# Community Detection and Discovery in Deterministic and Uncertain Networks

by

Mohammadmahdi Zafarmand

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 structures in different areas of science can be modeled with graphs containing nodes and edges, which represent the entities of the model and the relationship between them, respectively. Community detection and discovery are two important tasks in Social Network Analysis, which try to find groups of nodes within a graph such that they are densely connected inside and loosely connected to the rest of the network. A community detection algorithm finds all such characterized structures altogether. In contrast, a community discovery method takes an individual node in the network and finds all other network nodes that belong to the same group as the given node.

Before, networks were graphs with deterministic nodes and edges, which we were sure of whether they exist or not. Recently uncertainty in collected information makes us model the problems with probabilities (different types of uncertainty may exist in a network, but in this document, we focus only on the probability of edges). Furthermore, networks can have nodes that belong to more than one community. In such cases, we call them *Networks with Overlapping Communities*. In this document, we try to answer two essential questions of social network analysis, community detection and discovery, in social networks in the presence of deterministic and uncertain edges.

A part of this work is assigned to evaluate and improve on existing methods in order to make it possible to use them on networks that have probabilistic edges or have overlapping communities.

# Preface

A part of this thesis was published under the title *"Addressing the Resolution Limit and the Field of View Limit in Community Mining"* at Symposium on Intelligent Data Analysis (IDA) 2020 [1]. We are going to submit two other parts of this thesis for conferences and journals in the future. The first part contains our new proposed global and local approaches called *RSIWO* and *Local SIWO*. The second part has our new algorithm for networks with uncertainty called *USIWO* and the new uncertain network generator.

# Acknowledgements

I would like to thank my supervisors, Osmar Zaïane and Christine Largeron, for their dedicated support and guidance. They continuously gave me invaluable insights and were always willing and enthusiastic to assist in any way they could throughout my research. This work would not have been possible without their encouragement. I appreciate the opportunity to work with them, which was also an excellent chance for me to develop in research and character.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation and Background

Nowadays, we use complex networks to model entities of a group or an organization and their relationships in different domains, from sociology to biology or even computing science. In their purest form, these networks can be represented as graphs which consist of a finite number of nodes and edges that connect them. The nodes and edges of the network are not randomly distributed inside the graph and may have different attributes. There are specific characteristics of complex networks (or social networks) that separate them from random networks such as the power-law distribution [2], a high clustering coefficient [3], and the presence of community structure [4].

The distributions of a wide variety of physical and biological phenomena approximately follow a power-law over a wide range of magnitudes; these include the foraging pattern of various species [5], the sizes of activity patterns of neuronal populations [6], and the frequencies of words in most languages. In complex networks, node degrees often follow a power-law distribution, such that many nodes have low degrees, whereas few have high degree values. To demonstrate this feature in complex real-world networks, we compare the distribution of degrees of Email network [7] (which is a popular network in the field of social network analysis) against the power-law distribution in Fig 1.1 to show their similarity.

The clustering coefficient is a measure that shows which nodes in a graph tend to cluster together. There are two different versions of this measure:

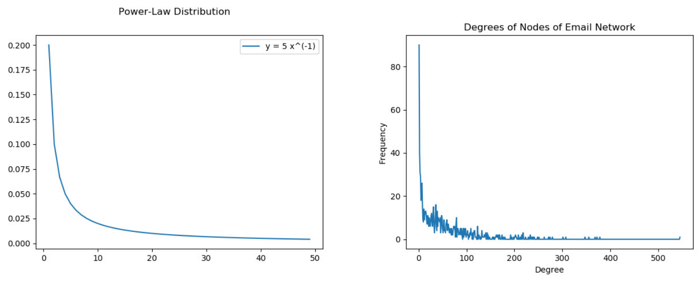


Figure 1.1: Power-Law Distribution and the Distribution of Email Network

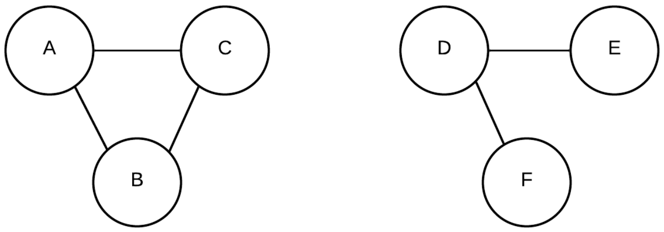


Figure 1.2: A Closed (On the Left) and an Open Triplet of Nodes (On the Right)

the global and the local. The global clustering coefficient is based on triplets of nodes. As Figure 1.2 shows, a triplet is a set of three nodes that are connected by either two (open triplet) or three (closed triplet) undirected ties [8]. Even though the average clustering coefficient (the average value of the clustering coefficient for all nodes in the network) can be used as a measure for the network, there is a direct global clustering coefficient as defined in the following:

$$CC = \frac{\text{number of closed triplets}}{\text{number of all triplets (open and closed)}} \tag{1.1}$$

The node's local clustering coefficient in a graph expresses how close its neighbors are to forming a clique (complete sub-graph). For the undirected graphs, given $G = (V, E)$, where $V$ is the set of nodes and $E$ is the set of edges between them, the local clustering coefficient of a node that has $n_i$ direct neighbors precisely, can be defined as:

$$CC_i = \frac{2|\{e_{j,k} : e_{i,j}, e_{i,k}, e_{j,k} \in E\}|}{n_i(n_i - 1)} \tag{1.2}$$

A clique or even a group of nodes with a high $CC$ value can be considered as a community but this definition is too strict in practice. Moreover, it is time-consuming to evaluate the $CC$ for all groups of nodes that can be built in a graph. Thus, other definitions for communities have been proposed. For example, a community is often defined as a group of nodes in a network that is densely connected, but their connection to the rest of the network is loose [9]. We are usually interested in finding such structures in the network. This definition also brings the significant benefit of discovering the nodes which share some characteristics. Also, by detecting them and splitting the whole network into smaller parts, any investigation could be done more efficiently. Such studies and analyses can be done in applications including but not limited to public health, recommendation systems, clustering users of a large service, link prediction, etc.

Many different methods have been proposed to find community structure in a complex network. The expected result of these methods over a graph $G = (V, E)$ in which $V$ is the set of nodes and $E$ is the set of edges, is usually a partition $P = \{c_1, c_2, ..., c_r\}$ in which $c_i$ is a non-empty set of nodes of $V$ that are highly connected and is called a community where $\cup_{i=0}^{r} c_i = V$. If the detected communities are supposed to be disjoint, there should be $c_i \cap c_j = \emptyset$ for every $c_i, c_j \in P$, if $i \neq j$, however, for communities with overlap, this condition is not necessarily established for each pair of communities.

In this study, we consider three aspects of community detection algorithms: global or local, disjoint or overlapping, and definite or uncertain graph. These three aspects can be noted as:

- If the focus is on the availability of network information, one can divide the community detection methods into **global** approaches (where all network information is accessible) and **local** approaches (where only a part of the network is required).

- In a network, nodes may be limited to belonging to only one community or multiple communities. Networks of the first kind consist of **disjoint** communities, whereas the second type of networks involves **overlapping**

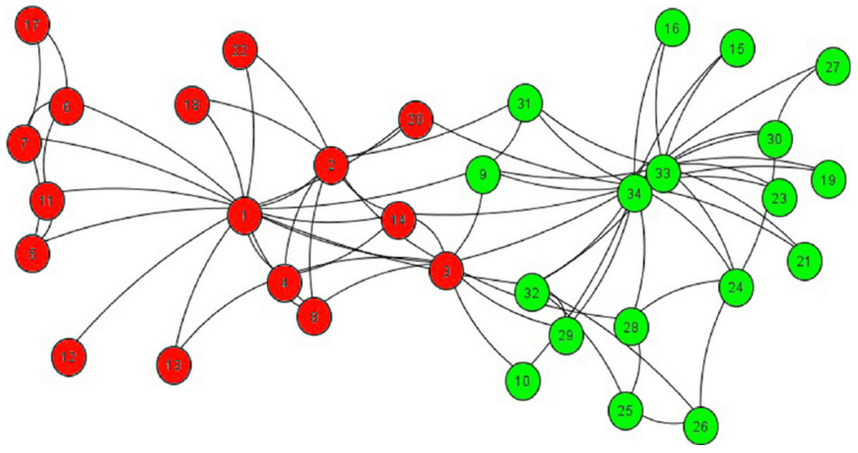


Figure 1.3: Zachary's Karate Club as an Undirected Unweighted Network [10]

communities.

- If the structure of a graph is deterministically known, in that case, the network is considered a **deterministic** network. Thus, the community detection algorithm used to reveal its community structure is regarded as a deterministic method. Otherwise, if the network structure involves any kind of indeterminacy, it would be analyzed using **uncertain** community detection algorithms.

## 1.2 Problem Definition and Challenges

Given a graph of $G = (V, E)$ where $V$ is the set of nodes, and $E$ is the set of edges. The edge between node $v_i$ and $v_j$ is denoted by $e_{i,j}$ or $e_{j,i}$ as the networks that are investigated in this study are undirected. The edges can also be weighted where their weight is denoted by $w_{i,j}$; weights are usually positive real values. Although negative weights are possible, they are not being investigated in this study. Edges can also be uncertain, where we represent the probability of the existence of edge $e_{i,j}$ with $p_{i,j}$. No nodes can have self-loops, and multiple edges between any two nodes of the graph are not allowed. Figure 1.3 shows the Zachary's Karate Club network as a simple example of deterministic unweighted networks.

