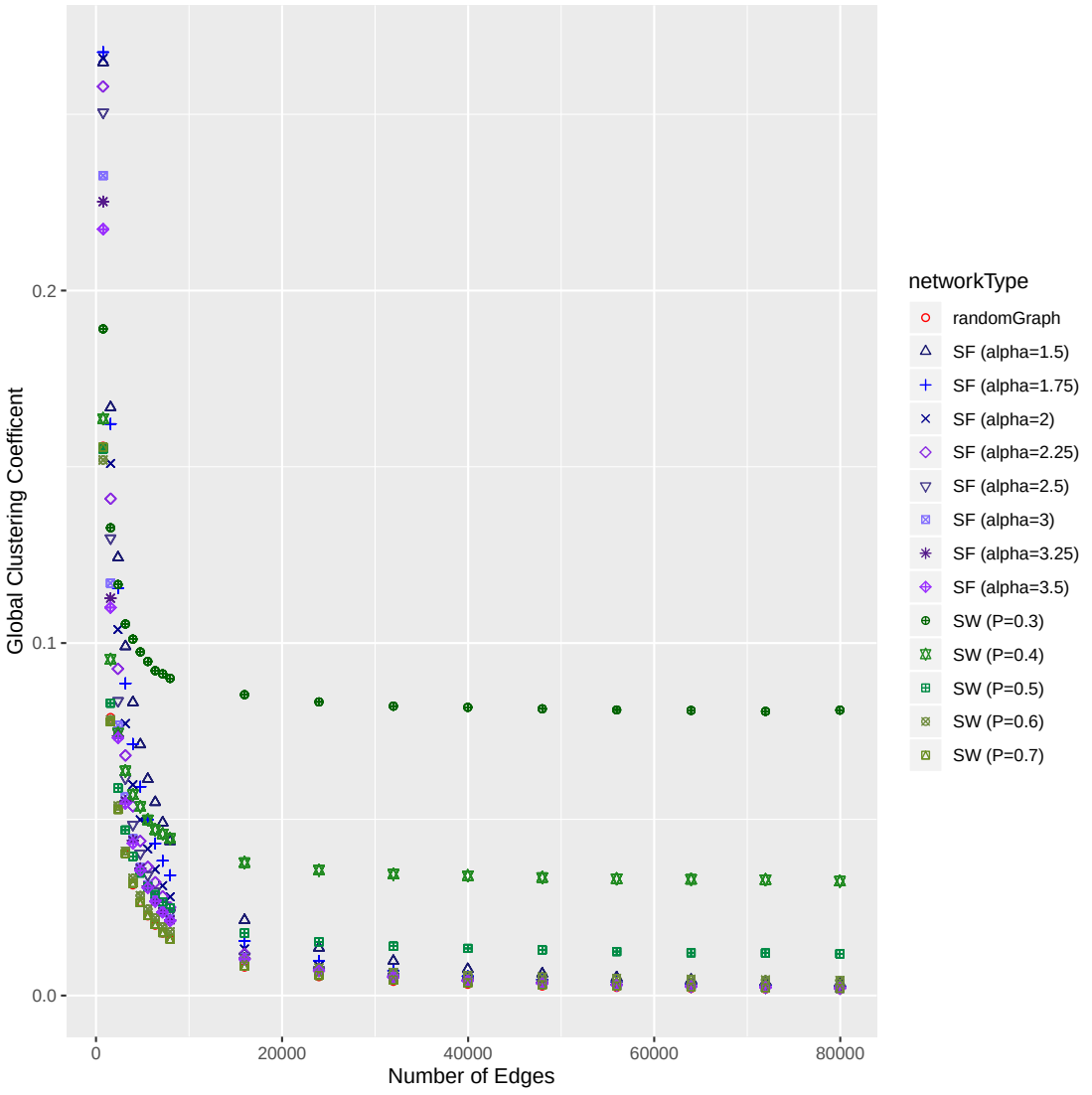


FIGURE A.15: S & Nei=8

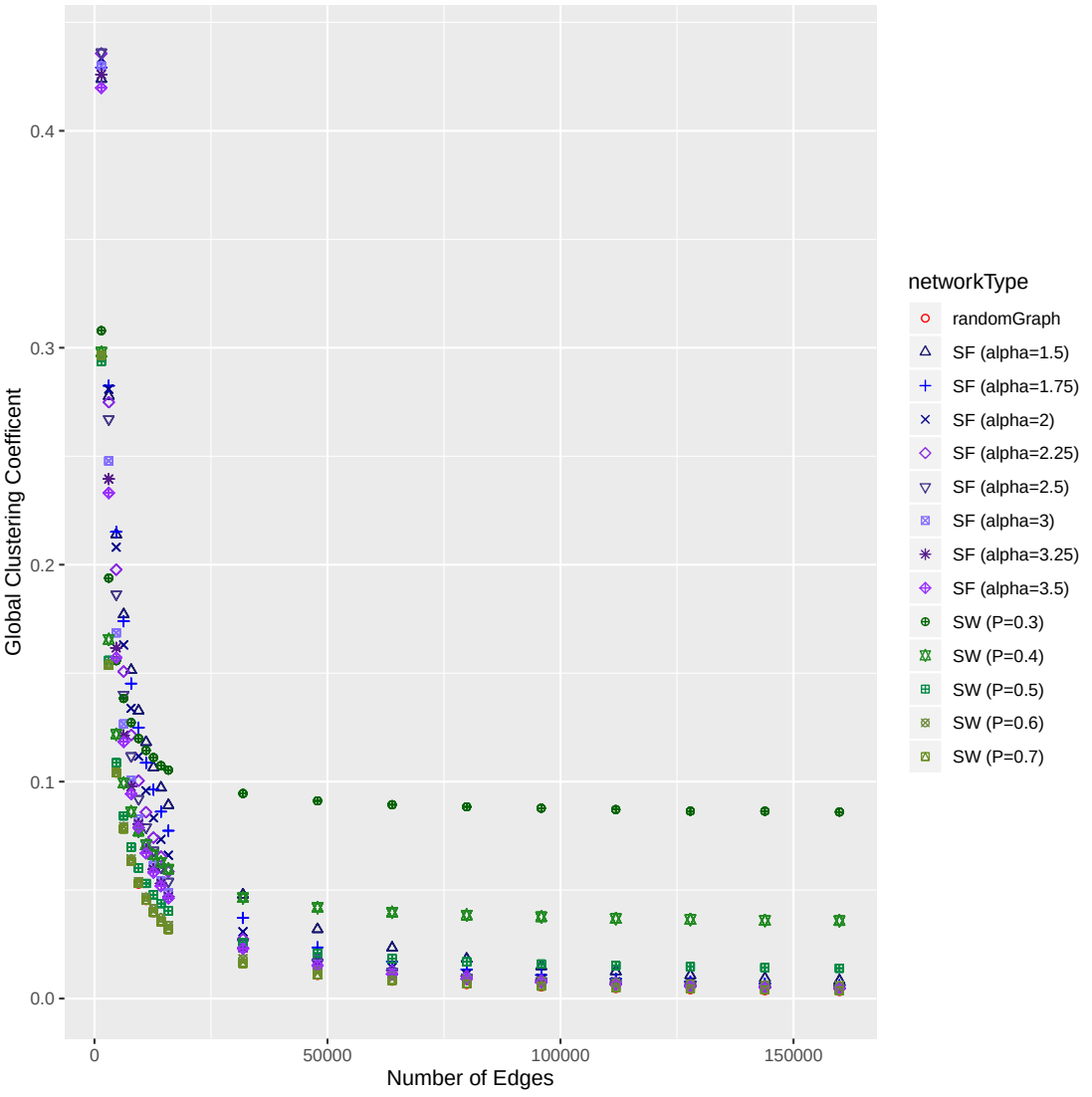


FIGURE A.16: S & Nei=16

A.2 Link Prediction Library

A Link prediction Library in R is developed to benchmark with most popular 12 different link prediction methods including our approach described in Chapter 5. It is a comprehensive library with the freedom to predict the unlimited number of parameters for each of the methods. The library contains 26 functions. This library supports multithreading and user can define the number of threads to be used during the experiment. The usage of multithreading radically reduces processing time. Although R is comparatively slow, this library utilises other highly optimised R libraries such as Matrix, igraph etc. Most of the time costly operations are done using different libraries/dependencies which are originally written in faster programming languages such as C/C++. Two main functions of the library are described below:

Function Name: calcPredictions()

Description: This function is a pipeline, and it calculates all the predictions from training datasets and also checks the prediction by comparing against the ground truth from the test set. The processed information in each steps is stored in HDD as serialised objects to reduce memory usage and enhance re-usability.

Arguements

FolderPath: This is the path of the folder where datasets are. For example if the root folder is E://experiment// , than inside the folder experiment there are subfulders such as: E://experiment//dataset1, E://experiment//dataset2 etc. Each of the subfolders, such as dataset1 should contain two text files named train.txt and test.txt for training and test set. For network input format, currently the library supports edge lists seperated by a comma.

KatzBeta: Katz parameter β. Any number of parameters could be selected. If auto is selected than the parameters are selected autometically.

rootedPageRankAlpha: α Parameter(s) for rooted PageRank method.

LocalPathsIndexEps: ϵ Parameter(s) for Local Path Index.

LeichtHolmeNewmanTheta: θ Parameter(s) for Leicht-Holme-Newman Global Index.

threads: Number of threads should be utilised.

Usage

calcPredictions(FolderPath="E://experiment//",KatzBeta =c("auto",3),
rootedPageRankAlpha =c(0.15,0.25),LocalPathsIndexEps =c(0.01,0.05),
LeichtHolmeNewmanTheta =c(0.5,0.7),threads=8)

Output: All the output files will be generated under each folder of the datasets. Results will also be evaluated using Precision-Recall (PR) and Receiver-Operating-characteristic (ROC) curves.

Function Name plotPredictions()

Description: This function plots heat maps, bar charts for all the predictors for every dataset. There are individual and also combined plots. This function also calculates statistics for all the networks. Performance of a random predictor is also evaluated and compared against all the predictors. This function only can be called once the calcPredictions function has been called.

Arguements

FolderPath: E://experiment//

Usage

plotPredictions(FolderPath=”E://experiment//”)

Output: Upon successful completion of the function, there will be three additional folders under the FolderPath. PR: which contains results for PR, ROC: which contains results for ROC, stats: which contains statistics of all the networks. Under PR and ROC folders there are two CSV files with the values of Area Under the Curve (AUC) of PR and ROC.

heatMapMatrix.csv: is the file with heatmap data. **heatMapMatrixRanked.csv:** each of the predictors are ranked for each of the datasets.

The library also generates interactive heat map, and bar plots for all the datasets for PR and ROC. Also, many other individual plots are available for better understanding the performance for the predictors.

There is also an progress bar while the calcPredictions() is called and running in R. Example of the output while this function runs an experiment is given below.

```
> calcPredictions("D:\\Experiment1\\")
[1] "Creating igraph graph object from dataset1 dataset"
[1] "calculating similarity for dataset1 dataset"
[1] "Calculating distance, and adjacency matrices..."
[1] "Calculating katz..."
[1] "Calculating AdamicAdar..."
[1] "calculating scoreCN"
[1] "Calculating Common Neighbor..."
[1] "Calculating Jaccard Coefficent ..."
[1] "Calculating Average Commute Time..."
solve(L - 1/n) worked
[1] "Calculating Average Commute Time Normalised..."
[1] "Calculating Rooted PageRank..."
[1] "Calculating Local Paths Index..."
[1] "Calculating Leicht-Holme-Newman Global Index..."
[1] "Calculating Matrix Forest Index..."
[1] "writing adding values in a table..."
  |=============================================================================================================================================| 100%
[1] "loop finished"
[1] "writing in HDD.."
[1] "Total time taken from creating table to writing in file: 8.39548087120056"
[1] "Adding ground truth from test set for dataset1 dataset"
```

FIGURE A.17: calcPredictions() running an experiment

```
> plotPredictions("D:\\Experiment1\\")
[1] "Generating Heatmap for AUC, ROC"
[1] "Ranked heatmap matrix in excel for ROC(file name: heatMapMatrixRanked.csv)"
[1] "Generating Barplot for AUC, ROC"

Attaching package: 'dplyr'

The following object is masked from 'package:Kmisc':

    matches

The following objects are masked from 'package:igraph':

    as_data_frame, groups, union

The following objects are masked from 'package:data.table':

    between, first, last

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

No trace type specified:
  Based on info supplied, a 'bar' trace seems appropriate.
  Read more about this trace type -> https://plot.ly/r/reference/#bar
[1] "Generating individual Barplots for AUC, ROC"
[1] "Generating Heatmap for AUC, PR"
[1] "Ranked heatmap matrix in excel for PR(file name: heatMapMatrixRanked.csv)"
[1] "Generating Barplot for AUC, PR"
No trace type specified:
  Based on info supplied, a 'bar' trace seems appropriate.
  Read more about this trace type -> https://plot.ly/r/reference/#bar
[1] "Generating individual Barplots for AUC, PR"
[1] "Calculatin AUC PR for random classifiers look for AUCofRandomclassifiers.csv file in the PR folder"
[1] "Comparing against random classifier for PR, look for heatMapMatrixRankedRandTested.csv file in the PR folder"
[1] "Calculating stats for all the networks, look for stats folder in the root dir"
Saving 7 x 7 in image
Saving 7 x 7 in image
Saving 7 x 7 in image
Saving 7 x 7 in image
Saving 7 x 7 in image
Saving 7 x 7 in image
Saving 7 x 7 in image
Saving 7 x 7 in image
Saving 7 x 7 in image
Saving 7 x 7 in image
Saving 7 x 7 in image
> |
```

FIGURE A.18: plotPredictions() auto-generating plots and statistics from experiment outputs that had been auto-generated by calcPredictions() function

All the heatmaps and bar charts and statistics in the Appendix A is generated using this library[1].

[1] the library could be shared for usage in academic purpose upon request via email

