Complex Network Analysis with Edge Uncertainty

by

Chi Zhang

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Many datasets can be represented as networks or graphs, where sets of nodes are joined together in pairs by links or edges. In the past, many works have been done on complex network analysis in deterministic graphs, graphs where the network structure is exactly and deterministically known. Recently, in many cases, uncertainty or imprecise information becomes a critical impediment to understanding and effectively utilizing the information contained in such graphs. There are many kinds of uncertainty in networks, such as edge uncertainty, node uncertainty, direction uncertainty and weight uncertainty. The problem of complex network analysis with uncertainty has become increasingly important. However, only a few studies take uncertainty into consideration.

In this thesis, we mainly focus on networks with edge uncertainty, which means the existence of some edges is uncertain. We propose efficient algorithms to solve problems such as entity ranking, link prediction and local community detection for networks with edge uncertainty. Due to the limited number of publicly available uncertain network datasets, we put forward a way to generate uncertain networks for evaluation purposes. Finally, we evaluate our algorithms using existing ground truth as well as based on common metrics to show the effectiveness of our proposed approaches.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

1.1.1 Background

Many datasets can be represented by networks consisting of a set of nodes and edges connecting these nodes. Examples include protein-protein interaction networks [30], food webs [63], social networks [58], air transportation networks, collaboration networks [33], [44] and the worldwide web (WWW) [4], [12]. The study of networked systems has a history stretching back several centuries. Because of its broad applications in different domains, network analysis has attracted increasing attention from computer scientists, biologists and physicists recently. There are a lot of interesting research topics in complex network analysis. Among them, problems such as link prediction, community detection and entity ranking have received a considerable amount of attention.

Entity Ranking

Entity ranking is the task of ordering sets of objects within a network based on the relations among them and the overall linking structure. Ranking nodes in networks can be useful in many applications. For example, in the context of web search, entity ranking methods can be used to rank webpages. To identify the most important nodes in a network, many centrality measures

are proposed, such as degree centrality, closeness centrality [22], betweenness centrality [21], eigenvector Centrality [10] and pagerank centrality [49].

Link Prediction

Link prediction is the problem of determining future or missing associations between entities in complex networks based on observed links. It can be categorized into two classes: one is forecasting the future links, which can be used to help on-line social network users find new friends; the other is determining the hidden or unobserved relationships between nodes, such as in crime networks, protein-protein interaction networks and food webs. The discovery of interaction links in biological networks is usually expensive, therefore, finding the most promising latent links instead of checking all possible links is important in reducing experimental costs.

In the past decade, many works have been done about link prediction in deterministic graphs, graphs where the network structure is exactly and deterministically known. There are many metrics available for computing the similarity of two nodes. According to the characteristics of these metrics, they can be divided into neighbor-based metrics [1], [27], [47], [56], [66], path-based metrics [28], [37] and random-walk-based metrics [24], [35]. Furthermore, there are some learning-based methods [36] that have been proposed in recent years. Among all approaches, neighbor-based metrics are effective and the simplest way to predict missing links. These metrics assume that two nodes are more likely to be connected if they have more common neighbors.

Community Detection

The nodes in many networks fall naturally into groups or communities. Nodes in the same community are densely connected, while the number of edges between nodes of different communities is much smaller. Community detection

is the task of finding such communities in complex networks based on edges between nodes. Detection of these communities is key to understanding the structure of complex networks and extracting useful information from them. The discovery of communities in networks can be useful in various applications. For example, the detection of groups within the worldwide web can be used to find sets of web pages on related topics [20]; the detection of groups within social networks can also be used to find social units or communities [23]. Besides these applications, community detection can also be used in analyzing trends in citation networks [8] and improving recommender systems [13].

In the past 15 years, a large number of community detection algorithms have been proposed for deterministic graphs. According to the characteristics of these algorithms, they can be divided into graph partitioning-based algorithms [29], [46], clustering-based algorithms [9], [18], [23], [45], genetic algorithms-based algorithms [53] and label propagation-based algorithms [54].

1.1.2 Challenges

Most previous studies on complex network analysis have focused on networks under where the structure is exactly known. Recently, we have an increasing number of networks which have uncertainty and imprecise information, which means the network structure is not exactly and deterministically known. For example, crime networks, in which we are uncertain about the existence of some edges; data collected through automated sensors [16], in which we are uncertain about the attributes of some nodes; anonymized communication data (e.g. e-mail headers [2]), in which we are uncertain about the existence of some nodes; and self-reporting/logging on Internet scale networks [11] as a proxy for real relationships and interactions causes some uncertainty. In many cases, uncertainty or imprecise information becomes a critical impediment to understanding and effectively utilizing the information contained in

such graphs. There are many kinds of uncertainty in the network:

1. We can be uncertain about whether there really exists a link between two nodes. Examples of such networks include protein-protein interaction networks with experimentally inferred links, sensor networks with uncertain connectivity links, or social networks, which are augmented with inferred friendship, similarity, or trust links.

2. We can be uncertain whether a given node really exist or not, i.e., about whether two found entities should be the same or not.

3. We know there is a link between two nodes, but we can be uncertain about the direction of the link.

4. We know there is a link between two nodes, but we can be uncertain about the weight over the link.

Among all these kinds of uncertain networks, the first one is the simplest. In this thesis, we mainly focus on the first kind of uncertain network. The graph is undirected, and we are certain about the existence of nodes. The only thing we are uncertain about is the existence of some edges, and these uncertain edges are assigned probabilities. The problem of complex network analysis with the presence of uncertainty has become increasingly important. However, only few studies take probabilities into consideration.

For the problem of entity ranking in uncertain networks, Sevon et al. [57] proposed to transform the probabilities into weights by taking the negative logarithm of the probabilities. Then standard algorithms for finding shortest paths can be applied. Shortest-path-based centralities, such as closeness centrality and betweenness centrality can also be calculated based on shortest paths. Pfeiffer and Neville [52] formulated a measure of centrality based on the most probable paths of communication, rather than shortest paths. They

developed a notion of probabilistic paths in uncertain networks, and used it as a foundation for computing probabilistic betweenness centrality in uncertain networks. A major shortcoming of both methods is that they can only be applied to unweighted probabilistic graphs. For weighted probabilistic graph, ignoring the original weights of the graph could cause the two methods to perform badly.

For the link prediction problem, Mallek et al. [41] put forward an approach that combined sampling techniques and information fusion and obtained good results in real-life settings. Ahmed and Chen [3] proposed the uncertain version of the random walk method for link prediction with edge uncertainty. Up to now, the uncertain version of the popular neighbor-based metrics have not been studied. Murata and Moriyasu [43] proposed weighted version of neighbor-based metrics. People may regard probabilities as weights and apply weighted variants of those metrics; however, it may lead to some problems. More details are presented in Section 5.1.1.

For the task of community detection, to solve the uncertain problem, Krogan et al. [30] converted the uncertain network into a conventional binary network by thresholding the likelihoods. Dahlin and Svenson [19] proposed a method which is based on sampling from an ensemble of deterministic networks that are consistent with the available information about the uncertain networks. Liu et al. [34] developed a novel k-means algorithm to solve the uncertain clustering problem. Martin, Ball and Newman [42] gave a principled maximum-likelihood method for inferring community structure. However, another issue exists in these algorithms is that they require knowledge of the entire graph structure to identify communities. The requirement of accessing the whole network can not be satisfied when networks become too large to know completely, for example, the WWW. In this scenario, it is hard to identify global communities, however, finding a local community for a certain

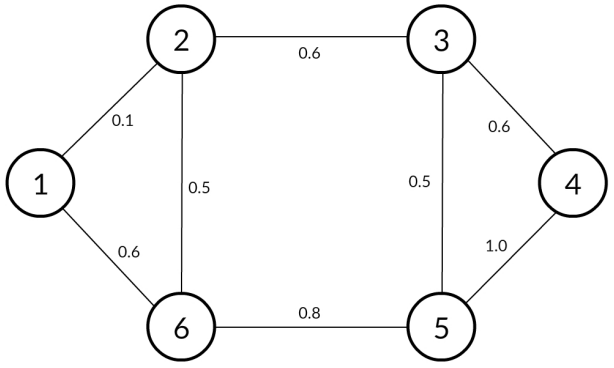

Figure 1.1: An example of uncertain network. The numbers are existential probabilities.

node is still useful. A local community is a community defined based on local information without having access to the entire network. For instance, we may want to quantify the local community of a person given his social network on Facebook. Though the problem of local community detection on deterministic graphs have been discussed by some researchers, and several different local modularity metrics [14], [15], [17] have been proposed to identify local community structure given limited information, the problem of local community detection on uncertain graphs is not yet solved.

1.2 Problem Definition

1.2.1 Networks with Edge Uncertainty

An uncertain graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{P})$ is defined over a set of nodes $\mathcal{V}$, a set of edges $\mathcal{E}$, and a set of probabilities $\mathcal{P}$ of edge existence. Note the probability over the edge between node $\mathcal{V}_i$ and node $\mathcal{V}_j$ can be represented as $\mathcal{P}_{i,j}$ or $\mathcal{P}_{j,i}$. The multiple links and self-connections are not allowed. In this thesis, we only focus on homogeneous networks, networks that have one class of nodes and one semantic for the edges. Figure 1.1 is an example of an uncertain network.

1.2.2 Centrality Measures for Uncertain Networks

Centrality measures are proposed to capture the relative importance of nodes in networks. In the context of uncertain networks, some fundamental requirements of proposing uncertain version of centrality measures should be noticed: (1) The uncertain version of centrality measures should also be able to rank nodes properly; (2) A deterministic network can be regarded as a special case of the uncertain network, therefore, applying the uncertain version and the original version of centrality measures on the same deterministic graph should get the same ranking results.

1.2.3 Link Prediction for Uncertain Networks

The task of link prediction is to discover missing, hidden or future associations between two nodes. Given a network and two unconnected nodes $\mathcal{V}_x$ and $\mathcal{V}_y \in \mathcal{V}$, link prediction is to predict the probability of the existence of a link between the node $\mathcal{V}_x$ and the node $\mathcal{V}_y$. To do this, for each pair of nodes, $\mathcal{V}_x, \mathcal{V}_y \in \mathcal{V}$, which are not directly connected, we assign a score, s_{xy}, according to a given similarity measure. A higher score means nodes $\mathcal{V}_x$ and $\mathcal{V}_y$ are more likely to have an edge. All the nonexistent links are sorted in a descending order according to their scores, and the links at the top are most likely to exist.

Generally, we do not know which links are the missing or future links, otherwise we do not need to do predictions. Therefore, to evaluate algorithms, we use known networks, hide some links, use link prediction algorithms to predict those hidden links and compare the prediction results. Based on the type of network, the observed edges E can be divided into training set E^T and probe set E^P randomly or according to the timestamp. If the known network is time-varying and we know the time each change happens, we can regard the network before a certain time as the training set and the remaining as the

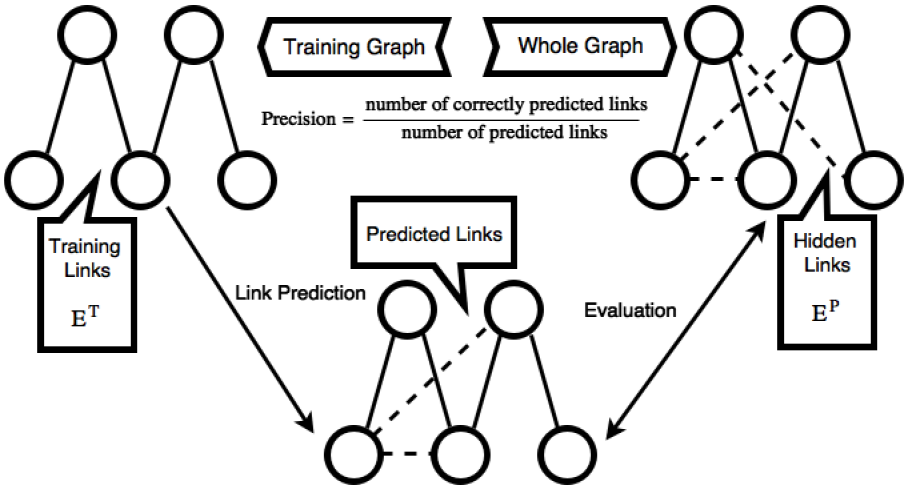

Figure 1.2: Evaluation procedures: Hidden true links are compared to predicted links. How many predicted links where truly there?

probe set. Otherwise, we can just divide the training set and the probe set randomly. To quantify the accuracy of prediction algorithms, we use Precision as our evaluation metric. Figure 1.2 shows the overall procedures for our experiments.

1.2.4 Local Community Detection for Uncertain Networks

As mentioned in the Section 1.1.1, local communities are densely-connected node sets which are discovered and evaluated based only on local information. The task of local community detection aims to find a local community for a certain start node. Clauset, Chen and Wu proposed local community detection problem settings for deterministic networks in [15], [17], [64]. Here, we reiterate them in the context of uncertain networks.

Suppose that in an undirected uncertain network $\mathcal{G}$, we start with one node and we know all its possible neighbors and the possibilities of edge existence between the start node and all possible neighbors. Note we use 'possible' here because there exists uncertainty in the existence of edges between the start node and neighbors. The possibility is in the range of (0,1]. We use $\mathcal{D}$ to

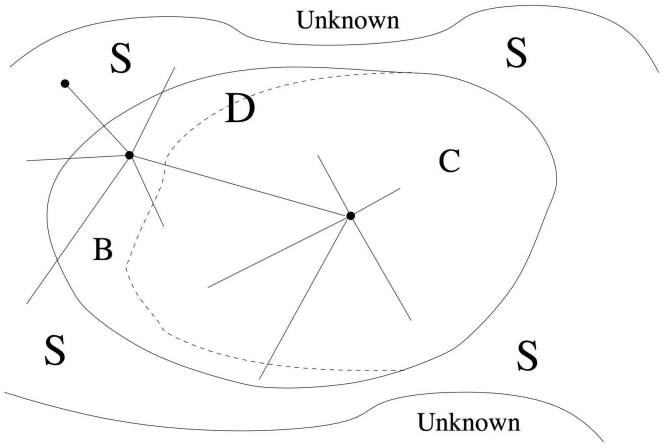

Figure 1.3: Local Community Definition

denote the known local community of the graph (for the start node). This necessarily implies that we also have limited information for another shell node set $\mathcal{S}$, which contains nodes that are possible neighbors of nodes in $\mathcal{D}$ but do not belong to $\mathcal{D}$ (note 'limited' means nodes in $\mathcal{S}$ may also have other possible neighbors that are not in $\mathcal{D}$, but we do not know this information until we visit them). In such circumstances, the only way to gain additional information about the network $\mathcal{G}$ is to visit possible neighbor nodes s_i of $\mathcal{D}$ (where $s_i \in \mathcal{S}$) and obtain the possible neighbors of s_i and the possibilities of edge existence between s_i and its neighbors. As a result, s_i is removed from S and becomes a member of $\mathcal{D}$ while additional neighbor nodes of s_i may be added to $\mathcal{S}$. Furthermore, nodes in $\mathcal{D}$ can be split into two groups: the core node set $\mathcal{C}$, where any node $c_i \in \mathcal{C}$ has no outward links, which means all possible neighbors of c_i belong to $\mathcal{D}$; and the boundary node set $\mathcal{B}$, where any node $b_i \in \mathcal{B}$ has at least one possible neighbor in $\mathcal{S}$. Figure 1.3 shows node sets $\mathcal{D}$, $\mathcal{S}$, $\mathcal{C}$ and $\mathcal{B}$ in a network.

1.3 Thesis Statement

For entity ranking, we state that by taking the inverse of edge probability with a hyper-parameter, we can use existing centrality measures to rank entities. For link prediction, we state that by taking all possible worlds of the uncertain network into account, the performance of link prediction can be improved. For local community detection, we state that we can redefine the notion of sharpness of community periphery while considering existential probabilities of edges.

1.4 Thesis Contributions

There are four major contributions in this work:

- **Uncertain Network Generator:** Due to the limited number of publicly available uncertain network datasets, we put forward a way to generate uncertain networks.
- **Centrality Measures:** We propose a conceptually straightforward as well as computationally efficient way to calculate shortest-path-based centrality measures in uncertain networks. The method can be applied to not only unweighted uncertain networks, but also weighted uncertain networks.
- **Link Prediction:** We propose the uncertain version of the popular common-neighbors-based metrics and efficient algorithms to calculate them. The metrics are developed by considering all possible worlds generated by the uncertain network.
- **Local Community Detection:** We propose a way to convert the uncertain community detection problem into the deterministic scenario. Then we illustrate with an example that periphery nodes tend to be

grouped into their neighbor communities in uncertain networks, and we propose a new measure $\mathcal{K}$ to tackle this problem.

1.5 Organization of the Thesis

Chapter 2 provides the background and related work. In Chapter 3, we put forward a way to generate uncertain networks based on deterministic networks. In Chapter 4, 5 and 6, we describe the methods of entity ranking, link prediction and local community detection in uncertain networks respectively. Chapter 7 concludes the thesis by summarizing our contributions and outlining future work.

Chapter 2

A Review of Complex Network Analysis

In this chapter, we review some of the related work relevant to this thesis. This thesis is mainly about the problems of entity ranking, link prediction and local community detection in uncertain networks. In deterministic networks, these problems have been fully studied, so we first review the existing algorithms in certain scenarios. There are also algorithms about complex network analysis in uncertain scenarios, we also summarize them briefly.

2.1 Centrality and Rank

A measure of centrality on a graph aims to assign a ranking or magnitude to each node that captures the relative importance of that node in the context of the graph's structure. Here are some of the key centrality measures from the literature.

2.1.1 Degree Centrality

Degree centrality is one of the simplest centrality measures. It is defined as the number of edges a node $\mathcal{V}_i$ has, $deg(i)$. It can be normalized by the maximal possible degree, $n-1$, to obtain a number between 0 and 1:

$$c^{deg}(i) = \frac{deg(i)}{n-1} \tag{2.1}$$

The degree centrality is a measure of the size of $\mathcal{V}_i$'s immediate network. It gives some insight into the popularity of the node $\mathcal{V}_i$, but misses potentially important aspects of the whole architecture of the network and a nodes position in the network.

2.1.2 Closeness Centrality

Closeness centrality [22] is based on the network distance between a node and each other reachable node. It can be regarded as a measure of how long it will take to spread information from a start node to all other nodes sequentially. A path from $\mathcal{V}_i$ to $\mathcal{V}_j$ is called a shortest path if it minimizes the number of steps in the sequence, and the distance from $\mathcal{V}_i$ to $\mathcal{V}_j$, denoted $d(i,j)$, is the number of steps in such a shortest path from $\mathcal{V}_i$ to $\mathcal{V}_j$. In a connected graph, the closeness centrality of $\mathcal{V}_i$ is defined as the reciprocal of the average shortest path distance to $\mathcal{V}_i$ over all $n-1$ reachable nodes.

$$c^{clo}(i) = \frac{n-1}{\sum_{i \neq j} d(i,j)} \tag{2.2}$$

When a graph is not strongly connected, Wasserman and Faust [62] proposed an improved formula, which is a ratio of the fraction of nodes in the network which are reachable, to the average distance from the reachable nodes. It is defined as:

$$c^{clo}(i) = \frac{n-1}{N-1} \frac{n-1}{\sum_{i \neq j} d(i,j)} \tag{2.3}$$

where N is the total number of nodes in the network.