### 1.2.1 Global Community Detection Algorithms

Many studies have been done concerning the community detection problem on social networks, most of which are global approaches that presume the full knowledge and access to the corresponding graph. The goal of such issues is to find the community structure of the whole network if it exists. Many algorithms have been suggested, classified into graph partitioning, hierarchical clustering, and modularity-based algorithms [11]. Among them, the focus is on modularity-based algorithms which have been proposed by Newman and Girvan [12] for the first time, because it covers many important aspects and points including the definition of community, the choice of a null model, the expression of the "strength" of communities, etc. [11]. However, they have plenty of limitations, including but not limited to the resolution limit, guarantee of communities to be connected, the computational complexity, etc.

### 1.2.2 Local Community Detection Algorithms

As the amount of data gathered for a problem grows, the size of the corresponding network increases as well. Because of the memory limitation, it is sometimes not feasible to read and load the whole network to analyze it. On the other hand, there exist problems in which we are interested in specific nodes (or their neighborhood) in the network, so we do not need to have all of the information regarding the entire network. In such cases, we prefer using local community detection approaches, which can serve with much fewer data and are often faster.

The local community detection algorithms usually have a one-node-at-one-step discovery process. This means that one node is added to the discovered community at each step, and the neighborhood of the detected part of the network expands; thus, the algorithm can explore more. The expected result consists of nodes that are highly connected to each other, even constructing maximal cliques. We can usually perform a local algorithm multiple times, starting with different initial nodes to cover the entire network and reveal all communities inside it, if needed.

With such advantages, these methods suffer from a low performance where they cannot grasp a community with only the nodes that should be inside it. Another problem emerges at the beginning of the community's expansion in the first steps when only a limited knowledge of the network is available. Since the early steps determines how the discovered community expands, lack of information of the known parts of the network may mislead the true path of community expansion.

### 1.2.3 Networks with Uncertainty

Most previous studies on social networks have been done in the context of deterministic graphs, which assumes a precise knowledge about the network structure. However, due to many causes, including but not limited to the data-collection process and the machine-learning techniques utilized to preprocess it, it is not always the case. Different kinds of uncertainty could exist in a network. These include uncertainty related to the existence of nodes, uncertainty in the attributes of the nodes, uncertainty with regards to the presence of edges, and uncertainty with respect to the characteristics of the edges.

In this study, the focus is on the third type of the mentioned uncertainties, which is highly applicable (i. e. in criminology analysis) and the uncertainty is encoded in the edges; e.g., a node in a social network has a connection and influences another node with some probability.

A straightforward approach to analyze probabilistic graphs, which is broadly used, is to heuristically cast the edges' probabilities into weights and apply existing methodologies on this weighted graph. This approach is problematic; not only there is no meaningful way to perform such a casting, but also there is no principled way to encode normal weights on the edges.

### 1.2.4 Networks with Overlapping Communities

The traditional node clustering methods have inherent drawbacks to discover overlapping communities where nodes can participate in the formation of more than one community. In reality, an entity can effectively belong to several groups; for example, a university student can be a collaborator of a research lab, a basketball player of their department team, and have their group of friends, where different individuals are involved in each group. It is also possible that a node's strength or membership value in different communities varies. Node clustering is inadequate to capture the pervasive overlaps, so overlapping community detection is still a difficult challenge that needs to be addressed.

### 1.2.5 Evaluation Methods

Community detection is a non-supervised task. How to evaluate the quality of detected communities is still a challenging problem that needs to be addressed. In this study, we review and assess some evaluation metrics such as internal measures, external measures, and relative measures and indicate how we can use them in the context of different community detection problems (global vs. local or disjoint vs. overlapping).

## 1.3 Thesis Statements

This thesis is composed of four main subjects: global community detection methods to find disjoint communities, local community detection and discovery methods to find the community of a given node (and continuing to cover all nodes in the network), detecting communities in uncertain networks, and finally, investigating evaluation metrics that usually score the performance of community detection algorithms. The thesis hypotheses are as follows:

*Thesis Statement 1:* The use of the local clustering coefficient of nodes in a network can be improved compared to some previous works such as SIWO [1]. It can be utilized to determine the strength of the edges in a neighborhood inside the network. Using the edges' local strength may improve the performance of some methods that find communities in a network on a global scale that can handle weighted edges.

*Thesis Statement 2:* A fast and reliable way to solve the potential disconnectedness of detected communities can be designed to perform in linear time without too many iterations. Previous works either cannot guarantee that the detected communities are composed of only connected nodes, or they need a significant number of iterations to do so. Louvain [13] and Leiden [14] are two examples of such practices.

*Thesis Statement 3:* A general community detection framework can be created such that it detects the communities faster than previous methods while maintaining accuracy. It is also able to ensure that all the identified communities comply with the definition of community.

*Thesis Statement 4:* A novel local community discovery framework can be devised to not merge the outlier nodes to the community. It is much faster than previous algorithms due to the novel approach of using local information and the notion of edge strength in the network.

*Thesis Statement 5:* A new local community discovery algorithm can be developed that addresses the problem in the early steps of community expansion and can perform without any input parameter. This process is faster than existing methods because of the new technique to calculate the expected number of inter-connections with the neighbors of the nodes in the discovered community.

*Thesis Statement 6:* A goodness function that evaluates the quality of a community or the entire partitioning of a network with uncertain edges. If the edges inside a community have higher existence probability, the quality of that community is higher. The quality of the detected partition is the average of the quality of the communities inside it.

## 1.4 Thesis Contribution

There are five major contributions in this work:

- **Global Community Detection** We propose a novel approach to detect community structure in a network using global information, which is more efficient and accurate than the previous works in this area. We also address the problem of resolution limit (a common problem in the modularity-based algorithms) and the poor connectedness of discovered communities that many community detection algorithms suffer from.

- **Local Community Detection** Since detecting the community using local information starts with one single node and its neighborhood, there are many issues with initiating the process. In this study, we address this problem, which leads to developing a new method that discovers more accurate communities.

- **Uncertain Network Generator** Due to the limited number of publicly available uncertain network datasets and the drawbacks of previous works in this area, we put forward a novel approach to generate networks with uncertain edges.

- **Community Detection for Uncertain Networks** Considering previous works in this area, two main problems need to be addressed; some approaches seem to be ad-hoc. More importantly, many algorithms consider the uncertainty of edges for uncertain networks like simple weights on graphs. These algorithms do not only indicate the probabilistic nature of such networks but also make it impossible to incorporate the probability and weight of edges to discover communities at the same time. We propose a novel approach that deals with uncertain networks and finds local community structure inside them.

## 1.5 Thesis Organization

Chapter 2 reviews different classes of community detection algorithms such as global methods, local methods, methods used for uncertain networks, and methods that can find overlapping communities. A set of internal, external, and relative measures for evaluating community detection algorithms are also provided. Chapter 3 improves several aspects used in former approaches, including a new notion of strong and weak edges, incorporating the idea of the smart local move when the nodes move to their neighbor communities to optimize the quality function and address the connectedness issue many previous algorithms suffer from. Chapter 4 highlights some critical issues in local community discovery algorithms, such as merging outliers, detections with low accuracy, and slow performance. A novel method is proposed that outperforms the previous method in terms of accuracy and speed, based on the notion of edges' local strength. Chapter 5 focuses on a specific and practical type of uncertain networks which can be represented with a probabilistic graph. After analyzing some related works, a new method is suggested to address the drawbacks of existing methods. A new goodness measure is formulated to evaluate the quality of the discovered communities in such networks.

# Chapter 2

# Background and Related Work

## 2.1 Social Network Analysis

### 2.1.1 Social Networks

Social Networks have been mainly studied in social and behavioral sciences that is useful for analyzing relationships between entities of the society. It is like a data structure made of a set of actors representing the individuals or entities and sets of ties such that each set represents a different kind of relation or interaction between the actors. The primary purpose of using social networks is to find the patterns of interactions between the entities in the structure [15]. These patterns become more visible as the network size grows. However, social networks are analyzed at the scale relevant to the researcher's question.

The actors in social networks are usually persons; however, we can expand the definition to other types of entities such as web pages, scientific documents, proteins, etc. Then, we may call these structures information networks. The relations inside these networks could also be in various types, such as friendship, weblinks, citations, protein-protein interactions, companies with business transactions, etc. These relations are demonstrated using links between the actors of the network. They may have properties such as directions (for relationships in which actors play different roles like giver and receiver), weights (for relations that need to be described by importance, frequency, etc.), and probabilities (for relations that are not certainly known). There are many exciting research topics in complex social network analysis, such as link prediction, community detection, and information diffusion, which have received considerable attention. In this study, our focus is on community detection, which is reviewed and further investigated.

### 2.1.2 Social Network Analysis

Social network analysis (SNA) is not a formal theory in sociology but rather a strategy for investigating social structures through the use of networks and graph theory [16]. It represents a network by a graph where the nodes correspond to the entities and the edges describe the relationships or interactions of the nodes that are connected to those edges.

Although it is rooted in sociology, nowadays it is more of an interdisciplinary field of study, including researchers from communications, computer science, economics, criminology, medicine, political science, and other disciplines [15]. For example, in communications, social network analysis is utilized to obtain measures of safety communication [17], in medicine, it can help us understand the progression of the spread of a contagious disease [18], in political science, it is used to effectively analyze the interdependence and flows of influence among individuals, groups, and institutions [19]. These networks are often visualized such that nodes are represented as points and edges are represented as lines that connect them. The study of these structures requires social network analysis to identify local and global patterns.

### 2.1.3 Terms and Definitions

Here we list the essential terms in the field of social network analysis, which we repeatedly use in the subsequent chapters. Then we define each one of them in a few sentences. Given a network represented by a graph $G = (V, E)$ where $V$ is the set of $n$ nodes ($|V| = n$) and $E$ is the set of $m$ edges ($|E| = m$) we can define as follows:

- **Adjacency Matrix** is an $n \times n$ matrix, in which element $a_{i,j}$ is a boolean value shows the connection between nodes $v_i, v_j \in V$ for an unweighted network or is a real value shows the weight of the edge $e_{i,j} \in E$ if the network is weighted. However, we often use $e_{i,j}$ instead of $a_{i,j}$ in this document, when we talk about the existence and the weight of that edge simultaneously and use $a_{i,j}$ to only show the edge's existence.

