Neighbor-based Link Prediction with Edge Uncertainty

Chi Zhang and Osmar R. Zaïane

Department of Computing Science, University of Alberta
Edmonton, AB, Canada
{chi7, zaiane}@ualberta.ca

Abstract. In this work, we are concerned with uncertain networks and focus on the problem of link prediction with *edge uncertainty*. Networks with edge uncertainty are networks where connections between nodes are observed with some probability. We propose the uncertain version of the popular neighbors-based metrics for link prediction. The metrics are developed by considering all possible worlds generated by the uncertain network. We state that by taking all possible worlds of the uncertain network into account, the performance of link prediction can be improved. Since uncertain edges result in a very large number of possible worlds, we propose an efficient divide and conquer algorithm to reduce time complexity and calculate these metrics. Finally, we evaluate our metrics using existing ground truth to show the effectiveness of our proposed approach against other popular link prediction methods.

Keywords: social network analysis, link prediction, uncertain networks

1 Introduction

Link prediction is the problem of determining future or missing associations between entities in networks based on observed links. Because of its broad applications in different domains, link prediction has attracted increasing attention.

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. Among all approaches, neighbor-based metrics [1–5] are effective and the simplest way to predict missing links. The other metrics include path-based metrics [6], random-walk-based metrics [7]. Furthermore, there are some learning-based methods [8] and embedding-based methods [9] that have been proposed in recent years.

Most previous studies on link prediction have focused on networks where the structure is exactly known. With the increasing number of applications in which the edges are constructed in the network through uncertain or statistical inference, the problem of link prediction with edge uncertainty has become increasingly important. 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. However, only few studies take probabilities into consideration. Ahmed et al. [10] proposed the uncertain version of the random walk method for link prediction with edge uncertainty. Mallek et al. [11] put forward an approach combined sampling techniques and information fusion and obtained good results in real-life settings. Up to now, the uncertain version of the popular neighbor-based metrics have not been studied. Murata and Moriyasu [12] proposed weighted similarity indices, including variants of some popular 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 4.

The uncertain scenario will make the problem of link prediction become more complex, and the uncertain version of the most basic neighbor-based methods are not yet studied. Therefore, in this work, we mainly focus on using neighbor-based algorithms to solve the problem of link prediction in the context of uncertain networks. We propose the uncertain version of the popular neighbors-based metrics and efficient algorithms to calculate them. The remainder of this paper is organized as follows. In Section 2, we provide the problem definition. In Section 3, we review related work. In Section 4, we show the limitation of previous work, propose the uncertain version of common-neighbors-based metrics and efficient algorithms to produce them. In Section 5, we present the experiment results and our evaluation metric. Finally, we conclude in Section 6.

2 Problem Definition

2.1 Uncertain Network

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.

2.2 Link Prediction Problem Definition

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 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. More experiment details can be found in Section 5.

3 Previous Work

As mentioned above, neighbor-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 [1]. The Resource Allocation (RA) metric [5] is regarded as one of the best neighbor-based metrics because of its performance. Therefore, in this paper, we concentrate on CN and RA indexes, whose definitions are as follows.

Common Neighbors (CN): 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 $\mathcal{V}_x$. The simplest measure of the neighborhood overlap is the directed count:

$$s_{xy} = |\Gamma(x) \cap \Gamma(y)| \tag{1}$$

Resource Allocation (RA): 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:

$$s_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{k(z)} \tag{2}$$

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

The above-mentioned similarity metrics, CN and RA, 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 [12] proposed weighted similarity metrics as variants of Common Neighbors and Resource Allocation:

$$\text{Weighted Common Neighbors: } s_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} w(x,z) + w(y,z) \tag{3}$$

$$\text{Weighted Resource Allocation: } s_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{w(x,z) + w(y,z)}{s(z)} \tag{4}$$

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 $\mathcal{V}_x$.