2.1.3 Betweenness Centrality

Betweenness centrality quantifies the number of times a node acts as a bridge along the shortest path between two other nodes. It was introduced as a measure for quantifying the control of a human on the communication between other humans in a social network by Linton Freeman [21]. It captures the role

of a node as an intermediary in the transmission of information or resources between other nodes in the network. Nodes that have a high probability to occur on a randomly chosen shortest path between two randomly chosen nodes have a high betweenness. The betweenness centrality can be represented as:

$$c^{bet}(i) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{2.4}$$

where σ_{st} is the total number of shortest paths from node $\mathcal{V}_s$ to node $\mathcal{V}_t$ and $\sigma_{st}(v)$ is the number of those paths that pass through $\mathcal{V}_v$.

2.1.4 Eigenvector Centrality

Eigenvector centrality [10] is a measure of the influence of a node in a network. It computes the centrality for a node based on the centrality of its neighbors. It assumes that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. The relative centrality score of node $\mathcal{V}_i$ can be defined as:

$$c^{eig}(i) = \frac{1}{\lambda} \sum_{\mathcal{V}_j \in \Gamma(i)} c^{eig}(j) \tag{2.5}$$

where $\Gamma(i)$ is a set of the neighbors of $\mathcal{V}_i$ and λ is a constant. It can also be rewritten in vector notation as the eigenvector equation:

$$AC^{eig} = \lambda C^{eig} \tag{2.6}$$

where $C^{eig} = (c^{eig}(1), c^{eig}(2), ..., c^{eig}(n))$; $A = (a_{i,j})$ is the adjacency matrix, i.e. $a_{i,j} = 1$ if node $\mathcal{V}_i$ is linked to node $\mathcal{V}_j$, and $a_{i,j} = 0$ otherwise.

The most famous variant of eigenvector centrality is Google's PageRank algorithm [49]: webpages rank highly in Googles search results if they are linked from other webpages of high rank. It is not the only algorithm used by Google to order search engine results, but it is the best-known.

2.1.5 Centrality Measures in Uncertain Graphs

In order to model the effect of relationship uncertainty on network connectivity, Sevon et al. [57] developed a method for biological databases. They proposed to transform the probabilities into weights by taking the negative logarithm of the probabilities:

$$w(e) = -\log(p(e)) \tag{2.7}$$

Then standard algorithms for finding shortest paths can be applied. Closeness centrality and betweenness centrality can also be calculated based on shortest paths.

Pfeiffer and Neville [52] formulated a measure of centrality based on the most probable paths of communication, rather than shortest paths. They developed a notion of probabilistic paths in uncertain networks, and used it as a foundation for computing probabilistic betweenness centrality in networks evolving over time. The motivation behind this approach is as follows: Conventional betweenness centrality uses shortest paths as an indication of how quickly information can potentially flow in the network. When adopting a probabilistic view of the network, information flowing across paths with fewer nodes is less important than whether the information is successfully transmitted. In this case, central nodes should correspond to nodes that have high probability of transferring information throughout the graph, regardless of path length.

2.2 Link Prediction

In the past decade, many works have been done about link prediction in deterministic graphs, graphs where the network structure is exactly and deterministically known. There are many metrics available for computing the similarity of two nodes. According to the characteristics of these metrics, they can be divided into neighbor-based metrics [1], [27], [47], [56], [66], path-based metrics [28], [37] and random-walk-based metrics [24], [35]. Furthermore, there are some learning-based methods [36] that have been proposed in recent years.

2.2.1 Neighbor-based Metrics

Among all approaches, neighbor-based metrics are the simplest yet effective to predict missing links. These metrics assume that two nodes are more likely to be connected if they have more common neighbors. Researchers design a lot of neighbor-based metrics for link prediction. Their definitions are as follows:

Common Neighbors (CN): Common Neighbors (CN) [47] is the simplest metric among all neighbor-based metrics. It simply counts the number of common neighbors between two nodes and ignores their total number of neighbors. Two nodes, $\mathcal{V}_x$ and $\mathcal{V}_y$, are more likely to form a link if they have many common neighbors. Let $\Gamma(x)$ denote the set of neighbors of node $\mathcal{V}_x$. This measure is defined as follows:

$$s_{xy} = |\Gamma(x) \cap \Gamma(y)| \tag{2.8}$$

CN ignores that different common neighbors have different contributions on the connection likelihood. To solve this problem, other variants such as Resource Allocation and Adamic-Adar metrics are proposed, where a common neighbor with low degree is advocated for by assigning more weight to it.

Resource Allocation (RA): Resource Allocation (RA) [66] metric is regarded as one of the best neighbor-based metrics because of its performance.

Considering a pair of nodes, $\mathcal{V}_x$ and $\mathcal{V}_y$, which are not directly connected. The node $\mathcal{V}_x$ can send some resource to $\mathcal{V}_y$, with their common neighbors playing the role of transmitters. In the simplest case, we assume that each transmitter has a unit of resource, and will evenly distribute to all its neighbors. As a results the amount of resource $\mathcal{V}_y$ received is defined as the similarity between $\mathcal{V}_x$ and $\mathcal{V}_y$, which is:

$$s_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{k(z)} \tag{2.9}$$

where $k(z)$ is the degree of node $\mathcal{V}_z$, namely $k(z) = |\Gamma(z)|$

Adamic-Adar Coefficient (AA): The AA metric [1] was firstly proposed by Adamic and Adar for computing similarity between two web pages, subsequent to which it has been widely used in social networks. Similarly to CN, common neighbors which have fewer neighbors are also weighted more heavily. It is defined as:

$$s_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log k(z)} \tag{2.10}$$

Since CN is not normalized, some neighbor-based metrics also consider how to normalize the CN metric reasonably.

Jaccard Coefficient (JC): Jaccard coefficient [27] normalizes the size of common neighbors. It assumes higher values for pairs of nodes which share a higher proportion of common neighbors relative to total number of neighbors they have. This measure is defined as:

$$s_{xy} = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|} \tag{2.11}$$

Other similar normalized metrics include:

Sørensen Index (SI) [59]

$$s_{xy} = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x)| + |\Gamma(y)|} \tag{2.12}$$

Other neighbor-based metrics include Hub Promoted Index (HPI) [55], Hub Depressed Index (HDI) [66], Leicht-Holme-Newman Index (LHNI) [32] and Preferential Attachment (PA) [6].

2.2.2 Path-based Metrics

Besides node and neighbors information, paths between two nodes can also be used for computing similarities of node pairs, and we call such methods path-based metrics.

Katz Index (KI): Katz Index [28] is based on the ensemble of all paths, which directly sums over the collection of paths and exponentially damped by length to give the short paths more weights. It is defined as:

$$s_{xy} = \sum_{l=1}^{\infty} \beta \cdot |\text{path}_{x,y}^{l}| \tag{2.13}$$

where $\text{path}_{x,y}^{l}$ is the set of all paths with length l connecting nodes $\mathcal{V}_x$ and $\mathcal{V}_y$, and β is a free parameter controlling the weights of the paths. Obviously, a very small β yields a measure close to CN because the long paths contribute very little.

Local Path (LP): Unlike Katz Index that considers paths with all possible length. To provide a good trade-off of accuracy and complexity, LP metric [37] only makes use of information of local paths with length 2 and length 3. Obviously, paths of length 2 are more relevant than paths of length 3, so there is an adjustment factor α applied in the measure. α should be a small number close to 0. (If $\alpha = 0$, LP is the same as CN.) The metric is defined as 2.14. Here, A^2 and A^3 denote the number of all paths with length 2 and 3 connecting nodes $\mathcal{V}_x$ and $\mathcal{V}_y$ respectively.

$$s_{xy} = A^2 + \alpha A^3 \tag{2.14}$$

2.2.3 Random-walk-based Metrics

Social interactions between nodes in social networks can also be modeled by random walks, which use transition probabilities from a node to its neighbors to denote the destination of a random walker from current node. The whole process is a Markov chain describing the sequence of nodes visited by a random walker. There exists some link prediction metrics which calculate similarities between nodes based on random walk.

Hitting Time (HT): [24] $HT(x, y)$ is the expected number of steps required for a random walk from node $\mathcal{V}_x$ to node $\mathcal{V}_y$. It is defined as follows:

$$HT(x, y) = 1 + \sum_{w \in \Gamma(x)} P_{x,w} HT(w, y) \tag{2.15}$$

Where $P_{x,w}$ is the probability of stepping on node $\mathcal{V}_w$ from node $\mathcal{V}_x$.

Commute Time (CT): Since the hitting time metric is not symmetric, commute time is used to count the expected steps both from $\mathcal{V}_x$ to $\mathcal{V}_y$ and from $\mathcal{V}_y$ to $\mathcal{V}_x$. It can be obtained as follows:

$$CT(x, y) = HT(x, y) + HT(y, x) \tag{2.16}$$

Local-random-walk-based Index (SRW): Local-random-walk-based Index [35] is based on a local random walk, which has lower computational complexity compared with other random-walk-based similarity metrics. It is defined as:

$$s_{xy}(t) = \frac{k(x)}{2|E|} \cdot \pi_{xy}(t) + \frac{k(y)}{2|E|} \cdot \pi_{yx}(t) \tag{2.17}$$

$$s_{xy}^{SRW} = \sum_{l=1}^{t} s_{xy}(t) \tag{2.18}$$

Here, $|E|$ is the number of links in the network. $k(x)$ is the degree of the node $\mathcal{V}_x$. $\pi_{xy}(t)$ is the probability that a random walker starts from node $\mathcal{V}_x$ and locates at node $\mathcal{V}_y$ after t steps. t is a tunable hyper-parameter. When we predict missing links, we use s_{xy}^{SRW} values ranking to predict the most promising latent links.

2.2.4 Learning-based Algorithms

Local-Naïve-Bayes-based Index (LNB): Liu et al. [36] proposed a probabilistic model called local Naïve Bayes (LNB) based on Bayes theorem. Different to traditional methods in which each common neighbor contributes equally to the link likelihood, LNB considers that different common neighbors may play different roles in link prediction. The characteristic of the model is that two node pairs with same number of common neighbors could have different connection likelihoods. It is defined as:

$$s_{xy} = \sum_{w \in \Gamma(x) \cap \Gamma(y)} f(k(w)) \log(sR_w) \tag{2.19}$$

Here, $s = \frac{M}{M^T} - 1$ ($M = \frac{|\mathcal{V}|(|\mathcal{V}|-1)}{2}$, $M^T = |\mathcal{E}|$), $R_w = \frac{N_{\triangle w}+1}{N_{\wedge w}+1}$ ($N_{\triangle w}$ and $N_{\wedge w}$ are respectively the number of connected and disconnected node pairs whose common neighbors include node $\mathcal{V}_w$). There are three forms of function f, namely $f(k(w)) = 1$, $f(k(w)) = \frac{1}{\log k(w)}$ and $f(k(w)) = \frac{1}{k(w)}$, which are corresponding to the Local Naïve Bayes (LNB) form of CN, AA and RA metrics respectively, and we name them as LNB-CN, LNB-AA and LNB-RA in Section 5.

2.2.5 Link Prediction Algorithms for Weighted Graphs

The above-mentioned similarity metrics only consider the binary relations among nodes; however, in the real world, links are naturally weighted, which may represent the amount of traffic load along connections in a transportation network or the number of co-authorized papers in a co-authorship network. Murata and Moriyasu [43] proposed weighted similarity metrics as variants of Common Neighbors, Resource Allocation and Adamic-Adar. The definition of the weighted metrics can also be examined in [38]:

Weighted Common Neighbors (WCN):

$$s_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} w(x,z) + w(y,z) \tag{2.20}$$

Weighted Resource Allocation (WRA):

$$s_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{w(x,z) + w(y,z)}{s(z)} \tag{2.21}$$

Weighted Adamic-Adar (WAA):

$$s_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{w(x,z) + w(y,z)}{\log(1 + s(z))} \tag{2.22}$$

Here, $w(x,y) = w(y,x)$ denotes the weight of the link between nodes $\mathcal{V}_x$ and $\mathcal{V}_y$, and $s(x) = \sum_{z \in \Gamma(x)} w(x,z)$ is the strength of node $\mathcal{V}_x$.

2.2.6 Link Prediction Algorithms for Uncertain Graphs

Mallek et al. [41] proposed to use belief function theory to deal with uncertain social networks. Belief function network is considered as a generalization of the probability theory, and one of its uses is the representation and management of missing information. It provides tools for combining evidence induced from several pieces of information. The approach proposed by Mallek combines sampling techniques and information fusion and returns good results in real-life settings. It used popular structural measures based on local graph information to compute distances between the links, and a fusion procedure was subsequently applied taking into account the reliability of the sources to predict missing links.

Ahmed and Chen [3] investigated the problem of link prediction in dynamic uncertain networks, they designed a new method based on a random walk in temporal uncertain networks. Their method first transformed the link prediction problem in uncertain networks to a random walk in a deterministic network. Then, the similarity scores between a node and its neighbors were calculated within a sub-graph around this node to reduce the computational time. They also extended the method for solving link prediction in temporal uncertain networks. The proposed method integrated time and global topological information and obtained accurate results.

2.3 Community Detection

In the past 15 years, a large number of community detection detection algorithms have been proposed for deterministic graphs. According to the characteristics of these algorithms, they can be divided into graph partitioning-based algorithms [29], [46], clustering-based algorithms [9], [18], [23], [45], genetic algorithms-based algorithms [53] and label propagation-based algorithms [54]. In this part, we mainly review graph partitioning-based algorithms and clustering-based algorithms, besides, we also review some local community detection algorithms.

2.3.1 Graph Partitioning-based Algorithms

Graph partitioning is the process of partitioning a graph into a predefined number of smaller components with specific properties. A common property to be minimized is called cut size. A cut is a partition of the vertex set of a graph into two disjoint subsets, and the size of the cut is the number of edges between the components. A multicut is a set of edges whose removal divides the graph into two or more components. It is necessary to specify the number of components one wishes to get in case of graph partitioning. There is a long tradition of research by computer scientists on graph partitioning, and a wide variety of heuristic algorithms have been developed that give acceptable good solutions in many cases.

The Kernighan-Lin [29] algorithm is one of the earliest methods proposed and is still frequently used, often in combination with other techniques. It partitions the nodes of the graph into subsets of given sizes so as to minimize the sum of costs on all edges cut. In each pass, the algorithm improves on a division of the network by optimizing of the number of within- and between-community edges using a greedy algorithm. Given a graph with n nodes, each

pass of the algorithm runs in time $O(n^2 \log n)$. A major disadvantage of this algorithm is that the number of groups has to be predefined, but we do not know how many communities there are, and there is no reason that they should be roughly the same size.

Another popular technique is the spectral bisection method [7], which is based on the properties of the spectrum of the Laplacian matrix. The spectral bisection method is quite fast, and it gives in general good partitions, that can be further improved by applying the KernighanLin algorithm.

2.3.2 Hierarchical Clustering-based Algorithms

Clustering is the process of grouping a set of similar items together in structures known as clusters. The hierarchical clustering and partitioning method of clustering are the commonly used clustering techniques that have been discussed in the literature. Here, we mainly focus on hierarchical clustering-based algorithms. In hierarchical clustering, a hierarchy of clusters is formed. The process of hierarchy creation or leveling can be agglomerative or divisive. In agglomerative clustering methods, a bottom-up approach to clustering is followed. A particular node is clubbed or agglomerated with similar nodes to form a cluster or a community. This aggregation is based on similarity. In divisive clustering approaches, a large cluster is repeatedly divided into smaller clusters.

Girvan and Newman [23] proposed a divisive algorithm based on edge-betweenness for a graph with undirected and unweighted edges. The algorithm detects communities by progressively removing edges from the original network. The connected components of the remaining network are the communities. Instead of trying to construct a measure that tells us which edges are the most central to communities, the Girvan-Newman algorithm focuses on edges that are most likely "between" communities. To find those edges,

the algorithm extends the definition of node betweenness to the case of edges, defining the "edge betweenness" of an edge as the number of shortest paths between pairs of nodes that run along it. The edges connecting communities will have high edge betweenness. By removing these edges, the groups are separated from one another and so the underlying community structure of the network is revealed.

To measure the quality of a particular division of a network, Newman and Girvan [48] first defined a measure known as modularity. The modularity was defined as $Q = \sum_i e_{ii} - a_i^2$, where e_{ij} denotes the fraction of all edges in the network that link nodes in community i to nodes in community j; while $a_i = \sum_j e_{ij}$, which represents the fraction of edges connect to nodes in community i. The value $Q = 1$ indicates a network with strong community structure. Later many researchers proposed clustering-based community detection methods based on the optimization of modularity Q.

Newman [45] has worked to maximize modularity so that the process of aggregating nodes to form communities leads to maximum modularity gain. The algorithm starts with each node in a separate community on its own and amalgamates communities in pairs, choosing at each step the pair whose amalgamation gives the greatest increase in Q.

Clauset et al. [18] used greedy optimization of modularity to detect communities for large networks. For a network structure with m edges and n nodes, the algorithm has a running time of $O(md \log n)$, where d denotes the depth of the dendrogram. For sparse real-world networks, the running time is $O(n \log n)$.

Blondel et al. [9] proposed a heuristic method known as the Louvain method. The algorithm is divided in two phases that are repeated iteratively. First, all nodes are placed into different communities. So, in the initial partition there are as many communities as there are nodes. Then for each node $\mathcal{V}_i$,

the change in modularity is calculated for removing $\mathcal{V}_i$ from its own community and moving it into the community of each neighbor $\mathcal{V}_j$ of $\mathcal{V}_i$. Once this value is calculated for all communities $\mathcal{V}_i$ is connected to, $\mathcal{V}_i$ is then placed in the community for which this gain is maximum (in case of a tie we use a breaking rule), but only if this gain is positive. If no positive gain is possible, $\mathcal{V}_i$ stays in its original community. This process is applied repeatedly and sequentially for all nodes until no further improvement can be achieved and the first phase is then complete. In the second phase of the algorithm, it groups all of the nodes in the same community and builds a new network where nodes are the communities from the previous phase. Any links between nodes of the same community are now represented by self loops on the new community node and links from multiple nodes in the same community to a node in a different community are represented by weighted edges between communities. Once the new network is created, the second phase has ended and the first phase can be re-applied to the new network. In Louvain algorithm, the modularity is defined as:

$$Q = \frac{1}{2m} \sum_{ij} [A_{ij} - \frac{k_i k_j}{2m}] \delta(c_i, c_j) \tag{2.23}$$

where A_{ij} represents the edge weight between nodes $\mathcal{V}_i$ and $\mathcal{V}_j$; k_i and k_j are the sum of the weights of the edges attached to nodes $\mathcal{V}_i$ and $\mathcal{V}_j$ respectively; m is the sum of all of the edge weights in the graph; c_i and c_j are the communities of the nodes; and δ is a simple delta function.