- **Directed / Undirected** describes if the adjacency matrix is symmetric or not, which means $a_{i,j}$ is not necessarily equal to $a_{j,i}$. In the second kind, $e_{i,j}$ and $e_{j,i}$ are two different edges, whereas in the first kind they are different representation of the same edge between nodes $v_i$ and $v_j$.

- **Distance** between two vertices in a graph is the number of edges in a shortest path (also called a path geodesic) connecting them. This is also known as the geodesic distance [20].

- **Neighbourhood** of a node consist of a set of nodes directly connected to it in an undirected graph. The neighbourhood of node $v_i$ is a set of all nodes $v_j \in V$, where $e_{i,j} \in E$ is not zero.

- **Community** consists of a subset of nodes that are related to each other fairly stronger than the rest of the network. This strong relation could be the higher frequency, strength or intensity. The communities are usually characterized by being highly connected internally and having only a few connections to the rest of the network.

- **Degree** of a node is the size of the neighbourhood of that node. In case of directed graphs, it could be divided into indegree and outdegree, and in case of networks with weighted edges $d_i = \sum_{j \in V} e_{i,j}$.

- **Density** is a measure for how connected the nodes are in a network i.e. the ratio of edges in the network divided by the total number of possible edges. The density for undirected networks is $d = \frac{2m}{n(n-1)}$, where $n$ is the number of nodes and $m$ is the number of edges in $G$.

- **Diameter** is the largest number of nodes which must be traversed in order to travel from one node to another when paths which backtrack, detour, or loop are excluded from consideration.

- **Clique** is a subset of nodes in $V$ of an undirected graph $G$ such that every two distinct nodes in the subset are adjacent; i.e. the induced subgraph is complete.
- **Triangle** is a clique of size three. It is more often noticed rather than cliques of other sizes, as it is the smallest and most frequent non-trivial clique in a network.
- **Maximal clique** is a clique that cannot be extended by including one more adjacent node, meaning it is not a subset of a larger clique.
- ***n*-clique** is a maximal subgraph such that the distance of each pair of its vertices is not larger than $n$ [21]. A normal clique that is defined above, can be considered as a $n$-clique when $n = 1$.
- ***n*-clan** is an $n$-clique whose diameter is not larger than $n$, i.e. a subgraph such that the distance between any two of its vertices, computed over shortest paths within the subgraph, does not exceed $n$ [11].

## 2.2 Community Detection in Social Networks

Since nowadays we have access to large datasets of information and are able to model them in the form of information networks, community detection becomes a popular research topic in computing science. Many different algorithms have been proposed to find community structures in a complex network. This line of research is similar to the clustering methods in data mining and machine learning. However, clustering methods in data mining and machine learning mainly use the attributes of data samples, whereas in social network analysis the focus is on the relations between the data entities.

Regarding the subject of the study, we may divide the community detection approaches into different groups. For example, if we focused on how a method finds the community structure, we could have divided the methods into groups such as graph partitioning-based methods, hierarchical clustering-based methods, propagation-based methods, modularity-based methods, etc [11]. However, in this study we are interested in three subjects:

- How the information of the network topology is gained and used, either the network structure is completely known or only local information is accessible.

- The networks are being investigated are either deterministically known or there is some kind of uncertainty about their edges.

- The community partitioning of the network which consists of disjoint communities or some nodes that cause overlap among the communities.

Because of these concerns, we divided the previous works into four categories: global community detection algorithms, local community detection algorithms, overlapping community detection algorithms, and community detection algorithms in uncertain networks. In the following, we generally review each class of algorithms separately.

### 2.2.1 Global Community Detection Algorithms

Some community detection approaches are called global because they exploit knowledge about the whole network topology. Global approaches yield accurate results but do not scale well on large networks. The time complexity of global approaches is usually high. Thus, these methods cannot be applied on very large networks consisting of millions of nodes and edges in a reasonable time [22].

Given a network represented by a graph $G = (V, E)$ where $V$ is the set of $n$ nodes ($|V| = n$) and $E$ is the set of $m$ edges ($|E| = m$), we aim to build a partition $P = \{c_1, c_2, ..., c_k\}$ where each $c_i$ is a subset of $V$, such that $c_i \cap c_j = \emptyset$ for $i \neq j$ and $\bigcup_{i=1}^{k} c_i = V$. However, for overlapping communities, the intersection of any two communities $c_i$ and $c_j$ is not necessarily empty. We also need a definition for the communities concerning the global scale. In general, we expect to see such structures in networks different from a random network, there are yet many definitions for the community. A common definition that is used as a basic idea in many popular graph clustering methods is established on the concept of the *null model*, which is a realization of a graph or subgraph devoid of community structures. The most popular null model is proposed by Newman and Girvan that is a randomized version of the original graph having the same number of edges and the same number of nodes. In this model edges are randomly assigned, such that the expected degree of each node remains the same as the degree of that node in the original graph [12]. This being said, a community is a subgraph if the number of edges inside the subgraph exceeds the expected number of internal edges that the same subgraph would have in the null model [11]. This is also a generic description of what is called *modularity*, as there exist many different null models. There are also plenty of modularity functions, some of them are reviewed in this study. The methods we review in this part of the study are agglomerative, which means they follow a bottom-up approach. In each one of them, all nodes are initially placed in their separate communities, then they iteratively merge to maximize a quality or objective function. This procedure ends when no further improvement on the quality function is possible. Each method has its unique tweaks to overcome the previous methods concerning time complexity, accuracy, connectedness, resolution limit, etc.

Many methods have been proposed respecting this idea, however, one objective function got more attention which is called Modularity Q [12] given as follows:

$$Q = \frac{1}{2m} \sum_{v_i, v_j \in V} [A_{i,j} - \frac{d_i d_j}{2m}] \delta(c_i, c_j) \tag{2.1}$$

where $A$ denotes the adjacency matrix, $m$ is the number of edges, $d_i$ and $c_i$ are respectively the degree and the community of node $i$ and $\delta(x, y)$ is the Kronecker function equal to 1 if $x = y$ and 0 otherwise.

Newman's approach [23] is the first one to utilize this idea of modularity $Q$. It has the computational complexity of $O(m^2 n)$ or $O(n^3)$ for sparse networks [24] which is not suitable to be used on large networks. Then, Clauset *et. al.* [25] showed that it is not necessary to compute the changes of the modularity function based on the adjacency matrix. They calculated $\Delta Q$ directly, and could improve on Newman's approach by achieving implementation of the same idea with the computational complexity of $O(m.d.\log n)$, where $d \sim \log n$ is the depth of the dendrogram. For sparse networks, the running time is $O(n \log^2 n)$ which could be considered as quasi-linear. To compute $\Delta Q$, they initialized the values of the community matrix as follows:

$$\Delta Q_{i,j} = \begin{cases} \frac{1}{2m} - \frac{k_i k_j}{(2m)^2}, & \text{if } c_i,\, c_j \text{ are connected} \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

Then if communities $c_i$ and $c_j$ are joined together, they label the combined community as $c_j$, update the $j^{th}$ row and column, and delete the $i^{th}$ row and column altogether. The update rules are as follows:

$$\Delta Q'_{j,k} = \begin{cases} \Delta Q_{i,k} + \Delta Q_{j,k}, & \text{if } c_k \text{ is connected to both } c_i \text{ and } c_j \\ \Delta Q_{i,k} - 2a_j a_k, & \text{if } c_k \text{ is connected to } c_i \text{ but not to } c_j \\ \Delta Q_{j,k} - 2a_i a_k, & \text{if } c_k \text{ is connected to } c_j \text{ but not to } c_i \end{cases} \tag{2.3}$$

$$a'_j = a_j + a_i \tag{2.4}$$

Blondel *et. al.* [13] could make it even faster by proposing Louvain. This method has two main phases that are repeated until no further improvement on modularity $Q$ is possible. The first phase consists of assigning a different community to each node, then moving them in a random order to a neighboring community that gains the greatest positive changes in the quality function. If there is no neighbor community with an improvement on $Q$, then the node remains in its community. In the second phase, a new weighted graph is created in which each node corresponds to a community detected in the first phase. Each edge in the new graph is assigned a weight equal to the sum of the weights of edges between the nodes in the corresponding communities. In this method, $\Delta Q$ is computed when a node leaves its former community and enters the new one. The new technique to compute $\Delta Q$ and the proper data structures in the implementation makes Louvain extremely fast with time complexity of $O(n \log n)$ [24].

$$\Delta Q = \left[\frac{\Sigma_{in} + k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right] \tag{2.5}$$

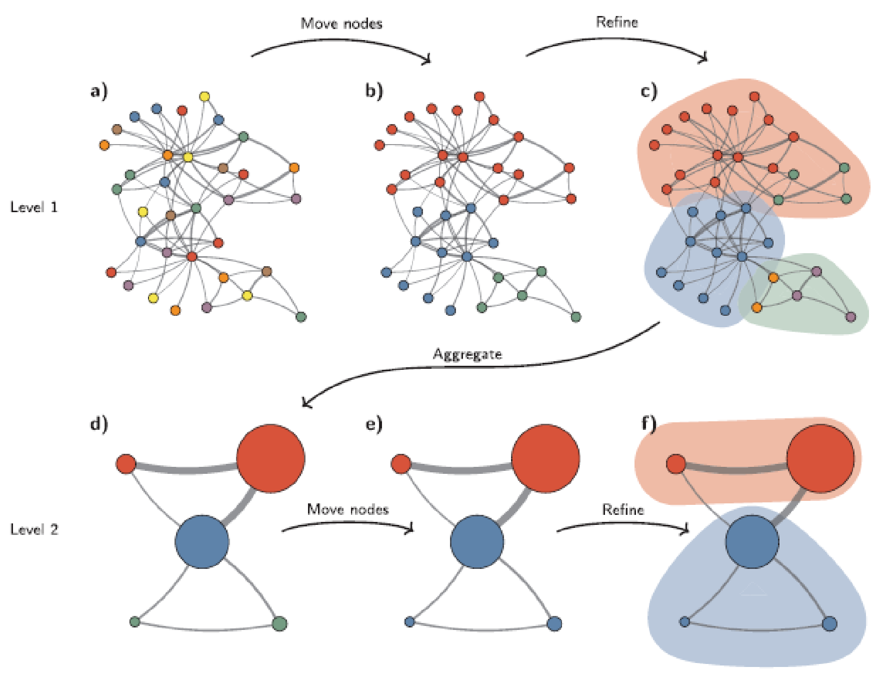


Figure 2.1: How Leiden algorithm works in two levels of aggregation [14].