A.3 AUC of Precision-Recall for 7 Datasets

A.3.1 Individual Barplots

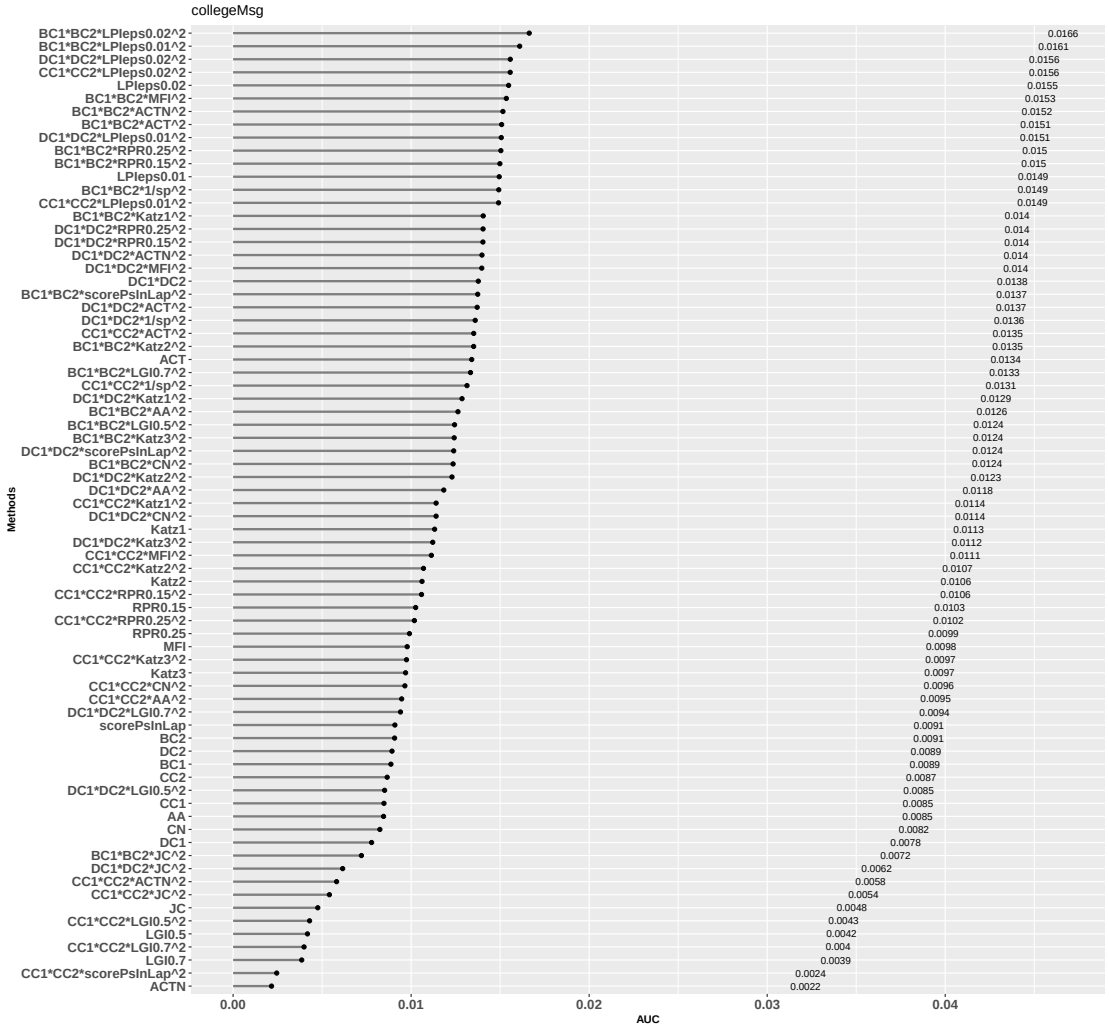


FIGURE A.19: PR AUC of *collegeMsg dataset*

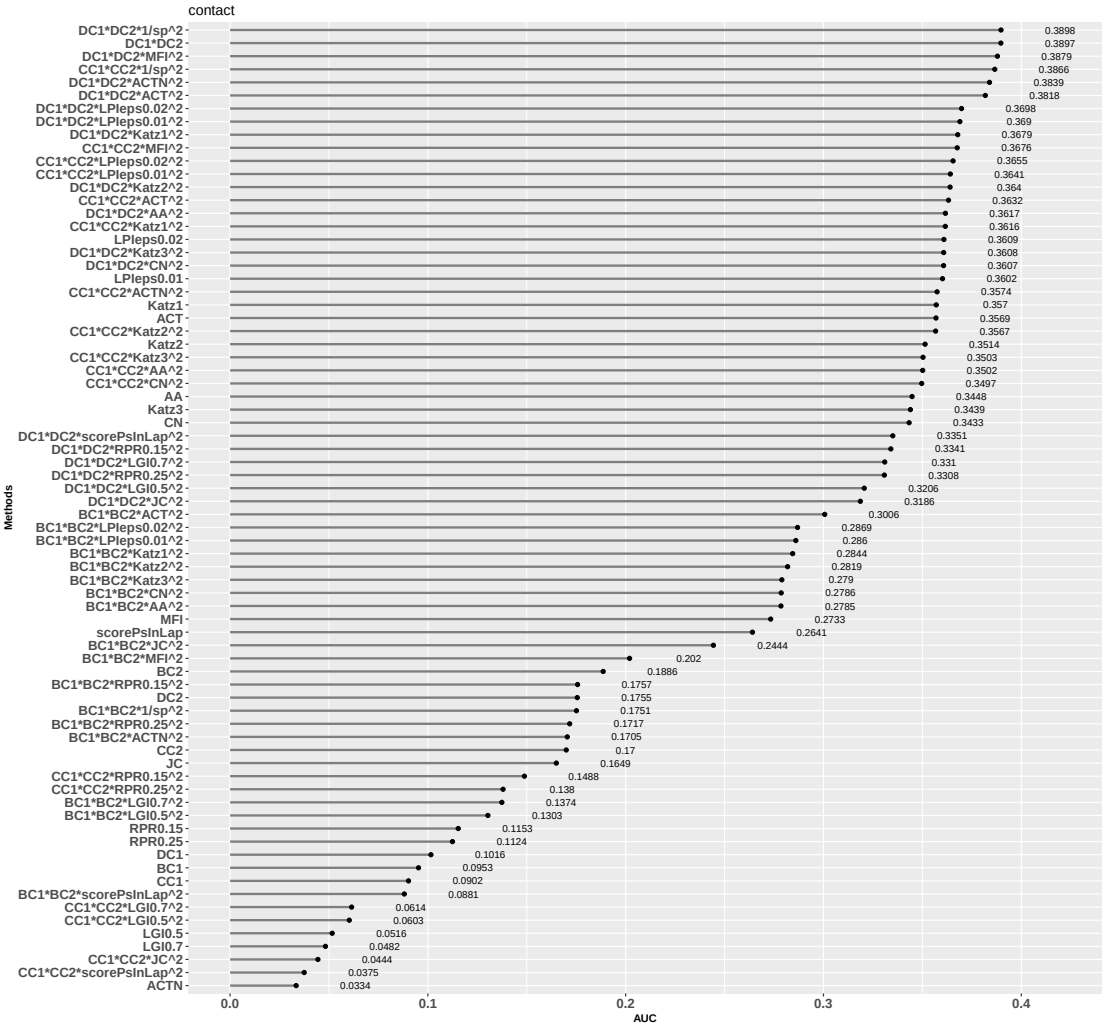


FIGURE A.20: PR AUC of *contact* dataset

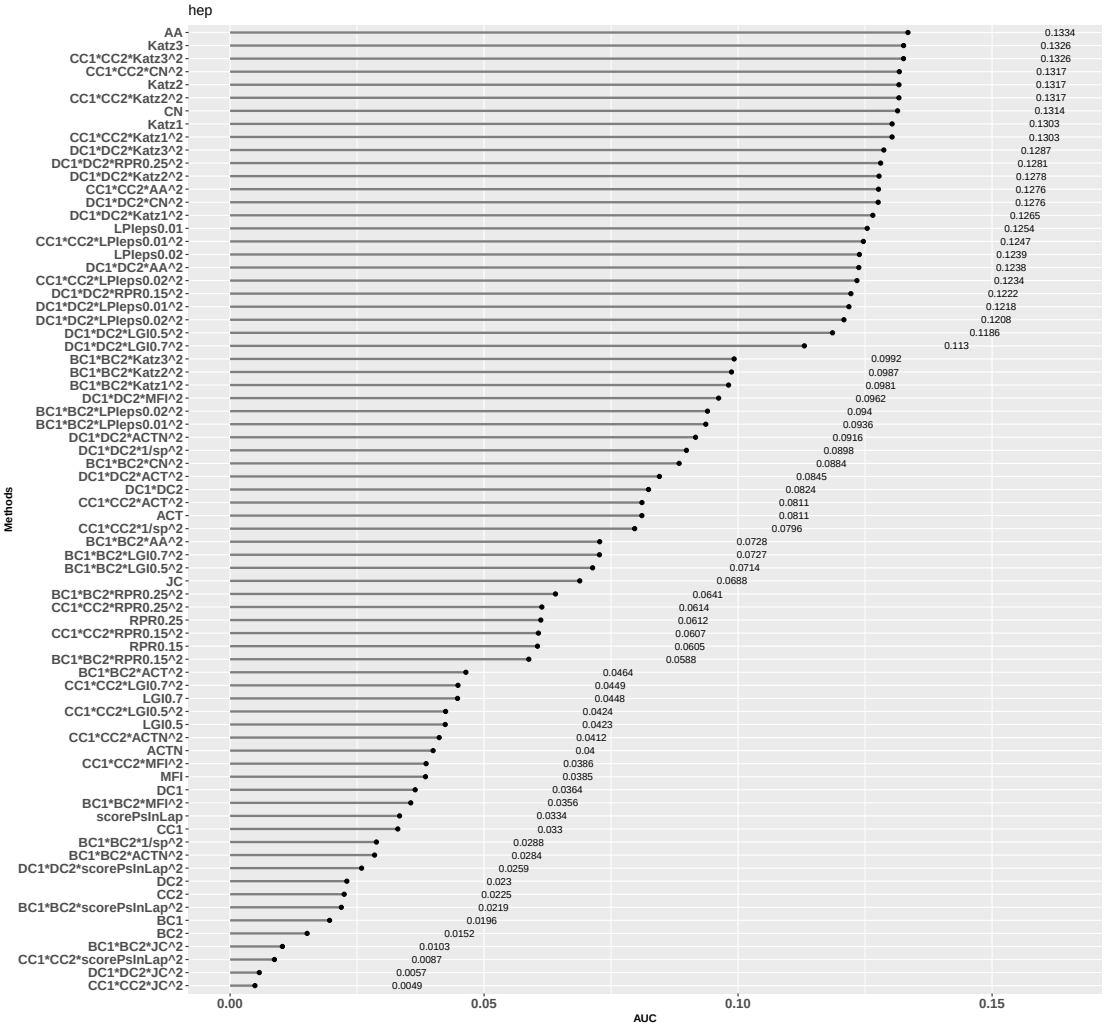


FIGURE A.21: PR AUC of *hep-th* dataset

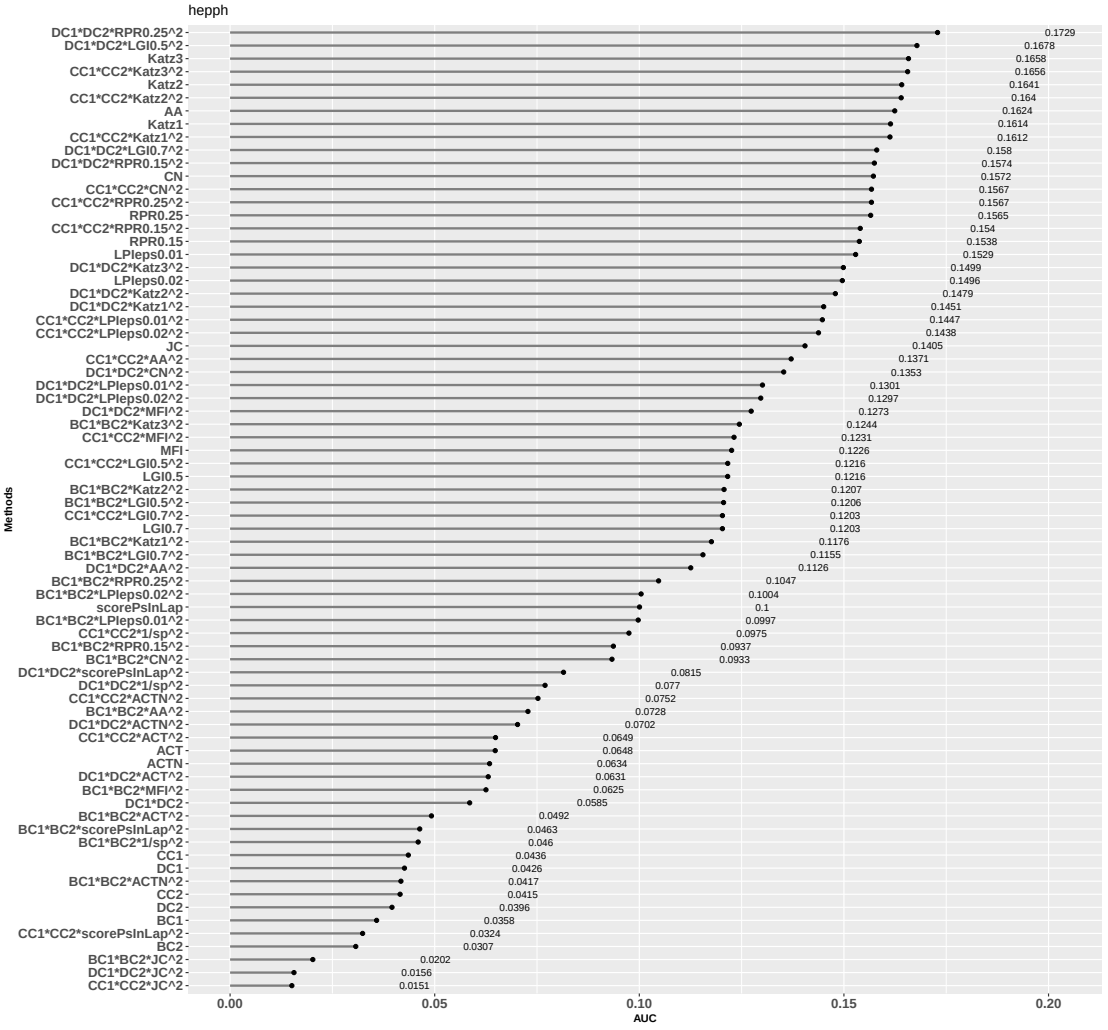


FIGURE A.22: PR AUC of *hep-ph* dataset

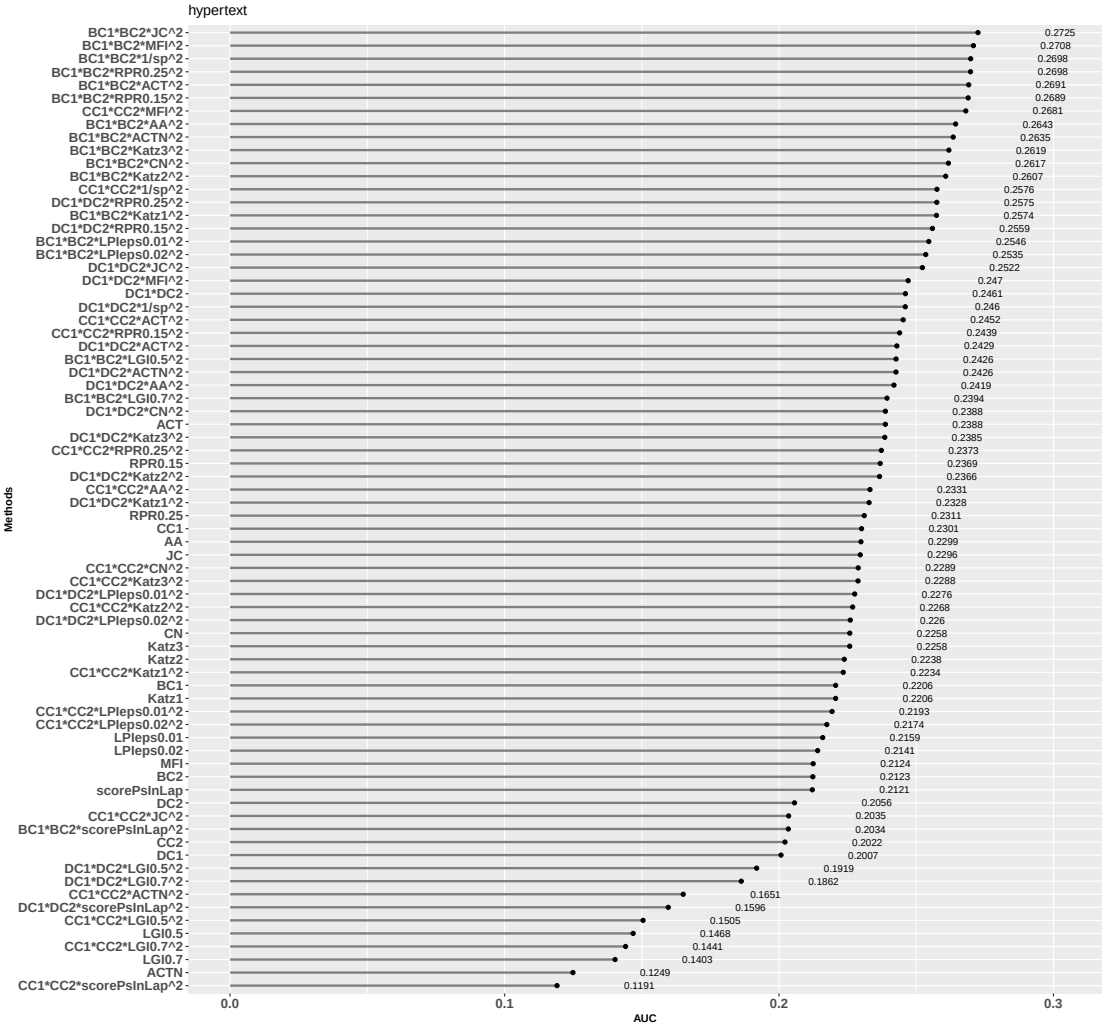


FIGURE A.23: PR AUC of *hypertext* dataset

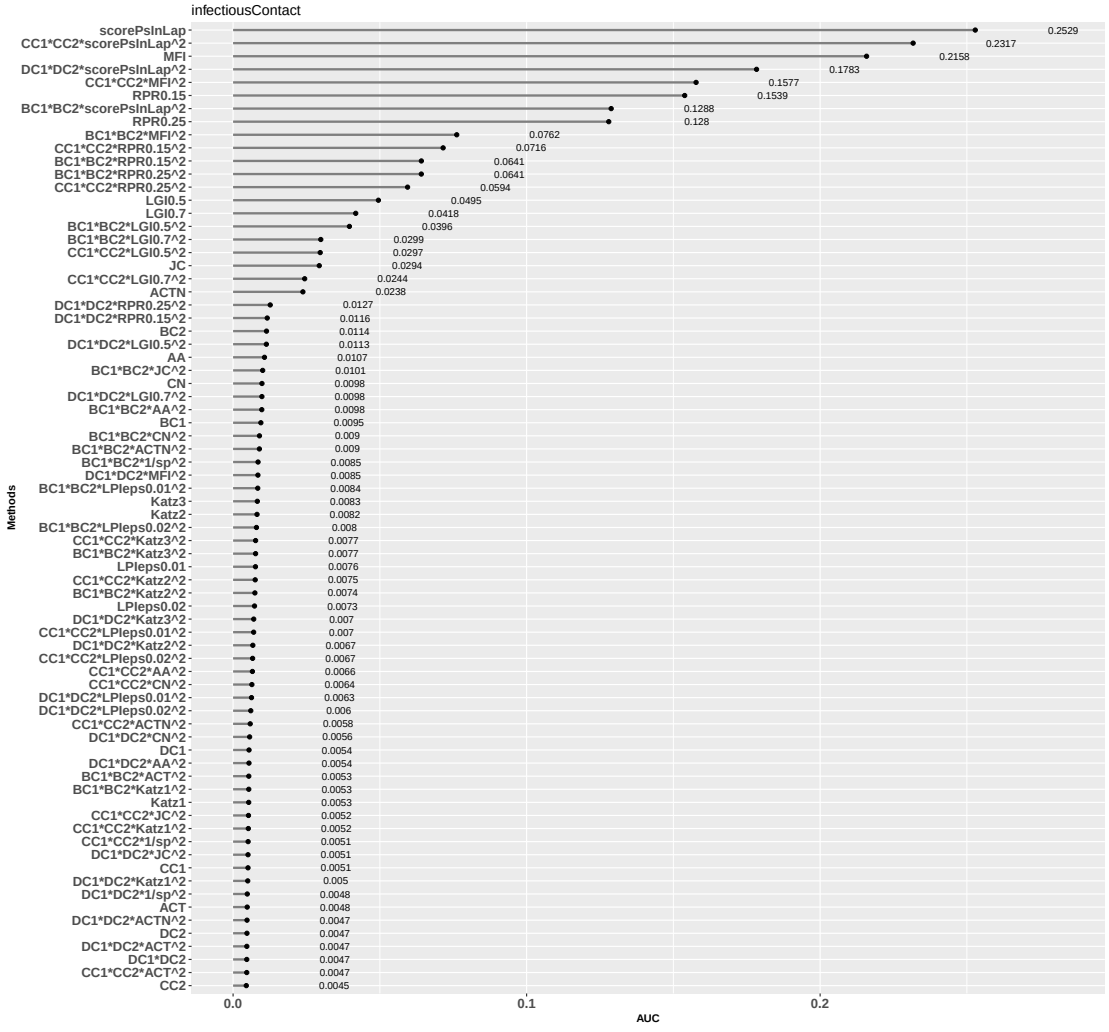


FIGURE A.24: PR AUC of *infectiousContact* dataset

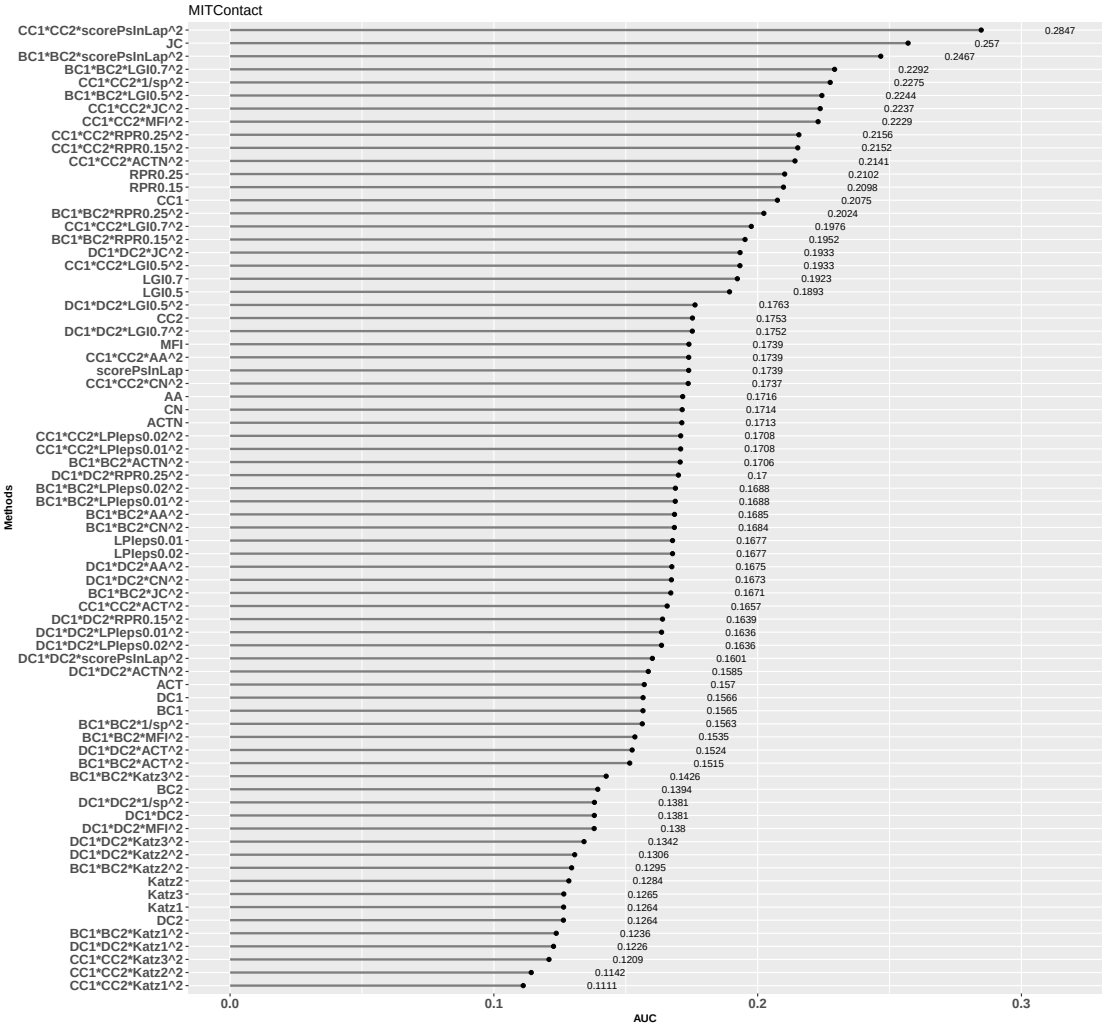


FIGURE A.25: PR AUC of *MITContact* dataset

A.3.2 Heatmap and Barplots

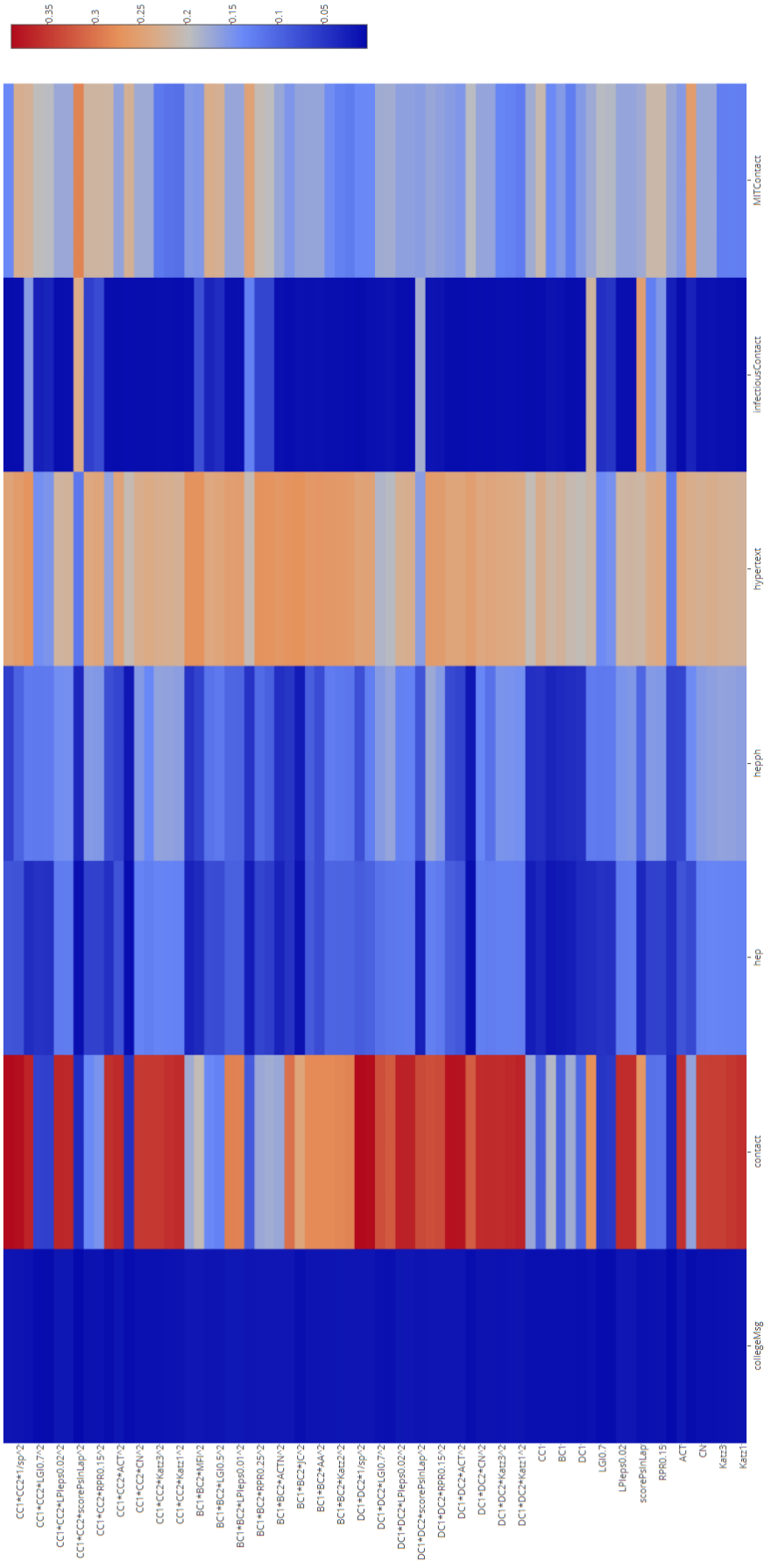


FIGURE A.26: PR AUC heatmap of 7 datasets

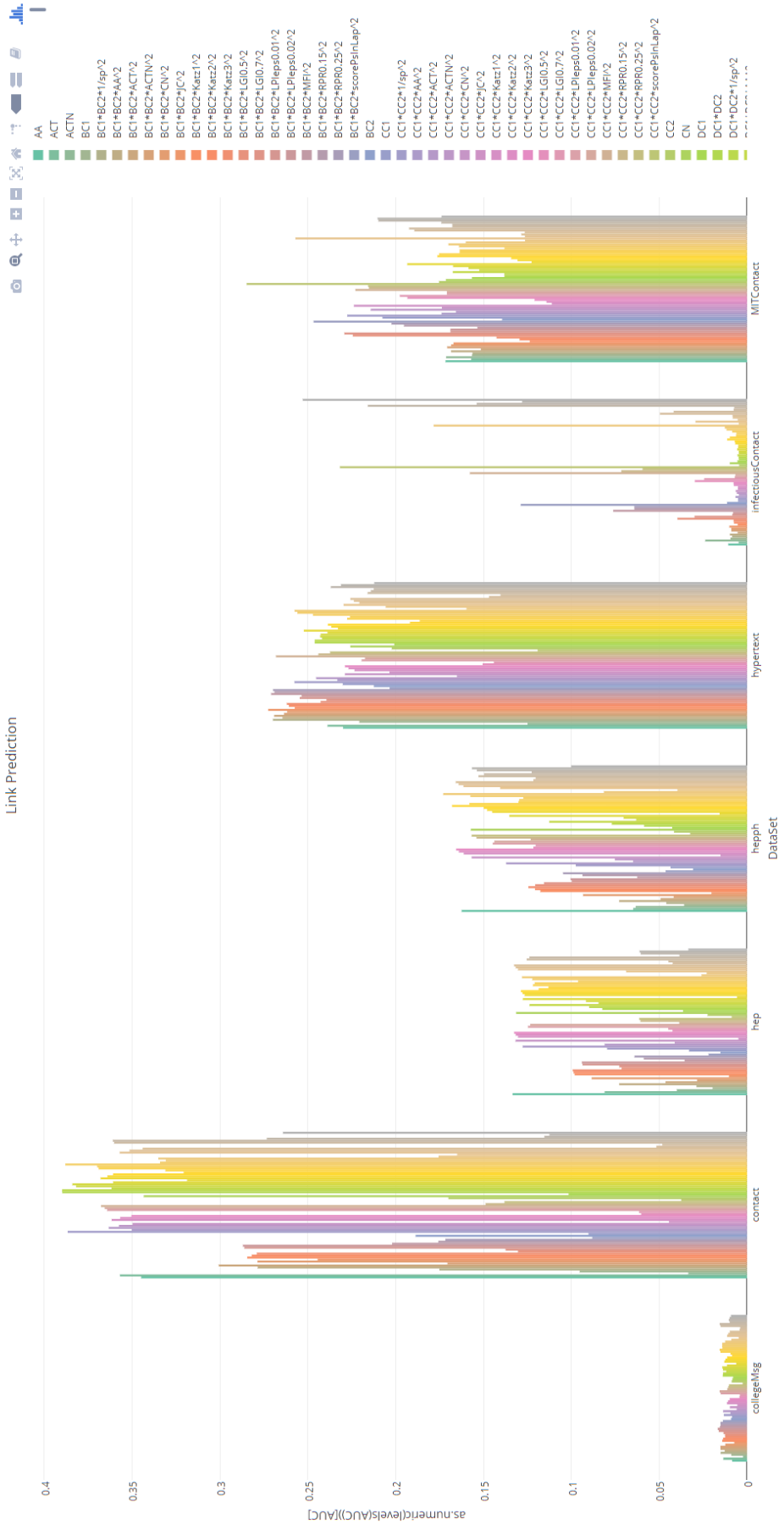


FIGURE A.27: PR AUC barplot of 7 datasets

A.4 AUC of Receiver-Operating-Characteristic for 7 Datasets

A.4.1 Individual Barplots

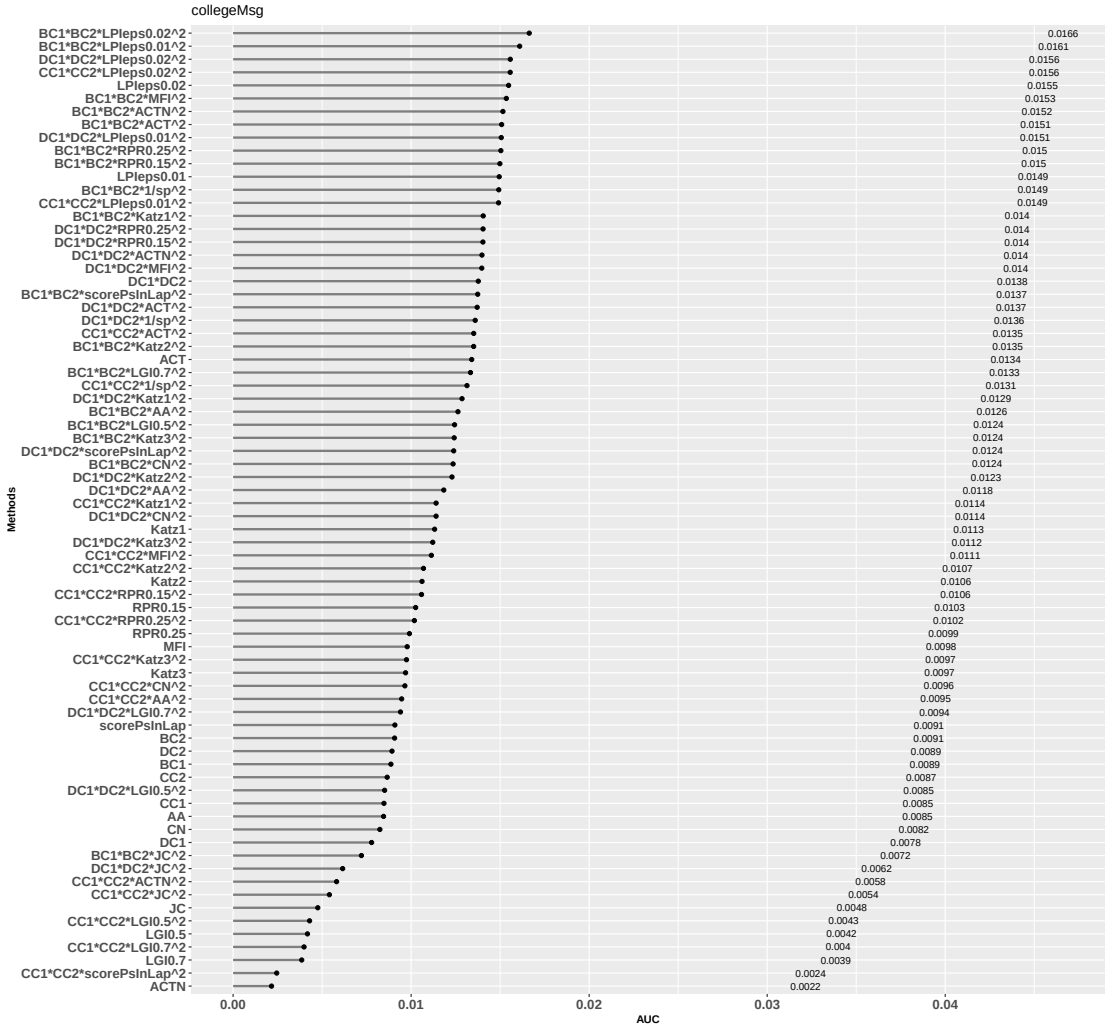


FIGURE A.28: PR AUC of *collegeMsg dataset*

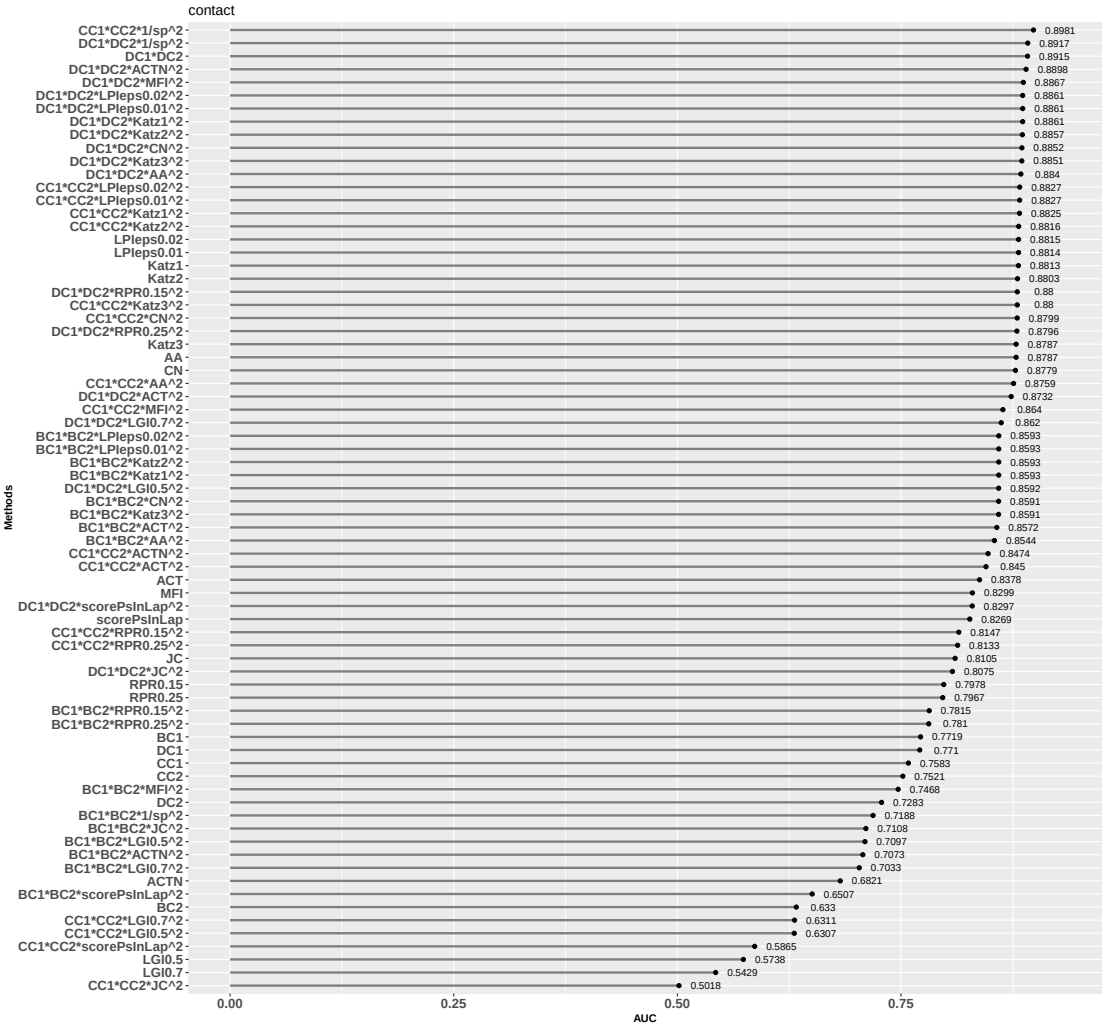


FIGURE A.29: ROC AUC of *contact* dataset

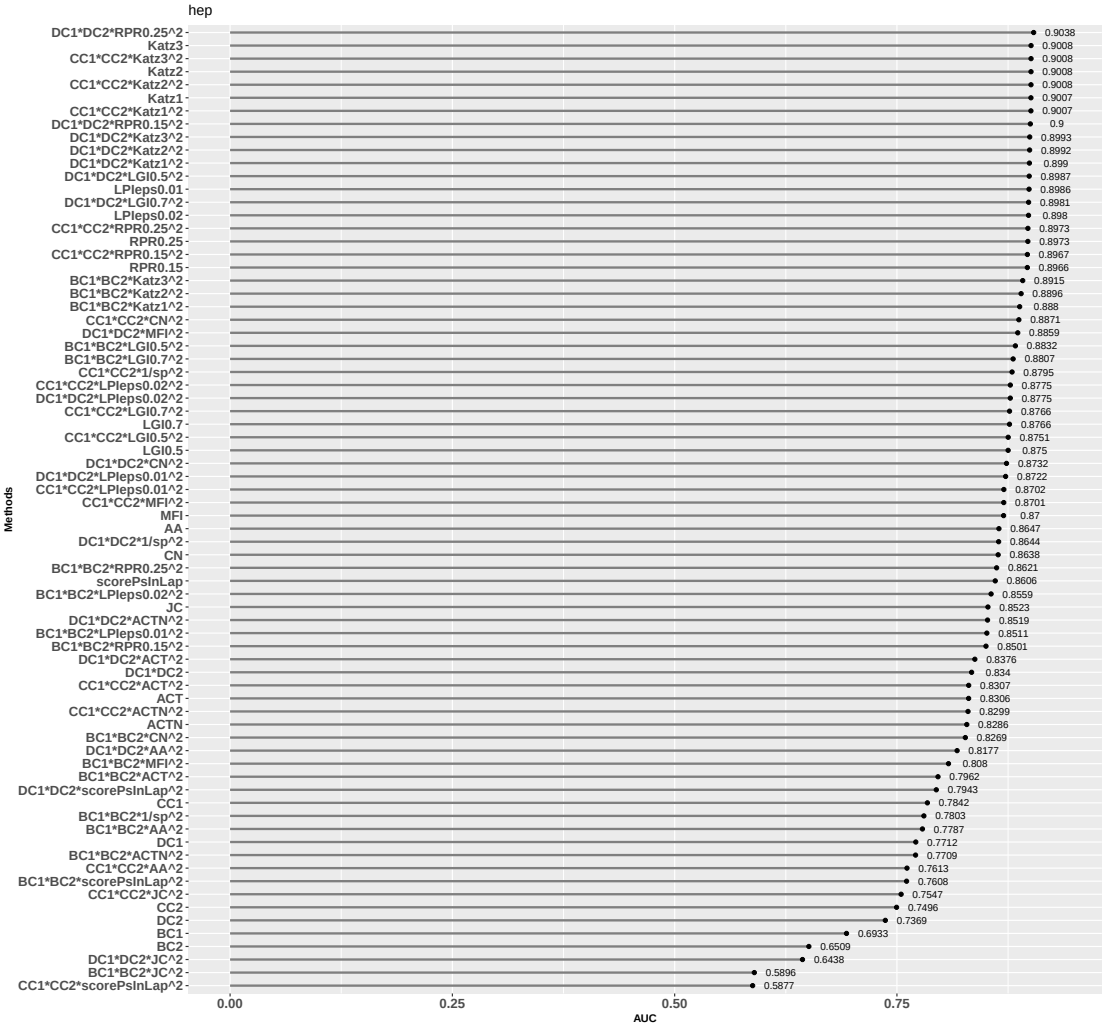


FIGURE A.30: ROC AUC of *hep-th* dataset

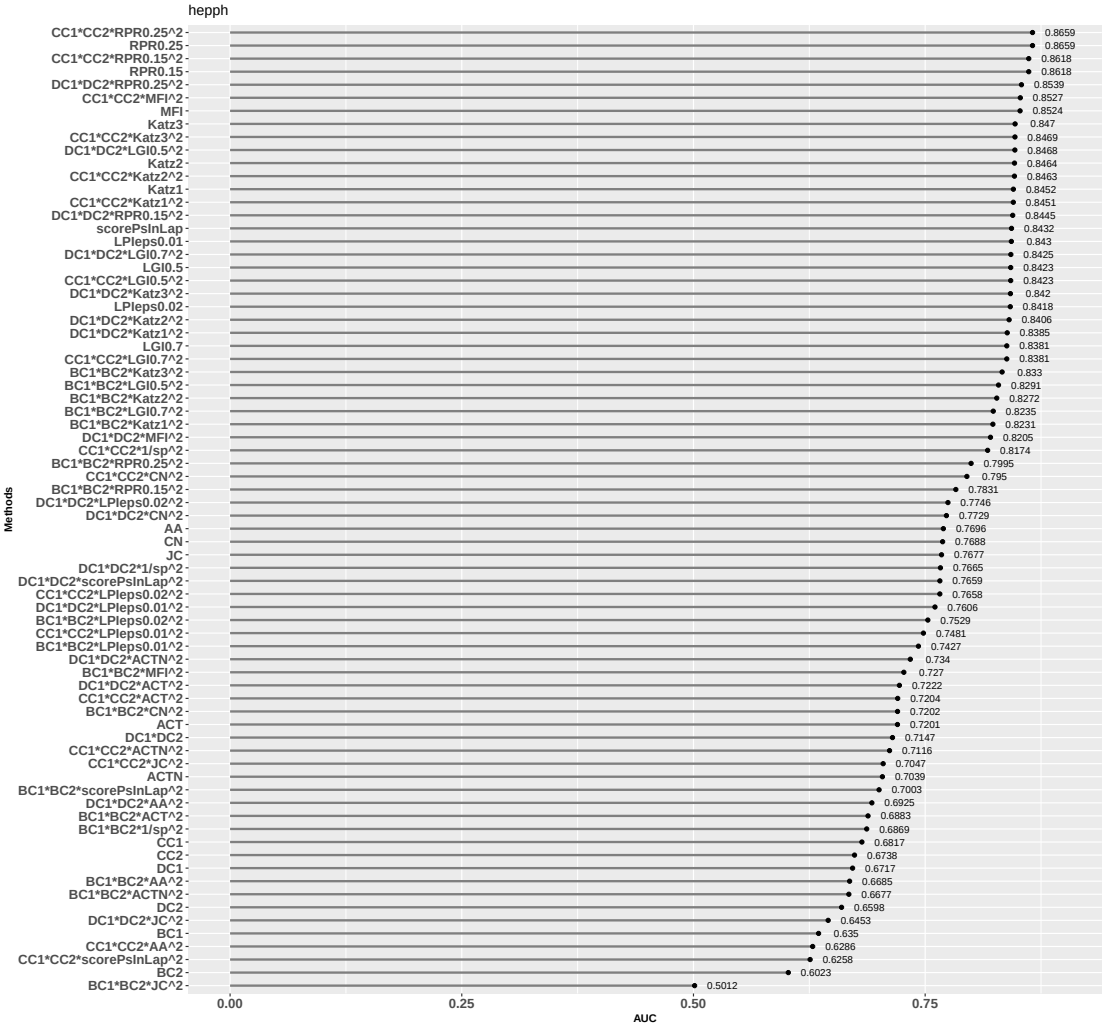


FIGURE A.31: ROC AUC of *hep-ph* dataset

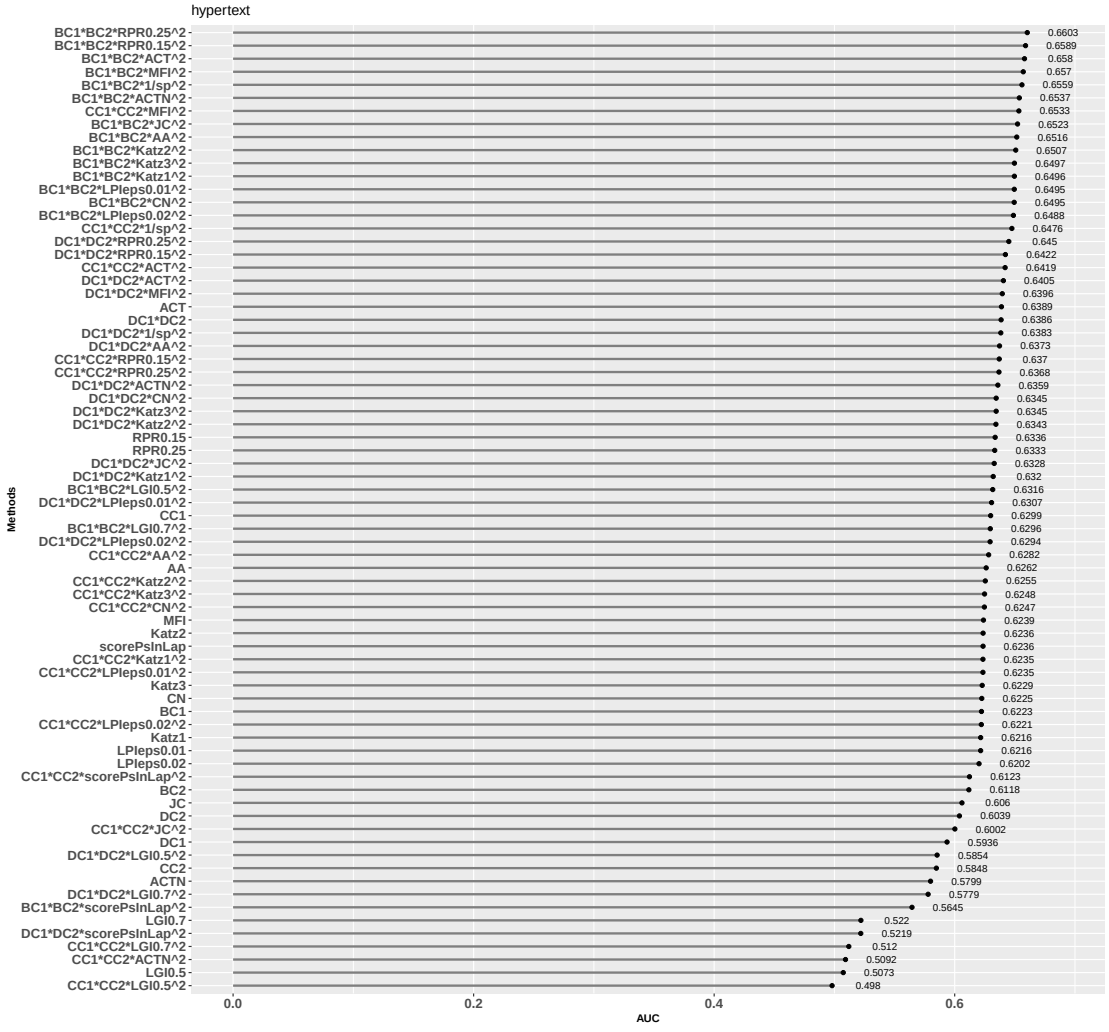


FIGURE A.32: ROC AUC of *hypertext* dataset

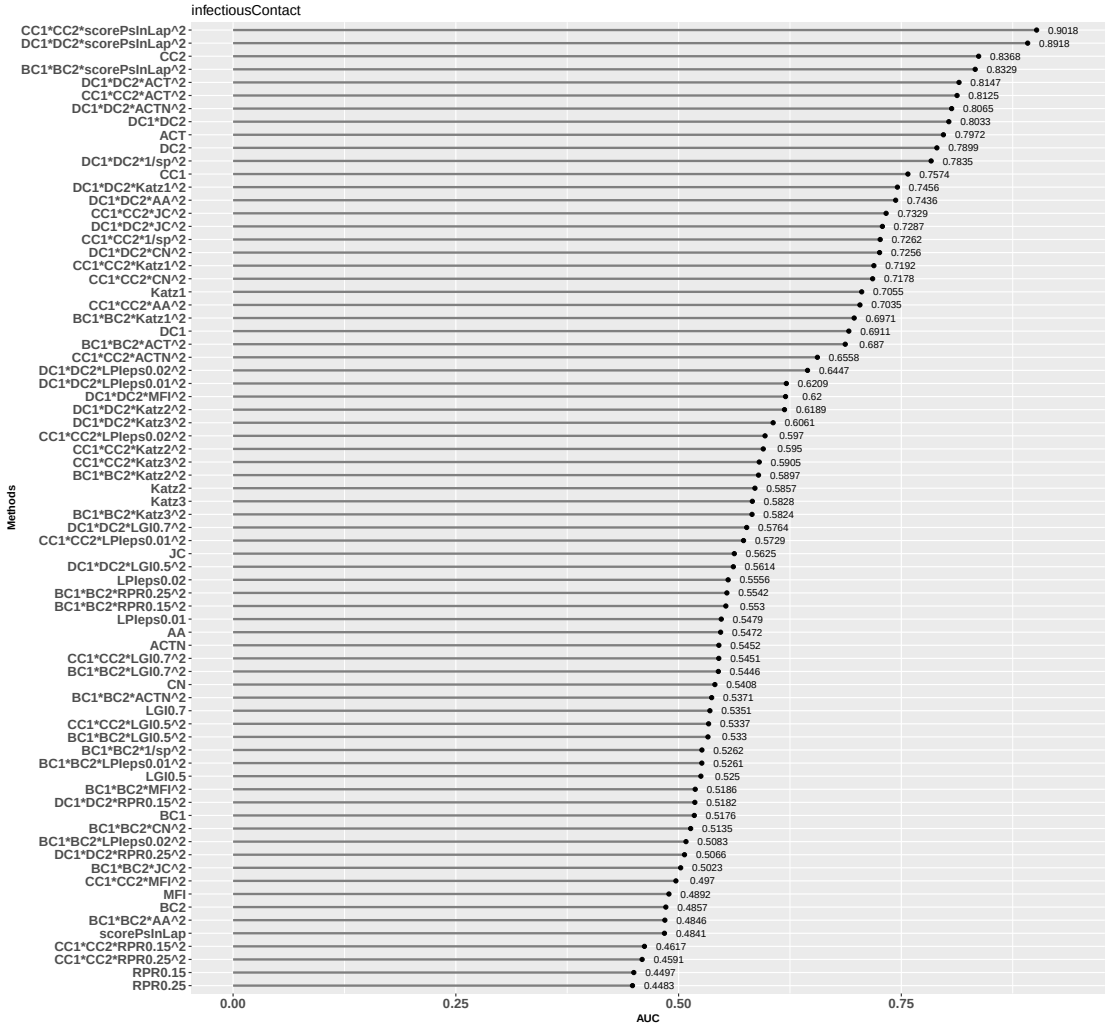


FIGURE A.33: ROC AUC of *infectiousContact* dataset

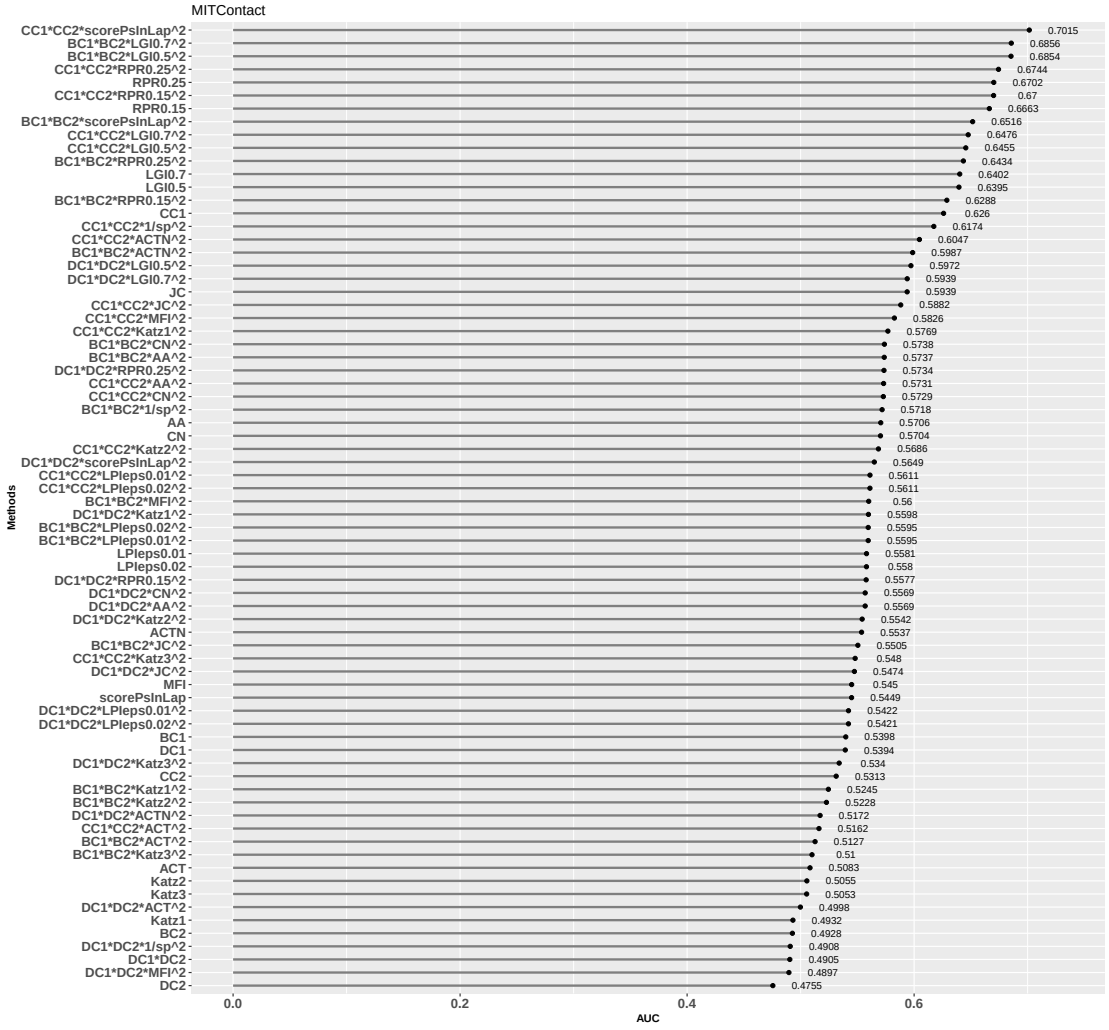


FIGURE A.34: ROC AUC of *MITContact* dataset

A.4.2 Heatmap and Barplots

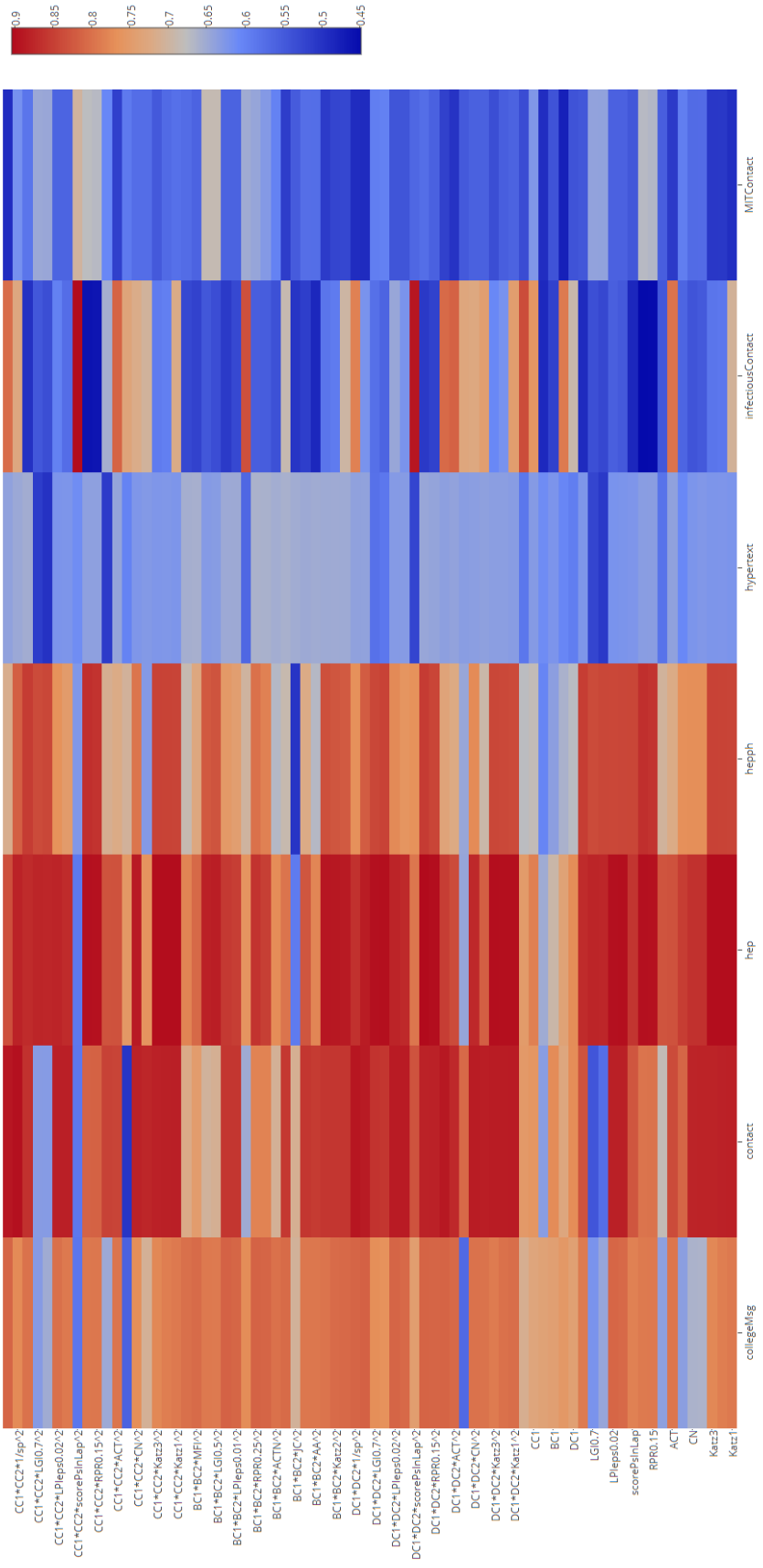


FIGURE A.35: ROC AUC heatmap of 7 datasets

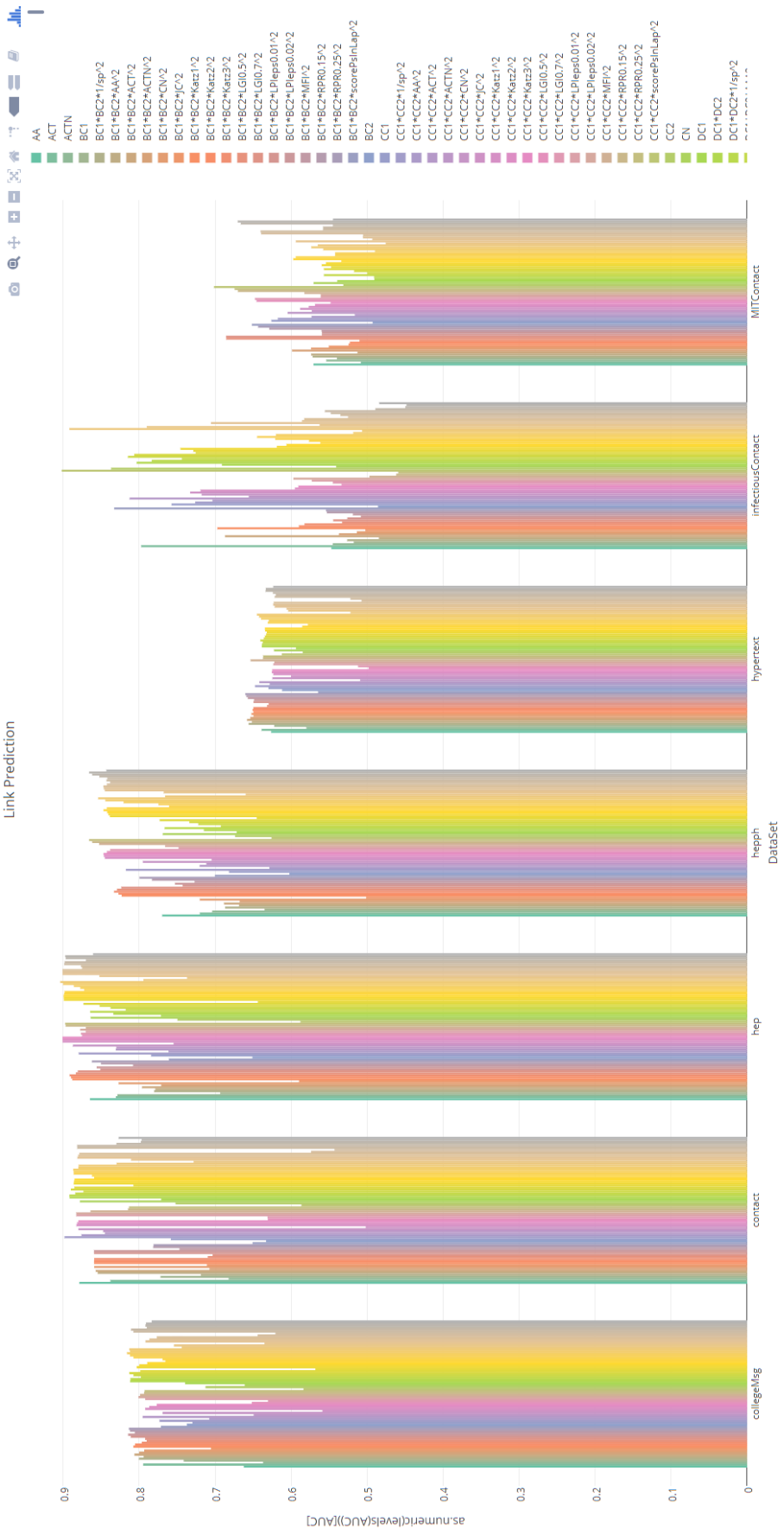


FIGURE A.36: ROC AUC barplot of 7 datasets

A.4.3 ROC AUC Ranked

Predictors	collegeMsg	rank(collegeMsg)	contact	rank(contact)	hep	rank(hep)	hepph	rank(hepph)	hypertext	rank(hypertext)	infectiousContact	rank(infectiousContact)	MITContact	rank(MITContact)
Katz1	0.791423633	39	0.881250389	19	0.900694574	6	0.845179608	13	0.621591957	55	0.705491306	21	0.493229451	69
Katz2	0.78621584	45	0.880277458	20	0.900767495	4	0.846400116	11	0.623603028	47	0.585679217	36	0.505505695	66
Katz3	0.776491836	48	0.878706814	25	0.900825839	2	0.847028363	8	0.622850093	51	0.582770727	37	0.50530841	67
AA	0.662028215	63	0.878683701	26	0.864703115	39	0.769634072	39	0.626170568	42	0.547150033	47	0.570602723	31
CN	0.660842181	64	0.877938882	27	0.863831123	41	0.768838409	40	0.622531068	52	0.540751792	51	0.570356912	32
JC	0.6348254	69	0.810494734	49	0.852334028	45	0.767654427	41	0.606032539	60	0.562523702	41	0.593859332	21