2.3.3 Local Community Detection Algorithms

The task of local community detection aims to find a local community for a certain start node. Numerous local community detection algorithms have been proposed. Many of these algorithms can be grouped as greedy community expansion, which also provides the basis of other (more complex) algorithms. These methods are listed as follows.

Clauset [17] proposed the local modularity $\mathcal{R}$ for the local community evaluation problem and used $\mathcal{R}$ in the expansion step to find the best local community.

$$\mathcal{R} = \frac{\mathcal{B}_{in\_edge}}{\mathcal{B}_{out\_edge} + \mathcal{B}_{in\_edge}} \tag{2.24}$$

where $\mathcal{B}_{in\_edge}$ is the number of edges that connect boundary nodes and other nodes in $\mathcal{D}$, while $\mathcal{B}_{out\_edge}$ is the number of edges that connect boundary nodes and nodes in $\mathcal{S}$. Intuitively, a good community should have a sharp boundary which has fewer connections from the boundary to the unknown portion of the graph, while having a greater number of connections from the boundary nodes back into the local community. Thus, $\mathcal{R}$ measures the fraction of those inside-community edges in all edges with one or more endpoints in $\mathcal{B}$ and community $\mathcal{D}$ is measured by the sharpness of the boundary given by $\mathcal{B}$.

To find a local community for a start node in deterministic networks, Clauset [17] and Chen [15] proposed the local community identification algorithm based on the local modularity $\mathcal{R}$. The algorithm firstly places the start node in the community and its neighbors in the shell node set. At each step, the algorithm adds the neighbor node which gives the largest increase of $\mathcal{R}$ to the community. Then the algorithm update the community set, the boundary set, the shell node set and the $\mathcal{R}$ value. This process will not finish until there are no candidate nodes that could increase $\mathcal{R}$.

Similarly, Luo et al. [40] proposed the modularity M for local community evaluation. Instead of measuring the internal edge fraction of boundary nodes, they directly compare the ratio of internal and external edges. The modularity M is defined as:

$$M = \frac{\text{number of internal edges}}{\text{number of external edges}} \tag{2.25}$$

This algorithm has both an addition step and a deletion step. Nodes will be added or removed from $\mathcal{D}$ only if it can cause an increase in M. This algorithm

turns out to result in high recall but low accuracy.

Chen et al. [14] later presented an alternative method to discover local communities, which aims at reducing outliers and improving detection accuracy. A new measure of local community structure called L was also proposed to help optimize the community hierarchy. The definition of the modularity L is:

$$L = \frac{\frac{\sum_{i \in \mathcal{D}} IK_i}{|\mathcal{D}|}}{\frac{\sum_{j \in \mathcal{B}} EK_j}{|\mathcal{B}|}} \tag{2.26}$$

Here, IK_i is the number of edges between node $\mathcal{V}_i$ and nodes in $\mathcal{D}$, and EK_j is the number of connections between node $\mathcal{V}_j$ and nodes in $\mathcal{S}$. However, this algorithm can hardly obtain a comparatively integrated community structure due to its strict criteria in agglomerating nodes

For weighted graphs, Huang et al. [26] defined structure similarity $s(u, v)$ between two adjacent nodes $\mathcal{V}_u$ and $\mathcal{V}_v$ as:

$$s(u, v) = \frac{\sum_{x \in \Gamma(u) \cap \Gamma(v)} w(u, x) \cdot w(v, x)}{\sqrt{\sum_{x \in \Gamma(u)} w^2(w, x)} \cdot \sqrt{\sum_{x \in \Gamma(v)} w^2(v, x)}} \tag{2.27}$$

When we consider an unweighted graph, the weight $w(u, v)$ of any edge can be set to 1. Based on it, in the agglomeration phase, the candidate node with the largest similarity value will be considered for adding to the community. To check whether a node can be added to the community, Jianbin defined a new measure called Tunable Tightness Gain. Its definition can be found in [26]. If the node with the largest similarity value can give positive value of the Tunable Tightness Gain, it will be added to the community.

Wu et al. [64] proposed a three-phase algorithm. In the agglomeration phase, the node with the highest link similarity value will first be considered for adding to the community. In the optimization phase, all the nodes in boundary will be judged to determine whether they should be removed from the community. After these two phases, a trimming phase is performed in

order to remove the outliers. Though this algorithm ensures that the result is a strong community, a lot of nodes will be reported as outliers in optimization phase and trimming phase.

There are also algorithms that work with a different strategy, which is not based on greedy community expansion. One of these is PageRank-Nibble [5], which works by locally approximating PageRank-vectors. It is a two-step algorithm: first, it approximates personalized PageRank vectors around the seed node and sorts all nodes with a positive score according to that score in decreasing order. Then it considers all communities that are a prefix of this sorted list and returns the prefix with minimum conductance as community.

2.3.4 Community Detection Algorithms for Uncertain Graphs

To deal with uncertain networks, one simple idea is thresholding: we assume that edges exist whenever their probability exceeds a certain threshold that we choose. Krogan et al. [30] first converted the uncertain network into a conventional binary network based on this idea, then applied traditional community detection algorithms to the conventional binary network. While this technique can certainly reveal useful information, it has some drawbacks. First, there is the issue of the choice of the threshold level. Krogan et al. used a value of 0.273 for their threshold, but there is little doubt that their results would be different if they had chosen a different value, and we have the question what threshold should we choose if we have a totally different uncertain network. Second, thresholding throws away potentially useful information. There is a substantial difference between an edge with probability 0.3 and an edge with probability 0.9, but the distinction is lost if one applies a threshold at 0.273.

Dahlin and Svenson [19] proposed a method which is based on sampling. The method mainly has three steps. Firstly, they samples candidate networks

from the ensemble of deterministic networks that are consistent with the available information about the uncertain networks using Monte Carlo methods. Then, standard community detection methods can be applied to candidate networks. Lastly, they merges candidate communities into the most probable community structure of the uncertain network. This method also has its shortcomings. To get reliable communities, one needs to sample enough candidate networks, which will result in high computation complexity.

Liu et al. [34] proposed a generalized reliability criterion from two basic intuitions (purity and size balance) to overcome the challenges from standard reliability criterion, and developed a novel k-means algorithm to solve the uncertain clustering problem. The criterion is designed from an information-theoretic perspective, and the use of such a criterion enables the design of an extremely simple and efficient version of the k-means algorithm, while retaining the desired qualitative properties.

Martin et al. [42] described a method for performing the common task of community detection on uncertain networks by fitting a generative network model to the data using a combination of an expectation maximization (EM) algorithm and belief propagation. They also shown how the resulting fit can be used to reconstruct the true underlying network by making predictions of which nodes are connected by edges.

Chapter 3

Uncertain Network Generator

Since most uncertain graphs are more often than not corporate and government assets and sensitive information, they are rarely disclosed to the public. Considering that there are not many publicly available uncertain network datasets, to conduct experiments in following chapters, we put forward a way to generate uncertain networks based on deterministic networks. The way we generate uncertain networks is mainly based on three assumptions: (1) Edges that exist in deterministic networks tend to have high probability in corresponding uncertain networks; (2) In uncertain networks, except existential edges, there should exist some edges which do not exist in deterministic networks, and they tend to have low probability; (3) Based on a power law distribution, nodes with high degree are more likely to have new added edges. Based on these three assumptions, we generate the uncertain network using Algorithm 1. The first loop in Algorithm 1 assigns high probabilities to existing edges (lines 1 to 5). The second loop creates a non existing edge and assigns it a low probability (lines 7 to 13). It is worth noting that the percentage of non-existential edges is adjustable (line 6), and we choose different values in different experiments in later chapters.

Algorithm 1: Uncertain Network Generator

Data: A deterministic network G, non-existential edge percentage p
Result: An uncertain network $\mathcal{G}$.

for *each edge* $e \in G.edges$ **do**
  Generate probability P according to a Gaussian distribution with mean 0.8 and variance 0.1. (If not in the range (0,1], regenerate it.);
  Assign probability P to edge e;
  Add edge e to the uncertain network $\mathcal{G}$;
end
$NonExistentialEdgesCount \leftarrow |G.edges| \times p$;
while $NonExistentialEdgesCount > 0$ **do**
  Generate edge e which is not in $\mathcal{G}.edges$ based on a power law distribution;
  Generate probability P according to a Gaussian distribution with mean 0.2 and variance 0.1. (If not in the range (0,1], regenerate it.);
  Assign probability P to edge e;
  Add edge e to the uncertain network $\mathcal{G}$;
  $NonExistentialEdgesCount \leftarrow NonExistentialEdgesCount - 1$;
end
return *uncertain network* $\mathcal{G}$

Chapter 4

Centrality and Rank

To rank entities in uncertain networks, as mentioned in Section 2.1.5, Sevon et al. [57] proposed to transform the probabilities into weights by taking the negative logarithm of the probabilities. Pfeiffer and Neville [52] formulated a measure of centrality based on the most probable paths of communication. To facilitate the comparison between our algorithm and these two methods, we rename these two methods Negative Logarithm (NL) and Most Likely Path (ML).

4.1 Inversed Probabilistic Graph

A major shortcoming of both NL and ML methods is that they can only be applied to unweighted probabilistic graphs. For a weighted probabilistic graph, ignoring the original weights of the graph could cause the two methods to perform badly, which we will show later in the experiment part. In order to solve this problem, we propose a new approach, Inversed Probabilistic Graph (IPG), defined as follow:

In Section 1.2.1, we define the uncertain network as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{P})$, which can be regarded as the definition of an unweighted uncertain network. In this chapter, as we aim to solve the problem of entity ranking in both unweighted and weighted networks, we redefine the uncertain network as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{P}, \mathcal{W})$,

where $\mathcal{V}$ is the collection of vertices, $\mathcal{E}$ the collection of edges, $\mathcal{P}$ is the probability distribution over the edges and $\mathcal{W}$ is the collection of original weights over the edges. For an arbitrary edge $e \in \mathcal{E}$, we divide the original weight by the corresponding edge probability to a power of λ,

$$w'(e) = \frac{w(e)}{p(e)^{\lambda}}. \tag{4.1}$$

In this way, the probabilistic graph $\mathcal{G}$ becomes the inversed probabilistic graph $IPG = (V, E, W)$ where W contains the adjusted edge weights that are computed from Equation 4.1.

Now similar to NL, we can apply standard shortest path algorithms used in deterministic graphs to the IPG, and betweenness and closeness centralities can be calculated accordingly. The intuition behind it is that the less distance between two nodes in the graph, the larger the probability that two nodes still have a link, hence the closer relationship two nodes should have. The adjusted weight can be calculated in $O(|E|)$, then we can calculate shortest paths for the IPG in $O(|V||E| + |V|^2 \log |V|)$ by running Dijkstra algorithm. The λ here is a tunable hyper-parameter. All experiments (except Section 4.2.3) described later use $\lambda = 0.4$, as we shall see later (Figure 4.9). $\lambda = 0.4$ gives good results.

It is worth noting that:

1. Unweighted probabilistic graph is a special case of weighted probabilistic graph: Compared to previously discussed two methods, we expand the applicability of our IPG method to weighted graphs. Actually an unweighted probabilistic graph can be regarded as a special case of weighted probabilistic graph. For weighted uncertain networks, $w(e)$ in Equation 4.1 can be replaced with 1.

2. Deterministic graph is a special case of probabilistic graph: Probability depicts our confidence/knowledge of the existence of an edge. If we are

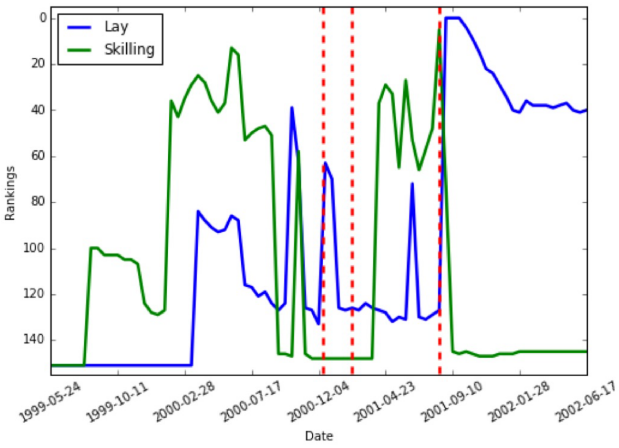

Figure 4.1: Betweenness centrality ranking evolution of Lay and Skilling

sure that an edge e_{uv} exists, then $p(e_{uv}) = 1$ and $w'(e_{uv}) = \frac{w(e_{uv})}{p(e_{uv})^\lambda} = w(e_{uv})$, meaning that in this case the adjusted weight is exactly the original weight of the graph. Conversely, if an e_{uv} is believed to be non-existent for certain, then $p(e_{uv}) = 0$ and $w'(e_{uv}) = \frac{w(e_{uv})}{p(e_{uv})^\lambda} = \infty$, indicating that the distance between nodes u and v is infinitely large, i.e., there is no interaction between these two nodes in the graph.

4.2 Experiment

4.2.1 Betweenness Centrality Ranking Evolution

To investigate the performance of our proposed IPG method, we first conduct an illustrative experiment as Pfeiffer [51], [52] did. We use the same method as Section 5 in [52] to assign each edge with a possibility of occurrence. We also perform our experiment on the same dataset as in [51], [52], which is a subset of Enron, comprised of emails sent between employees, resulting in a dataset with 50,572 emails among 151 employees. We pick two pairs of employees in the dataset, visualizing the evolution of betweenness centrality rankings (BCR) for each pair respectively.

Lay and Skilling

First, we analyze two key figures at Enron: Kenneth Lay and Jeffrey Skilling, whose centrality ranking evolutions are shown in Figure 4.1. The background is that Lay was the CEO of the company first, then he handed it over to Skilling. Several months later, Skilling resigned as CEO, relinquishing control back to Lay. We analyze the change of their centrality rankings during the transition periods. We expect that the trends in the figure can reflect what really happened at that period of time.

The first vertical red line in Figure 4.1 marks the time *Dec.* 13^{th} *2000*, when it was announced that Skilling would assume the CEO position at Enron, with Lay retiring but remaining as a chairman. In Figure 4.1, our IPG approach identifies a spike in BCR for both Lay and Skilling. This can be naturally explained as Skilling and Lay should be informing other executives about the transition right before it was about to be announced. The second event is during *February, 2001* (marked by the second vertical red line), when Skilling made the official transition to CEO. We can see in the figure that the BCR of Skilling has a steep increase afterwards. Whereas during that period, the BCR of Lay dropped significantly. Seven months later, on *Aug.* 14^{th}*, 2001*, Skilling resigned as CEO and Lay took over again. Hence it is no surprise in Figure 4.1 near the third vertical red line, that Lay's BCR rose to a high level again and Skilling's went down.

Moreover, the overall trends of BCR evolution correspond to the real development of the two people. For example, during Skilling's tenure as CEO, he remained a quite high level of centrality ranking; We also notice that the Lay's BCR is higher in the transition period than that during his tenure as CEO. This indicates that a secretary or someone was handling the generic communications with other people for Lay, while during eventful times like

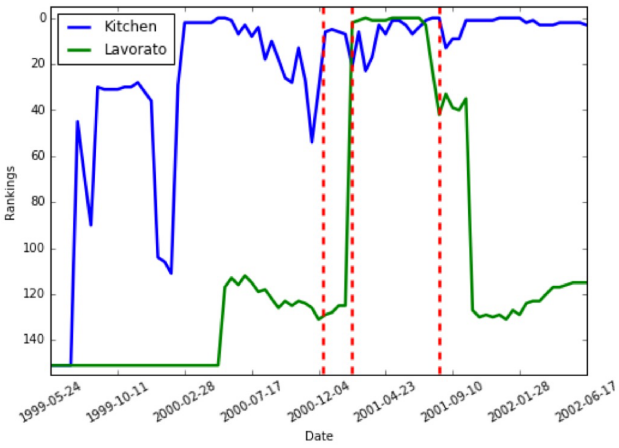

Figure 4.2: Betweenness centrality ranking evolution of Kitchen and Lavorato

job transition, he would communicate with others directly; The third observation is that during the gap between Lay's first and second tenure as CEO, he still kept a certain level of centrality, which can be explained by the fact that he remained as a chairman during his retirement.

Kitchen and Lavorato

The second pair of individuals that we are interested in is Louise Kitchen and John Lavorato, who were executives for Enron America. We can see in Figure 4.2 BCR of Kitchen and Lavorato. The first salient discovery is that Lavorato's ranking is extremely high only during Skilling's tenure as CEO. His ranking drops noticeably otherwise. In comparison, Kitchen keeps a relatively high ranking over time. This indicates that Lavorato has a close relationship with Skilling, whereas Kitchen could have been playing a key role at Enron throughout the time scope.

4.2.2 Linear Correlations

To consolidate the correctness of our method, we also compare our IPG method against NL and ML. We carry out the comparative experiments on Enron dataset, synthetic unweighted graph, and synthetic weighted graph, respec-

tively. The way we evaluate each method is to visualize the linear relationship between centrality rankings computed by each method and the average centrality rankings computed by sampling. Since all three datasets are either dynamic network or generated dataset, the ground truth of entity rankings is not available in these datasets. However, the Law of Large Numbers shows that, the more samples, the closer the estimation is to the expectation. Since the variance of the rankings of the same node at different sampled deterministic graphs is small, we contend that in the case where there is no ground truth, a strong linear correlation with sampling can be a reasonable indicator of a good method.