Traag *et. al.* [14] brought up some problems of Louvain, including disconnectedness or badly connectedness in the detected communities, an optimization function leading to the resolution limit, etc. They proposed their method called Leiden to address these issues by using a new quality function. They had introduced this quality function in [26], which they used instead of the well-known modularity $Q$. This new function is called Constant Potts Model, or CPM, given as follows:

$$\mathbf{H} = \sum_c [e_c - \gamma \binom{n_c}{2}] \tag{2.6}$$

where $e_c$ is the actual number of edges in community $c$, $\gamma > 0$ is the resolution parameter, and $n_c$ is the number of nodes in community $c$.

They make the community detection process faster by introducing a new way of moving nodes which is called *smart local move* [27], in which a node is put forward to move only if at least one of its direct neighbors changes its community. This new approach removes all unnecessary moves that Louvain does in each cycle. To overcome the problem of bad connectedness, this algorithm adds a refinement phase between Louvain's two steps. This phase has two steps. First, every node in the temporary community (after the first phase) goes into a separate sub-community. Then, each node joins to a sub-community randomly only if this makes an improvement based on the quality function. However, the best increase is not intended in this step. Having this new middle phase results in breaking the temporary sub-communities if nodes have been merged too much, which amends the resolution limit and provides more opportunities for nodes to explore in the communities that may improve on the quality of the ultimately detected communities. Figure 2.1 shows how this method works. Leiden has two major problems; it adds another parameter to the process, required to compute the probabilities of candidate sub-communities a node can merge into in the refinement step. The second and more critical issue exists because of the randomness in the refinement step and the fact that the most significant improvement is not selected all the time. So, the algorithm may need a lot of iterations to be stable. There also is no guarantee for stability, so we need a threshold value to use as the stopping condition.

The last method that we inspect in this part is *SIWO* [1], which is an improvement over Louvain. It adds a preprocessing step, in which the dangling nodes (the nodes connected to precisely one other node in the network) are removed, and new strength values are assigned to the edges of the network. SIWO calculates these strength values based on local information of the edges and their end-nodes' clustering coefficients. It also exploits a new quality function given below to overcome the well-known problem of modularity-based approaches, the resolution limit.

$$SIWO = \sum_{v_i, v_j \in V} \frac{w(i,j)\delta(c_i, c_j)}{2} \tag{2.7}$$

where $w(i,j)$ is the weight of the edge between nodes $v_i$ and $v_j$ (also denoted by $e_{uv}$) and $\delta(c_i, c_j) = 1$ only if nodes $v_i$ and $v_j$ belong to the same community, otherwise it is 0. This method also guarantees the detected communities are qualified which means both following conditions are satisfied for any detected community $c_i$:

|  | Fast Greedy | Louvain | Leiden | SIWO | RSIWO |
|---|---|---|---|---|---|
| resolution limit free | X | X | ✓ | ✓ | ✓ |
| smart local move | X | X | ✓ | X | ✓ |
| preprocessing | X | X | X | ✓ | ✓ |
| connectedness | X | X | ✓ | X | ✓ |

Table 2.1: Comparison of Global Community Detection Algorithms

$$\sum_{v \in c_i} |N_v^{c_i}| > \sum_{v \in c_i} |N_v - N_v^{c_i}| \tag{2.8}$$

$$\frac{1}{2}\sum_{v \in c_i} |N_v^{c_i}| > \sum_{v \in c_i} |N_v^{c_j}|, j \in [1..t], j \neq i \tag{2.9}$$

where $N_v$ is the set of the neighbors of node $v$ and $N_v^c$ is the set of the neighbors of node $v$ that are also in community $c$. This method is the only one among the previous works that takes advantage of the local information of the edges and its neighbourhood to assign strength values which could be useful when it comes to detect strong (edges that are considered to be inside the communities) and weak (edges that are considered to be between communities) edges of the network. However, the empirical results show little improvement comparing with Louvain, such that only in some cases SIWO performs better. Table 2.1 briefly compares the previous algorithms and RSIWO, the new algorithm that we suggest in the next chapter. It shows there are some features in each method that can be improved. It is worth mentioning once again that the time complexity of global approaches is usually high. Thus, these methods cannot be applied on very large networks consisting of millions of nodes and edges in a reasonable time [22]. Due to this fact, we are inclined to use the local community detection algorithms, which can handle networks with millions of nodes and edges.

### 2.2.2 Local Community Detection Algorithms

We call some clustering approaches local because they only require partial knowledge of the network topology like the knowledge of the neighbors of each node [22]. They scale well on large networks as their time complexity is often near linear in the number of graph's edges. However, their results are often not as accurate as of the results of global methods. Given a network represented by a graph $G = (V, E)$ where $V$ is the set of $n$ nodes ($|V| = n$), and $E$ is the set of $m$ edges ($|E| = m$), and a node $v_x$, we aim to find all nodes of $V$ that belong to the same community as the given node $v_x$ does.

Like the global approaches, we need a definition for communities concerning the local scale. Different criteria could be used to reach the goal included but not limited to reachability or comparison of internal versus external connections. In this type of community detection algorithms, we can observe only the subgraph under analysis and presumably its direct neighborhood. Thus, the final result consists of nodes that are highly connected to each other, oftentimes constructing maximal cliques [11]. Although connectedness is a key criterion when a subgraph is considered a community, a strong linkage between the subgraph and the rest of the graph makes us less interested in labeling such a group of nodes as a community. Therefore, it is essential to compare the internal and external connections of a subgraph, which is why definitions of community do it. Communities can also be identified by a fitness measure, denoting to what extent a subgraph provides a given property related to its connectedness, as the more established communities correspond with more significant fitness measures [11]. In the following, we review some of the well-known local community detection algorithms.

Most local community detection algorithms work in a one-node-at-one-step discovery process. To gain more information from the unknown part of the network, we first need to expand the current community and visit the new direct neighbors of already seen nodes. We usually represent the set of community-assigned nodes with $D$, the set of $D$'s neighborhood with $S$, the nodes in $D$ that are only connected to nodes inside it with $C$, and $B = D - C$ is the boundary set. Figure 2.2 shows these sets in a network.

Branting [29] introduced the local community detection algorithms in three main steps; selecting the next nodes to be added to the community, when to stop adding nodes to the community, and removing nodes from the community if necessary. The main focus of this work is on improving the selection phase

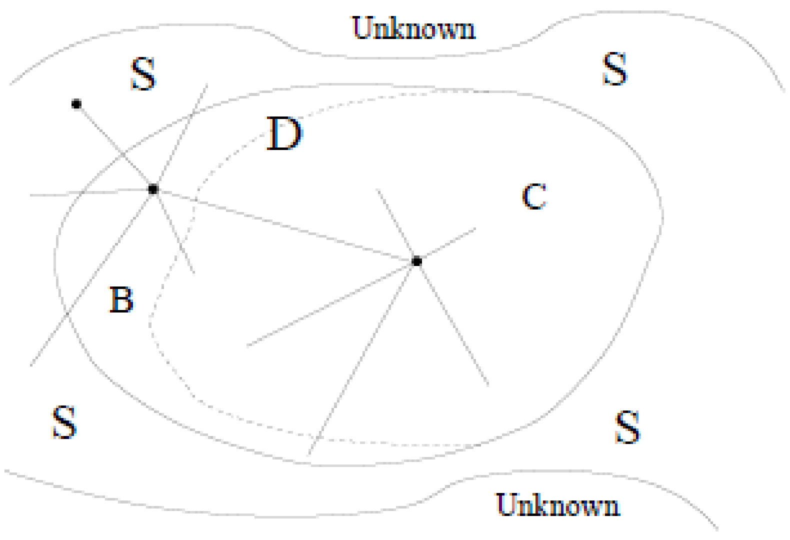


Figure 2.2: Different Parts of the Network Considering Local Approaches [28].

independent of the choice of termination and filtering.