ACT	0.794570247	30	0.837778054	43	0.830614571	52	0.72010683	54	0.638887766	22	0.797229838	9	0.508280419	65
ACTN	0.636716884	68	0.682126631	66	0.828576186	54	0.703880941	58	0.579914812	66	0.545202686	48	0.553706977	47
RPR0.15	0.791491805	38	0.797822009	51	0.896641641	19	0.861768809	4	0.633624703	32	0.449738153	73	0.666255125	7
RPR0.25	0.790762269	40	0.796660713	52	0.897256248	17	0.865851037	2	0.6332894	33	0.448252684	74	0.670186514	5
scorePsInLap	0.783366944	46	0.826922013	46	0.860563744	43	0.843169604	16	0.623571593	48	0.484131201	70	0.544903911	52
LPleps0.01	0.807325474	14	0.881449771	18	0.898583216	13	0.842980361	17	0.621582227	56	0.547904365	46	0.558063166	41
LPleps0.02	0.810331766	12	0.881465108	17	0.897991403	15	0.841841209	22	0.620236336	57	0.555575831	43	0.558012254	42
LGI0.5	0.64393342	67	0.573809024	72	0.875043133	33	0.842311044	19	0.507332887	73	0.525025896	58	0.639497495	13
LGI0.7	0.620414916	71	0.54285086	73	0.876561355	31	0.838059898	25	0.522021864	69	0.535140974	53	0.64018163	12
MFI	0.789726197	42	0.829858087	44	0.869950839	38	0.852396475	7	0.623925233	46	0.489156462	67	0.544967551	51
DC1	0.739507171	57	0.770973743	56	0.77118249	63	0.671685478	65	0.593635301	63	0.691054288	24	0.539364805	56
DC2	0.754045609	54	0.728253356	60	0.736938474	69	0.659795743	68	0.603938765	61	0.789890729	10	0.475549336	74
BC1	0.741083646	56	0.771903687	55	0.693282977	70	0.634993479	70	0.622256015	53	0.517617364	61	0.539765739	55
BC2	0.736834546	58	0.632999245	68	0.650948027	71	0.602327892	73	0.611787892	59	0.48567401	68	0.492768058	70
CC1	0.729762916	59	0.758282224	57	0.784245521	60	0.681738308	63	0.629878363	38	0.757359627	12	0.629957192	15
CC2	0.712437893	60	0.75209664	58	0.74957383	68	0.673775111	64	0.584762676	65	0.836763149	3	0.531327015	58
DC1*DC2*Katz1^2	0.802978 94	20	0.886061471	8	0.898990168	11	0.838535465	24	0.632007314	35	0.74558325	13	0.559754411	38
DC1*DC2*Katz2^2	0.799515639	24	0.88569295	9	0.899210922	10	0.840586953	23	0.634278844	31	0.6188726	30	0.554225647	46
DC1*DC2*Katz3^2	0.789132682	43	0.885075363	11	0.899325874	9	0.842012993	21	0.634480924	30	0.606101747	31	0.53397048	57
DC1*DC2*AA^2	0.797648969	27	0.884027043	12	0.817652877	56	0.692492727	60	0.637278236	25	0.743619358	14	0.556877863	45
DC1*DC2*CN^2	0.798292754	26	0.88519806	10	0.873244964	34	0.772930497	38	0.634488034	29	0.72556907	18	0.556885818	44
DC1*DC2*JC^2	0.567855959	73	0.807491048	50	0.643759321	72	0.645321354	69	0.632838649	34	0.728735547	16	0.547376343	50
DC1*DC2*ACT^2	0.807195456	15	0.873188068	29	0.837630777	49	0.722201883	51	0.640496174	20	0.814727981	5	0.499821011	68
DC1*DC2*ACTN^2	0.812583826	5	0.889845619	4	0.851896522	46	0.733970457	49	0.635893239	28	0.806450195	7	0.51715349	61