So, let $\mathcal{G}$ be the given probabilistic graph, with probabilistic distribution $\mathcal{P}$ over the edges, we sample N discrete, deterministic graphs $G_1, G_2, ..., G_N$ from $\mathcal{G}$. (In Section 4.2.2 and Section 4.2.3, we choose $N = 500$.) We use $CR_i(G_j)$ to denote the centrality ranking of a given node $\mathcal{V}_i$ on a given discrete, (un)weighted graph G_j. For a given node $\mathcal{V}_i$, the expected centrality ranking of that node in $\mathcal{G}$ is:

$$\mathbb{E}(CR_i(\mathcal{G})) = \frac{\sum_{j=1}^{N} CR_i(G_j)}{N} \tag{4.2}$$

After calculating the expected centrality ranking for each node, we rank all the nodes in $\mathcal{G}$ again according to their expected ranking values. We refer to this newly computed ranking as *overall ranking*. As for experiments, we apply each target method to probabilistic graph $\mathcal{G}$ to get the overall rankings for all nodes in $\mathcal{G}$. We visualize the result by drawing a 2D plot of overall rankings of nodes computed by each target method against the average rankings of nodes computed from sampling.

$$r = \frac{\sum_{i=1}^{m} |x_i - y_i|}{\sqrt{2}m} \tag{4.3}$$

We use Equation 4.3 as the metric of the linear relationship strength, where m is the number of total nodes in $\mathcal{G}$, i indexes of the target node and (x_i, y_i) is

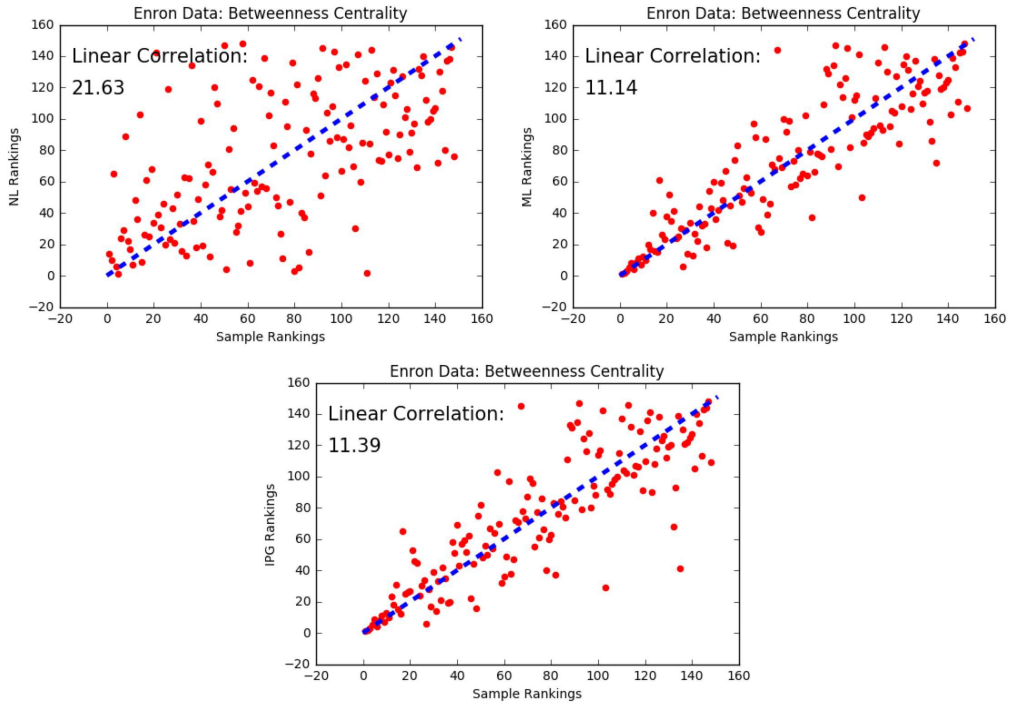

Figure 4.3: Betweenness centralities computed by different methods on Enron dataset

the coordinates of the i^{th} node in the plot. Hence x_i is the average centrality ranking of N sampled graphs, and y_i is the overall ranking computed by a certain method.

Enron Dataset

Figure 4.3 shows the betweenness centrality rankings on Enron dataset computed by NL, ML, and IPG respectively, at a particular point in time: November 14^{th}, 2001. The linear correlation values are displayed on the plots as well. We can easily tell that our IPG method outperforms the NL method, and is slightly worse than ML method.

Figure 4.4 is the same experiment as Figure 4.3 except that we compute closeness rather than betweenness. In terms of closeness centrality on the same dataset, IPG can outperform both NL and ML.

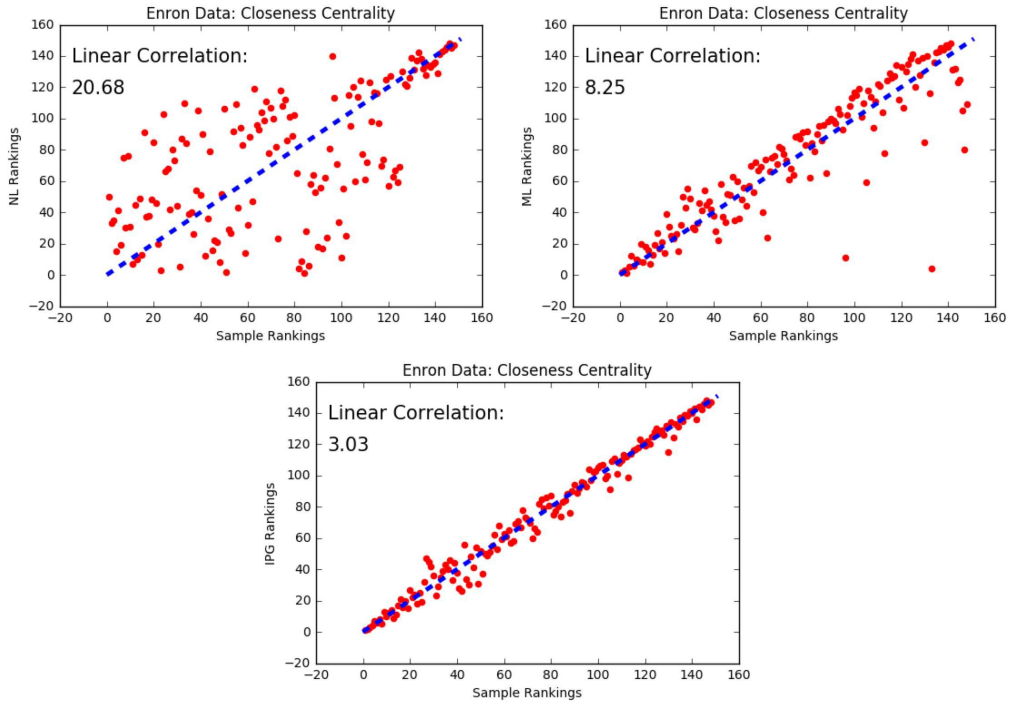

Figure 4.4: Closeness centralities computed by different methods on Enron dataset

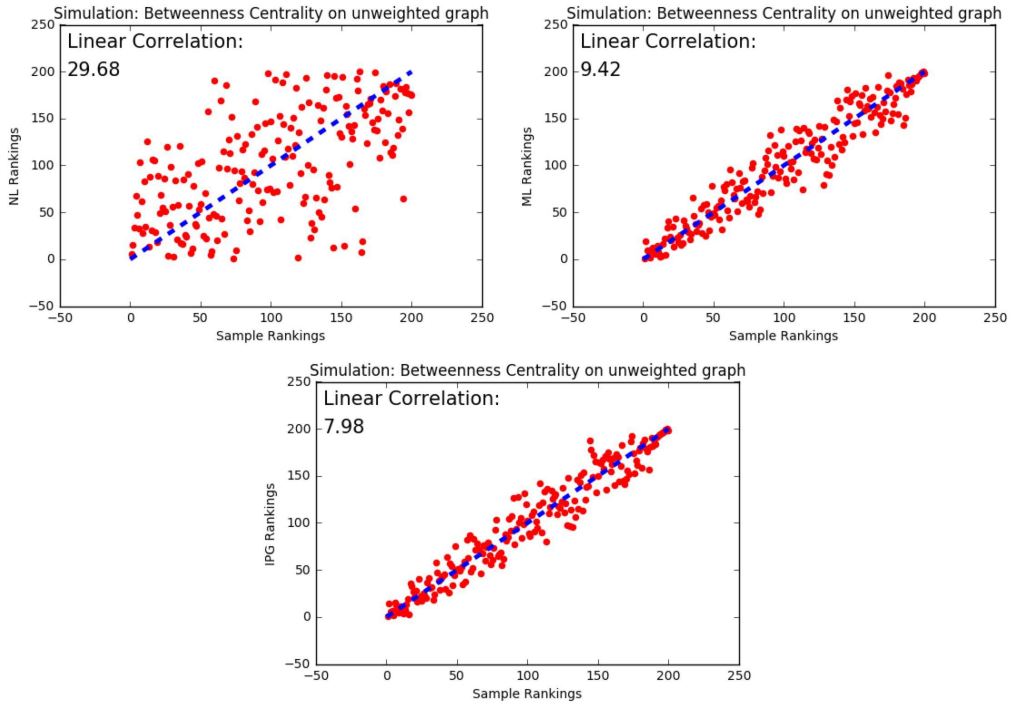

Figure 4.5: Betweenness centralities computed by different methods on synthetic unweighted graphs

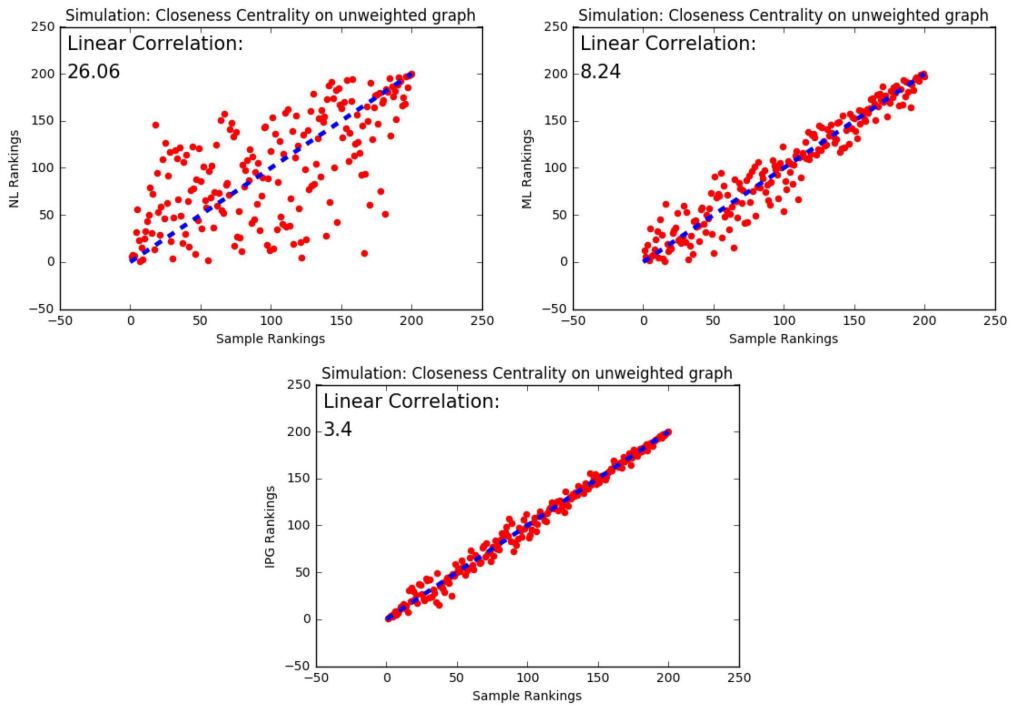

Figure 4.6: Closeness centralities computed by different methods on synthetic unweighted graphs

Unweighted Graphs

We also implement an experiment on a synthetic unweighted graph. We generate three groups with the number of nodes in each group being 30, 50 and 120, respectively. Nodes in the same group are connected with probability 0.2, and nodes of different groups are connected with probability 0.02. Then we use the uncertain network generator (Algorithm 1 as mentioned in Chapter 3) to generate an uncertain network based on the given deterministic network. In this experiment, the percentage of non-existential edges we choose to add is 20%.

The betweenness centrality results of three methods are displayed in Figure 4.5 and the closeness results are in Figure 4.6. For all three methods, the linear correlation in the closeness experiment are stronger than that in the betweenness experiment. But no matter in which case, our IPG always exhibits

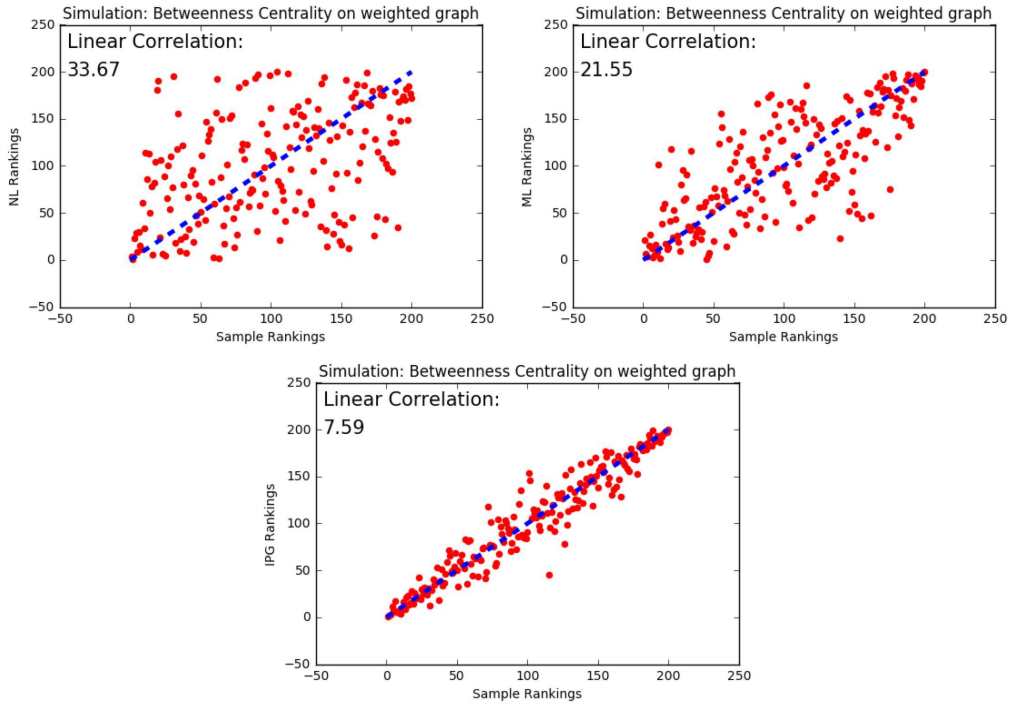

Figure 4.7: Betweenness centralities computed by different methods on synthetic weighted graphs

the best performance among the three and NL has the worst performance.

Weighted Graphs

We pointed out earlier that one major advantage of IPG over NL and ML is that our method extends well on weighted graphs. We show this advantage in this section via an experiment. The experimental parameters are inherited from the former Unweighted Graphs experiment, and the weights assigned to edges are generated based on a uniform distribution on the interval $[1, 3]$. Figure 4.7 and Figure 4.8 respectively show the betweenness and closeness of three methods on weighted graphs. As the plots in two figures indicate, IPG still has the best performance among the three methods in terms of both betweenness and closeness centrality rankings.

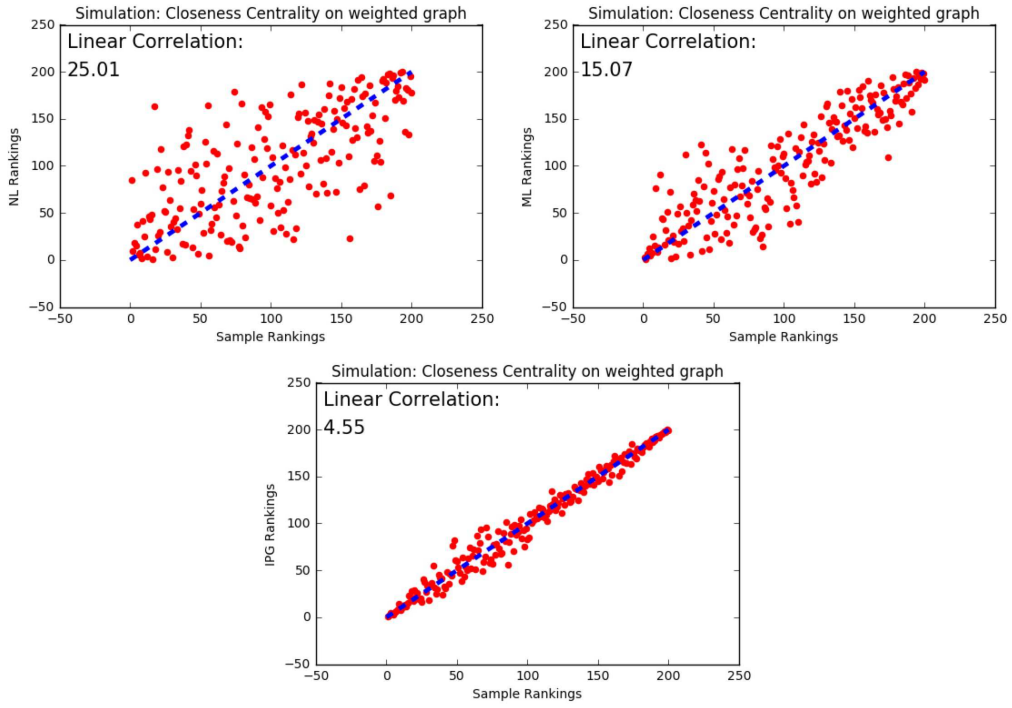

Figure 4.8: Closeness centralities computed by different methods on synthetic weighted graphs

4.2.3 Hyper-parameter Validation

As we mentioned in Section 4.1, the choice of hyper-parameter λ is another crucial topic in our experiment. To validate that the λ value we use is a reasonable choice, we conduct experiments in this section to find optimal λ for different centrality measures on different graphs. We do this by repeating previous experiments on synthetic unweighted and weighted graphs over a range of different λ values, as shown in Figure 4.9.

Each plot is obtained in the following way. First, we generated 20 probabilistic graphs and candidate λ value, which ranges from 0.01 to 0.8; Next, on each probabilistic graph, we computed linear correlation strength under each λ, using our IPG method; Finally, we took the average linear correlation over 20 probabilistic graphs. From these four experiments we can conclude that the optimal choice of the hyper-parameter λ varies with tasks, whereas all the

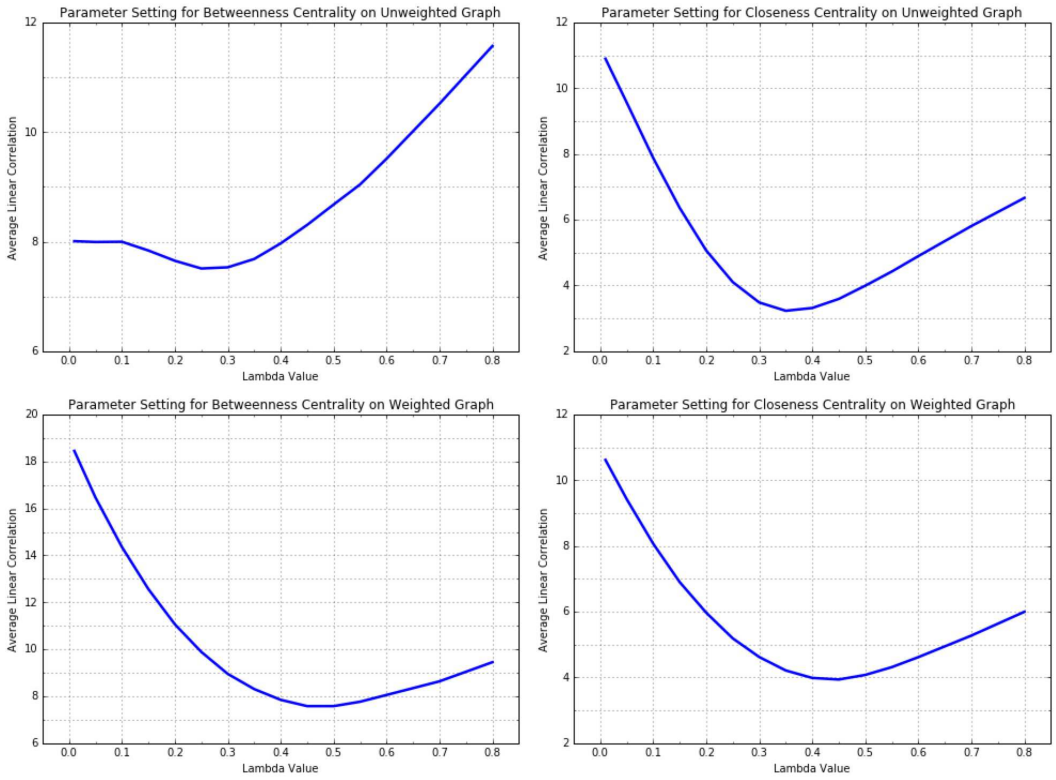

Figure 4.9: Hyper-parameter validation

optimal $\lambda's$ lie in the interval from 0.25 to 0.45. Though $\lambda = 0.4$ is not always the optimal choice in different tasks, the results achieved by $\lambda = 0.4$ is still competitive. Therefore, it is reasonable to choose $\lambda = 0.4$.

