Detecting Local Communities in Networks with Edge Uncertainty

Chi Zhang
Department of Computing Science
University of Alberta
Edmonton, Canada
chi7@ualberta.ca

Osmar R. Zaïane
Department of Computing Science
University of Alberta
Edmonton, Canada
zaiane@ualberta.ca

***Abstract*—In this work, we focus on the problem of local community detection with *edge uncertainty*. We use an estimator to cope with the intrinsic uncertainty of the problem. 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 address this problem. Due to the very limited publicly available uncertain network datasets, we also put forward a way to generate uncertain networks. Finally, we evaluate our algorithm using existing ground truth as well as based on common metrics to show the effectiveness of our proposed approach.**

***Index Terms*—uncertain network, local community detection, social network analysis**

I. Introduction

Many datasets can be represented by networks consisting of a set of nodes and edges connecting these nodes. Examples include protein-protein interaction networks [1], food webs [2], social networks [3], air transportation networks, collaboration networks [4], [5] and the worldwide web (WWW) [6], [7]. 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 [8]; the detection of groups within social networks can also be used to find social units or communities [9]. Besides these applications, community detection can also be used in analyzing trends in citation networks [10] and improving recommender systems [11]. Because of its broad applications in different domains, community detection has attracted increasing attention from computer scientists, biologists and physicists recently.

In the past, a large number of community detection algorithms have been proposed for deterministic graphs, where the existence of edges is known and certain. According to the characteristics of these algorithms, they can be divided into graph partitioning-based algorithms [12], [13], clustering-based algorithms [9], [14]–[16], genetic algorithms-based algorithms [17] and label propagation-based algorithms [18].

Most previous approaches on community detection have focused on networks where the structure is deterministically known. Recently, we have an increasing number of networks which have edge uncertainty, which means the network structure is not exactly and deterministically known. The edges are constructed through uncertain or statistical inference, so we only know the connections between nodes with a certain probability. 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. To deal with uncertain networks, Krogan and his colleagues converted the uncertain network into a conventional binary network by thresholding the likelihoods [1]. Dahlin and Svenson 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 [19]. Liu et al. developed a novel k-means algorithm to solve the uncertain clustering problem [20]. Martin, Ball and Newman gave a principled maximum-likelihood method for inferring community structure [21]. Kollios et al. [22] and Ceccarello et al. [23] proposed approximation algorithms to cluster uncertain graphs. However, all these approaches require knowledge of the entire graph structure. 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 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/her social network on Facebook. Though several different local modularity metrics [24]–[26] have been proposed to identify local community structure given limited information in deterministic networks. The problem of local community detection with edge uncertainty is not yet solved.

In this work, we mainly focus on the problem of local community detection in the context of uncertain networks. We propose a way to convert the uncertain community detection

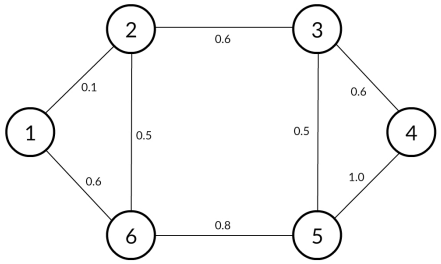

Fig. 1. An example of uncertain network. The numbers are existential probabilities.

problem into the deterministic scenario, and put forward a new measure to tackle the problem existing in uncertain networks. The remainder of this paper is organized as follows. In Section II, we provide the problem definition. In Section III, we review related works. In Section IV, we show the limitation of previous works and develop a new algorithm to find local communities in uncertain networks. In Section V, we present our evaluation metrics and the experiment results. We also demonstrate how we find the best hyper-parameter in this section. Finally, we conclude the results in Section VI.

II. Problem Definition

A. 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. Figure 1 is an example of an uncertain network.

B. Local Community Detection

As mentioned in the introduction, 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 [24], [25], [27]. Here, we reiterate them in the context of uncertain networks.