DC1*DC2*RPR0.15^2	0.811833042	8	0.880010248	21	0.900042059	8	0.844469183	15	0.642155476	18	0.518247691	60	0.557740191	43
DC1*DC2*RPR0.25^2	0.812684355	4	0.879584699	24	0.903769756	1	0.853917529	5	0.64500705	17	0.506631572	64	0.573445065	27
DC1*DC2*scorePsInLap^2	0.74394484	55	0.829690891	45	0.794270134	59	0.76586516	43	0.521911468	70	0.891786836	2	0.5649427	34
DC1*DC2*LPleps0.01^2	0.807186206	16	0.886082424	7	0.872241047	35	0.760576261	45	0.630669843	37	0.620916297	28	0.542173735	53
DC1*DC2*LPleps0.02^2	0.812031818	6	0.886113963	6	0.877478233	29	0.774595062	37	0.629420689	40	0.644720468	27	0.542148279	54
DC1*DC2*LGI0.5^2	0.76515432	53	0.859183827	36	0.898749617	12	0.84679619	10	0.585378084	64	0.561446664	42	0.597198865	19
DC1*DC2*LGI0.7^2	0.769110143	51	0.862001092	31	0.898062101	14	0.842471342	18	0.577898876	67	0.57643161	39	0.593916609	20
DC1*DC2*MFI^2	0.815668178	1	0.886740189	5	0.885882992	24	0.820489626	32	0.639570348	21	0.620043069	29	0.489702179	73
DC1*DC2*1/sp^2	0.81116952	10	0.891730132	2	0.864370868	40	0.766485859	42	0.638329426	24	0.783453073	11	0.490834979	71
BC1*BC2*Katz1^2	0.807608742	13	0.859255328	35	0.888025843	22	0.823078816	31	0.649629071	12	0.697059588	23	0.524467369	59
BC1*BC2*Katz2^2	0.805088961	19	0.859263752	34	0.889641356	21	0.827237785	29	0.650749493	10	0.589694591	35	0.522766578	60
BC1*BC2*Katz3^2	0.796252731	28	0.859093101	38	0.891477673	20	0.833029118	27	0.649669487	11	0.582410795	38	0.51002894	64
BC1*BC2*AA^2	0.793793627	31	0.854425668	40	0.778684867	62	0.668470962	66	0.651594112	9	0.484590078	69	0.573669397	26
BC1*BC2*CN^2	0.793214907	33	0.859120319	37	0.826901638	55	0.720201072	53	0.649461419	14	0.51354465	62	0.573822134	25
BC1*BC2*JC^2	0.705174291	62	0.710788774	62	0.58958455	73	0.501227524	74	0.652261351	8	0.502319688	65	0.550450177	48
BC1*BC2*ACT^2	0.80586703	17	0.857158472	39	0.796170339	58	0.688342959	61	0.65804159	3	0.687038751	25	0.512727296	63
BC1*BC2*ACTN^2	0.799936747	23	0.707346362	64	0.770936983	64	0.667657458	67	0.653731671	6	0.537102351	52	0.598729418	18
BC1*BC2*RPR0.15^2	0.812000822	7	0.781513106	53	0.850140792	48	0.783107275	36	0.658907166	2	0.553020622	45	0.628836127	14
BC1*BC2*RPR0.25^2	0.813295809	3	0.781031177	54	0.862096144	42	0.799545143	34	0.660332205	1	0.554190117	44	0.643412974	11
BC1*BC2*scorePsInLap^2	0.771175021	50	0.650737734	67	0.760835024	66	0.700259215	59	0.564493857	68	0.832866022	4	0.65157009	8
BC1*BC2*LPleps0.01^2	0.810969302	11	0.859282762	33	0.851104428	47	0.742730044	48	0.649532147	13	0.526057193	57	0.559533261	40
BC1*BC2*LPleps0.02^2	0.81435901	2	0.85934411	32	0.855882307	44	0.752896301	46	0.648812892	15	0.508299927	63	0.559538034	39