Different approaches suggest different fitness measures that can be used to expand the community by adding new nodes from the neighbourhood of the current state of the community. Many fitness functions are defined based on the sum of all edge weights interior to the community $C$ and sum of all edge weights on the border of it, shown by $K_{in}^{C}$ and $K_{out}^{C}$ given by:

$$K_{in}^{C} = \sum_{e_{i,j} \in E | v_i \in C, v_j \in C} e_{i,j} \tag{2.10}$$

$$K_{out}^{C} = \sum_{e_{i,j} \in E | v_i \in C, v_j \notin C} e_{i,j} \tag{2.11}$$

Modularity [30] compares the number of intra-community edges to the expected number under a null model, given by:

$$Q(C) = \frac{1}{|E|}(K_{in}^{C} - \frac{(2K_{in}^{C} + K_{out}^{C})^2}{4|E|}) \tag{2.12}$$

Conductance [30] is another popular fitness function which measures the community cut, given by:

$$\phi(C) = \frac{K_{out}^{C}}{min(2K_{in}^{C} + K_{out}^{C}, 2K_{in}^{V \setminus C} + K_{out}^{V \setminus C})} \tag{2.13}$$

Yixuan *et. al.* [31] suggested a method by making two fundamental changes to the traditional spectral clustering methods. Such methods find the first few singular vectors of the Laplacian matrix of a graph proportional to the number of communities in the network (for more details refer to [32]). Then an $n \times d$ latent space matrix is formed. Vertices are clustered using some methods such as the k-means clustering algorithm. This method is not likely to work well if the communities are small and heavily overlapped. The first change is to overcome the drawback of computing the singular vectors, and the second modification is to handle the overlapping situation.

Two methods to detect local communities are modularity $R$ [33] and modularity $M$ [34], they focus on the sharpness of the boundary set $B$ and the whole community $D$ respectively. The formulas for them are as follows:

$$R = \frac{|B_{in\_edge}|}{|B_{in\_edge}| + |B_{out\_edge}|} \tag{2.14}$$

$$M = \frac{\text{number of internal edges}}{\text{number of external edges}} \tag{2.15}$$

where $R$ measures the fraction of inside-community edges to all edges with one or more endpoints in $B$. Therefore, the quality of community $D$ is measured by the sharpness of the boundary given by $B$. However, $M$ looks at both $B$ and $D$, as internal edges are the edges that have both endpoints in $D$, and external edges are the ones that have only one endpoint in $B$. Algorithms using these metrics can detect communities in complex networks. However, their results usually include many outliers, i.e., the discovered communities have high recall but low accuracy, which reduces the overall community quality.

Chen *et. al.* [28] proposed a new measure of local community structure $L$ to evaluate the local community structure when we lack global information. They also presented a community discovery process for large networks that iteratively finds communities based on the suggested measure. In contrast to many other approaches, the metric $L$ is robust against outliers. The missing feature in former metrics is the connection density as if $N$ is a set of nodes that does not have any outward edges, but every node in $N$ connects only one or two neighbors, then $N$ should not be considered as a community. The absolute number of connections does not matter, but the density should be taken into account, which is the basic idea of modularity $L$ is as follows:

$$L = \frac{L_{in}}{L_{ex}} \tag{2.16}$$

$$L_{in} = \frac{\sum_{v_i \in D} IK_i}{|D|} \tag{2.17}$$

$$L_{ex} = \frac{\sum_{v_j \in B} EK_j}{|B|} \tag{2.18}$$

where $IK_i$ is the number of edges between node $v_i$ and nodes in $D$ and $EK_j$ is the number of connections between node $v_j$ and nodes in $S$. Since they only considered the nodes in $S$ that increase $L$ value for the detected community, three different cases may happen where $L^{'}$ is the updated measure:

1. $L'_{in} > L_{in}$ and $L'_{ex} < L_{ex}$
2. $L'_{in} < L_{in}$ and $L'_{ex} < L_{ex}$
3. $L'_{in} > L_{in}$ and $L'_{ex} > L_{ex}$

The first equation represents the nodes that strengthen the community and weaken the external connections, so the nodes definitely belong to the community. The second equation describes the nodes that are weakly connected to both community and the rest of the network, so they are outliers. And finally, the third equation shows the potential hub nodes, but it is not possible to say anything about them at this step. Since the method adds nodes that may be outliers and hubs to the community, there is an examination step at the end that keeps nodes in the community only if by removing them the $L$ value drops.

A community exists for a given node $v_x$ if it remains in $D$ at the end of the process (it may vanish during the examination step). We usually begin such operations with one node $v_x$, but the method can be initialized with multiple nodes to allow a larger starting $D$, $C$, $B$, and $S$. We can also repeatedly run the same procedure, every time with a new initial node in the remaining part of the network, to find all communities inside it. If not all the communities are needed in the network (but some of them), parameters as stopping criteria could be useful that prevents exploring the whole network as it is not unnecessary or impractical.

### 2.2.3 Overlapping Community Detection Algorithms

We assumed each node belongs to exactly one community that made a disjoint partitioning in the network in previous methods. However, this is not always the case. In real-world networks, it most surely happens when one node or more contributes to different groups in the same society, which results in overlap among the network's communities. For example, a university student is probably a member of different social groups. These include a group of participants in a specific class, researchers in a particular lab, department basketball team, etc. Considering a real network of entities, overlapping community structure seems inevitable and an inherent characteristic of social networks, so it is important to deal with this property where nodes would be able to be a member of more than exactly one community. Given a network represented by a graph $G = (V, E)$ where $V$ is the set of $n$ nodes ($|V| = n$) and $E$ is the set of $m$ edges ($|E| = m$), we aim to build a partition $P = \{c_1, c_2, ..., c_k\}$ where each $c_i$ is a subset of $V$, such that $\bigcup_{i=1}^{k} c_i = V$. However, $c_i \cap c_j = \emptyset$ for $i \neq j$ is not necessary, as $c_i$ and $c_j$ may contain same nodes. Figure 2.3 represents a community detection with overlapping communities for the Karate network, where the green nodes are shared between the two communities.

Many different approaches, such as hierarchical clustering and optimization-based algorithms, have been proposed to discover community structure in networks. These methods restrict a node to belonging to only one community. To overcome this limitation, overlapping community detection algorithms have drawn lots of attention, mainly divided into two groups of node-based algorithms and edge-based algorithms [36]. In this section, we offer a review of some previous works in this domain.

Palla *et. al.* [37] proposed the Clique Percolation Method, also known as

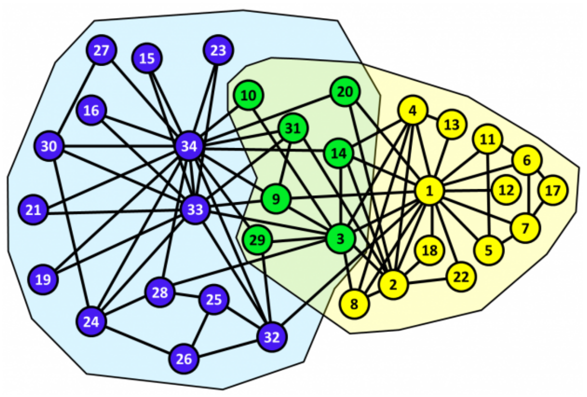
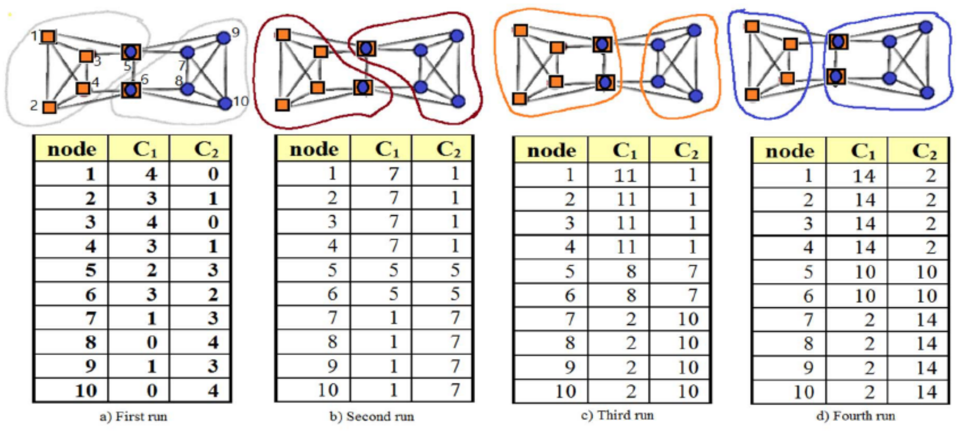


Figure 2.3: Karate Network Represented with Overlapping Communities [35]

CPM, which is a popular approach for analyzing the overlapping community structure of networks. This method starts by placing a clique of size $k$ in the community and finds other $k$-cliques, which are neighbors of any previously detected clique inside the community. Finally, CPM expands the community using such neighbor cliques. Two cliques of size $k$ are neighbors if they have $k - 1$ shared nodes. This method finishes the community detection when no further neighboring $k$-clique exists. Since any node may belong to many cliques, it provides the possibility of detection of overlapping communities.

Whang *et. al* [38] suggests another method which is capable of detecting overlapping communities. This method consists of four phases: filtering, seeding, seed set expansion, and propagation. In the filtering phase, some regions of the graph that are trivially separable from the rest of the graph are removed, and consequently, they do not participate in overlapping communities. In the seeding phase, the seeds in the filtered graph are found, then in the seed set expansion phase, the seed sets are expanded using a personalized PageRank clustering scheme. Finally, in the propagation phase, the communities expand further to the regions removed in the filtering phase.