Suppose that in an undirected 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 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 2 shows node sets $\mathcal{D}$, $\mathcal{S}$, $\mathcal{C}$ and $\mathcal{B}$ in a network.

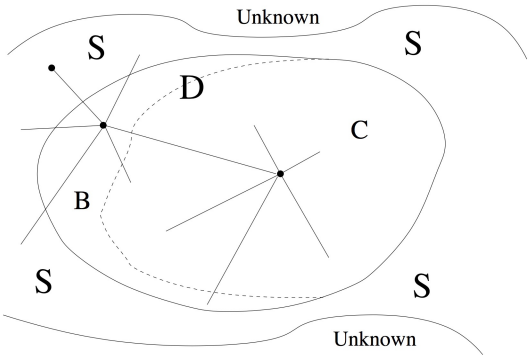

Fig. 2. Local Community Definition

III. Previous Work

A. Local Modularity $\mathcal{R}$ in Deterministic Networks

In deterministic networks, Clauset 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 [24].

$$\mathcal{R} = \frac{\mathcal{B}_{in\_edge}}{\mathcal{B}_{out\_edge} + \mathcal{B}_{in\_edge}} \tag{1}$$

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 $\mathcal{B}$.

B. Local Community Detection Algorithm for Deterministic Networks

To find a local community for a start node in deterministic networks, Chen and his colleagues proposed a local community identification algorithm based on the local modularity $\mathcal{R}$ [25]. 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 $\mathcal{R}$.

IV. Local Community Detection Algorithm for Uncertain Networks

A. Local Modularity $\mathcal{UR}$ in Uncertain Networks

Inspired by the local modularity $\mathcal{R}$ in deterministic networks, 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:

$$\mathcal{UR} = \frac{\mathbb{E}(\mathcal{B}_{in\_edge})}{\mathbb{E}(\mathcal{B}_{in\_edge}) + \mathbb{E}(\mathcal{B}_{out\_edge})} \tag{2}$$

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:

$$\mathbb{E}(\mathcal{B}_{in\_edge}) = \frac{1}{2} \sum_{\mathcal{V}_i \in \mathcal{B}, \mathcal{V}_j \in \mathcal{B}, i \neq j} \mathcal{P}_{i,j} + \sum_{\mathcal{V}_i \in \mathcal{B}, \mathcal{V}_j \in \mathcal{C}} \mathcal{P}_{i,j} \tag{3}$$

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:

$$\mathbb{E}(\mathcal{B}_{out\_edge}) = \sum_{\mathcal{V}_i \in \mathcal{B}, \mathcal{V}_j \in \mathcal{S}} \mathcal{P}_{i,j} \tag{4}$$

After replacing $\mathcal{R}$ with $\mathcal{UR}$, we can use the algorithm mentioned in [25] to find the local community for the input node.

B. Reviews of the Previous Method

However, simply using $\mathcal{UR}$ to replace $\mathcal{R}$ and applying the original local community detection algorithm (as mentioned in Section III-B) 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 3 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 [25] 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 $\mathcal{UR}$ to take the place of $\mathcal{R}$. $\mathcal{V}_1$ to $\mathcal{V}_5$ and node $\mathcal{V}_7$ are all neighbors of $\mathcal{V}_6$, so they are all candidates.

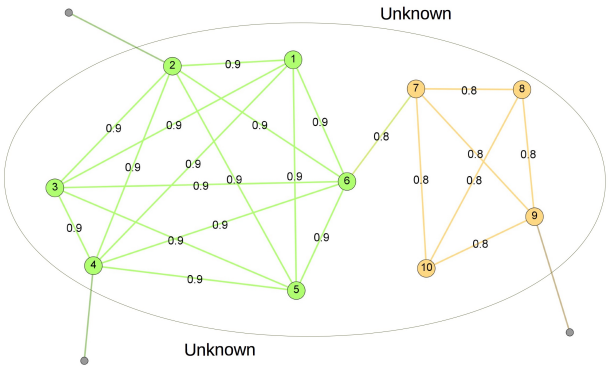

Fig. 3. An example showing the problem of only using $\mathcal{UR}$ to find the local community.