BC1*BC2*LGI0.5^2	0.789918178	41	0.709705243	63	0.883215071	25	0.829133765	28	0.631639453	36	0.532952834	55	0.685355217	3
BC1*BC2*LGI0.7^2	0.791841636	35	0.703344903	65	0.880682809	26	0.823481956	30	0.629587218	39	0.544615019	50	0.685606597	2
BC1*BC2*MFI^2	0.805534761	18	0.746839919	59	0.807982855	57	0.726999747	50	0.657000877	4	0.518648823	59	0.559981926	37
BC1*BC2*1/sp^2	0.800129557	22	0.718769657	61	0.780291941	61	0.686910265	62	0.655919375	5	0.526168789	56	0.571788822	30
CC1*CC2*Katz1^2	0.791719549	37	0.882504787	15	0.900686509	7	0.845112776	14	0.623487393	49	0.719201811	19	0.576891196	24
CC1*CC2*Katz2^2	0.786586922	44	0.881566635	16	0.900765045	5	0.846335556	12	0.625451686	43	0.595033717	33	0.568571797	33
CC1*CC2*Katz3^2	0.776639666	47	0.87998735	22	0.900813851	3	0.84688539	9	0.624830477	44	0.590536595	34	0.54803343	49
CC1*CC2*AA^2	0.707704314	61	0.875851746	28	0.761296576	65	0.628621673	71	0.628220745	41	0.703528225	22	0.573117316	28
CC1*CC2*CN^2	0.768911008	52	0.879904184	23	0.887125909	23	0.79502564	35	0.624691641	45	0.717774816	20	0.572865937	29
CC1*CC2*JC^2	0.558703236	74	0.501825541	74	0.754676066	67	0.704663358	57	0.600151074	62	0.732898852	15	0.588212037	22
CC1*CC2*ACT^2	0.795056478	29	0.845042633	42	0.830688734	51	0.720356668	52	0.64191036	19	0.812519404	6	0.516184564	62
CC1*CC2*ACTN^2	0.649013945	66	0.847363712	41	0.829940714	53	0.711569674	56	0.50921298	72	0.655780468	26	0.604671846	17
CC1*CC2*RPR0.15^2	0.793400636	32	0.814666845	47	0.896689227	18	0.861778341	3	0.637045844	26	0.461673166	71	0.6700337 77	6
CC1*CC2*RPR0.25^2	0.79259886	34	0.81332172	48	0.897305078	16	0.865855814	1	0.636811206	27	0.45905486	72	0.674350191	4
CC1*CC2*scorePsInLap^2	0.583609585	72	0.586459068	71	0.587709356	74	0.625842576	72	0.6122712	58	0.901816594	1	0.701465799	1
CC1*CC2*LPleps0.01^2	0.791783574	36	0.882693152	14	0.870248805	36	0.748137162	47	0.623485896	50	0.572874544	40	0.561103589	35
CC1*CC2*LPleps0.02^2	0.800592224	21	0.882705249	13	0.877521639	28	0.765814141	44	0.622126721	54	0.597025509	32	0.561081315	36
CC1*CC2*LGI0.5^2	0.651667267	65	0.630674493	70	0.875102683	32	0.842308293	20	0.49804394	74	0.533668318	54	0.645513905	10
CC1*CC2*LGI0.7^2	0.630210727	70	0.631064832	69	0.876627328	30	0.838054053	26	0.511963515	71	0.545099848	49	0.647637905	9
CC1*CC2*MFI^2	0.798634388	25	0.863971579	30	0.870146681	37	0.852672156	6	0.653349964	7	0.496991916	66	0.582625201	23
CC1*CC2*1/sp^2	0.772911413	49	0.898138643	1	0.879473979	27	0.817410474	33	0.647560182	16	0.726247329	17	0.617368913	16
DC1*DC2	0.811538855	9	0.891498564	3	0.833976232	50	0.714749357	55	0.638588014	23	0.803304886	8	0.490494502	72