4.3 Conclusion

In this chapter, we provide a novel approach to deal with edge uncertainty in graph mining problems. Specifically, by taking the inverse of edge probability with a hyper-parameter, we can use existing existing centrality measures to rank entities. We empirically show the effectiveness of our IPG method by experiments.

Chapter 5

Link Prediction

As we discussed in Section 2.2.1, common-neighbors-based metrics are the simplest yet effective to predict missing links. They assume that two nodes are more likely to be connected if they have more common neighbors. Common neighbors (CN) is one of the most widespread measure used in the link prediction problem mainly due to its simplicity [47]. The Resource Allocation (RA) metric [66] is regarded as one of the best neighbor-based metrics because of its performance. Therefore, in this chapter, we concentrate on CN and RA indexes.

5.1 Link Prediction for Uncertain Graphs

5.1.1 Uncertain Version of Neighbor-based Metrics

To solve the problem of link prediction for uncertain graphs, one very naïve/intuitive way is to regard the probability as a weight and apply weighted similarity metrics. However, this is not necessarily appropriate and may mislead the prediction of absent connections. Figure 5.1 is an example illustrating such a problem.

Nodes $\mathcal{V}_A$ and $\mathcal{V}_B$ are more likely to be connected than nodes $\mathcal{V}_D$ and $\mathcal{V}_E$ based on Equation 2.20 for Weighted Common Neighbors.

$$s_{AB} = 0.2 + 0.9 = 1.1 \tag{5.1}$$

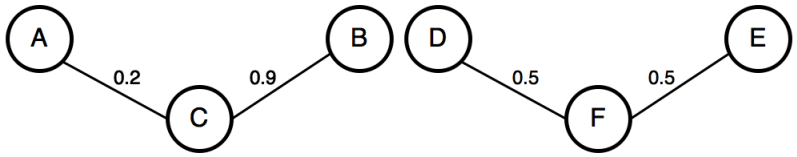

Figure 5.1: An example showing the problem when considering the probability as a weight. A-B would seem to have a higher probability to exist than D-E

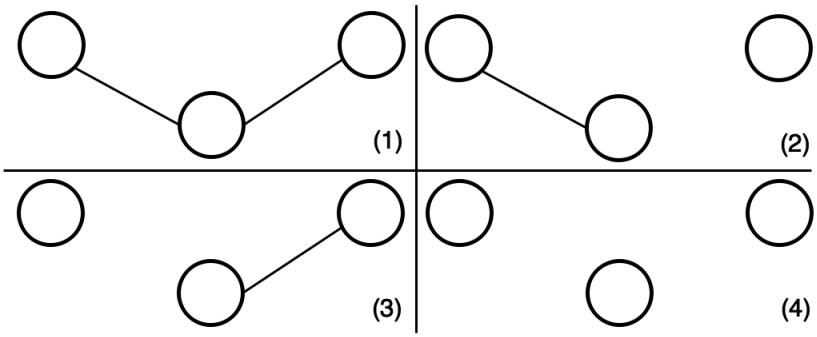

Figure 5.2: Possible worlds for two uncertain links between three nodes

$$s_{DE} = 0.5 + 0.5 = 1.0 < 1.1 \tag{5.2}$$

However, because each edge may exist or not exist in the real world, both of these two uncertain graphs have four possible worlds, as can be seen in Figure 5.2: both links may exist, both may be absent, or either one is present.

Only when both edges $\mathcal{E}_{AC}$ and $\mathcal{E}_{BC}$ exist, node $\mathcal{V}_C$ is the common neighbor of nodes $\mathcal{V}_A$ and $\mathcal{V}_B$, as Figure 5.2(1), the probability for this case is $0.2 \times 0.9 = 0.18$ (In this Chapter, we assume that the existence of edges are independent with each other). In this case, $s_{AB} = 1$ based on Equation 2.8. If node $\mathcal{V}_C$ is not the common neighbor of nodes $\mathcal{V}_A$ and $\mathcal{V}_B$, as Figure 5.2(2, 3 and 4), then $s_{AB} = 0$. The probability for this case is 0.82. In comparison, the probability that $s_{DE} = 1$ is $0.5 \times 0.5 = 0.25$, while the probability of $s_{DE} = 0$ is 0.75. Therefore, nodes $\mathcal{V}_D$ and $\mathcal{V}_E$ are more likely to be connected than nodes $\mathcal{V}_A$ and $\mathcal{V}_B$, because the probability of $s_{DE} = 1$ is larger than the probability of $s_{AB} = 1$.

From this example, we can find that each uncertain edge in an uncertain graph may exist or not exist in a real world. If an uncertain graph has $|\mathcal{E}|$

uncertain edges, there will be $2^{|\mathcal{E}|}$ possible worlds in total, since each edge provides us with a binary sampling decision.

Given an uncertain network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{P})$, we can sample each edge in $\mathcal{G}$ according to the probability $\mathcal{P}(e)$ to generate the possible graph $G = (V_G, E_G)$. We have $E_G \in \mathcal{E}$ and $V_G \in \mathcal{V}$. The probability $Pr(G)$ of sampling the possible graph is as follows:

$$Pr(G) = \prod_{e \in E_G} \mathcal{P}(e) \prod_{e \in \mathcal{E}, e \notin E_G} (1 - \mathcal{P}(e)) \tag{5.3}$$

For each possible world, its corresponding similarity measure may differ. When we calculate its similarity measures, we should take all possible worlds and their possibilities into account. Therefore, Common Neighbor and Resource Allocation in uncertain graphs can be represented as follows.

Uncertain Common Neighbors (UCN)

$$s_{xy} = \sum_{G \in \mathcal{G}} (Pr(G) \times |\Gamma_G(x) \cap \Gamma_G(y)|) \tag{5.4}$$

Uncertain Resource Allocation (URA)

$$s_{xy} = \sum_{G \in \mathcal{G}} (Pr(G) \times \sum_{z \in \Gamma_G(x) \cap \Gamma_G(y)} \frac{1}{k_G(z)}) \tag{5.5}$$

Here, $\Gamma_G(x)$ denotes the set of neighbors of node $\mathcal{V}_x$ in the possible world G; $k_G(x)$ is the degree of node $\mathcal{V}_x$ in the possible world G.

5.1.2 UCN Complexity Analysis

We have a total of $2^{|\mathcal{E}|}$ possible worlds, and we can calculate CN value for each possible world in $O(k)$, where k is nodes' average degree in the possible world. Therefore, the time complexity of calculating the Common Neighbors value based on Equation 5.4 is $O(2^{|\mathcal{E}|}k)$.

Assume $\Gamma_{xy} = \Gamma(x) \cap \Gamma(y)$ is the common neighbors set of nodes $\mathcal{V}_x$ and $\mathcal{V}_y$ in uncertain graph $\mathcal{G}$. Whether a node $\mathcal{V}_z \in \Gamma_{xy}$ is a common neighbor

of nodes $\mathcal{V}_x$ and $\mathcal{V}_y$ in a possible world is independent of other nodes because it is determined by the existence of edges $\mathcal{E}_{xz}$ and $\mathcal{E}_{yz}$ in the possible world. Therefore, each node in Γ_{xy} can be considered independently. If the existence probability over uncertain edges $\mathcal{E}_{xz}$ and $\mathcal{E}_{yz}$ are $\mathcal{P}_{x,z}$ and $\mathcal{P}_{y,z}$ respectively, only in $\mathcal{P}_{x,z} \times \mathcal{P}_{y,z}$ of all possible worlds, node $\mathcal{V}_z$ is the common neighbor of nodes $\mathcal{V}_x$ and $\mathcal{V}_y$. Therefore, Equation 5.4 can also be represented as:

$$\begin{aligned} s_{xy} &= \sum_{G \in \mathcal{G}} (Pr(G) \times |\Gamma_G(x) \cap \Gamma_G(y)|) \\ &= \sum_{G \in \mathcal{G}} Pr(G) \sum_{z \in \Gamma(x) \cap \Gamma(y)} \mathbb{I}_{\Gamma_G(x) \cap \Gamma_G(y)}(z) \\ &= \sum_{z \in \Gamma(x) \cap \Gamma(y)} \sum_{G \in \mathcal{G}} Pr(G) \times \mathbb{I}_{\Gamma_G(x) \cap \Gamma_G(y)}(z) \\ &= \sum_{z \in \Gamma(x) \cap \Gamma(y)} \mathcal{P}_{x,z} \times \mathcal{P}_{y,z} \end{aligned}$$

When $z \in \Gamma_G(x) \cap \Gamma_G(y)$, $\mathbb{I}_{\Gamma_G(x) \cap \Gamma_G(y)}(z) = 1$, otherwise, $\mathbb{I}_{\Gamma_G(x) \cap \Gamma_G(y)}(z) = 0$.

By doing so, the time complexity for calculating s_{xy} can be reduced to $O(K)$, where K is the nodes' average degree in the uncertain network.

5.1.3 URA Complexity Analysis

We have a total of $2^{|\mathcal{E}|}$ possible worlds, and nodes' average degree in the possible world is k, then we can calculate RA value for each possible world in $O(k)$, so the time complexity of calculating Resource Allocation value based on Equation 5.5 is $O(2^{|\mathcal{E}|}k)$.

As mentioned in Section 5.1.2, whether a node $\mathcal{V}_z \in \Gamma_{xy}$ is a common neighbor of nodes $\mathcal{V}_x$ and $\mathcal{V}_y$ in a possible world is independent of other nodes. Besides, the number of edges each common neighbor has is also independent of other nodes. Therefore, each common neighbor can also be considered independently in this case. For the common neighbor node $\mathcal{V}_z$, when we generate possible worlds, we can consider only edges connecting to it, because the existence of other edges will not have an impact on $\mathbb{I}_{\Gamma_G(x) \cap \Gamma_G(y)}(z)$ and $k_G(z)$. The

nodes' average degree in the uncertain network is K, so we can consider 2^K possible worlds for the node $\mathcal{V}_z$, and the time complexity can be reduced to $O(2^K t)$, where $t = |\Gamma_G(x) \cap \Gamma_G(y)|$.

$$\begin{aligned} s_{xy} &= \sum_{G \in \mathcal{G}} (Pr(G) \times \sum_{z \in \Gamma_G(x) \cap \Gamma_G(y)} \frac{1}{k_G(z)}) \\ &= \sum_{z \in \Gamma(x) \cap \Gamma(y)} \sum_{G \in \mathcal{G}} Pr(G) \times \mathbb{I}_{\Gamma_G(x) \cap \Gamma_G(y)}(z) \times \frac{1}{k_G(z)} \\ &= \sum_{z \in \Gamma(x) \cap \Gamma(y)} \sum_{G_z \in \mathcal{G}_z} Pr(G_z) \times \mathbb{I}_{\Gamma_{G_z}(x) \cap \Gamma_{G_z}(y)}(z) \times \frac{1}{k_{G_z}(z)} \end{aligned}$$

$\mathcal{G}_z$ here stands for the uncertain sub-graph formed by edges connecting to node $\mathcal{V}_z$, and G_z is the possible world based on the uncertain sub-graph $\mathcal{G}_z$.

5.1.4 An Efficient Algorithm for URA

Only when both edges $\mathcal{E}_{xz}$ and $\mathcal{E}_{yz}$ exist, node $\mathcal{V}_z$ is the common neighbor of node $\mathcal{V}_x$ and node $\mathcal{V}_y$ in the possible world G, which means $\mathbb{I}_{\Gamma_G(x) \cap \Gamma_G(y)}(z) = 1$. When node $\mathcal{V}_z$ is not the common neighbor of node $\mathcal{V}_x$ and node $\mathcal{V}_y$, $\mathbb{I}_{\Gamma_G(x) \cap \Gamma_G(y)}(z) = 0$, it means those possible worlds will not have an impact on the value of s_{xy}. Edges $\mathcal{E}_{xz}$ and $\mathcal{E}_{yz}$ belong to the edge set which connects to node $\mathcal{V}_z$, so for those possible worlds which have an impact on the value of s_{xy}, node $\mathcal{V}_z$ at least has two edges $\mathcal{E}_{xz}$ and $\mathcal{E}_{yz}$.

Assume node $\mathcal{V}_z$ has m extra edges in an uncertain graph except edges $\mathcal{E}_{xz}$ and $\mathcal{E}_{yz}$. Although it will result in 2^m possible worlds, the number of its edges in possible worlds will only range from 0 to m (the number of edges node $\mathcal{V}_z$ has in total ranges from 2 to $m + 2$), which means some of the possible worlds share the same number of edges. To calculate s_{xy}, one way is to iterate through all possible worlds, calculate each possible world's possibility based on Equation 5.3 and its corresponding count of edges. The other way is to iterate through all the possible number of edges and calculate their corresponding

probability, which can be seen as follows:

$$s_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \sum_{G_z \in \mathcal{G}_z} Pr(G_z) \times \mathbb{I}_{\Gamma_{G_z}(x) \cap \Gamma_{G_z}(y)}(z) \times \frac{1}{k_{G_z}(z)}$$
$$= \sum_{z \in \Gamma(x) \cap \Gamma(y)} \mathcal{P}_{x,z} \times \mathcal{P}_{y,z} \times \sum_{n=0}^{m} (P^n_{1 \to m} \times \frac{1}{n+2})$$

For the common neighbor $\mathcal{V}_z$, assume there are m edges connecting to it except edges $\mathcal{E}_{xz}$ and $\mathcal{E}_{yz}$, so we can index them from 1 to m. $P^n_{1 \to m}$ here stands for from edges e_1 to e_m, the probability that exactly n among them exist in possible worlds. For the node with m edges in the uncertain graph, the number of its edges in possible worlds will range from 0 to m, and in other words, we need to compute $P^0_{1 \to m}, P^1_{1 \to m}, ..., P^m_{1 \to m}$.

We propose an efficient way to compute them, which can be regarded as a divide and conquer algorithm. Conceptually, it works as follows:

1) Divide the probability list into n sublists, each containing 1 element, and compute the probability of having and not having this item respectively.

2) Repeatedly merge sublists to compute probabilities for sublists with more than 1 element. Here is the equation for merging the left half sublist and the right half sublist.

$$P^n_{1 \to m} = \sum_{i=max(0, n-\lceil m/2 \rceil)}^{min(n, \lfloor m/2 \rfloor)} P^i_{1 \to \lfloor m/2 \rfloor} P^{n-i}_{\lfloor m/2 \rfloor + 1 \to m} \tag{5.6}$$

It can be implemented recursively. The result probability list has the length of $m+1$ and $P^0_{1 \to m}, P^1_{1 \to m}, ..., P^m_{1 \to m}$ are saved sequentially in the result probability list. The full algorithm description can be found in Algorithm 2.

Based on the description of Algorithm 2, we can find that $T(m) = 2 \times T(\frac{1}{2}m) + (1+2+...+m)$, so the time complexity of Algorithm 2 is $O(m^2)$. After calculating the probability list, we can easily calculate node $\mathcal{V}_z$'s contribution for s_{xy}.

It is reasonable to calculate $\mathcal{V}_z$'s contribution for s_{xy} in $O(m^2)$. However,

Algorithm 2: kEdgeProbability

Data: Probability List $uncertainEdgeList$

Result: The probability list $probList$ of existing n among m edges, $n \in [0, m]$

```
uncertainEdgeListLength ← len(uncertainEdgeList);
return kEdge(0, uncertainEdgeListLength − 1);
// Inner Function;
Function kEdge(i, j)
    length ← j − i + 1;
    if length = 1 then
        return [1 − uncertainEdgeList[i], uncertainEdgeList[i]]
    else
        leftLength ← length//2;
        rightLength ← length − leftLength;
        // Divide Phase;
        left ← kEdge(i, i + leftLength − 1);
        right ← kEdge(i + leftLength, j);
        probList ← [0] × (length + 1);
        for each n ∈ [0, length] do
            for each k ∈ [0, n] do
                if k <= leftLength and n − k <= rightLength then
                    // Merge Phase;
                    probList[n] ← probList[n] + left[k] × right[n − k];
                end
            end
        end
        return probList;
    end
```

because the node $\mathcal{V}_z$ has $(m+2)$ neighbors in total, then any two of these neighbors (except those that are already connected, assume u of them are already connected) will regard the node $\mathcal{V}_z$ as a common neighbor when calculating their similarity measures. Then node $\mathcal{V}_z$ will be calculated $(\frac{(m+2)(m+1)}{2} - u)$ times.

For each pair of unconnected nodes which have common neighbor $\mathcal{V}_z$, when we calculate node $\mathcal{V}_z$'s contribution on the similarity measure value, we firstly pick out edges which do not connect to the pair of nodes we are considering, then we apply Algorithm 2 to the list of these edges to get the probability list. For different pairs of unconnected nodes, the list of edges differ. To calculate $\mathcal{V}_z$'s contribution for one pair of nodes, the time complexity is $O(m^2)$. We have $(\frac{(m+2)(m+1)}{2} - u)$ pairs of unconnected nodes which have common neighbor $\mathcal{V}_z$, so the total time complexity will be $O(m^4)$.

This kind of time complexity is still very large. We can use the similar idea as we mentioned in Algorithm 2 to reduce the time complexity. In Algorithm 2, we use the probability lists of the left half sublist and the right half list to compute the probability list of the full list. Actually, Equation 5.6 has a more general form, which can be represented as follow:

$$P^n_{1\to m} = \sum_{i=max(0,n+k-m)}^{min(n,k)} P^i_{1\to k} P^{n-i}_{k+1\to m} \tag{5.7}$$

In Equation 5.6, we choose $k = \lfloor m/2 \rfloor$.

$P^n_{1\to m}$ stands for from edges e_1 to e_m, the probability that exactly n among them exist in possible worlds. To have n edges in possible worlds, i of n (i should be in the range of [0, n]) can be generated from edges e_1 to e_k (k has to be smaller than m), the probability for this case can be represented as $P^i_{1\to k}$; while the other $n-i$ of n can be generated from e_{k+1} to e_m, and the probability for this case is $P^{n-i}_{k+1\to m}$.