Besides the aforementioned metrics, we will also use the Local Naïve Bayes model [8] and the Local Random Walk metric [7] to assess our metrics as they are popular. The Local Naïve Bayes model is considered as the state-of-the-art in neighbor-based link prediction algorithms. Due to limited space, we will not cover them in details here.

4 Link Prediction for Uncertain Graphs

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, there exists some problems. Figure 1 is an example.

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 (3) for Weighted Common Neighbors.

$$s_{AB} = 0.2 + 0.9 = 1.1 \tag{5}$$

$$s_{DE} = 0.5 + 0.5 = 1.0 < 1.1 \tag{6}$$

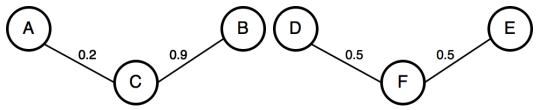

Fig. 1. An example showing the problem when considering the probability as a weight.

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 2: both links may exist, both may be absent, or either one is present.

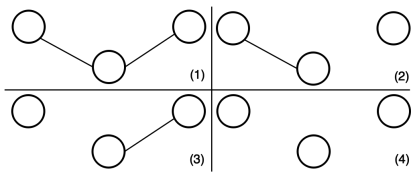

Fig. 2. Possible worlds for two uncertain links between three nodes

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 2(1), the probability for this case is $0.2 \times 0.9 = 0.18$ (we assume that the existence of edges are independent with each other). In this

case, $s_{AB} = 1$ based on Equation (1). If node $\mathcal{V}_C$ is not the common neighbor of nodes $\mathcal{V}_A$ and $\mathcal{V}_B$, as Figure 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:

$$Pr(G) = \prod_{e \in E_G} \mathcal{P}(e) \prod_{e \in \mathcal{E}, e \notin E_G} (1 - \mathcal{P}(e)) \tag{7}$$

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.

$$\text{Uncertain Common Neighbors: } s_{xy} = \sum_{G \in \mathcal{G}} (Pr(G) \times |\Gamma_G(x) \cap \Gamma_G(y)|) \tag{8}$$

$$\text{Uncertain Resource Allocation: } s_{xy} = \sum_{G \in \mathcal{G}} (Pr(G) \sum_{z \in \Gamma_G(x) \cap \Gamma_G(y)} \frac{1}{k_G(z)}) \tag{9}$$

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.

4.1 Time Complexity Analysis for the Calculation of Common Neighbors in Uncertain Networks

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 (8) 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 (8) can also be represented as:

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

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.

4.2 Time Complexity Analysis for the Calculation of Resource Allocation in Uncertain Networks

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 (9) is $O(2^{|\mathcal{E}|}k)$.

As mentioned in Section 4.1, 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 $t = |\Gamma_G(x) \cap \Gamma_G(y)|$.

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

$\mathcal{G}_z$ 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 $\mathcal{G}_z$.

4.3 An Efficient Algorithm for the Calculation of Resource Allocation

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 (7) 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:

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

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.

$$P^n_{1\to m} = \sum_{i=max(0,n-\lceil m/2\rceil)}^{min(n,\lfloor m/2\rfloor)} P^i_{1\to\lfloor m/2\rfloor} P^{n-i}_{\lfloor m/2\rfloor+1\to m} \tag{10}$$

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 1.

Algorithm 1: kEdgeProbability

```
Data: Probability List uncertainEdgeList
Result: The probability list probList of existing n among m edges,
        n ∈ [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;
        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
                    probList[n] ← probList[n] + left[k] × right[n − k];
                end
            end
        end
        return probList;
    end
```

Based on the description of Algorithm 1, we can find the time complexity of Algorithm 1 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, 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, 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 1 to reduce the time complexity. In Algorithm 1, 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 (10) has a more general form, which can be represented as follow:

$$P^n_{1\to m} = \sum_{i=max(0,n+k-m)}^{min(n,k)} P^i_{1\to k} P^{n-i}_{k+1\to m} \tag{11}$$

In Equation (10), we choose $k = \lfloor m/2 \rfloor$.