TABLE A.1: ROC AUC Ranked

A.5 More Detail Properties of the 7 Datasets

dataset	Vertices	Edges	GlobalClusteringCoefficent	ClusteringCoefficentSD	AvgGeodesicPath	MeanDegreeCentrality	SDDegreeCentrality	MeanClosenessCentrality	SDClosenessCentrality	MeanBetweenessCentrality	SDBetweenessCentrality
collegeMsg-Training	1260	29918	0.0557	0.1363	3.0564	47.4889	95.5567	0.000158	0.000014	1290.3714	4870.9555
collegeMsg-Full	1899	59835	0.0568	0.1255	3.0552	63.0174	131.768	0.000058	0.000004	1938.0453	7689.691
contact-Training	131	14122	0.6798	0.3708	2.0957	215.6031	317.8374	0.003812	0.000769	71.2214	156.4402
contact-full	274	28244	0.5664	0.4063	2.424	206.1606	461.7712	0.00155	0.000271	194.3796	689.7984
hep-Training	2293	50943	0.4671	0.2988	3.4859	44.4335	74.8863	0.000002	0	2293.3314	10927.3927
hep-full	6776	290484	0.3327	0.3237	3.2238	85.7391	186.0342	0	0	6859.9293	54183.6514
hepph-Training	2157	43627	0.4092	0.3536	3.6496	40.4516	72.7137	0.000008	0.000001	2709.898	9172.3323
hepph-full	10324	955423	0.3509	0.3374	2.9455	185.0878	336.9404	0.000003	0	9989.4466	53544.1013
hypertext-Training	110	10409	0.3961	0.0829	1.7766	189.2545	184.5619	0.005209	0.000515	42.3273	93.1187
hypertext-full	113	20818	0.4952	0.0425	1.6563	368.4602	324.4265	0.005451	0.000606	36.7522	72.9689
infectiousContact-Training	241	8649	0.4164	0.0598	3.3572	71.7759	52.3614	0.001293	0.000229	282.8631	665.9181
infectiousContact-full	410	17298	0.4357	0.0965	3.6309	84.3805	58.9714	0.000688	0.000098	538.0098	1786.4198
MITContact-Training	96	543202	0.6892	0.1083	1.5169	11316.7083	8550.6648	0.007068	0.000942	24.5521	52.1285
MITContact-full	96	1086405	0.7254	0.1083	1.4447	22633.4375	21929.0061	0.007426	0.001005	21.125	44.2452

TABLE A.2: Properties of the 7 datasets

A.6 VirtualSoc

```
from Transfer import WriteToFile
from Networks import RandomSocialGraphAdvanced
import os
import sys
sys.setrecursionlimit(100000000)  # this is needed for a large network
# if keep history is true and   WriteToFile(G).easySaveEverything(folder)
# is needed. This is due to python's deep copy problems. One way to avoid this is to write to file
# after each of the evolution and setting keepHistory = False
if __name__ == '__main__':
    folder = 'D:/cpu_virtualsoc_single2/'  # just a folder name to write the networks
    os.makedirs(folder)    # create the folder
    G = RandomSocialGraphAdvanced(labelSplit=[10, 20, 30, 40], # 1st 10 with the same sDNA-1, 2nd 20 with the same sDNA-2 (each 10 with
                                  # same sDNA)

                                  connectionPercentageWithMatchedNodes=5, # the top fraction of the nodes to be connected, see paper
                                  connectionPercentageWithMatchedNodesWithRandomness=1, # this param is not implemented
                                  explorationProbability=0.3, # exploration probablity, see paper
                                  addTraidtionalFeatures=False, # weather to add traditional age, gender, location features
                                  additionalFeatureLen=1000, # feature len. If the addTraidtionalFeatures is true then 1003 features for each
                                  # nodes
                                  npDistFunc=['np.random.randint(3, high=500)','np.random.randint(3, high=500)'], # the distribution function
                                  # from where the features will be generated. Usually it is a numpy random number generator
                                  #function. However, the size parameter should not be filled while passing
                                  # the function as a string.
                                  #  If the number of features equals to the number of
                                  # passed distribution functions then each of them will be used with the corresponding
                                  # feature. Otherwise a features will be generated based on a randomly selected passed function.
                                  # A feature for all the nodes for a single network will be generated from a single function.
                                  popularityPreferenceIntensity=0.5, # preferential attachment parameter
                                  mutualPreferenceIntensity=[0.9, 0.3, 0.1], # the path length preference parameter
                                  genFeaturesFromSameDistforAllLabel=False, # Wheter or not to generate features from the same dist.
                                  # for all labels. Otherwise different labels will have a different dist. for the selected feature dist.
                                  #numpy function
                                  keepHistory=True, # whether or past history for each evaliation is kept.  If
                                  #   WriteToFile(G).easySaveEverything(folder) is used then this should be set to true.
                                  # However, for a large network the recursion limit needs to be set high for this to be used.
                                  # This is for Python's complicated deep copy procedure. A easy way to solve this is to
                                  #use the WriteToFile after each evolution of the network and settin this as param as False.
                                  useGPU=False, # whether GPU should be used. Only valid for the GPU version
                                  numberofProcesses=None, # The number of processed should be used. Set it to None
                                  # if that is not desired.
                                  createInGPUMem=False, # whether the features and sDNAs should be created
                                  #in GPU memory. This is faster as the transition from system memory to
                                  # video memory is not required if useGPU is set to True. Only
                                  # valid for the the GPU version
                                  socialiseOnCreation=True # if  a socialisation is performed right after instantiate the network.
                                  # if False, the socialise agolrithm can be called later using the socialise function.
                                  # see paper for more information.
                                  )
    G.socialise() # a socialise method to socialise the graph. The exp explorationProbability=None, connectionPercentageWithMatchedNodes=None
    # can also be reassigned here again. If not assigned then it will use the same value for these two parameters when
    # the network was first created.
    G.mutateDNA(mutationIntensity=0.0001, mutatePreference=True, mutatePreferenceProbability=True) # wheather sDNA should be mutated
    G.socialise()
    G.mutateDNA(mutationIntensity=0.01, mutatePreference=True, mutatePreferenceProbability=True)
    G.socialise()
    WriteToFile(G).easySaveEverything(folder) # save everything to a folder
```

FIGURE A.37: *VirtualSoc*, Simulation of a single network

```
import sys
from MultipleNetworksSimulation import Simulate, SalibPreprocessGetParamsForSobol
import os
sys.setrecursionlimit(100000000)
if __name__ == '__main__':
    folder = 'H:/cpu_Networksim_test_multiprocess4/' # path where the folder will be created with all the generated networks
    os.makedirs(folder) # make dir if not created already
    params = SalibPreprocessGetParamsForSobol(numberOfSamples=3, # number of samples. To generate sample parameters SALib's sobol
                                              # method is used (https://salib.readthedocs.io/en/latest/) N * (D + 2) , where D is the number of parameters and
                                              #N = numberOfSamples
                                              folderPathToSaveParamsAndProblem=folder, # the path where the params will be generated and the graph will also be
                                              #generated there
                                              labelSplit=[100, 200, 300], ## 1st 100 with the same sDNA-1, 2nd 200 with the same sDNA-2 (each 100 with
                                    # same sDNA)
                                              features= 15,
                                              npDistFunc=['np.random.randint(18, high=80)',
                                                          'np.random.binomial(2, 0.5)'],# the distribution function
                                    # from where the features will be generated. Usually it is a numpy random number generator
                                    #function. However, the size parameter should not be filled while passing
                                    # the function as a string.
                                    #  If the number of features equals to the number of
                                    # passed distribution functions then each of them will be used with the corresponding
                                    # feature. Otherwise a features will be generated based on a randomly selected passed function.
                                    # A feature for all the nodes for a single network will be generated from a single function.
                                              bounds=[[0.1,1.0], [0.1, 10], [1, 80],[0.7, 0.9], [0.3, 0.6],[0.1, 0.2]] )
    if __name__ == '__main__':
        Simulate(processes=6, # number of cpu process that should be used
                 params=params, # params generated by SalibPreprocessGetParamsForSobol function. Or a user defined
                 # list of parameters as a pandas dataframe as the following row and col format:
                 #explorationProbabilityV,popularityPreferenceIntensityV,connectionPercentageWithMatchedNodesV,mutualPreferenceIntensityV2,
                 #mutualPreferenceIntensityV3,mutualPreferenceIntensityV4,index,labelSplit,features,npDistFuncs,path
                 # 0.14130859375000002,1.05712890625,41.9658203125,0.8353515625,0.38408203124999996,0.19072265625,1.0,
                 #"[100, 200, 300]",15,"['np.random.randint(18, high=80)', 'np.random.binomial(2, 0.5)']",H:/cpu_Networksim_test_multiprocess3/
                 # 0.29775390625,9.00419921875,41.9658203125,0.8353515625,0.38408203124999996,0.19072265625,2.0,"[100, 200, 300]",15,
                 #"['np.random.randint(18, high=80)', 'np.random.binomial(2, 0.5)']",H:/cpu_Networksim_test_multiprocess3/
                 evalParam='graph.socialise()' # after the graph is created and socialise once, if mutation or further socialise is needed
                 # the instruction should be passed as a string. The string will be evaluated for each of the networks.
                 # for example the follwoing will result in a mutation before socialising:
                 #     'G.mutateDNA(mutationIntensity=0.0001, mutatePreference=True, mutatePreferenceProbability=True) # wheather sDNA should be mutated
                 #     G.socialise()'
                 )
```

FIGURE A.38: *VirtualSoc*, Simulation of multiple networks

A.7 Parameters for the Simulated Networks

exploration probability p	popularity Preference Intensity r	node-pair fraction connection t	Path 2 Preference Intensity c_1	Path 3 Preference Intensity c_2	Path 4 Preference Intensity c_3
2.97753906249999778e-01	1.05712890625000000e+00	4.19658203125000000e+01	8.35351562499999778e-01	3.84082031249999556e-01	1.90722656250000111e-01
1.41308593750000167e-01	1.05712890625000000e+00	4.19658203125000000e+01	8.35351562499999778e-01	3.84082031249999556e-01	1.90722656250000111e-01
2.97753906249999778e-01	9.00419921874999289e+00	4.19658203125000000e+01	8.35351562499999778e-01	3.84082031249999556e-01	1.90722656250000111e-01
2.97753906249999778e-01	1.05712890625000000e+00	4.05771484375000000e+01	8.35351562499999778e-01	3.84082031249999556e-01	1.90722656250000111e-01
2.97753906249999778e-01	1.05712890625000000e+00	4.19658203125000000e+01	7.13867187500000000e-01	3.84082031249999556e-01	1.90722656250000111e-01
2.97753906249999778e-01	1.05712890625000000e+00	4.19658203125000000e+01	8.35351562499999778e-01	3.25488281250000111e-01	1.90722656250000111e-01
2.97753906249999778e-01	1.05712890625000000e+00	4.19658203125000000e+01	8.35351562499999778e-01	3.84082031249999556e-01	1.25488281250000000e-01
2.97753906249999778e-01	9.00419921874999289e+00	4.05771484375000000e+01	7.13867187500000000e-01	3.25488281250000111e-01	1.25488281250000000e-01
1.41308593750000167e-01	1.05712890625000000e+00	4.05771484375000000e+01	7.13867187500000000e-01	3.25488281250000111e-01	1.25488281250000000e-01
1.41308593750000167e-01	9.00419921874999289e+00	4.19658203125000000e+01	7.13867187500000000e-01	3.25488281250000111e-01	1.25488281250000000e-01

TABLE A.3: Input parameters for the simulated networks in Chapter 7, Figure 7.5 and Table 7.3. 1400 networks are simulated using this sampling method and the first ten networks with the above first ten parameters are used in 7. Each of the networks is consists of three snapshots, thus 30 networks in total

A.8 Properties for the Simulated Networks

Network	Edges	Global Clustering Coefficient	Clustering Coefficient (SD)	Centrality Degree Mean	Centrality Closeness (Mean)	Centrality Betweeness (Mean)	Centrality Degree (SD)	Centrality Closeness (SD)	Centrality Betweeness SD	Avg Geodesic Path
9-2	83698	0.219124075	0.071425715	167.396	0.000548037	415.802	101.1033683	3.21E-05	601.8456563	1.832436436
9-1	57486	0.152901691	0.052800973	114.972	0.000531865	442.023	72.04887408	2.12E-05	648.8455707	1.884930931
9-0	29621	0.082840954	0.030642846	59.242	0.000508835	484.069	37.65037831	1.56E-05	703.5184044	1.969107107
8-2	81091	0.222944217	0.054173577	162.182	0.000545928	418.409	85.38837187	2.61E-05	506.8817393	1.837655656
8-1	55640	0.157881191	0.040593909	111.28	0.000530371	444.271	61.02983074	1.77E-05	547.9090429	1.889431431
8-0	28641	0.084102147	0.025682035	57.282	0.000498833	504.957	32.81510791	2.25E-05	616.6126177	2.010924925
7-2	160056	0.549580434	0.054636218	320.112	0.000598187	348.572	183.261038	6.85E-05	391.864325	1.697841842
7-1	113408	0.397361212	0.054629529	226.816	0.000562905	396.091	135.3515134	4.95E-05	443.3466403	1.792974975
7-0	60350	0.207346288	0.040059126	120.7	0.000525569	455.297	71.93003496	3.08E-05	493.5133176	1.911505506
6-2	164823	0.466574488	0.054261799	329.646	0.000604125	334.699	151.7404885	5.24E-05	316.9432457	1.670068068
6-1	117031	0.358238793	0.039804195	234.062	0.000568386	384.403	118.376878	3.84E-05	382.2600459	1.769575576
6-0	62415	0.207176101	0.040172281	124.83	0.000340653	471.052	68.88622803	2.06E-05	483.6098929	1.944936921
5-2	164823	0.505990506	0.062575234	329.646	0.000605197	335.005	170.6617183	5.89E-05	412.9360756	1.670680681
5-1	117031	0.380975226	0.050934659	234.062	0.000567227	387.767	129.5004035	4.46E-05	500.8345596	1.77631031
5-0	62415	0.212415381	0.062794896	124.83	0.000339141	478.502	72.64691786	2.19E-05	620.5696644	1.959881725
4-2	164823	0.433290951	0.05869506	329.646	0.000603651	334.677	140.2080775	5.08E-05	296.5044142	1.670024024
4-1	117031	0.318333188	0.044090972	234.062	0.000568943	382.472	105.2961226	3.41E-05	347.261219	1.76570971
4-0	62415	0.173959255	0.025864001	124.83	0.000533961	437.885	58.63585305	1.74E-05	408.683651	1.876646647
3-2	160056	0.38632966	0.040308863	320.112	0.000598738	339.444	111.8447706	4.14E-05	283.2581963	1.679567568
3-1	113408	0.280341641	0.030453832	226.816	0.000565885	386.094	83.69893461	2.74E-05	328.5046171	1.772960961
3-0	60350	0.153332385	0.017879546	120.7	0.000532688	439.777	46.9460321	1.40E-05	376.9181835	1.880434434
2-2	164823	0.408590055	0.061389134	329.646	0.000603355	334.677	133.1836047	4.98E-05	321.2547438	1.670024024
2-1	117031	0.297291235	0.044219401	234.062	0.000568705	382.469	98.17840473	3.23E-05	369.4063686	1.765703704
2-0	62415	0.166012792	0.021268557	124.83	0.00053398	437.692	53.85526427	1.59E-05	394.484531	1.87626026
1-2	83698	0.241401105	0.031849318	167.396	0.000546733	417	81.9318498	2.52E-05	432.7960615	1.834834835
1-1	57486	0.165305367	0.027296065	114.972	0.000528786	447.464	57.12449934	2.01E-05	472.288996	1.895823824
1-0	29621	0.084831963	0.021858391	59.242	0.000492388	519.966	30.34409663	2.94E-05	551.4941557	2.040972973
0-2	164823	0.47262035	0.047635739	329.646	0.000604325	334.748	155.2453407	5.37E-05	338.9488688	1.670166166
0-1	117031	0.341618112	0.041529809	234.062	0.000569068	382.914	115.4338301	3.69E-05	401.9915505	1.766594595
0-0	62415	0.184678735	0.028370755	124.83	0.000531694	442.613	64.34217136	2.24E-05	478.6616234	1.886112112

TABLE A.4: Properties for the simulated networks used in Figure 7.5 and Table 7.3. Each of the networks contains 1000 nodes

A.9 Citation Networks

A.9.1 Random Weight Initialisation

For the two citation networks in Chapter 7, Table 7.4, following randomly selected seeds are used to initialise the weights:

seeds = {222, 479, 534, 569, 264, 649, 382, 871, 239, 654, 287, 928, 833, 474, 963, 378, 298, 411 , 292, 893, 240, 226, 644, 831, 780, 527, 326, 809, 343, 564, 708, 750, 906, 467, 422, 333 , 555, 99, 204, 271, 231, 705, 276, 685, 267, 375 }

Bibliography

Adamic, L. A. and Adar, E. (2003), 'Friends and neighbors on the web', *Social networks* **25**(3), 211–230.