Yakoubi *et. al.* [39] proposed another method being able to detect overlapping communities. The basic idea underlying the proposed algorithm is that a community is composed of two types of nodes: Leaders and Followers. First, it searches for nodes in the network that are likely to be leaders in a community. The list of leaders is then reduced by grouping leaders estimated to be



| node | $C_1$ | $C_2$ |
|---|---|---|
| 1 | 4 | 0 |
| 2 | 3 | 1 |
| 3 | 4 | 0 |
| 4 | 3 | 1 |
| 5 | 2 | 3 |
| 6 | 3 | 2 |
| 7 | 1 | 3 |
| 8 | 0 | 4 |
| 9 | 1 | 3 |
| 10 | 0 | 4 |

| node | $C_1$ | $C_2$ |
|---|---|---|
| 1 | 7 | 1 |
| 2 | 7 | 1 |
| 3 | 7 | 1 |
| 4 | 7 | 1 |
| 5 | 5 | 5 |
| 6 | 5 | 5 |
| 7 | 1 | 7 |
| 8 | 1 | 7 |
| 9 | 1 | 7 |
| 10 | 1 | 7 |

| node | $C_1$ | $C_2$ |
|---|---|---|
| 1 | 11 | 1 |
| 2 | 11 | 1 |
| 3 | 11 | 1 |
| 4 | 11 | 1 |
| 5 | 8 | 7 |
| 6 | 8 | 7 |
| 7 | 2 | 10 |
| 8 | 2 | 10 |
| 9 | 2 | 10 |
| 10 | 2 | 10 |

| node | $C_1$ | $C_2$ |
|---|---|---|
| 1 | 14 | 2 |
| 2 | 14 | 2 |
| 3 | 14 | 2 |
| 4 | 14 | 2 |
| 5 | 10 | 10 |
| 6 | 10 | 10 |
| 7 | 2 | 14 |
| 8 | 2 | 14 |
| 9 | 2 | 14 |
| 10 | 2 | 14 |

Figure 2.4: Updating the Belonging Matrix Before Normalization [40]

in the same community and set of communities $C$ is generated. Each node in the network (a leader or a follower) computes its membership degree to each community in $C$. Finally, each node is assigned to top-ranked communities in its final obtained membership preference list.

Elyasi *et. al.* [40] proposed a method that used the stochasticity of disjoint community detection algorithms, as by running such algorithms multiple times, the detected disjoint communities would be different. They selected Louvain as the disjoint community detection algorithm because it is well-known and works fast. After the first run of Louvain, an $N \times C$ belonging matrix is made ($N$ is the number of nodes in the network and $C$ is the number of disjoint communities detected), such that the number in row $i$ and column $j$ shows the number of edges inside community $c_j$ that is connected to node $v_i$. For any next iterations of Louvain, they also update the values of the belonging matrix, Figure 2.4 shows how the updates occur. After all the iterations are done, the matrix values are normalized in the range of $[0, 1]$. These normalized values show how much a node belongs to a community in the network. Finally, the belonging of a node to a community is determined based on a user-defined threshold. Communities could merge if they share more nodes than a certain number to increase the modularity and establish better communities. However, this step in the method is not recommended because it is highly costly ($O(n^3)$ if n represents the number of nodes).

The main idea of this method is to use strong and weak components of the network communities to find the nodes that belong to more than just one community. It is also scalable because it is useful on networks with larger sizes. We also can substitute Louvain with any other algorithm that produces non-deterministic communities and has a reasonable execution time. However, this method suffers from some ambiguities, e.g., it is not clear how many times we need to run Louvain before normalizing the final belonging matrix. This method also assumes that Louvain's stochastic results always consist of the same number of communities, whereas there is no guarantee. It is also unclear how a specific community goes to the belonging matrix's same column, over different runs. The algorithm uses a threshold value to determine if a node belongs to a community. However, there is no exact way to find an acceptable threshold. The final step that merges the found communities is supposed to enhance the communities' modularity measure and quality. Although it is computationally expensive and they prefer not to use it.

Considering the mentioned strengths and weaknesses, we learn that using strong/weak community components is a way to find communities with overlap. However, the proposed method suffers from some issues (including but not limited to ambiguity in both algorithm and parameters, high time complexity, etc.) that make us believe there is much room in this research domain.

### 2.2.4 Community Detection Algorithms in Uncertain Networks

Regardless of the network's communities and the fact that they may intersect or not, we always take the structure of the corresponding graph for granted. However, in many real datasets, the network structure is not precisely and deterministically known. There are different types of uncertainty in networks; for example, we may not be sure about the existence of nodes, edges, attributes, or even combinations of them. In this study, we focus on uncertainty on edges of the network, which means we are entirely sure of the network's node-set. However, rather than knowing the edge set of the network exactly, we know the connections between nodes only with a certain probability.

The uncertainty in the edges of the network may come from different sources. For example, if the data is acquired using not perfect measurement tools, there always is the chance of inaccuracy and noise for the network dataset. We review some previous works concerning networks with uncertain edges in this section.

Zhang *et. al.* [41] proposed a method by expanding on the original modularity $R$ [33] to address the problem of detecting local communities for networks with uncertain edges, where the probability of edge $e_{ij}$ is denoted by $P_{ij}$. Although modularity $R$ only considers networks with deterministic edges, Zhang proposed a variant of this called $UR$ (Uncertain $R$) to cover the uncertainty in a network. This new measure is given by:

$$UR = \frac{\mathbb{E}(B_{in\_edge})}{\mathbb{E}(B_{in\_edge}) + \mathbb{E}(B_{out\_edge})} \tag{2.19}$$

$$\mathbb{E}(B_{in\_edge}) = \frac{1}{2} \sum_{V_i \in B, V_j \in B, i \neq j} P_{ij} + \sum_{V_i \in B, V_j \in C} P_{ij} \tag{2.20}$$

$$\mathbb{E}(B_{out\_edge}) = \sum_{V_i \in B, V_j \in S} P_{ij} \tag{2.21}$$

They provided the network of Figure 2.5 as an example to demonstrate that using $UR$ is not enough if they want the community to expand appropriately. This network has two connected cliques of size 4 and 6, which cannot be detected using $UR$ alone. This failure demonstrates the need for another factor to help $UR$ in order to obtain the correct communities. A new measure $K$ is introduced to address this issue, which pays attention to nodes in both $B$ and $S$. It is given by:

$$\mathbb{K}_i = \mathbb{E}(N_{i,in\_edge}) + \mathbb{E}(N_{i,shell\_edge}) \tag{2.22}$$

where $\mathbb{E}(N_{i,in\_edge})$ is the expected number of edges that connect the candidate node $V_i$ and the nodes in $D$, and $\mathbb{E}(N_{i,shell\_edge})$ is the expected number of edges that connect the candidate node $V_i$ and the other nodes in $S$.

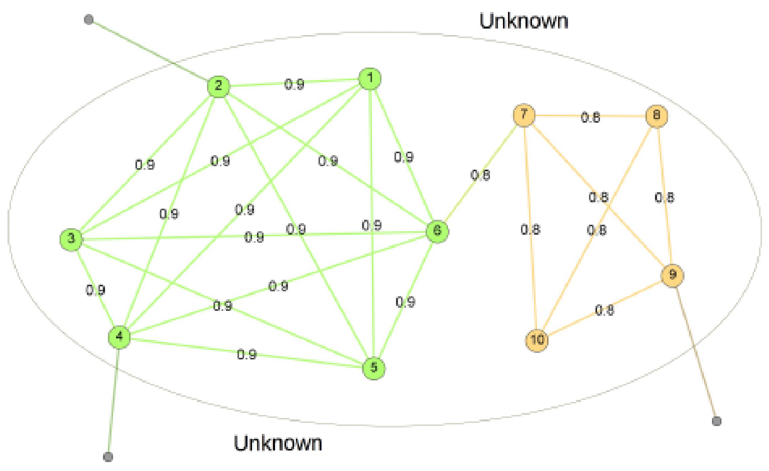


Figure 2.5: An example showing the problem of only using $UR$ to find the local community [41].

The proposed algorithm works in this way, given an uncertain network, an initial node $v_x$, and a parameter $\lambda$ which represents the number of early steps that $K$ should be prioritized over $UR$, $v_x$ goes to set $D$, and all its direct neighbors go to $S$. Then at each step, all nodes in $S$ are ranked based on both $K$ and $UR$ measures. If the number of nodes in $D$ is smaller than $\lambda$, the ranking should be based on the first $K$, then $UR$, otherwise $UR$ would be the first criteria (if there are some nodes with same measures, tie breaks randomly). They showed if $\lambda = 3$, communities tend to be more similar to the ground-truth. The first node from $S$ that improves the quality criteria should move from $S$ to $D$, then $B$ and $S$ should be updated. This procedure continues until either no nodes remain in $S$ or no node in $S$ can improve the quality criteria. Set $D$ at this step would be the community that we were looking for. They compared their method against modularity $R$, the uncertain version of modularity $R$, Louvain (which is a community mining algorithm based on global information), and the uncertain version of Louvain.

Since there is no proper dataset for uncertain networks, Zhang *et. al.* [41] suggested an algorithm that adds noise to the deterministic edges and adds new noisy edges, which results in an uncertain network. For this purpose, they create a new network with the same number of nodes, such that if an edge exists between nodes $i$ and $j$ in the original graph, in the new one, a probabilistic edge will be added between the same nodes. The algorithm also determines the number of newly added noisy edges (as a ratio of total edges in the original network). New probabilistic edges will be added to the network as well. The probability distribution for both types of edges is normal, while the mean value of the distribution function for the first kind is $\mu_1 = 0.8$, and for the second kind, it is $\mu_2 = 0.2$. They considered different mean values to emphasize on the edges that initially exist in the deterministic network, called the existential edges. The problem with this method is it cannot keep the community structure of the network intact.

## 2.3 Overview of Evaluation Methods

Considering there are many community detection algorithms which have their own perspective and method of finding community structures in the networks, Fortunato [11] demonstrated that they might outperform one another in some class of networks and they also have various computational complexities. Thus, we need to have solid criteria and evaluation metrics to compare each community detection algorithm's results to perceive which one provides more meaningful clustering for a given network.

The first way that one may think of is to ask a human expert in the field of the intended problem to validate the results [34]. However, since the human expert validation is limited and the networks under study are usually large, it is not feasible to do it this way. Therefore, we need structured methods to evaluate different results. In the following, we review and discuss different evaluation methods divided into three main types: external evaluation, internal evaluation, and relative evaluation.