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 1. 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 (11). The full equations can be represented as follows:

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

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 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. After we calculate nodes $\mathcal{V}_x$ and $\mathcal{V}_y$'s each common neighbor's contribution for s_{xy}, we can calculate s_{xy} easily.

5 Experiments

5.1 Datasets

Protein-Protein Interaction Network: We used the protein-protein interaction network (PPI) created by Krogan [13]. 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: 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 [14] to assign each edge with a possibility of occurrence.

Synthetic Uncertain Network Based on Deterministic Network: Considering that there are not many publicly available uncertain network datasets

on the web, we also generated an uncertain network based on deterministic networks. The dataset we used here is USAir. The US air transportation network contains 332 airports and 2126 airlines. Based on this network, we use an uncertain network generator to generate its corresponding uncertain network. The uncertain network generator used here is adopted from [15]. The percentage of non-existential edges we choose to add in this experiment is 20%.

5.2 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 of 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. 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 [14]. 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 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.

5.3 Results and Evaluation

As the literature suggested [8], 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. To evaluate our metrics, we mainly focus on the comparison between the uncertain version of graph proximity measures with weighted and unweighted ones. We also compare our metrics with LNB and

SRW. 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 [7]). 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 1.

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	Ignore probabilities and run local random walk algorithm [7], choose $t = 2$
SRW3	Ignore probabilities and run local random walk algorithm [7], choose $t = 3$
LNB-CN	Ignore probabilities and use Local Naïve Bayes form of Common Neighbors [8]
LNB-RA	Ignore probabilities and use Local Naïve Bayes form of Resource Allocation [8]

Table 1. Comparative Results for Different Algorithms

Datasets	Common Neighbor			Resource Allocation			SRW2	SRW3	LNB-CN	LNB-RA
	CN	WCN	UCN	RA	WRA	URA				
PPI	0.472	0.5045	**0.5288**	0.4123	0.45	**0.5728**	0.4136	0.5284	0.4856	0.4992
Enron	0.49	0.52	**0.61**	0.51	0.47	**0.52**	0.43	0.45	0.55	0.46
Synthetic Network	0.5812	0.5954	**0.6043**	0.6075	0.6124	**0.6233**	0.5852	0.5992	0.5962	0.5885

From Table 1, 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.

From Table 1, we can also observe that our metrics (UCN and URA) can outperform the other four baseline methods on PPI and Synthetic datasets. 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.

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.

6 Conclusion

In this paper, 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. In this work, we only focus on the neighbor-based algorithms because they are simple, effective but not yet have been studied, and 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, learning-based metrics and embedding-based algorithms, 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.

References

1. M. E. Newman, "Clustering and preferential attachment in growing networks," *Physical review E*, vol. 64, no. 2, p. 025102, 2001.
2. G. Salton and M. J. McGill, "Introduction to modern information retrieval," 1986.
3. P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 547–579, 1901.
4. L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social networks*, vol. 25, no. 3, pp. 211–230, 2003.
5. 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.
6. 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. 046122, 2009.
7. W. Liu and L. Lü, "Link prediction based on local random walk," *EPL (Europhysics Letters)*, vol. 89, no. 5, p. 58007, 2010.
8. 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. 48007, 2011.
9. 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*, pp. 701–710, ACM, 2014.
10. 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.
11. 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*, pp. 274–285, Springer, 2016.
12. 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*, pp. 85–88, IEEE Computer Society, 2007.
13. 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.
14. J. J. Pfeiffer and J. Neville, "Probabilistic paths and centrality in time," in *In Proceedings of the 4th SNA-KDD Workshop, KDD*, 2010.
15. C. Zhang and O. R. Zaïane, "Detecting local communities in networks with edge uncertainty," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 9–16, IEEE, 2018.