Adelberger, E. G., Heckel, B. R. and Nelson, A. E. (2003), 'Tests of the gravitational inverse-square law', *Annual Review of Nuclear and Particle Science* **53**(1), 77–121.

Aher, S. N. and Walunj, S. M. (2019), Accelerate the execution of graph processing using gpu, *in* 'Information and Communication Technology for Intelligent Systems', Springer, pp. 125–132.

Ahn, J.-w., Plaisant, C. and Shneiderman, B. (2014), 'A task taxonomy for network evolution analysis', *IEEE transactions on visualization and computer graphics* **20**(3), 365–376.

Al Hasan, M., Chaoji, V., Salem, S. and Zaki, M. (2006), Link prediction using supervised learning, *in* 'SDM06: workshop on link analysis, counter-terrorism and security'.

Al Hasan, M. and Zaki, M. J. (2011), A survey of link prediction in social networks, *in* 'Social network data analytics', Springer, pp. 243–275.

Anthonisse, J. M. (1971), 'The rush in a directed graph', *Stichting Mathematisch Centrum. Mathematische Besliskunde* (BN 9/71), 1–10.

Ashraf, A. W.-U. (2016), Detection and analysis of the causal relation between tourism and greenhouse gas emission: an empirical approach., Master's thesis, Bournemouth University.

Backstrom, L. and Leskovec, J. (2011), Supervised random walks: predicting and recommending links in social networks, *in* 'Proceedings of the fourth ACM international conference on Web search and data mining', ACM, pp. 635–644.

Bahulkar, A., Szymanski, B. K., Baycik, N. O. and Sharkey, T. C. (2018), Community detection with edge augmentation in criminal networks, *in* '2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)', IEEE, pp. 1168–1175.

Bannister, M. J., Eppstein, D., Goodrich, M. T. and Trott, L. (2012), Force-directed graph drawing using social gravity and scaling, *in* 'International Symposium on Graph Drawing', Springer, pp. 414–425.

Barabási, A.-L. and Albert, R. (1999), 'Emergence of scaling in random networks', *science* **286**(5439), 509–512.

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

Barabâsi, A.-L., Jeong, H., Néda, Z., Ravasz, E., Schubert, A. and Vicsek, T. (2002), 'Evolution of the social network of scientific collaborations', *Physica A: Statistical mechanics and its applications* **311**(3), 590–614.

Barabasi, A.-L. and Oltvai, Z. N. (2004), 'Network biology: understanding the cell's functional organization', *Nature reviews genetics* **5**(2), 101–113.

Bassett, D. S. and Bullmore, E. T. (2016), 'Small-world brain networks revisited', *The Neuroscientist* p. 1073858416667720.

Bastami, E., Mahabadi, A. and Taghizadeh, E. (2019), 'A gravitation-based link prediction approach in social networks', *Swarm and evolutionary computation* **44**, 176–186.

Bellamy, R. and Hulme, M. (2011), 'Beyond the tipping point: understanding perceptions of abrupt climate change and their implications', *Weather, Climate, and Society* **3**(1), 48–60.

Bennett, C. J. (2018), 'The european general data protection regulation: An instrument for the globalization of privacy standards?', *Information Polity* **23**(2), 239–246.

Berkhin, P. (2005), 'A survey on pagerank computing', *Internet Mathematics* **2**(1), 73–120.

Bianconi, G. and Barabási, A.-L. (2001), 'Bose-einstein condensation in complex networks', *Physical review letters* **86**(24), 5632.

Bliss, C. A., Frank, M. R., Danforth, C. M. and Dodds, P. S. (2014), 'An evolutionary algorithm approach to link prediction in dynamic social networks', *Journal of Computational Science* **5**(5), 750–764.

Bonacich, P. (1987), 'Power and centrality: A family of measures', *American journal of sociology* pp. 1170–1182.

Borgatti, S. P. and Everett, M. G. (2000), 'Models of core/periphery structures', *Social networks* **21**(4), 375–395.

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

Brandes, U. (2001), 'A faster algorithm for betweenness centrality', *Journal of mathematical sociology* **25**(2), 163–177.

Brandes, U., Robins, G., McCranie, A. and Wasserman, S. (2013), 'What is network science?', *Network Science* **1**(01), 1–15.

Bressler, S. L. and Seth, A. K. (2011), 'Wiener–granger causality: a well established methodology', *Neuroimage* **58**(2), 323–329.

Brin, S. and Page, L. (2012), 'Reprint of: The anatomy of a large-scale hypertextual web search engine', *Computer networks* **56**(18), 3825–3833.

Broido, A. D. and Clauset, A. (2019), 'Scale-free networks are rare', *Nature communications* **10**(1), 1017.

Bruch, E. and Atwell, J. (2015), 'Agent-based models in empirical social research', *Sociological methods & research* **44**(2), 186–221.

Budka, M., Juszczyszyn, K., Musial, K. and Musial, A. (2013), 'Molecular model of dynamic social network based on e-mail communication', *Social Network Analysis and Mining* **3**(3), 543–563.

Butt, R. S. (1992), 'Structural holes: The social structure of competition'.

Cadwalladr, C. and Graham-Harrison, E. (2018), 'Revealed: 50 million facebook profiles harvested for cambridge analytica in major data breach', *The Guardian* **17**.

Carey, H. C. (1867), *Principles of social science*, Vol. 3, JB Lippincott & Company.

Chaintreau, A., Hui, P., Crowcroft, J., Diot, C., Gass, R. and Scott, J. (2007), 'Impact of human mobility on opportunistic forwarding algorithms', *IEEE Transactions on Mobile Computing* **6**(6).

Chebotarev, P. and Shamis, E. (2006), 'The matrix-forest theorem and measuring relations in small social groups', *Autom. Remote Control.* **58**(math. CO/0602070), 1505–1514.

Chen, J., Geyer, W., Dugan, C., Muller, M. and Guy, I. (2009), Make new friends, but keep the old: recommending people on social networking sites, *in* 'Proceedings of the SIGCHI Conference on Human Factors in Computing Systems', ACM, pp. 201–210.

Cohen, J. E., Briand, F. and Newman, C. M. (2012), *Community food webs: data and theory*, Vol. 20, Springer Science & Business Media.

Costa, L. d. F., Rodrigues, F. A., Travieso, G. and Villas Boas, P. R. (2007), 'Characterization of complex networks: A survey of measurements', *Advances in physics* **56**(1), 167–242.

Courrieu, P. (2005), 'Fast computation of moore-penrose inverse matrices', *Neural Information Processing-Letters and Reviews* **8**(2).

Crombie, A. (1957), 'Newton's conception of scientific method', *Physics Bulletin* **8**(11), 350.

Csardi, G. and Nepusz, T. (2006), 'The igraph software package for complex network research', *InterJournal, Complex Systems* **1695**(5), 1–9.

Da Silva, M. R., Ma, H. and Zeng, A.-P. (2008), 'Centrality, network capacity, and modularity as parameters to analyze the core-periphery structure in metabolic networks', *Proceedings of the IEEE* **96**(8), 1411–1420.

Davidsen, J., Ebel, H. and Bornholdt, S. (2002), 'Emergence of a small world from local interactions: Modeling acquaintance networks', *Physical Review Letters* **88**(12), 128701.

Dawson, S., Bakharia, A. and Heathcote, E. (2010), Snapp: Realising the affordances of real-time sna within networked learning environments, Networked Learning.

De Choudhury, M., Lin, Y.-R., Sundaram, H., Candan, K. S., Xie, L. and Kelliher, A. (2010), How does the data sampling strategy impact the discovery of information diffusion in social media?, *in* 'Fourth International AAAI Conference on Weblogs and Social Media'.

de Coulomb, C. (1785), 'Premiere memoire sur l'electricite et le magnetism. second memoire sur l'electricite et le magnetism. troisieme memoire sur l'electricite et le magnetism', *Histoire de l'Académie Royal des Sciences* pp. 569–638.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. and Harshman, R. (1990), 'Indexing by latent semantic analysis', *Journal of the American society for information science* **41**(6), 391.

Dreyfus, S. E. (1969), 'An appraisal of some shortest-path algorithms', *Operations research* **17**(3), 395–412.

Drossel, B. and Schwabl, F. (1992), 'Self-organized critical forest-fire model', *Physical review letters* **69**(11), 1629.

Eagle, N. and Pentland, A. S. (2006), 'Reality mining: sensing complex social systems', *Personal and ubiquitous computing* **10**(4), 255–268.

Edunov, S., Diuk, C., Filiz, I. O., Bhagat, S. and Burke, M. (2016*a*), 'Three and a half degrees of separation', *Research at Facebook* .

Edunov, S., Diuk, C., Filiz, I. O., Bhagat, S. and Burke, M. (2016*b*), 'Three and a half degrees of separation'.
URL: *https://research.fb.com/blog/2016/02/three-and-a-half-degrees-of-separation/*

Erdős, P. and Rényi, A. (1960), 'On the evolution of random graphs', *Publ. Math. Inst. Hung. Acad. Sci* **5**(17-61), 43.

Erdös, P. and Rényi, A. (1959), 'On random graphs, i', *Publicationes Mathematicae (Debrecen)* **6**, 290–297.

Erdős, P. and Rényi, A. (1961), 'On the strength of connectedness of a random graph', *Acta Mathematica Hungarica* **12**(1-2), 261–267.

Essen, U. and Steinbiss, V. (1992), Cooccurrence smoothing for stochastic language modeling, *in* 'Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on', Vol. 1, IEEE, pp. 161–164.

Euler, L. (1999), 'The königsberg bridge problem', *Commentarii academiae scientiarum Petropolitanae* **8**, 128–140.

Facebook (nips) network dataset KONECT (2017), http://konect.uni-koblenz.de/networks/ego-facebook. Accessed: November 2017.

Fire, M., Tenenboim, L., Lesser, O., Puzis, R., Rokach, L. and Elovici, Y. (2011), Link prediction in social networks using computationally efficient topological features, *in* 'Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on', IEEE, pp. 73–80.

Firth, W. (1991), 'Chaos–predicting the unpredictable.', *BMJ: British Medical Journal* **303**(6817), 1565.

Floyd, R. W. (1962), 'Algorithm 97: shortest path', *Communications of the ACM* **5**(6), 345.

Fortunato, S. (2010), 'Community detection in graphs', *Physics reports* **486**(3-5), 75–174.

Foster, K. C., Muth, S. Q., Potterat, J. J. and Rothenberg, R. B. (2001), 'A faster katz status score algorithm', *Computational & Mathematical Organization Theory* **7**(4), 275–285.

Fouss, F., Pirotte, A., Renders, J.-M. and Saerens, M. (2007), 'Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation', *IEEE Transactions on knowledge and data engineering* **19**(3).

Freeman, L. C. (1977), 'A set of measures of centrality based on betweenness', *Sociometry* pp. 35–41.

Freeman, L. C., Romney, A. K. and Freeman, S. C. (1987), 'Cognitive structure and informant accuracy', *American anthropologist* **89**(2), 310–325.

Getoor, L. and Diehl, C. P. (2005), 'Link mining: a survey', *Acm Sigkdd Explorations Newsletter* **7**(2), 3–12.

Girvan, M. and Newman, M. E. (2001), 'Community structure in social and biological networks', *Proc. Natl. Acad. Sci. USA* **99**(cond-mat/0112110), 8271–8276.

Gleich, D. (2007), 'Matlab bgl v2.'.

Goldberg, D. S. and Roth, F. P. (2003), 'Assessing experimentally derived interactions in a small world', *Proceedings of the National Academy of Sciences* **100**(8), 4372–4376.

Gong, M., Fu, B., Jiao, L. and Du, H. (2011), 'Memetic algorithm for community detection in networks', *Physical Review E* **84**(5), 056101.

Google+ network dataset – KONECT (2017), http://konect.uni-koblenz.de/networks/ego-gplus. Accessed: November 2017.

Granger, C. W. (1988), 'Causality, cointegration, and control', *Journal of Economic Dynamics and Control* **12**(2-3), 551–559.

Granovetter, M. (1978), 'Threshold models of collective behavior', *American journal of sociology* **83**(6), 1420–1443.

Grau, J., Grosse, I. and Keilwagen, J. (2015), 'Prroc: computing and visualizing precision-recall and receiver operating characteristic curves in r', *Bioinformatics* **31**(15), 2595–2597.

Gregersen, H. and Sailer, L. (1993), 'Chaos theory and its implications for social science research', *Human relations* **46**(7), 777–802.

Griesinger, D. W. (1979), 'Reconsidering the theory of social gravity', *Journal of Regional Science* **19**(3), 291–302.

Gutiérrez, C. E. and Sabra, A. (2014), 'The reflector problem and the inverse square law', *Nonlinear Analysis: Theory, Methods & Applications* **96**, 109–133.

Hamsterster full network dataset KONECT (2017), http://konect.uni-koblenz.de/networks/petster-hamster. Accessed: November 2017.

Hand, D. J. (2018), 'Aspects of data ethics in a changing world: Where are we now?', *Big data* **6**(3), 176–190.

Harish, P. and Narayanan, P. (2007), Accelerating large graph algorithms on the gpu using cuda, *in* 'International conference on high-performance computing', Springer, pp. 197–208.

Haveliwala, T., Kamvar, S., Klein, D., Manning, C. and Golub, G. (2003), Computing pagerank using power extrapolation, Technical report, Stanford.

Helbing, D. (2012), Systemic risks in society and economics, *in* 'Social Self-Organization', Springer, pp. 261–284.

Herman, J., Usher, W., Mutel, C., Tridade, B., Hadka, D., Woodruff, M., Rios, F. and Hyams, D. (2014), 'Salib: Sensitivity analysis library in python'.

Holme, P. (2019), 'Rare and everywhere: Perspectives on scale-free networks', *Nature communications* **10**(1), 1–3.

Hristova, D., Noulas, A., Brown, C., Musolesi, M. and Mascolo, C. (2016), 'A multilayer approach to multiplexity and link prediction in online geo-social networks', *EPJ Data Science* **5**(1), 24.

Isella, L., Stehlé, J., Barrat, A., Cattuto, C., Pinton, J.-F. and Van den Broeck, W. (2011), 'What's in a crowd? analysis of face-to-face behavioral networks', *Journal of theoretical biology* **271**(1), 166–180.

Jeh, G. and Widom, J. (2002), Simrank: a measure of structural-context similarity, *in* 'Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 538–543.

Jeong, H., Mason, S. P., Barabási, A.-L. and Oltvai, Z. N. (2001), 'Lethality and centrality in protein networks', *Nature* **411**(6833), 41–42.

Johnson, D. B. (1977), 'Efficient algorithms for shortest paths in sparse networks', *Journal of the ACM (JACM)* **24**(1), 1–13.

Jost, J. and Joy, M. P. (2002), 'Evolving networks with distance preferences', *Physical Review E* **66**(3), 036126.

Juszczyszyn, K., Budka, M. and Musial, K. (2011), The dynamic structural patterns of social networks based on triad transitions, *in* '2011 International Conference on Advances in Social Networks Analysis and Mining', IEEE, pp. 581–586.

Juszczyszyn, K., Musial, A., Musial, K. and Bródka, P. (2009), Molecular dynamics modelling of the temporal changes in complex networks, *in* 'Evolutionary Computation, 2009. CEC'09. IEEE Congress on', IEEE, pp. 553–559.

Juszczyszyn, K., Musial, K. and Budka, M. (2011), Link prediction based on subgraph evolution in dynamic social networks, *in* 'Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom)', IEEE, pp. 27–34.

Katz, L. (1953), 'A new status index derived from sociometric analysis', *Psychometrika* **18**(1), 39–43.

Kavak, H., Padilla, J. J., Lynch, C. J. and Diallo, S. Y. (2018), Big data, agents, and machine learning: towards a data-driven agent-based modeling approach, *in* 'Proceedings of the Annual Simulation Symposium', Society for Computer Simulation International, p. 12.

Keilwagen, J., Grosse, I. and Grau, J. (2014), 'Area under precision-recall curves for weighted and unweighted data', *PLOS ONE* **9**(3).

Kellert, S. H. (1993), *In the wake of chaos: Unpredictable order in dynamical systems*, University of Chicago press.

Kipf, T. N. and Welling, M. (2017), Semi-supervised classification with graph convolutional networks, *in* '5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings'.

Klein, D. J. and Randić, M. (1993), 'Resistance distance', *Journal of mathematical chemistry* **12**(1), 81–95.

Kleinberg, J. M. (1999), 'Authoritative sources in a hyperlinked environment', *Journal of the ACM (JACM)* **46**(5), 604–632.

Krioukov, D., Kitsak, M., Sinkovits, R. S., Rideout, D., Meyer, D. and Boguñá, M. (2012), 'Network cosmology', *Scientific reports* **2**.

Kunegis, J. (2013*a*), 'Haggle network dataset konect', http://konect.uni-koblenz.de/networks/contact. Accessed: April 2017.

Kunegis, J. (2013*b*), Konect: the koblenz network collection, *in* 'Proceedings of the 22nd International Conference on World Wide Web', ACM, pp. 1343–1350.