Initially, $\mathcal{UR} = 0$. In the first step, $\mathcal{V}_6$ is added to $\mathcal{D}$ and $\mathcal{B}$; $\mathcal{V}_1$ to $\mathcal{V}_5$ and $\mathcal{V}_7$ are added to $\mathcal{S}$. The $\mathcal{UR}$ of the community, after adding $\mathcal{V}_1, \mathcal{V}_3$ or $\mathcal{V}_5$, are all

$$\frac{0.9}{4 \times 0.9 + (4 \times 0.9 + 0.8) + 0.9} = 0.1011 \tag{5}$$

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

$$\frac{0.8}{5 \times 0.9 + 3 \times 0.8 + 0.8} = 0.1039 \tag{6}$$

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 [26], but they regarded these start nodes as periphery nodes and did not provide local communities for them. 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, based on our experiments, we find 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.

C. 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 [28] propose a 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 $\mathcal{S}$.

$$\mathcal{K}_i = \mathbb{E}(N_{i,in\_edge}) + \mathbb{E}(N_{i,shell\_edge}) \tag{7}$$

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:

$$\mathbb{E}(N_{i,in\_edge}) = \sum_{\mathcal{V}_j \in \mathcal{D}} \mathcal{P}_{i,j} \tag{8}$$

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:

$$\mathbb{E}(N_{i,shell\_edge}) = \sum_{\mathcal{V}_j \in \mathcal{S}, i \neq j} \mathcal{P}_{i,j} \cdot \mathcal{P}_{j,shell} \tag{9}$$

$$\mathcal{P}_{j,shell} = 1 - \prod_{\mathcal{V}_m \in \mathcal{D}} (1 - \mathcal{P}_{j,m}) \tag{10}$$

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 3, 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:

$$\mathcal{K}_{i=1,2,3,4,5} = 0.9 + 0.9 \times 0.9 \times 4 = 4.14 \tag{11}$$

$$\mathcal{K}_{i=7} = 0.8 \tag{12}$$

$\mathcal{K}_{i=7}$ is smaller than $\mathcal{K}_{i=1,2,3,4,5}$, so we will first exclude node $\mathcal{V}_7$. As mentioned in equation (5), 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 $\mathcal{V}_5$.

D. Full Algorithm Description

Similar to the algorithm mentioned in Section III-B, our algorithm also firstly places 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 V-C. The full algorithm description can be found in Algorithm 1.

V. Experiments

A. 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 [29] describes friendships between members of a karate club at a U.S. university in 1977. The core network consists of 34 nodes and 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 [9] 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 paper is adopted from [30]. 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

Algorithm 1: 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 (10);
      break for loop
    end
  end
until *no new node is added to* $\mathcal{D}$;
return $\mathcal{D}$

TABLE I
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

different input parameters, shown in the Table I [1], together with the values used in this paper. These parameters allow for the generation of families of networks with desired properties.

[1]The parameters shown in Table I are randomly chosen. We also generate other synthetic networks based on other parameters and get similar results in the experiments. To avoid redundancy, we only show the results when executed using the parameters in Table I.

Generating Uncertain Networks

Since most uncertain graphs are more often than not corporate and government assets and sensitive information, they are rarely disclosed to the public. Since there are not many publicly available uncertain network datasets, we proposed a way to generate uncertain networks based on deterministic networks. It 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 by the Algorithm 2.

Algorithm 2: 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 $\mathcal{G}.edges$;
  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* $\mathcal{G}$

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.

B. 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 II. We mainly focus on the comparison between our algorithm and the other two local community detection algorithms (R and UR). Though Louvain algorithm [15] is not a local community detection algorithm, we also

TABLE II
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 (2)
UR+K	our algorithm as mentioned in Algorithm1
Louvain	Louvain algorithm
ULouvain	regard probability as weight and run weighted version of the Louvain algorithm

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 V-C.