In the following, we assume the graph $G = (V, E)$ is given, where $V$ is the set of $n$ nodes ($n = |V|$) and $E$ is the set of edges (represented by $(u, v, w = 1, p = 1)$ where $u$ and $v$ are the end-nodes, $w$ is the weight and $p$ is the probability of an edge) of the graph. The detected communities are shown as $C = \{c_1, c_2, ..., c_k\}$ on $V$ such that $V = \cup_{i=1}^{k} c_i$ and $c_i \cap c_j = \emptyset$ for all $i \neq j$.

### 2.3.1 External Evaluation

In some cases, we know the actual community structure of the network, which we call the *ground-truth communities*. There are two different classes of networks that we could have such information about them. Such networks are one of the prespecified small networks (such as Zachary karate club [42]) that are frequently used in many studies in this field or a synthesized network made by network generators (such as LFR benchmark [43]) where we impose the community structure. However, in real-world networks, we barely have such prior information about the network's community structure. Thus, the discovered communities cannot be validated based on external evaluation.

Girvan and Newman [9] proposed the first synthetic network generator for community evaluation, called the GN benchmark. The output of this network generator is a graph that has 128 nodes with an expected degree of 16, which is divided into four groups of equal sizes. The existence probabilities of the edges between pairs of nodes in the same group and different groups are $z_{in}$ and $1 - z_{in}$, respectively. The main problem of this benchmark is the simplicity of its community structure, which makes most algorithms perform well. Lancichinetti *et. al.* [43] improved on the GN benchmark and proposed the well-known LFR benchmark. LFR considers power-law distribution for the degrees of nodes and community sizes. These modifications result in properties more similar to real-world networks. The nodes of these generated graphs share a fraction of $1 - \mu$ of their edges with the other nodes of their community and a fraction of $\mu$ with the nodes of the rest of the network.

With the ground-truth information $G = \{G_1, G_2, ..., G_k\}$, it is easy to validate the detected communities only by comparing them against the known communities. We introduce five different metrics that could be used for the evaluation of the detected communities, such Jaccard Index [44], Purity [45], Adjusted Rank Index (ARI) [46], Normalized Mutual Information (NMI) [47]. Another simple but important metric is the ratio of the number of detected communities over the network's actual number of communities, which is depicted by ${}^{C}/_{C_r}$. To evaluate the detected communities using this metric is informative because it shows if the algorithm suffers from either resolution limit (if the ratio is close to 0) or field of view limit (if the ratio is relatively greater than 1). Jaccard index can compare a pair of communities, one from set $C$ and the other from set $G$. Thus, we can replace it with other functions in the different methods anywhere, a comparison between $C_i$ and $G_j$ is needed.

**Jaccard Index** [44], also known as the Jaccard similarity coefficient, is a statistic used to show how much two sets of items are similar and is formally defined as the size of the intersection divided by the size of the union of the sample sets as given:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.23}$$

**Purity** [45] is the number of correctly assigned nodes divided by the total number of nodes in $V$. Purity ranges from 0 that shows no agreement at all and 1, which means a full agreement. The mathematical representation of purity is as follows:

$$purity(C,G) = \frac{1}{n} \times \sum_i max_j |C_i \cap G_j| \tag{2.24}$$

**Adjusted Rand Index (ARI)** [46] penalizes both false negatives and false positives. ARI ranges between $-1$, which indicates no agreement at all, and $+1$, which means a full agreement. Let $a$, $b$, $c$, and $d$ denote the number of pairs of nodes that are respectively in the same community in both $G$ and $C$, in the same community in $G$ but in different communities in $C$, in different communities in $G$ but in the same community in $C$, and in different communities in both $G$ and $C$. These four values are computed as follows:

$$a = \sum_{i,j} \delta(C^i, C^j)\delta(G^i, G^j)$$
$$b = \sum_{i,j} (1 - \delta(C^i, C^j))\delta(G^i, G^j)$$
$$c = \sum_{i,j} \delta(C^i, C^j)(1 - \delta(G^i, G^j))$$
$$d = \sum_{i,j} (1 - \delta(C^i, C^j))(1 - \delta(G^i, G^j))$$

where $C^n = \{C_i | n \in C_i\}$ and $\delta(x, y)$ is 1 if $x = y$ and 0 otherwise. Then ARI is computed by the following formula:

$$ARI(C, G) = \frac{\binom{n}{2}(a + d) - [(a + b)(a + c) + (c + d)(b + d)]}{\binom{n}{2}^2 - [(a + b)(a + c) + (c + d)(b + d)]} \tag{2.25}$$

**Normalized Mutual Information (NMI)** [47] is based on defining a confusion matrix $N$, where the rows correspond to the ground-truth communities, and the columns correspond to the detected communities. The element $N_{i,j}$ in this matrix is the number of nodes in the ground-truth community $c_i$ that appear in the detected community $c_j$. A measure of similarity between the partitions, based on information theory, is then:

$$NMI(C, G) = \frac{-2 \sum_{i=1}^{n_C} \sum_{j=1}^{n_G} N_{ij} \log(\frac{N_{ij} N}{N_{i.} N_{.j}})}{\sum_{i=1}^{n_C} N_{i.} \log(\frac{N_{i.}}{N}) + \sum_{j=1}^{n_G} N_{.j} \log(\frac{N_{.j}}{N})} \tag{2.26}$$

where the number of real communities is denoted $n_G$ and the number of detected communities is denoted $n_C$, the sum over row $i$ of matrix $N_{i,j}$ is denoted $N_{i.}$ and the sum over column $j$ is denoted $N_{.j}$. Suppose the detected communities are identical to the ground-truth. In that case, $NMI(C, G)$ takes its maximum value of 1 if the partition detected by the algorithm is totally independent of the ground-truth $NMI(C, G) = 0$.

### 2.3.2 Internal Evaluation

As mentioned before, in most cases of real-world networks, we cannot access the ground-truth community information to evaluate the communities detected by an algorithm. Thus, we need evaluation methods that can do this task using only the network's built-in information. These methods need to act based on the internal criterion that measures the similarity between the detected community structure and the structure of the data. An internal criterion can also be considered as a quality measure to compare different community results. In the following, we introduce four evaluation metrics such as Modularity $Q$ [12], Conductance [48], Coverage [49], and Performance [50].

**Modularity Q** [12] is an important measure that falls into this group of techniques. We use it to validate a single community detection result and also to compare different community detection results. Modularity is defined as the fraction of edges within communities minus the expected value of this fraction derived based on the null model. It can be computed with the formula given in Equation (2.1).

**Conductance** [48] compares the size of a cut (i. e., the number of edges cut) and the weight of the edges in either of the two subgraphs induced by that cut. Consider a cut that divides the graph into $K$ communities $\{c_1, c_2, ..., c_K\}$. The conductance of any given community $\phi(c_i)$ can be obtained as shown in Equation (2.27), where $a(c_i) = \sum_{v_j \in c_i} \sum_{v_k \in V} w(v_j, v_k)$ is the sum of the weights of all edges with at least one endpoint in $c_i$. This $\phi(c_i)$ value represents the cost of one cut that bisects the graph into two node sets $c_i$ and $V - c_i$. Since we detected $K$ communities, we need $K - 1$ cuts to achieve that number. It is also assumed the conductance for the whole partition to be the average value of those $K - 1$ cuts.

$$\phi(c_i) = \frac{\sum_{v_j \in c_i} \sum_{v_k \notin c_i} w(v_j, v_k)}{\min(a(c_i), a(\bar{c_i}))} \tag{2.27}$$

$$\phi(C) = avg(\phi(c_i)) = \frac{1}{K} \sum_{i=1}^{K} \phi(c_i) \tag{2.28}$$

**Coverage** [49] of a clustering $C$ (where $C = \{c_1, c_2, ..., c_k\}$) is given as the fraction of the weight of all intra-cluster edges with respect to the total weight of all edges in the whole graph, as shown in the following. Coverage values usually range from 0 to 1. Higher values of coverage mean that there are more edges inside the communities than edges linking different communities, which translates to a better partitioning.

$$coverage(C) = \frac{\sum_{v_i,v_j \in V} w(v_i, v_j)\delta(c_i, c_j)}{\sum_{v_i,v_j \in V} w(v_i, v_j)} \tag{2.29}$$

**Performance** [50] counts the number of internal edges in a community along with the edges that do not exist between the community's nodes and other nodes in the graph. Performance ranges from 0 to 1, and higher values indicate that a community is both internally dense and externally sparse and, therefore, a better partitioning.

$$performance(C) = \frac{f(C) + g(C)}{\binom{n}{2}} \tag{2.30}$$

$$\begin{aligned} f(C) &= \sum_{i=1}^{K} \sum_{u,v \in C_i} w(u, v) \\ g(C) &= \sum_{i=1}^{K} \sum_{j>i} |\{(u, v) \notin E | u \in C_i, v \in C_j\}| \end{aligned}$$

### 2.3.3 Relative Evaluation

The methods in this group are quality functions that would be used to compare different results that come from various community detection algorithms. It is not an easy task to define such measures as there is no consensus over the term "community" in the first place, which makes it more difficult to judge different detected communities. However, using some inspiration from well-studied clustering validation measures in machine learning, some criteria are adapted for the problem of community detection, including but not limited to Dunn Index [51], Silhouette [52], and C-Index [53]. In addition to being used as a goodness measure, we can use them instead of modularity $Q$ to design novel community detection algorithms.

Before introducing relative evaluation metrics, as they are adopted from machine learning evaluation approaches, an adjusted definition for the distance function is required. The distance function is strongly related to the similarity measure between two nodes of the graph. There are many similarity measures, and as a result, there are many distance functions too. In the following, we introduce some of them.

**Shortest Path** is the distance between two nodes of the graph, which could be computed using the well-known Dijkstra's Shortest Path algorithm.