When we consider different pairs of unconnected nodes, node $\mathcal{V}_z$'s total

edges remain the same, what differs is the set of two edges which connects to the pair of nodes we are considering, and it results in the difference of the remaining edges list which will be used in Algorithm 2. To reduce the time complexity, the idea is to calculate the full edges list's corresponding probability list, which can be represented as $A = [P^0_{1\to m+2}, P^1_{1\to m+2}, ..., P^{m+2}_{1\to m+2}]$. For each pair of unconnected nodes, we want to calculate the remaining edges list's corresponding probability list, which can be represented as $B = [P^0_{1\to m}, P^1_{1\to m}, ..., P^m_{1\to m}]$. We can firstly find the two edges connecting to the pair of unconnected nodes, and calculate these two edges' corresponding probability list, which can be represented as $C = [P^0_{m+1\to m+2}, P^1_{m+1\to m+2}, P^2_{m+1\to m+2}]$. Then we can use A and C to calculate B based on the Equation 5.7. The full equations can be represented as follows:

$$\begin{cases} P^2_{m+1\to m+2}P^m_{1\to m} = P^{m+2}_{1\to m+2} \\ P^1_{m+1\to m+2}P^m_{1\to m} + P^2_{m+1\to m+2}P^{m-1}_{1\to m} = P^{m+1}_{1\to m+2} \\ P^0_{m+1\to m+2}P^m_{1\to m} + P^1_{m+1\to m+2}P^{m-1}_{1\to m} + P^2_{m+1\to m+2}P^{m-2}_{1\to m} = P^m_{1\to m+2} \\ P^0_{m+1\to m+2}P^{m-1}_{1\to m} + P^1_{m+1\to m+2}P^{m-2}_{1\to m} + P^2_{m+1\to m+2}P^{m-3}_{1\to m} = P^{m-1}_{1\to m+2} \\ ... \\ P^0_{m+1\to m+2}P^3_{1\to m} + P^1_{m+1\to m+2}P^2_{1\to m} + P^2_{m+1\to m+2}P^1_{1\to m} = P^3_{1\to m+2} \\ P^0_{m+1\to m+2}P^2_{1\to m} + P^1_{m+1\to m+2}P^1_{1\to m} + P^2_{m+1\to m+2}P^0_{1\to m} = P^2_{1\to m+2} \end{cases}$$

These equations are easy to solve, after we get A and C, we can calculate the probability list $[P^0_{1\to m}, P^1_{1\to m}, ..., P^m_{1\to m}]$ in $O(m)$. Though it takes $O(m^2)$ time to calculate A, when we consider different pairs of unconnected nodes which have common neighbor $\mathcal{V}_z$, A only needs to be calculated once. To calculate different pairs of unconnected nodes' corresponding probability list B, we can firstly calculate their probability C in constant time, and then use A and C to calculate B in $O(m)$. Because we have $(\frac{(m+2)(m+1)}{2} - u)$ pairs of

unconnected nodes, the time complexity of calculating A can be ignored. To calculate node $\mathcal{V}_z$'s contribution for all unconnected nodes which have common neighbor $\mathcal{V}_z$, the time complexity is $O(m^3)$, and the average time complexity of calculating node $\mathcal{V}_z$'s contribution for a certain pair of unconnected nodes will be $O(m)$. To calculate s_{xy}, we can calculate nodes $\mathcal{V}_x$ and $\mathcal{V}_y$'s each common neighbor's contribution for s_{xy} in $O(m)$, because nodes $\mathcal{V}_x$ and $\mathcal{V}_y$ have t common neighbors in total. The time complexity of computing s_{xy} is $O(mt)$. The overall algorithm for calculating s_{xy} can be found in Algorithm 4. Before running Algorithm 4, we need to do the initialization step, which is described in Algorithm 3. In Algorithm 3, we iterate through all nodes in graph $\mathcal{G}$, calculate their probability lists A and save them in a hash table. The hash table will be used in Algorithm 4.

Algorithm 3: Initialization

Data: An uncertain graph $\mathcal{G}$.
Result: Probability lists for all nodes in $\mathcal{G}$ saved in a hash table *dict*

```
1 Hash table dict ← {};
2 for each node V_z ∈ G do
3 |   Array uncertainEdgeList ← all uncertain edges connecting to node
  |     V_z;
4 |   dict[V_z] ← kEdgeProbability(uncertainEdgeList);
5 end
6 return dict;
```

5.2 Experiments

We, first, present the real datasets we used in our experiments and the approach we adopt to generate uncertain networks. Then, we evaluate our approach in contrast with the original CN, RA and their weighted versions, as well as other state-of-art link prediction approaches.

Algorithm 4: Resource Allocation Value Calculation

Data: Nodes $\mathcal{V}_x$, $\mathcal{V}_y$, an uncertain graph $\mathcal{G}$ and the hash table *dict* after running Algorithm 3

Result: Resource Allocation value for nodes $\mathcal{V}_x$ and $\mathcal{V}_y$ in $\mathcal{G}$

```
result ← 0;
for each node V_z ∈ Γ(x) ∩ Γ(y) do
    Array uncertainEdgeList ← [];
    probValue ← 1;
    for each node V_m connecting to node V_z do
        if V_m = V_x or V_m = V_y then
            probValue ← probValue × P_{m,z};
            add P_{m,z} to uncertainEdgeList;
        end
    end
    Array probListC ← kEdgeProbability(uncertainEdgeList);
    Array probListA ← dict[V_z];
    Array probListB ← use probListA and probListC to calculate
     probListB based on the Equation 5.7;
    oneNodeResult ← 0;
    for each i ∈ [0, len(probListB)) do
        oneNodeResult ← oneNodeResult + probListB[i] × 1/(i+2);
    end
    result ← result + probValue × oneNodeResult;
end
return result;
```

5.2.1 Datasets

Protein-Protein Interaction Network

We used the protein-protein interaction network (PPI) created by Krogan [30]. Two proteins are linked if it is likely that they interact. The core network consists of 2708 proteins and 7123 interactions labeled with probabilities.

Enron Network

We used the Enron dataset compiled by Shetty and Adibi [58]. The dataset is a subset of Enron employees, comprised of emails sent between employees, resulting in a dataset with 50,572 emails among 151 employees. We used the same method as Pfeiffer and Neville in [52] to assign each edge with a possibility of occurrence.

Synthetic Uncertain Network Based on Deterministic Network

Since there are not many publicly available uncertain network datasets, we also generated an uncertain network based on a deterministic network. The dataset we used here is USAir. The US air transportation network contains 332 airports and 2126 airlines. Based on this network, we use the uncertain network generator (Algorithm 1 as mentioned in Chapter 3) to generate its corresponding uncertain network. In this experiment, the percentage of non-existential edges we choose to add is 20%.

5.2.2 Experiments

To test the prediction performance of an algorithm, the observed edges, E, are divided into two separate sets: training set E^T, is regarded as known information; and probe set E^P, is used for testing and no information therein is allowed to be used for prediction. Clearly, we have $E^T \cup E^P = E$ and

$E^T \cap E^P = \emptyset$.

For the protein-protein interaction network and the synthetic uncertain network, we only know their connection information, so the training set E^T and the probe set E^P can be randomly divided. In this paper, the training set E^T and the probe set E^P are assumed to contain 90% and 10% of the links respectively. To get more reliable result, each value is obtained by averaging over 100 independent runs of random divisions into the training set and probe set.

Link prediction algorithms should be capable of detecting the dynamic relationships between members in a temporal social network. Because the Enron dataset is time-evolving, the relations among social members change continuously over time, and links are constantly varying and evolving. Using link prediction algorithms, we should be able to predict newly added links in future networks. In the experiment, we predict new communications between two employees in Enron Corporation after Jan. 16, 2001, based on historical data. The idea is that, if two employees have email records before Jan. 16, 2001, we generate a potential edge between them. Then we assign these edges with a probability following the method described in [58]. The resulting probabilistic graph consists of 113 nodes and 419 edges, and this graph is regarded as the training set. The testing set is formed by taking in all the edges formed after Jan. 16, 2001. After discarding employees that have not appeared in the list of the 113 employees, as well as the edges that have appeared both before and after Jan. 16, 2001, we obtained 578 ground-truth edges with 113 distinct employees.

To evaluate the performance of prediction algorithms, we apply a standard Precision metric to quantify the accuracy of the prediction, which focuses on top-ranked latent links. It is defined as L_r/L, where among top-L candidate links, L_r is the number of accurate predicted links actually appearing in the

testing period.

5.2.3 Results and Evaluation

As the literature suggested [36], [38], [39], [60], [67], the top L is set to 100 in our experiments. In this section, we compare our metrics (UCN and URA) and other metrics/algorithms using existing ground truth. The name for these metrics/algorithms and their corresponding descriptions are shown in Table 5.1. To evaluate our metrics, we mainly focus on the comparison between the uncertain version of graph proximity measures with weighted and unweighted ones. Besides, we also use the LNB model and the SRW metric (with different parameters) to assess our metrics as they are popular and considered as state-of-the-art. LNB is a local Naïve Bayes model which is based on neighbor-based metrics, and SRW is a local-random-walk-based algorithm (we choose $t = 2$ and $t = 3$ in our experiments because they are the optimal choices based on Liu and Lü's experiments in [35]). Since LNB and SRW algorithms are for deterministic networks, in our experiments we ignore the edge probabilities and consider uncertain networks as normal deterministic networks.

The prediction accuracies on the three networks are shown in Table 5.2 and Figures 5.3, 5.4. PPI illustrates the efficacy for predicting connections in real uncertain networks, Enron for predicting future connections in temporal networks, and a synthetic network.

From Table 5.2 and Figure 5.3, we can observe that our uncertain version of the Common Neighbor and Resource Allocation metrics can significantly outperform their original and weighted ones when dealing with uncertain networks. This shows that in the task of link prediction with edge uncertainty, it is worthwhile to take every possible worlds into account. Considering uncertainties as weights and applying a weighted version of the metrics, while it improves over the original approach, it does not really take advantage of the

Table 5.1: Algorithm List

Algorithm Name	Description
CN/RA	Pay no attention to probabilities and use the original metrics.
WCN/WRA	Regard probability as weight and use weighted metrics.
UCN/URA	Use our uncertain version of graph proximity measures.
SRW2	Pay no attention to probabilities and run local random walk algorithm as mentioned in [35], and we choose $t = 2$
SRW3	Pay no attention to probabilities and run local random walk algorithm as mentioned in [35], and we choose $t = 3$
LNB-CN	Pay no attention to probabilities and use Local Naïve Bayes form of Common Neighbors as mentioned in [36]
LNB-RA	Pay no attention to probabilities and use Local Naïve Bayes form of Resource Allocation as mentioned in [36]

Table 5.2: Comparative Results for the original, weighted and uncertain versions of Common Neighbors and Resource Allocation

Datasets	Common Neighbor			Resource Allocation		
	CN	WCN	UCN	RA	WRA	URA
PPI	0.472	0.5045	**0.5288**	0.4123	0.45	**0.5728**
Enron	0.49	0.52	**0.61**	0.51	0.47	**0.52**
Synthetic Network	0.5812	0.5954	**0.6043**	0.6075	0.6124	**0.6233**

Datasets	SRW2 [35]	SRW3 [35]	LNB-CN [36]	LNB-RA [36]
PPI	0.4136	0.5284	0.4856	0.4992
Enron	0.43	0.45	0.55	0.46
Synthetic Network	0.5852	0.5992	0.5962	0.5885

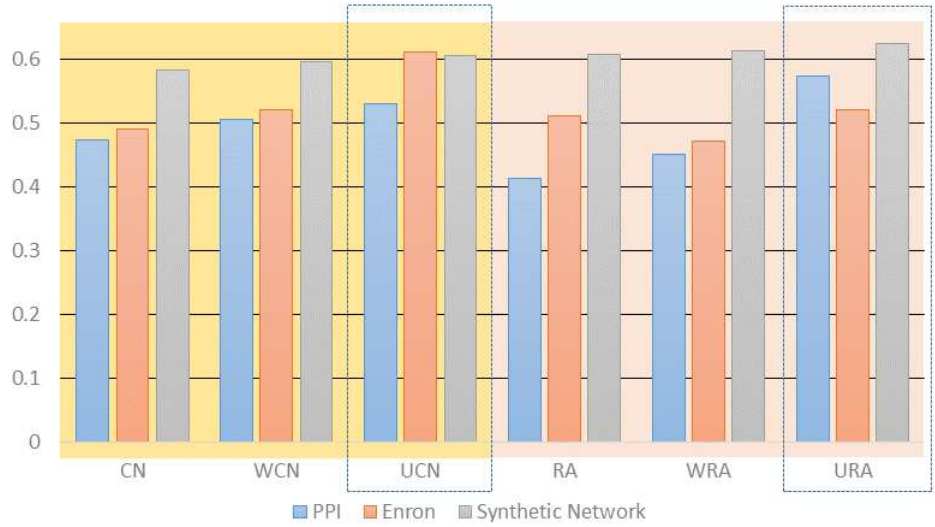

Figure 5.3: Visual comparison of the accuracy for link prediction with UCN and URA with wighted and unweighted versions.

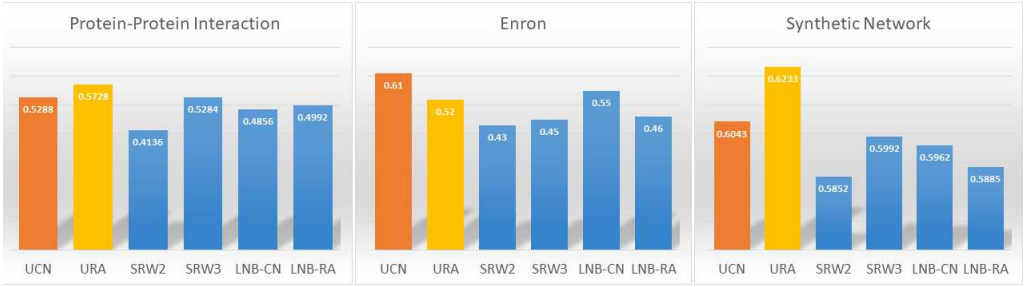

Figure 5.4: Comparison of the accuracy for link prediction with UCN and URA against SRW [35] and LNB [36].

notion of possible parallel worlds as our metric does.

From Table 5.2 and Figure 5.4, we can observe that our metrics (UCN and URA) can outperform the other four baseline methods on PPI and Synthetic datasets. The contenders are again the Local Random Walk with two different parameters and the Local Naïve Bayes model with either CN or RA. The Enron dataset allows the following observation: the Common Neighbor-based metrics seems to outperform the Resource Allocation-based counterparts on this dataset. It seems that the Resource Allocation metrics are not good choices for Enron dataset. While our UCN wins the contest for Enron, it is understandable that LNB-CN, leveraging on Common Neighbors, slightly outshines our URA metric.

For run time, based on our experiments, we find UCN to be just a little bit slower than CN, but it has almost the same run time as WCN; and URA is around 2 to 3 times slower than RA and WRA.

5.3 Conclusion

In this chapter, we propose an uncertain version of graph proximity measures for the link prediction problem in uncertain networks. We propose a new algorithm to reduce the time complexity of computing the uncertain version of graph proximity measures. By taking all possible worlds into consideration, the performance of link predictions are improved compared with the original and weighted proximity measures.

We have also shown the superiority of our approach for link prediction in uncertain networks compared to the state-of-the-art in link prediction. The contenders, however, do not take into account the existential probabilities of the edges and future work is to investigate how the local random walk or the local Naïve Bayes model could do that. We have shown the effectiveness of considering all possible worlds when using neighbor-based metrics to do link

prediction. When proposing the uncertain version of other link prediction metrics, such as path-based and learning-based metrics, all possible worlds should also be considered, which would also be very time-consuming. To reduce time complexity, some variants of our algorithm may then be considered.

Chapter 6

Local Community Detection

6.1 Local Community Detection Algorithm for Uncertain Networks

6.1.1 Local Modularity $\mathcal{UR}$ in Uncertain Networks

In Section 2.3.3, we review some local community detection algorithms for deterministic networks. Inspired by the local modularity $\mathcal{R}$, in order to solve the problem of detecting local communities with edge uncertainty, one intuitive approach is to convert the uncertain community detection problem into the deterministic scenario by using edge probability. In the uncertain scenario, the local modularity $\mathcal{UR}$ for uncertain networks can be defined as follows:

$$\mathcal{UR} = \frac{\mathbb{E}(\mathcal{B}_{in\_edge})}{\mathbb{E}(\mathcal{B}_{in\_edge}) + \mathbb{E}(\mathcal{B}_{out\_edge})} \tag{6.1}$$

where $\mathbb{E}(\mathcal{B}_{in\_edge})$ is the expected number of edges that connect boundary nodes and other nodes in $\mathcal{D}$, which can be represented as:

$$\mathbb{E}(\mathcal{B}_{in\_edge}) = \frac{1}{2} \sum_{\mathcal{V}_i \in \mathcal{B}, \mathcal{V}_j \in \mathcal{B}, i \neq j} \mathcal{P}_{i,j} + \sum_{\mathcal{V}_i \in \mathcal{B}, \mathcal{V}_j \in \mathcal{C}} \mathcal{P}_{i,j} \tag{6.2}$$

while $\mathbb{E}(\mathcal{B}_{out\_edge})$ is the expected number of edges that connect boundary nodes and nodes in $\mathcal{S}$, which can be represented as:

$$\mathbb{E}(\mathcal{B}_{out\_edge}) = \sum_{\mathcal{V}_i \in \mathcal{B}, \mathcal{V}_j \in \mathcal{S}} \mathcal{P}_{i,j} \tag{6.3}$$

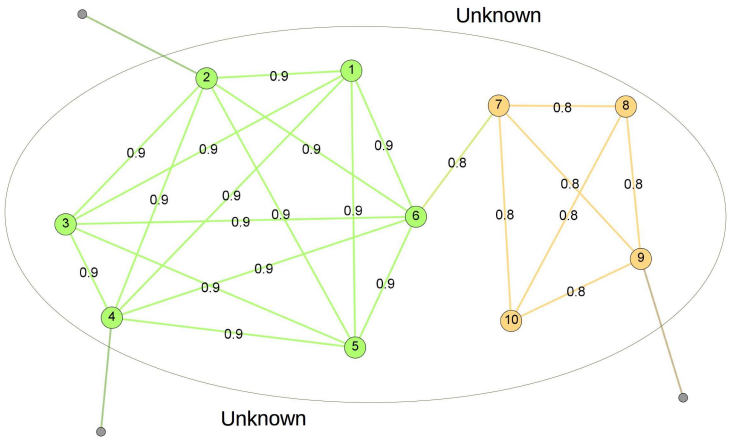

Figure 6.1: An example showing the problem when only using $\mathcal{UR}$ to find the local community.

After replacing $\mathcal{R}$ with $\mathcal{UR}$, we can use the algorithm mentioned in [15], [17] to find the local community for the input node.

6.1.2 Reviews of the Previous Method

However, simply using $\mathcal{UR}$ to replace $\mathcal{R}$ and applying Clauset and Chen's original local community detection algorithm (as mentioned in Section 2.3.3) will cause some problems. In uncertain networks, there are some noise edges between nodes, which may be generated by missed observations, misreporting or wrong inference. Although some of them are assigned low probability, they do not actually exist and can be regarded as noise. Due to these kinds of noise, if the algorithm starts from a node $\mathcal{V}_i$ in community A, the expansion step might fall into a different neighbor community B. Figure 6.1 shows an example.