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.

Supervised Evaluation

One way to compare community detection results is supervised evaluation. We use a similar evaluation method as mentioned in [26]. 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:

$$\text{precision} = \frac{\sum_{\mathcal{V}_i \in \mathcal{G}} |\mathcal{D}_i \cap \mathcal{D}_{i'}|}{\sum_{\mathcal{V}_i \in \mathcal{G}} |\mathcal{D}_i|} \tag{13}$$

$$\text{recall} = \frac{\sum_{\mathcal{V}_i \in \mathcal{G}} |\mathcal{D}_i \cap \mathcal{D}_{i'}|}{\sum_{\mathcal{V}_i \in \mathcal{G}} |\mathcal{D}_{i'}|} \tag{14}$$

$$\text{F1-measure} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} \tag{15}$$

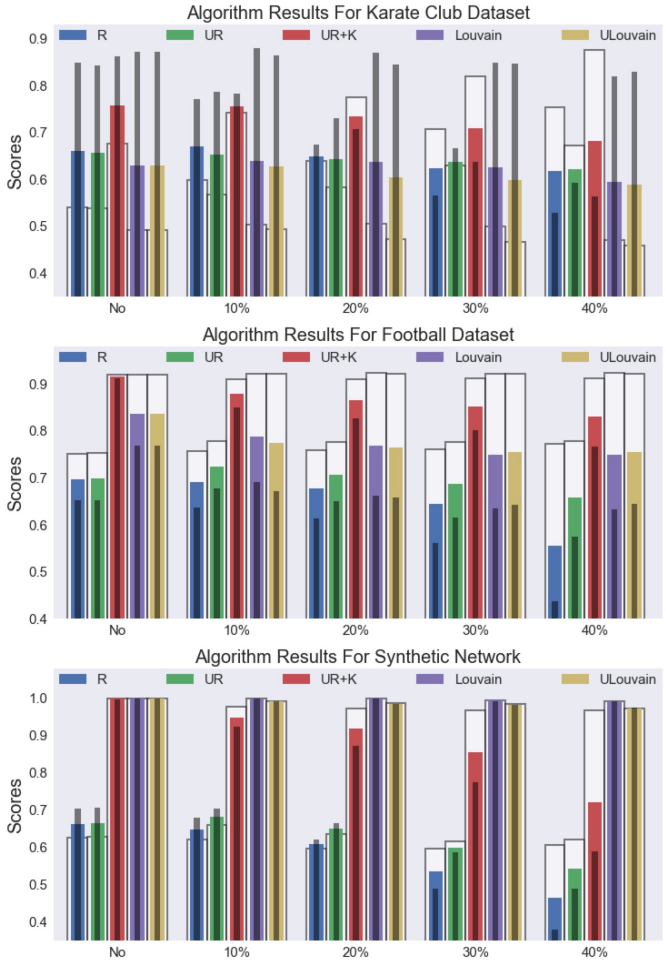

Fig. 4. Algorithm results on karate club, football and synthetic networks. For each network, we add 10%, 20%, 30% and 40% non-existential edges to the original network. The original network is also regarded as a special case of uncertain network. For each uncertain network, we compare our algorithm (UR+K) 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 precision and recall values are represented by its inner black bar and external white bar respectively.

The results on three networks are shown in Figure 4.

From Figure 4, 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 III-A, 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, 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 III, IV, V.

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

C. Hyper-Parameter Evaluation

Our algorithm has a hyper-parameter λ. As we mentioned in Section IV-D, 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 5.