Kunegis, J. (2017*a*), 'arxiv hep-ph network dataset konect', http://konect.uni-koblenz.de/networks/ca-cit-HepTh. Accessed: November 2017.

Kunegis, J. (2017*b*), 'arxiv hep-th network dataset konect', http://konect.uni-koblenz.de/networks/ca-cit-HepTh. Accessed: November 2017.

Kunegis, J. (2017*c*), 'Hypertext 2009 network dataset konect', http://konect.uni-koblenz.de/networks/contact. Accessed: April 2017.

Kunegis, J. (2017*d*), 'Infectious network dataset konect', http://konect.uni-koblenz.de/networks/sociopatterns-infectious. Accessed: November 2017.

Kunegis, J. (2017*e*), 'Reality mining network dataset konect', http://konect.uni-koblenz.de/networks/contact. Accessed: April 2017.

Kunegis, J. and Lommatzsch, A. (2009), Learning spectral graph transformations for link prediction, *in* 'Proceedings of the 26th Annual International Conference on Machine Learning', ACM, pp. 561–568.

Lancichinetti, A., Kivelä, M., Saramäki, J. and Fortunato, S. (2010), 'Characterizing the community structure of complex networks', *PloS one* **5**(8).

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

Lazer, D., Pentland, A. S., Adamic, L., Aral, S., Barabasi, A. L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M. et al. (2009), 'Life in the network: the coming age of computational social science', *Science (New York, NY)* **323**(5915), 721.

Lee, D.-S., Chang, C.-S., Ye, W.-G. and Cheng, M.-C. (2014), Analysis of clustering coefficients of online social networks by duplication models, *in* '2014 IEEE International Conference on Communications (ICC)', IEEE, pp. 4095–4100.

Lee, L. (1999), Measures of distributional similarity, *in* 'Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics', Association for Computational Linguistics, pp. 25–32.

Leicht, E. A., Holme, P. and Newman, M. E. (2006), 'Vertex similarity in networks', *Physical Review E* **73**(2), 026120.

Leskovec, J., Kleinberg, J. and Faloutsos, C. (2005), Graphs over time: densification laws, shrinking diameters and possible explanations, *in* 'Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining', ACM, pp. 177–187.

Leskovec, J., Kleinberg, J. and Faloutsos, C. (2007), 'Graph evolution: Densification and shrinking diameters', *ACM Transactions on Knowledge Discovery from Data (TKDD)* **1**(1), 2.

Leskovec, J. and Mcauley, J. J. (2012), Learning to discover social circles in ego networks, *in* 'Advances in neural information processing systems', pp. 539–547.

Leskovec, J. and Sosič, R. (2016), 'Snap: A general-purpose network analysis and graph-mining library', *ACM Transactions on Intelligent Systems and Technology (TIST)* **8**(1), 1.

Levy, M. and Goldenberg, J. (2014), 'The gravitational law of social interaction', *Physica A: Statistical Mechanics and its Applications* **393**, 418–426.

Li, Q., Han, Z. and Wu, X.-M. (2018), Deeper insights into graph convolutional networks for semi-supervised learning, *in* 'Thirty-Second AAAI Conference on Artificial Intelligence'.

Liben-Nowell, D. and Kleinberg, J. (2007), 'The link-prediction problem for social networks', *journal of the Association for Information Science and Technology* **58**(7), 1019–1031.

Lichtenwalter, R. N. and Chawla, N. V. (2011), 'Lpmade: Link prediction made easy', *Journal of Machine Learning Research* **12**(Aug), 2489–2492.

Lichtenwalter, R. N., Lussier, J. T. and Chawla, N. V. (2010), New perspectives and methods in link prediction, *in* 'Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 243–252.

Lindsay, R. and Zhang, J. (2005), 'The thinning of arctic sea ice, 1988–2003: Have we passed a tipping point?', *Journal of Climate* **18**(22), 4879–4894.

Lorenz, E. N. (1963), 'Deterministic nonperiodic flow', *Journal of the atmospheric sciences* **20**(2), 130–141.

Lovász, L. (1993), 'Random walks on graphs', *Combinatorics, Paul erdos is eighty* **2**, 1–46.

Lü, L., Jin, C.-H. and Zhou, T. (2009), 'Similarity index based on local paths for link prediction of complex networks', *Physical Review E* **80**(4), 046122.

Lü, L. and Zhou, T. (2011), 'Link prediction in complex networks: A survey', *Physica A: statistical mechanics and its applications* **390**(6), 1150–1170.

Marten, K. and Marler, P. (1977), 'Sound transmission and its significance for animal vocalization', *Behavioral ecology and sociobiology* **2**(3), 271–290.

Martínez, V., Berzal, F. and Cubero, J.-C. (2016), 'A survey of link prediction in complex networks', *ACM Computing Surveys (CSUR)* **49**(4), 69.

Mazzocchi, F. (2008), 'Complexity in biology', *EMBO reports* **9**(1), 10–14.

Merrill, D., Garland, M. and Grimshaw, A. (2012), Scalable gpu graph traversal, *in* 'ACM SIGPLAN Notices', Vol. 47, ACM, pp. 117–128.

Michalski, R., Szymański, B. K., Kazienko, P., Lebiere, C., Lizardo, O. and Kulisiewicz, M. (2018), 'Social networks through the prism of cognition', *arXiv preprint arXiv:1806.04658* .

Mislove, A., Marcon, M., Gummadi, K. P., Druschel, P. and Bhattacharjee, B. (2007), Measurement and analysis of online social networks, *in* 'Proceedings of the 7th ACM SIGCOMM conference on Internet measurement', ACM, pp. 29–42.

Moreno, J. L. (1934), 'Who shall survive?: A new approach to the problem of human interrelations'.

Mutton, P. (2004), Inferring and visualizing social networks on internet relay chat, *in* 'Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on', IEEE, pp. 35–43.

Narayanan, A., Shi, E. and Rubinstein, B. I. (2011), Link prediction by de-anonymization: How we won the kaggle social network challenge, *in* 'Neural Networks (IJCNN), The 2011 International Joint Conference on', IEEE, pp. 1825–1834.

Newman, M. (2010*a*), 'Networks: an introduction', *United Slates: Oxford University Press Inc., New York* pp. 1–2.

Newman, M. (2010*b*), *Networks: an introduction*, Oxford university press.

Newman, M. (2018), *Networks*, Oxford university press.

Newman, M. E. (2001*a*), 'Clustering and preferential attachment in growing networks', *Physical review E* **64**(2), 025102.

Newman, M. E. (2001*b*), 'Scientific collaboration networks. i. network construction and fundamental results', *Physical review E* **64**(1), 016131.

Newman, M. E. (2003), 'Properties of highly clustered networks', *Physical Review E* **68**(2), 026121.

Newman, M. E. (2016), 'Equivalence between modularity optimization and maximum likelihood methods for community detection', *Physical Review E* **94**(5), 052315.

Newman, M. E. and Park, J. (2003), 'Why social networks are different from other types of networks', *Physical Review E* **68**(3), 036122.

Newman, M. E. and Watts, D. J. (1999), 'Scaling and percolation in the small-world network model', *Physical Review E* **60**(6), 7332.

Newton, I. (1987), 'Philosophiæ naturalis principia mathematica (mathematical principles of natural philosophy)', *London (1687)* .

Nielsen, E., Fedorov, D. V., Jensen, A. S. and Garrido, E. (2001), 'The three-body problem with short-range interactions', *Physics Reports* **347**(5), 373–459.

Nvidia (2019), 'nvgraph'.
URL: *https://docs.nvidia.com/cuda/nvgraph/index.html nvgraph-api-reference*

NVIDIA (n.d.), 'Virtual-based safety testing for self-driving cars from nvidia drive constellation'.
URL: *https://www.nvidia.com/en-gb/self-driving-cars/drive-constellation/*

Okamoto, K., Chen, W. and Li, X.-Y. (2008), Ranking of closeness centrality for large-scale social networks, *in* 'International Workshop on Frontiers in Algorithmics', Springer, pp. 186–195.

Panzarasa, P., Opsahl, T. and Carley, K. M. (2009), 'Patterns and dynamics of users' behavior and interaction: Network analysis of an online community', *Journal of the Association for Information Science and Technology* **60**(5), 911–932.

Papadimitriou, A., Symeonidis, P. and Manolopoulos, Y. (2011), Predicting links in social networks of trust via bounded local path traversal, *in* 'Proceedings 3rd Conference on Computational Aspects of Social Networks (CASON'2011), Salamanca, Spain'.

Papadimitriou, A., Symeonidis, P. and Manolopoulos, Y. (2012), 'Fast and accurate link prediction in social networking systems', *Journal of Systems and Software* **85**(9), 2119–2132.

Papadopoulos, F., Kitsak, M., Serrano, M. Á., Boguná, M. and Krioukov, D. (2012), 'Popularity versus similarity in growing networks', *Nature* **489**(7417), 537–540.

Pecli, A., Cavalcanti, M. C. and Goldschmidt, R. (2018), 'Automatic feature selection for supervised learning in link prediction applications: a comparative study', *Knowledge and Information Systems* **56**(1), 85–121.

Popescul, A. and Ungar, L. H. (2003), Statistical relational learning for link prediction, *in* 'IJCAI workshop on learning statistical models from relational data', Vol. 2003, Citeseer.

Price, D. d. S. (1976), 'A general theory of bibliometric and other cumulative advantage processes', *Journal of the American society for Information science* **27**(5), 292–306.

Quinn, M. (2018), 'Social media's year of falling from grace'.
URL: *https://www.voanews.com/a/social-medias-year-of-falling-from-grace/4720871.html*

Ravasz, E. and Barabási, A.-L. (2003), 'Hierarchical organization in complex networks', *Physical review E* **67**(2), 026112.

Rombach, M. P., Porter, M. A., Fowler, J. H. and Mucha, P. J. (2014), 'Core-periphery structure in networks', *SIAM Journal on Applied mathematics* **74**(1), 167–190.

Rubinov, M. and Sporns, O. (2010), 'Complex network measures of brain connectivity: uses and interpretations', *Neuroimage* **52**(3), 1059–1069.

Salha, G., Limnios, S., Hennequin, R., Tran, V. A. and Vazirgiannis, M. (2019), 'Gravity-inspired graph autoencoders for directed link prediction', *arXiv preprint arXiv:1905.09570* .

Saltelli, A. (2002), 'Making best use of model evaluations to compute sensitivity indices', *Computer physics communications* **145**(2), 280–297.

Saltelli, A., Annoni, P., Azzini, I., Campolongo, F., Ratto, M. and Tarantola, S. (2010), 'Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index', *Computer Physics Communications* **181**(2), 259–270.

Salton, G. and McGill, M. J. (1986), 'Introduction to modern information retrieval'.

Sapountzi, A. and Psannis, K. E. (2018), 'Social networking data analysis tools & challenges', *Future Generation Computer Systems* **86**, 893–913.

Scellato, S., Noulas, A. and Mascolo, C. (2011), Exploiting place features in link prediction on location-based social networks, *in* 'Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 1046–1054.

Scott, J. (2017), *Social network analysis*, Sage.

Seidel, R. (1992), On the all-pairs-shortest-path problem, *in* 'Proceedings of the twenty-fourth annual ACM symposium on Theory of computing', ACM, pp. 745–749.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B. and Eliassi-Rad, T. (2008), 'Collective classification in network data', *AI magazine* **29**(3), 93–93.

Serafino, M., Cimini, G., Maritan, A., Suweis, S., Banavar, J. R. and Caldarelli, G. (2019), 'Scale-free networks revealed from finite-size scaling', *arXiv preprint arXiv:1905.09512* .

Shi, X., Zheng, Z., Zhou, Y., Jin, H., He, L., Liu, B. and Hua, Q.-S. (2018), 'Graph processing on gpus: A survey', *ACM Computing Surveys (CSUR)* **50**(6), 81.

Siek, J. G., Lee, L.-Q. and Lumsdaine, A. (2001), *The Boost Graph Library: User Guide and Reference Manual, Portable Documents*, Pearson Education.

Simini, F., González, M. C., Maritan, A. and Barabási, A.-L. (2012), 'A universal model for mobility and migration patterns', *Nature* **484**(7392), 96–100.

Simon, H. A. (1955), 'On a class of skew distribution functions', *Biometrika* **42**(3/4), 425–440.

Smith, R. D. (1998), 'Social structures and chaos theory', *Sociological Research Online* **3**(1), 82–102.

Sobol, I. M. (2001), 'Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates', *Mathematics and computers in simulation* **55**(1-3), 271–280.

Solomonoff, R. and Rapoport, A. (1951), 'Connectivity of random nets', *The bulletin of mathematical biophysics* **13**(2), 107–117.

Sporns, O. and Zwi, J. D. (2004), 'The small world of the cerebral cortex', *Neuroinformatics* **2**(2), 145–162.

Stewart, J. Q. (1948), 'Demographic gravitation: evidence and applications', *Sociometry* **11**(1/2), 31–58.

Sugihara, G. and May, R. M. (1990), 'Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series', *Nature* **344**(6268), 734.

Symeonidis, P., Tiakas, E. and Manolopoulos, Y. (2010), Transitive node similarity for link prediction in social networks with positive and negative links, *in* 'Proceedings of the fourth ACM conference on Recommender systems', ACM, pp. 183–190.

Tan, R., Gu, J., Chen, P. and Zhong, Z. (2013), Link prediction using protected location history, *in* 'Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on', IEEE, pp. 795–798.

Taskar, B., Wong, M.-F., Abbeel, P. and Koller, D. (2004), Link prediction in relational data, *in* 'Advances in neural information processing systems', pp. 659–666.

Thietart, R.-A. and Forgues, B. (1995), 'Chaos theory and organization', *Organization science* **6**(1), 19–31.

Thwe, P. (2013), 'Proposed approach for web page access prediction using popularity and similarity based page rank algorithm', *International Journal of Scientific & Technology Research* **2**(3), 240–246.

Tiropanis, T., Hall, W., Crowcroft, J., Contractor, N. and Tassiulas, L. (2015), 'Network science, web science, and internet science', *Communications of the ACM* **58**(8), 76–82.

Townsend, L. and Wallace, C. (2016), 'Social media research: A guide to ethics', *University of Aberdeen* pp. 1–16.

Twitter lists network dataset – KONECT, (2017), http://konect.uni-koblenz.de/networks/munmun_twitter_social. Accessed: November 2020.

Tylenda, T., Angelova, R. and Bedathur, S. (2009), Towards time-aware link prediction in evolving social networks, *in* 'Proceedings of the 3rd workshop on social network mining and analysis', ACM, p. 9.

Urry, J. (2004), 'Small worlds and the new 'social physics'', *Global networks* **4**(2), 109–130.

Voitalov, I., van der Hoorn, P., van der Hofstad, R. and Krioukov, D. (2019), 'Scale-free networks well done', *Physical Review Research* **1**(3), 033034.

Wahid-Ul-Ashraf, A., Budka, M. and Musial-Gabrys, K. (2017), Newton's gravitational law for link prediction in social networks, *in* 'International Workshop on Complex Networks and their Applications', Springer, pp. 93–104.

Wahid-Ul-Ashraf, A., Budka, M. and Musial-Gabrys, K. (2018), Newton's Gravitational Law for Link Prediction in Social Networks, *in* C. Cherifi, M. K. Hocine Cherifi and M. Musolesi, eds, 'Complex Networks & Their Applications VI. COMPLEX NETWORKS 2017', Springer, Cham, pp. 93–104.

Wahid-Ul-Ashraf, A., Budka, M. and Musial, K. (2018), 'Netsim–the framework for complex network generator', *Procedia Computer Science* **126**, 547–556.

Wahid-Ul-Ashraf, A., Budka, M. and Musial, K. (2019), 'How to predict social relationships—physics–inspired approach to link prediction', *Physica A: Statistical Mechanics and its Applications* .

Wang, P., Xu, B., Wu, Y. and Zhou, X. (2015), 'Link prediction in social networks: the state-of-the-art', *Science China Information Sciences* **58**(1), 1–38.

Wang, Y., Davidson, A., Pan, Y., Wu, Y., Riffel, A. and Owens, J. D. (2016), Gunrock: A high-performance graph processing library on the gpu, *in* 'ACM SIGPLAN Notices', Vol. 51, ACM, p. 11.

Wasserman, S. and Faust, K. (1994), *Social network analysis: Methods and applications*, Vol. 8, Cambridge university press.

Watts, D. J. and Strogatz, S. H. (1998), 'Collective dynamics of 'small-world'networks', *nature* **393**(6684), 440–442.

Waymo (n.d.), 'Waymo safety report on the road to fully self-driving', https://storage.googleapis.com/sdc-prod/v1/safety-report/SafetyReport2018.pdf.

Wickham, H. (2011), 'ggplot2', *Wiley Interdisciplinary Reviews: Computational Statistics* **3**(2), 180–185.

Xu, S., Han, K. and Xu, N. (2018), A supervised learning approach to link prediction in dynamic networks, *in* 'International Conference on Wireless Algorithms, Systems, and Applications', Springer, pp. 799–805.

Yang, Z., Cohen, W. W. and Salakhutdinov, R. (2016), 'Revisiting semi-supervised learning with graph embeddings', *arXiv preprint arXiv:1603.08861* .

You, K., Tempo, R. and Qiu, L. (2017), 'Distributed algorithms for computation of centrality measures in complex networks', *IEEE Transactions on Automatic Control* **62**(5), 2080–2094.

Zachary, W. W. (1977), 'An information flow model for conflict and fission in small groups', *Journal of anthropological research* **33**(4), 452–473.

Zhang, X., Martin, T. and Newman, M. E. (2015), 'Identification of core-periphery structure in networks', *Physical Review E* **91**(3), 032803.

Zhong, J. and He, B. (2014), 'Medusa: Simplified graph processing on gpus', *IEEE Transactions on Parallel and Distributed Systems* **25**(6), 1543–1552.

Zhou, D. and Schölkopf, B. (2004), 'Learning from labeled and unlabeled data using random walks', *Lecture notes in computer science* pp. 237–244.

Zhou, T., Lü, L. and Zhang, Y.-C. (2009), 'Predicting missing links via local information', *The European Physical Journal B-Condensed Matter and Complex Systems* **71**(4), 623–630.

Zipf, G. K. (1949), 'Human behaviour and the principle of least effort: an introduction to human ecology'.