Community A is a 6-vertex clique, and the probability over edges in community A are all 0.9. Community B is a 4-vertex clique, and the probability over edges in community B are all 0.8. Besides these edges, there is an edge between node $\mathcal{V}_6$ and node $\mathcal{V}_7$ with a probability of 0.8. There are also some other edges between other nodes of community A and B and the unknown part

of the network. If the start node is $\mathcal{V}_6$, we want to find the local community for node $\mathcal{V}_6$. The algorithm mentioned in [15], [17] starts the expansion step from the start node's neighbors and adds the node which results in the largest increase in $\mathcal{R}$ to the community. In this uncertain example, we use $\mathcal{UR}$. $\mathcal{V}_1$ to $\mathcal{V}_5$ and node $\mathcal{V}_7$ are all neighbors of $\mathcal{V}_6$, so they are all candidates. The $\mathcal{UR}$ of the community, after adding $\mathcal{V}_1, \mathcal{V}_3$ or $\mathcal{V}_5$, are all

$$\frac{0.9}{4 \times 0.9 + (4 \times 0.9 + 0.8) + 0.9} = 0.1011 \tag{6.4}$$

After adding node $\mathcal{V}_2$ or $\mathcal{V}_4$, the $\mathcal{UR}$ will be less than 0.1011 due to extra outward edges. The $\mathcal{UR}$ of the community after adding node $\mathcal{V}_7$ is

$$\frac{0.8}{5 \times 0.9 + 3 \times 0.8 + 0.8} = 0.1039 \tag{6.5}$$

The algorithm will add node $\mathcal{V}_7$ to $\mathcal{C}$ because it results in the largest increase in $\mathcal{UR}$. Then nodes $\mathcal{V}_8, \mathcal{V}_9, \mathcal{V}_{10}$ (or $\mathcal{V}_{10}, \mathcal{V}_9, \mathcal{V}_8$, because nodes $\mathcal{V}_8$ and $\mathcal{V}_{10}$ are exactly the same) will be added to $\mathcal{C}$ one by one. In this example, when we input the node $\mathcal{V}_6$, the algorithm will regard nodes $\mathcal{V}_6$ to $\mathcal{V}_{10}$ as node $\mathcal{V}_6$'s local community, while it should be nodes $\mathcal{V}_1$ to $\mathcal{V}_6$.

In this example, the structure of the network is clear, but the algorithm still makes a wrong decision. Chen et al. also reported a similar problem in the deterministic network in [14], but they regarded these start nodes as periphery nodes and did not provide local communities for these nodes. In other uncertain networks, there are more (noise) edges between communities, and the probability over some edges are low while others are high, which make the situation even more complex. In complex uncertain networks, more nodes will encounter the aforementioned problem and be grouped into their neighbor communities. We can not just simply regard these nodes as periphery nodes and report no community for them.

6.1.3 Introduction of the New Measure $\mathcal{K}$

The reason why the original algorithm does not perform well in uncertain networks is that $\mathcal{UR}$ only measures the sharpness of the boundary. However, in the uncertain scenario, the boundary between communities becomes less clear due to the appearance of noise edges. One main drawback of the local modularity $\mathcal{UR}$ is that it only cares about the nodes in $\mathcal{D}$ and pays no attention to the difference between shell nodes and unknown area's nodes. At this moment, though shell nodes are not part of the community, they can be regarded as the neighbors of the community, and they can give us extra information about the community. In the research of link prediction, people propose the measure called Common Neighbor (CN) to find the potential links. Researchers find that two nodes are more likely to form a link if they have many common neighbors. This idea can also be used in the local community detection problem. It is easy to understand that nodes in the same community share some common neighbors, even though they do not have direct links with each other. Inspired by this idea, in order to solve the existing problem in uncertain scenarios mentioned previously, a new measure $\mathcal{K}$ is introduced, which not only pays attention to nodes in $\mathcal{B}$, but also nodes in $\mathcal{S}$.

$$\mathcal{K}_i = \mathbb{E}(N_{i,in\_edge}) + \mathbb{E}(N_{i,shell\_edge}) \tag{6.6}$$

where $\mathbb{E}(N_{i,in\_edge})$ is the expected number of edges that connect candidate node $\mathcal{V}_i$ and other nodes in $\mathcal{D}$, which can be represented as:

$$\mathbb{E}(N_{i,in\_edge}) = \sum_{\mathcal{V}_j \in \mathcal{D}} \mathcal{P}_{i,j} \tag{6.7}$$

while $\mathbb{E}(N_{i,shell\_edge})$ is the expected number of edges that connect candidate node $\mathcal{V}_i$ and other nodes in $\mathcal{S}$, which can be represented as:

$$\mathbb{E}(N_{i,shell\_edge}) = \sum_{\mathcal{V}_j \in \mathcal{S}} \mathcal{P}_{i,j} \cdot \mathcal{P}_{j,shell} \tag{6.8}$$

$$\mathcal{P}_{j,shell} = 1 - \prod_{\mathcal{V}_m \in \mathcal{D}} (1 - \mathcal{P}_{j,m}) \tag{6.9}$$

The new measure $\mathcal{K}$ aims to measure how close the relationship is between the candidate node and the existing community. The larger the value $\mathcal{K}$ is, the closer the relationship between the community and the candidate node will be. This measure will be used to choose which neighboring node should be added to $\mathcal{C}$ (and to $\mathcal{B}$, if necessary) in the first few steps. It is worth noting that:

1. $\mathbb{E}(N_{i,shell\_edge})$ is not simply the sum of probability over edges between candidate node $\mathcal{V}_i$ and nodes in $\mathcal{D}$, but it also cares about the true probability of at least one edge between shell node $\mathcal{V}_j$ and the existing community being present, which is denoted by $\mathcal{P}_{j,shell}$.

2. The start node is more likely to merge other communities' nodes at the first few steps of the discovery phase. With the increase in the number of nodes, the possibility of wrongly adding other communities' nodes will be significantly reduced, so $\mathcal{K}$ will only be mainly considered in the first few steps of the discovery phase ($\mathcal{K}$ will also be considered when $\mathcal{UR}$ ties in future steps).

In the example mentioned in Figure 6.1, the $\mathcal{K}$ value for nodes $\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \mathcal{V}_4, \mathcal{V}_5$ and $\mathcal{V}_6$ are calculated as follows:

$$\mathcal{K}_{i=1,2,3,4,5} = 0.9 + 0.9 \times 0.9 \times 4 = 4.14 \tag{6.10}$$

$$\mathcal{K}_{i=7} = 0.8 \tag{6.11}$$

$\mathcal{K}_{i=7}$ is smaller than $\mathcal{K}_{i=1,2,3,4,5}$, so we will first exclude node $\mathcal{V}_7$. As mentioned in Equation 6.4, the $\mathcal{UR}$ of the community after adding $\mathcal{V}_1, \mathcal{V}_3$ or $\mathcal{V}_5$ is 0.1011 while the $\mathcal{UR}$ of the community after adding $\mathcal{V}_2$ or $\mathcal{V}_4$ is less than

0.1011. The node added to the community $\mathcal{C}$ will be randomly chosen from nodes $\mathcal{V}_1, \mathcal{V}_3$ and $\mathcal{V}_5$.

6.1.4 Full Algorithm Description

Similar to Clauset and Chen's greedy algorithm mentioned in Section 2.3.3, our algorithm also firstly place the start node in the community. At each step, we sort candidate nodes based on their $\mathcal{K}$ (first few steps) or $\mathcal{UR}$ values (other steps). After all candidate nodes are sorted, the algorithm will add the first node (which can increase the community's $\mathcal{UR}$) to the community $\mathcal{C}$. This process will stop when there are no remaining nodes in $\mathcal{S}$ which can increase the community's $\mathcal{UR}$. It is worth noting that candidate nodes are sorted based on $\mathcal{K}$ value in the first few steps. However, the number of steps is not yet decided. It is a tunable hyper-parameter, and we use λ to represent it. We will demonstrate how to find the optimal λ in Section 6.2.3. The full algorithm description can be found in Algorithm 5.

6.2 Experiments

6.2.1 Datasets

To compare methods and evaluate them for use in practical applications, both synthetic and real-world networks are used in the experiments.

Real-World Networks

The two real-world networks used are classics in network science and describe different types of networks: human friendships and football matches. These networks all have a known community structure which is supplied by external labels.

The karate network [65] describes friendships between members of a karate club at a U.S. university in 1977. The core network consists of 34 nodes and

Algorithm 5: Local Community Identification with Edge Uncertainty

Data: A network $\mathcal{G}$, a start node $\mathcal{V}_0$ and number of steps λ.
Result: A local community for $\mathcal{V}_0$

Add $\mathcal{V}_0$ to $\mathcal{D}$ and $\mathcal{B}$, add all $\mathcal{V}_0$'s neighbors to S, $\mathcal{UR} \leftarrow 0$;
repeat
  Array *nodelist* $\leftarrow []$;
  for *each* $\mathcal{V}_i \in S$ **do**
    Compute $\mathcal{UR}_i$;
    // $\mathcal{UR}_i$ represents the $\mathcal{UR}$ value after adding node $\mathcal{V}_i$;
    Compute $\mathcal{K}_i$;
    Add $\mathcal{V}_i$ to *nodelist*;
  end
  if $|\mathcal{D}| < \lambda$ **then**
    Sort *nodelist* first by $\mathcal{K}_i$, then by $\mathcal{UR}_i$;
  else
    Sort *nodelist* first by $\mathcal{UR}_i$, then by $\mathcal{K}_i$;
  end
  // If some nodes have same $\mathcal{K}_i$ and $\mathcal{UR}_i$, break ties randomly;
  for *each* $\mathcal{V}_i \in$ *nodelist* **do**
    if $\mathcal{UR}_i > \mathcal{UR}$ **then**
      $\mathcal{UR} \leftarrow \mathcal{UR}_i$;
      Add $\mathcal{V}_i$ to $\mathcal{D}$;
      Remove $\mathcal{V}_i$ from $\mathcal{S}$;
      Update $\mathcal{B}$, $\mathcal{D}$;
      Update shell nodes possibility based on Equation 6.9;
      break for loop
    end
  end
until *no new node is added to* $\mathcal{D}$;
return $\mathcal{D}$

Table 6.1: Parameters for Generating Synthetic Networks

Variable	Value	Description
N	100	number of nodes
k	10	average degree
k_{max}	30	maximum degree
μ	0.2	mixing parameter
c_{min}	15	minimum for the community sizes
c_{max}	25	maximum for the community sizes

78 edges. The club fractured into two parts during the study and the resulting two groups are the labels used for the external evaluation. It is assumed that the community structure can be recovered using a good community detection algorithm.

The football network [23] contains all the Division IA college football teams and the edges indicate games during the fall of 2000. The total number of teams is 115 and the total number of matches is 613. The labels are the conferences to which each team belongs and matches are most often played between teams from the same conference. Therefore communities detected in this network should indicate the different conferences.

Synthetic Networks

The synthetic network model used in this thesis is adopted from [31]. The authors have constructed algorithms to generate synthetic networks with community structures, which has become a standard benchmark for community detection using synthetic networks. The networks are generated using six different input parameters, shown in the Table 6.1, together with the values used in this thesis. These parameters allow for the generation of families of networks with desired properties.

Generating Uncertain Networks

Since there are not many publicly available uncertain network datasets, we use Algorithm 1 as mentioned in Chapter 3 to generate uncertain networks based on the former 3 deterministic networks.

In experiments, the percentage of non-existential edges we choose to add range from 10% to 40%. Besides, we also evaluate our algorithm on original deterministic networks, which can be regarded as a special case of uncertain networks.

6.2.2 Evaluation

The most important step is to evaluate our algorithm on real-world networks and synthetic networks. In this section, we compare our algorithm (UR+K) and other algorithms by supervised evaluation and unsupervised evaluation. The name for these algorithms and their corresponding descriptions are shown in Table 6.2. We mainly focus on the comparison between our algorithm and the other two local community detection algorithms (R and UR). Though Louvain algorithm is not a local community detection algorithm, we also compare our algorithm with it and its variant because it is also a greedy optimization method and it is always regarded as a baseline in the research of community mining. As mentioned previously, in our algorithm, λ is a tunable hyper-parameter. In this section, we choose $\lambda = 3$, and we will demonstrate how we find the optimal λ in Section 6.2.3.

All uncertain networks are randomly generated based on deterministic networks. To get more reliable results, for each deterministic network, we randomly generate 100 uncertain networks, and all values shown in result tables/figures are average values over 100 uncertain networks.

Table 6.2: Algorithm List

Algorithm	Description
R	original local community detection algorithm based on the local modularity R
UR	original local community detection algorithm based on the uncertain version local modularity UR as mentioned in Equation 6.1
UR+K	our algorithm as mentioned in Algorithm 5
Louvain	Louvain algorithm
ULouvain	regard probability as weight and run weighted version of the Louvain algorithm

Supervised Evaluation

One way to compare community detection results is supervised evaluation. We use a similar evaluation method as mentioned in [14]. We provide networks with absolute community ground truth to the algorithm, but limit its access to network information to local nodes only (Louvain and ULouvain algorithms are allowed to use global information). The only way for the algorithm to obtain more network knowledge is to expand the community, one node at a time. Therefore, we can evaluate our algorithm based on the comparison between ground truth and results obtained by our algorithm, while satisfying limitations for local community identification.

We compare our algorithm with other algorithms on 2 real-world networks and 1 synthetic network. For each network, each node is taken as the start point for algorithms one by one. Assume the start point is $\mathcal{V}_i$, we use $\mathcal{D}_i$ to represent the local community for the start point $\mathcal{V}_i$ after running local community detection algorithms, and we use $\mathcal{D}_{i'}$ to represent the set of nodes which have the same label as the start node $\mathcal{V}_i$. To quantify the accuracy of local community detection algorithms, based on the ground truth, we use F1-measure as our evaluation metrics. F1-measure is defined as the harmonic mean of precision and recall. The definition of precision, recall and F1-measure

are as follows:

$$\text{precision} = \frac{\sum_{\mathcal{V}_i \in \mathcal{G}} |\mathcal{D}_i \cap \mathcal{D}_{i\prime}|}{\sum_{\mathcal{V}_i \in \mathcal{G}} |\mathcal{D}_i|} \tag{6.12}$$

$$\text{recall} = \frac{\sum_{\mathcal{V}_i \in \mathcal{G}} |\mathcal{D}_i \cap \mathcal{D}_{i\prime}|}{\sum_{\mathcal{V}_i \in \mathcal{G}} |\mathcal{D}_{i\prime}|} \tag{6.13}$$

$$\text{F1-measure} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} \tag{6.14}$$

The results on three networks are shown in Figure 6.2.

From Figure 6.2, we can observe that our algorithm (UR+K) can significantly outperform other algorithms in terms of F1-measure in karate club dataset and football dataset. Though our algorithm cannot beat Louvain and ULouvain algorithms in synthetic network, it also shows competitive detection accuracy, and it still performs better than algorithms R and UR on synthetic dataset. Considering our algorithm only uses local information, while Louvain and ULouvain use global information, these results are still satisfying.

Unsupervised Evaluation

Since we generate uncertain networks based on deterministic networks, when we evaluate our algorithm in the supervised way, we assume that uncertain networks have the same community ground truth as their corresponding deterministic networks. This assumption is true if we only add a few noise edges to uncertain networks because a small number of noise edges will not have an impact on the community structure. However, with the increase in the number of noise edges, some communities may merge into one community. In this scenario, using the original community labels will cause problems.

To solve this problem, we also propose an unsupervised way to evaluate our algorithm. As mentioned in Section 2.3.3, the local community modularity $\mathcal{R}$ can be used to measure the quality of the local community. In the uncertain networks scenario, we can use $\mathcal{UR}$. Therefore, to compare different algorithms,

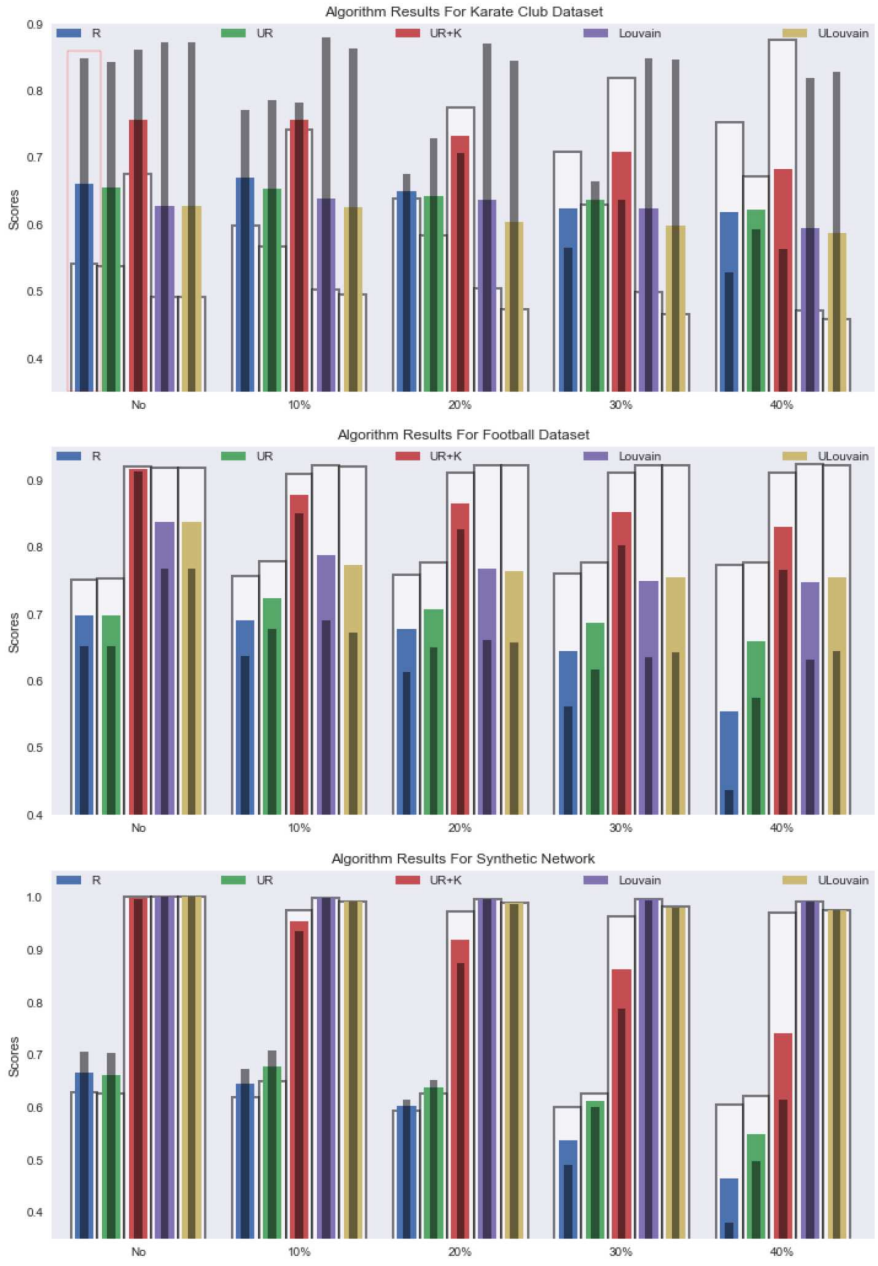

Figure 6.2: Algorithm results on karate club, football and synthetic networks. For each network, we add 10%, 20%, 30% and 40% non-existential edges to the original network. The original network is also regarded as a special case of uncertain network. For each uncertain network, we compare our algorithm (UR+K) with the other 4 algorithms. The precision, recall and F1-measure values of each algorithm are all shown in the graphs. The F1-measure values gained by different algorithms are represented by different colored bars. Each F1-measure value's corresponding precision and recall values are represented by its inner black bar and external white bar respectively.

we can compare local community modularity $\mathcal{UR}$ values after running different algorithms.