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

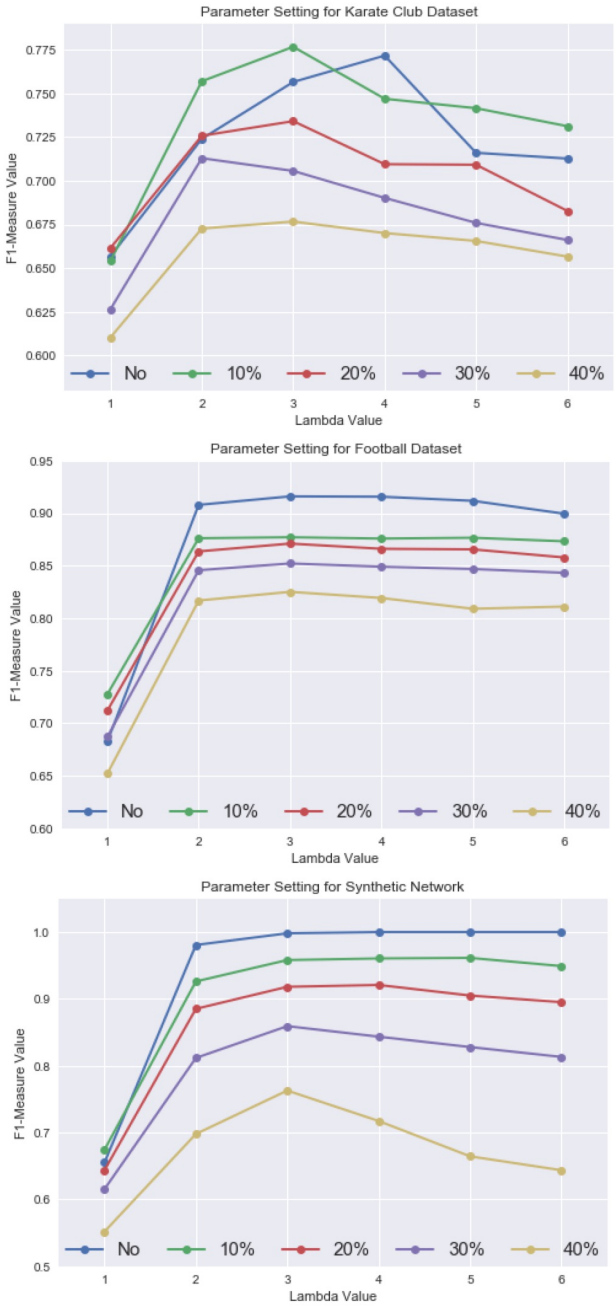

Fig. 5. Hyper-parameter Validation. For each network, we add 10%, 20%, 30% and 40% non-existential edges to the original network. The original network is also regarded as a special case of uncertain network.

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.

VI. Conclusion

In this work, 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 IV-B, and our measure $\mathcal{K}$ may then be considered.

References

[1] 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, pp. 637–643, 2006.

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

[3] J. Shetty and J. Adibi, "The enron email dataset database schema and brief statistical report,"

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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*, pp. 1–8, ACM, 2015.

[12] 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.

[13] M. E. Newman, "Community detection and graph partitioning," *EPL (Europhysics Letters)*, vol. 103, no. 2, p. 28003, 2013.

[14] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical review E*, vol. 69, no. 6, p. 066133, 2004.

[15] 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, p. P10008, 2008.

[16] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.

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

[18] 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. 036106, 2007.

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

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

[21] T. Martin, B. Ball, and M. E. Newman, "Structural inference for uncertain networks," *Physical Review E*, vol. 93, no. 1, p. 012306, 2016.

[22] G. Kollios, M. Potamias, and E. Terzi, "Clustering large probabilistic graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 2, pp. 325–336, 2013.

[23] M. Ceccarello, C. Fantozzi, A. Pietracaprina, G. Pucci, and F. Vandin, "Clustering uncertain graphs," *Proceedings of the VLDB Endowment*, vol. 11, no. 4, pp. 472–484, 2017.

[24] A. Clauset, "Finding local community structure in networks," *Physical review E*, vol. 72, no. 2, p. 026132, 2005.

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

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

[27] 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.

[28] M. E. Newman, "Clustering and preferential attachment in growing networks," *Physical review E*, vol. 64, no. 2, p. 025102, 2001.

[29] 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.

[30] 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. 016118, 2009.