In this part, we also compare our algorithm (UR+K) with other algorithms on 2 real-networks and 1 synthetic network. The results are shown in Tables 6.3, 6.4, 6.5.

Table 6.3: $\mathcal{UR}$ on Karate Club Data

Noise	R	UR	UR+K	Louvain	ULouvain
No	0.5787	0.5789	**0.654**	0.5604	0.5604
10%	0.584	0.5902	**0.7026**	0.5176	0.5258
20%	0.6235	0.5942	**0.7462**	0.4737	0.4858
30%	0.7065	0.6232	**0.7858**	0.4326	0.4583
40%	0.7628	0.6641	**0.8601**	0.3965	0.4281

Table 6.4: $\mathcal{UR}$ on Football Data

Noise	R	UR	UR+K	Louvain	ULouvain
No	0.5065	0.5057	0.5327	**0.5497**	**0.5497**
10%	0.4714	0.4826	0.4902	0.5152	**0.5208**
20%	0.4378	0.4491	0.454	0.4783	**0.4802**
30%	0.415	0.4189	0.4221	0.4384	**0.4463**
40%	0.4121	0.4008	0.4017	0.4134	**0.4164**

Table 6.5: $\mathcal{UR}$ on Synthetic Data

Noise	R	UR	UR+K	Louvain	ULouvain
No	0.6018	0.6019	**0.6537**	**0.6537**	**0.6537**
10%	0.5476	0.5361	**0.5928**	0.5924	0.5924
20%	0.5016	0.4941	**0.5484**	0.5415	0.5414
30%	0.4964	0.4685	**0.5306**	0.5003	0.5
40%	0.5232	0.4735	**0.5742**	0.4635	0.4642

From Tables 6.3, 6.4, 6.5, we can find our algorithm (UR+K) performs the best on karate club and synthetic datasets, and the $\mathcal{UR}$ values achieved on football dataset by our algorithm are very close to the best results achieved by the other algorithms. It is worth noting that, in the expansion steps, though the UR algorithm always chooses the node which gives the largest increase of $\mathcal{UR}$, while $\mathcal{UR}$ value is not mainly considered in the first few steps in

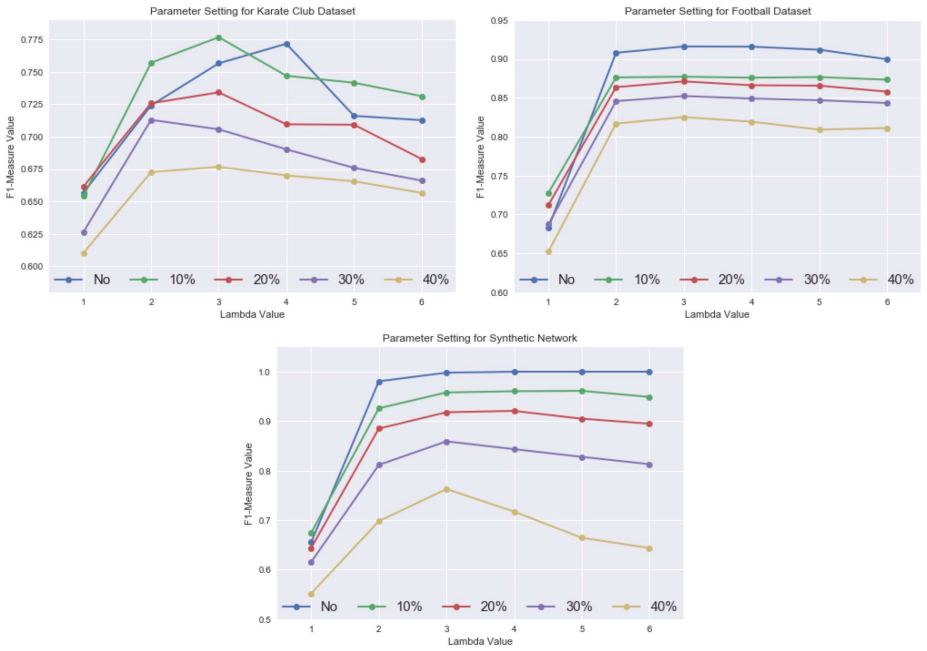

Figure 6.3: Hyper-parameter Validation. For each network, we add 10%, 20%, 30% and 40% non-existential edges to the original network. The original network is also regarded as a special case of uncertain network.

our algorithm, our algorithm finally achieved higher $\mathcal{UR}$ values than the UR algorithm on all datasets.

6.2.3 Hyper-Parameter Evaluation

Our algorithm has a hyper-parameter λ. As we mentioned in Section 6.1.4, the choice of hyper-parameter λ is another crucial topic in our experiment. To validate that the λ value we use is a reasonable choice, we conduct experiments in this section to find the optimal λ. We do this by repeating previous supervised evaluation experiments on karate club, football and synthetic networks over a range of different λ values, as shown in Figure 6.3.

By using different λ values, we run our algorithm (UR+K) on three networks and get their corresponding F1-measure values.

From these experiments, we can conclude that though the optimal choice of the hyper-parameter λ varies with networks, the best choice of the hyper-

parameter λ is 3 or 4 in almost all cases. In most cases, $\lambda = 3$ performs the best compared to the other values. Even in the other cases when $\lambda = 3$ is not the optimal choice, the results achieved by $\lambda = 3$ is also competitive compared to the results achieved by the optimal λ value. Therefore, it is reasonable to choose $\lambda = 3$ when running our algorithm.

6.3 Conclusion

In this chapter, we provide a novel approach which is able to detect local communities in uncertain networks. By taking our new measure $\mathcal{K}$ into consideration, our algorithm can avoid the problem in which periphery nodes tend to be grouped into their neighbor communities, and we experimentally show that our algorithm can outperform the other local community detection algorithms. However, one drawback is that using $\mathcal{K}$ in the algorithm inevitably results in additional computation. The $\mathcal{K}$ value is mainly considered in the first few steps, after that, $\mathcal{K}$ will only be considered when $\mathcal{UR}$ has ties. To reduce computation, when $steps \geqslant \lambda$, we can ignore $\mathcal{K}$ and no longer calculate it at those steps. When $\mathcal{UR}$ has ties, we can just randomly pick one node from all nodes which gives the equally largest increase of $\mathcal{UR}$ to the community. Though it will affect the performance of our algorithm, the extent of the impact is not significant, since $\mathcal{UR}$ rarely has ties in uncertain networks. Even though $\mathcal{UR}$ can sometimes have ties, choosing a node which does not have the largest $\mathcal{K}$ value will only have a slight impact on the final detection result, because the detected local community is stable enough at that stage.

We have shown the effectiveness of applying $\mathcal{K}$ to the original local community detection algorithm. A future direction is that the new measure $\mathcal{K}$ may also be applied in global community detection algorithms that deal with edge existential uncertainty. Many hierarchical clustering-based algorithms, such as Louvain, also use the greedy strategy to maximize the modularity gain in

the agglomeration phase. When devising an equivalent approach for uncertain graphs, when detecting global communities, we may encounter some similar problems as we mentioned in Section 6.1.2, and our measure $\mathcal{K}$ may then be considered.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we first study existing algorithms for deterministic networks, then we try to develop their uncertain versions. However, for different tasks, we encounter different problems. To solve these problems, we propose different algorithms.

For the entity ranking task, we aim to propose a algorithm which is able to work effectively in both unweighted and weighted uncertain networks. By taking the inverse of edge probability with a hyper-parameter, we can use existing centrality measures to rank nodes. For the link prediction task, uncertain edges result in a very large number of possible worlds, and we propose a divide and conquer algorithm to reduce time complexity. For the local community detection task, we find periphery nodes tend to be grouped into their neighbor communities in uncertain networks, and we introduce a new measure $\mathcal{K}$ to help our algorithm find reliable local communities.

In the evaluation part, we use supervised, unsupervised and illustrative experiments to show the effectiveness of our methods.

7.2 Future Work

As mentioned in Section 1.1.2, there are at least four types of uncertain networks. In this thesis, we only research on networks with edge uncertainty. For future study, we may can consider to solve complex network analysis problems in the context of other types of uncertainty.

Another interesting topic is about graph representation. Recently, there has been a surge of approaches that seek to learn representations that encode structural information about the graph. The idea behind these representation learning approaches is to learn a mapping that embeds nodes, or entire (sub)graphs, as points in a low-dimensional vector space. The goal is to optimize this mapping so that geometric relationships in this learned space reflect the structure of the original graph. After optimizing the embedding space, the learned embeddings can be used as feature inputs for downstream machine learning tasks, such as node classification and link prediction. Many recent successful methods belong to random walk approaches, such as DeepWalk [50], LINE [61] and node2vec [25]. The task of graph representation has attracted a lot of attention recently, however, the same problem is that they all focused on deterministic networks. Graph representation with edge uncertainty is also an important task but not yet solved. Therefore, this may be a direction we can further explore.

References

[1] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social networks*, vol. 25, no. 3, pp. 211–230, 2003. 2, 16, 17

[2] L. Adamic and E. Adar, "How to search a social network," *Social networks*, vol. 27, no. 3, pp. 187–203, 2005. 3

[3] N. M. Ahmed and L. Chen, "An efficient algorithm for link prediction in temporal uncertain social networks," *Information Sciences*, vol. 331, pp. 120–136, 2016. 5, 21

[4] R. Albert, H. Jeong, and A.-L. Barabási, "Internet: Diameter of the world-wide web," *Nature*, vol. 401, no. 6749, pp. 130–131, 1999. 1

[5] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, IEEE, 2006, pp. 475–486. 28

[6] A.-L. Barabâsi, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek, "Evolution of the social network of scientific collaborations," *Physica A: Statistical mechanics and its applications*, vol. 311, no. 3, pp. 590–614, 2002. 18

[7] E. R. Barnes, "An algorithm for partitioning the nodes of a graph," *SIAM Journal on Algebraic Discrete Methods*, vol. 3, no. 4, pp. 541–550, 1982. 23

[8] P. Bedi and C. Sharma, "Community detection in social networks," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 6, no. 3, pp. 115–135, 2016. 3

[9] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: Theory and experiment*, vol. 2008, no. 10, P10008, 2008. 3, 22, 24

[10] P. Bonacich, "Power and centrality: A family of measures," *American journal of sociology*, vol. 92, no. 5, pp. 1170–1182, 1987. 2, 14

[11] D. M. Boyd, "Friendster and publicly articulated social networking," 2004. 3

[12] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Computer networks*, vol. 33, no. 1, pp. 309–320, 2000. 1

[13] C. Cao, Q. Ni, and Y. Zhai, "An improved collaborative filtering recommendation algorithm based on community detection in social networks," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ACM, 2015, pp. 1–8. 3

[14] J. Chen, O. R. Zaïane, and R. Goebel, "Local community identification in social networks," in *Social Network Analysis and Mining, 2009. ASONAM'09. International Conference on Advances in*, IEEE, 2009, pp. 237–242. 6, 27, 64, 71

[15] J. Chen, O. R. Zaïane, and R. Goebel, "Detecting communities in large networks by iterative local expansion," in *Computational Aspects of Social Networks, 2009. CASON'09. International Conference on*, IEEE, 2009, pp. 105–112. 6, 8, 26, 63, 64

[16] T. Choudhury, M. Philipose, D. Wyatt, and J. Lester, "Towards activity databases: Using sensors and statistical models to summarize people's lives.," 2006. 3

[17] A. Clauset, "Finding local community structure in networks," *Physical review E*, vol. 72, no. 2, p. 026 132, 2005. 6, 8, 26, 63, 64

[18] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066 111, 2004. 3, 22, 24

[19] J. Dahlin and P. Svenson, "A method for community detection in uncertain networks," in *Intelligence and Security Informatics Conference (EISIC), 2011 European*, IEEE, 2011, pp. 155–162. 5, 28

[20] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee, "Self-organization and identification of web communities," *Computer*, vol. 35, no. 3, pp. 66–70, 2002. 3

[21] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977. 2, 13

[22] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social networks*, vol. 1, no. 3, pp. 215–239, 1978. 2, 13

[23] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002. 3, 22, 23, 69

[24] F. Göbel and A. Jagers, "Random walks on graphs," *Stochastic processes and their applications*, vol. 2, no. 4, pp. 311–336, 1974. 2, 16, 19

[25] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2016, pp. 855–864. 79

[26] J. Huang, H. Sun, Y. Liu, Q. Song, and T. Weninger, "Towards online multiresolution community detection in large-scale networks," *PloS one*, vol. 6, no. 8, e23829, 2011. 27

[27] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," *Bulletin del la Société Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901. 2, 16, 17

[28] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953. 2, 16, 18

[29] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell system technical journal*, vol. 49, no. 2, pp. 291–307, 1970. 3, 22

[30] N. J. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, A. P. Tikuisis, *et al.*, "Global landscape of protein complexes in the yeast saccharomyces cerevisiae," *Nature*, vol. 440, no. 7084, p. 637, 2006. 1, 5, 28, 55

[31] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Physical Review E*, vol. 80, no. 1, p. 016 118, 2009. 69

[32] E. A. Leicht, P. Holme, and M. E. Newman, "Vertex similarity in networks," *Physical Review E*, vol. 73, no. 2, p. 026 120, 2006. 18

[33] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 2, 2007. 1

[34] L. Liu, R. Jin, C. Aggarwal, and Y. Shen, "Reliable clustering on uncertain graphs," 5, 29

[35] W. Liu and L. Lü, "Link prediction based on local random walk," *EPL (Europhysics Letters)*, vol. 89, no. 5, p. 58 007, 2010. 2, 16, 19, 57–59

[36] Z. Liu, Q.-M. Zhang, L. Lü, and T. Zhou, "Link prediction in complex networks: A local naïve bayes model," *EPL (Europhysics Letters)*, vol. 96, no. 4, p. 48 007, 2011. 2, 16, 20, 57–59

[37] L. Lü, C.-H. Jin, and T. Zhou, "Similarity index based on local paths for link prediction of complex networks," *Physical Review E*, vol. 80, no. 4, p. 046 122, 2009. 2, 16, 18

[38] L. Lü and T. Zhou, "Link prediction in weighted networks: The role of weak ties," *EPL (Europhysics Letters)*, vol. 89, no. 1, p. 18 001, 2010. 20, 57

[39] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011. 57

[40] F. Luo, J. Z. Wang, and E. Promislow, "Exploring local community structures in large networks," *Web Intelligence and Agent Systems: An International Journal*, vol. 6, no. 4, pp. 387–400, 2008. 26

[41] S. Mallek, I. Boukhris, Z. Elouedi, and E. Lefevre, "Evidential missing link prediction in uncertain social networks," in *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Springer, 2016, pp. 274–285. 5, 21

[42] T. Martin, B. Ball, and M. E. Newman, "Structural inference for uncertain networks," *Physical Review E*, vol. 93, no. 1, p. 012 306, 2016. 5, 29

[43] T. Murata and S. Moriyasu, "Link prediction of social networks based on weighted proximity measures," in *Proceedings of the IEEE/WIC/ACM international conference on web intelligence*, IEEE Computer Society, 2007, pp. 85–88. 5, 20

[44] M. A. Nascimento, J. Sander, and J. Pound, "Analysis of sigmod's co-authorship graph," *ACM Sigmod record*, vol. 32, no. 3, pp. 8–10, 2003. 1

[45] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical review E*, vol. 69, no. 6, p. 066 133, 2004. 3, 22, 24

[46] M. E. Newman, "Community detection and graph partitioning," *EPL (Europhysics Letters)*, vol. 103, no. 2, p. 28 003, 2013. 3, 22

[47] M. E. Newman, "Clustering and preferential attachment in growing networks," *Physical review E*, vol. 64, no. 2, p. 025 102, 2001. 2, 16, 44

[48] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026 113, 2004. 24

[49] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," Stanford InfoLab, Tech. Rep., 1999. 2, 14

[50] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2014, pp. 701–710. 79

[51] J. J. Pfeiffer III and J. Neville, "Methods to determine node centrality and clustering in graphs with uncertain structure.," in *ICWSM*, 2011. 34

[52] J. J. Pfeiffer and J. Neville, "Probabilistic paths and centrality in time," in *In Proceedings of the 4th SNA-KDD Workshop, KDD*, 2010. 4, 15, 32, 34, 55

[53] C. Pizzuti, "Ga-net: A genetic algorithm for community detection in social networks.," Springer. 3, 22

[54] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical review E*, vol. 76, no. 3, p. 036 106, 2007. 3, 22

[55] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási, "Hierarchical organization of modularity in metabolic networks," *Science*, vol. 297, no. 5586, pp. 1551–1555, 2002. 18

[56] G. Salton and M. J. McGill, "Introduction to modern information retrieval," 1986. 2, 16

[57] P. Sevon, L. Eronen, P. Hintsanen, K. Kulovesi, and H. Toivonen, "Link discovery in graphs derived from biological databases," in *International Workshop on Data Integration in the Life Sciences*, Springer, 2006, pp. 35–49. 4, 15, 32

[58] J. Shetty and J. Adibi, "The enron email dataset database schema and brief statistical report," 1, 55, 56

[59] T. Sørensen, "A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons," *Biol. Skr.*, vol. 5, pp. 1–34, 1948. 17

[60] F. Tan, Y. Xia, and B. Zhu, "Link prediction in complex networks: A mutual information perspective," *PloS one*, vol. 9, no. 9, e107056, 2014. 57

[61] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077. 79

[62] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*. Cambridge university press, 1994, vol. 8. 13

[63] R. J. Williams and N. D. Martinez, "Simple rules yield complex food webs," *Nature*, vol. 404, no. 6774, pp. 180–183, 2000. 1

[64] Y.-J. Wu, H. Huang, Z.-F. Hao, and F. Chen, "Local community detection using link similarity," *Journal of computer science and technology*, vol. 27, no. 6, p. 1261, 2012. 8, 27

[65] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of anthropological research*, vol. 33, no. 4, pp. 452–473, 1977. 67

[66] T. Zhou, L. Lü, and Y.-C. Zhang, "Predicting missing links via local information," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 71, no. 4, pp. 623–630, 2009. 2, 16, 18, 44

[67] B. Zhu and Y. Xia, "Link prediction in weighted networks: A weighted mutual information model," *PloS one*, vol. 11, no. 2, e0148265, 2016. 57