**Adjacency** similarity between the two nodes $v_i$ and $v_j$ is considered to be their incident edge weight $A_{i,j}$ taken from the (weighted) adjacency matrix $A$. Accordingly, the distance measure between the mentioned nodes can be computed as follows:

$$d^A_{i,j} = A_{max} - A_{i,j} \tag{2.31}$$

where $A_{max}$ is the maximum edge weight in the graph; i.e., $A_{max} = \max_{ij} A_{i,j}$, which is considered to avoid any negative distance values.

**Adjacency Relation** of two nodes measures their structural dissimilarity, which is computed by the difference between their instant neighborhoods [54] as:

$$d^{AR}_{i,j} = \sqrt{\sum_{k \neq j,i} (A_{i,k} - A_{j,k})^2} \tag{2.32}$$

Based on each distance function, various evaluation metrics can be defined. One of which is C-Index [53], which is used as a validity measure for clustering tasks. Rabbany *et. al.* [55] generalized it to measure the quality of a network's partitioning. C-Index can be defined as follows:

$$CIndex = \frac{\theta - \min\theta}{\max\theta - \min\theta}, \quad \text{where } \theta = \frac{1}{2}\sum_{l=1}^{K} \sum_{i,j \in C_l} d(i,j) \tag{2.33}$$

where $\min\theta/\max\theta$ is the sum of $n$ smallest/largest distances between nodes in the graph, where $n = \sum_{l=1}^{K} \binom{|C_l|}{2}$. The best/worst case is when the within community distances are the shortest/longest distances in the graph [55]. These two scenarios correspond with C-Index equal to 0 and 1, respectively.

# Chapter 3

# Global Community Detection Methods

## 3.1 Motivation

Q-modularity [12] is a well-known objective function in community mining tasks. We can use it to detect communities of a network and evaluate the discovered community structure when we have no ground-truth information about the network. Newman proposed a greedy agglomerative algorithm to maximize $Q$-modularity [23]. This algorithm starts by placing each node in single communities and then moving nodes towards a neighboring community, resulting in the maximum gain in $Q$-modularity. This algorithm runs in $O(n(n+m))$, where $n$ is the number of nodes, and $m$ is the number of edges of the network. Later, Clauset *et al.* [25] improved Newman's algorithm by utilizing efficient data structures and reduced the running time of the algorithm to $O(nlog^2n)$ for sparse networks. Blondel *et al.* [13] proposed Louvain, the fastest algorithm that could almost be executed in linear time to optimize $Q$-modularity. This method repeats two main steps iteratively until no further improvement in $Q$-modularity is achievable.

Although $Q$-modularity has been widely used, Fortunato and Barthelemy [56] showed that $Q$-Modularity suffers from the resolution limit, which means that by optimizing $Q$-modularity, communities that are smaller than a scale cannot be resolved. The field of view limit [57] is in contrast to the resolution limit, which results in over-partitioning the communities with a large diameter.

Schaub *et al.* [57] showed that $Q$-modularity and the map equation (as they both are instances of optimal community detection methods) are affected by the field of view limit. The general idea behind the Map equation algorithm, proposed by Rosvall and Bergstrom [58], is to find a binary code with unique code-words for each node within a community that can be used to describe the position of a random walker in the network compactly [57].

Several propositions have been made to overcome the resolution limit of $Q$-modularity, notably by [1] and [26], who took two different directions to address this issue. The first idea is a greedy method called SIWO (refer to 2.2.1), with a critical feature different from Louvain, which is to differentiate the links. Louvain does not consider any strength value for the edges of the network when the quality function optimization runs. However, SIWO utilizes the local information of neighborhoods inside the network and computes strength values based on the nodes' local clustering coefficients. Using its innovative quality function, this SIWO tries to put strong edges as much as possible inside the communities and weak edges among them. The second idea is a novel quality function that can be used instead of $Q$-modularity. It is called Constant Potts Model, and Leiden [14] is the state-of-the-art method that uses this quality function. By changing the null model, which is compared while the quality function is optimizing, this algorithm shows improvements regarding the resolution limit problem of Louvain. In our experiments, we realized that although some ideas in these two most recent methods are useful, there is more room to get them to produce better results. Moreover, these methods are not problem-free, so we propose some alterations to amend them.

In Section 3.2, we consider the definition of strong and weak edges from SIWO, bring up the issues in this method, and try to address them as much as possible. In Section 3.3, we introduce the smart move's notion when a node needs to move locally to a neighboring community while optimizing the quality function. It has been shown that this particular adjustment may result in a significant reduction in terms of run-time. In Section 3.4, we discuss the guarantees in Louvain and the guarantees that should have existed when a credible subgraph was detected as a community. We propose a solution that ensures the connectedness of the identified communities without too many iterations or convergence ambiguity. Finally, in Section 3.5, extensive experiments show that the modifications and enhancements mentioned in this chapter significantly improve the results compared with the communities detected by the well-known Louvain method or its recent successors.

## 3.2 Strong and Weak Edges

When the network's communities are detected, some edges fall inside the communities, and the other edges lie between them. Regarding the goal of finding dense subgraphs (defined by comparing against a null model), we need to put as many edges as possible into the communities. However, to avoid putting all the nodes in a single community, one may penalize the missing edges within the communities.

SIWO assumes there are two kinds of edges, strong edges that reside inside communities and weak edges that lie between them. Then, instead of penalizing the missing links, its criterion encourages adding strong links to the communities while avoiding weak links. The weak edges have a more crucial role in the graph connectivity compared to the strong edges. By removing any strong edge, the graph connectivity would not notably change, while removing a weak edge may even divide the graph into disconnected subgraphs because it locates between two communities [1]. As nodes in the same community are more likely to have shared neighbors than nodes in different communities, weight values in the range of $(-1, +1)$ are assigned to the edges representing their strength.

SIWO suggested some steps to calculate the strength of the edges. First, $S_{i,j}$ needs to be computed for any edge $e_{i,j}$ which is the number of shared neighbors of nodes $v_i$ and $v_j$ using the following formula:

$$S_{i,j} = |\{v_k \in V : e_{i,k} \in E, e_{j,k} \in E\}| \tag{3.1}$$

where $V$ is the set of nodes of the network, and $E$ is the set of edges. We only can make the comparison of the edges' strengths locally, such that we can only compare the edges that are incident to a shared end-node. Assuming the shared node is $v_i$, we need to find $S_i^{max} = \max_{j:e_{i,j} \in E} S_{i,j}$. Then any $S$ value should be normalized using the following formula:

$$w_{i,j}^{i} = S_{i,j} \frac{2}{S_i^{max} + 1} + \frac{1}{S_i^{max} + 1} - 1 \tag{3.2}$$

$w_{i,j}^{j}$ should be computed similarly, too. Finally, a comparison between the local clustering coefficient of nodes $v_i$ and $v_j$ is needed to decide if $w_{i,j}^{i}$ or $w_{i,j}^{j}$ is better to represent the strength of the edge $e_{i,j}$. The local clustering coefficient can be computed as defined below:

$$CC(i) = \frac{|\{e_{j,k} : v_j \in N_i, v_k \in N_i, e_{j,k} \in E\}|}{\binom{d_i}{2}} \tag{3.3}$$

where $d_i$ and $N_i$ are respectively the degree and the set of neighbours of node $v_i$. $CC(i)$ is in the range of $[0, 1]$ with 1 for nodes whose neighbours form cliques, and 0 for nodes whose neighbours are not directly connected to each other. Here, each edge weight is scaled from the viewpoint of the endpoint that is more likely to be in a dense neighbourhood characterized by a large $CC$. $w_{i,j} = w_{i,j}^{i}$ if $CC(i) > CC(j)$ or otherwise $w_{i,j} = w_{i,j}^{j}$.

This scheme is used as a preprocessing step in the SIWO method. SIWO objective function (given in Equation 2.7) is optimized by putting edges with larger strengths inside the community and edges with smaller strength between them. An essential goal of SIWO was to address the resolution limit of Louvain, which was achieved using its novel weighting scheme. Although this scheme works well when the SIWO objective function is to optimize, it has two significant issues in a general case that can be improved:

1. This approach uses a relatively complex normalization formula (given in Equation 3.2), which may lead to negative values. The experiments show that any normalization (i.e., the regular min-max normalization) is sufficient. On the other hand, negative strength values make the whole preprocessing step futile in cases that we want to use this step for different community detection algorithms that are mostly designed to work with positive values.

2. Another problem of SIWO's current weighting scheme is picking the

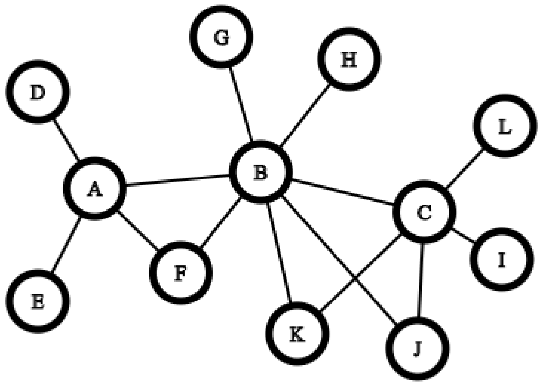


Figure 3.1: Relative Strengths of Nodes of the Graph

larger weight value for each edge among two calculated weights $w^i_{i,j}$ and $w^j_{i,j}$. Selection based on the larger clustering coefficient of their end-nodes may cause misrepresentations of the edges' strengths. To clarify the problem, we provide an example in Figure 3.1 to compare the strength of two edges $e_{a,b}$ and $e_{b,c}$, where nodes $v_A$, $v_B$, and $v_C$ are three nodes of the network that are probably connected to other nodes in the network. Suppose $w^a_{a,b}$ is a relatively large value like 0.9, but $w^b_{a,b}$ is a relatively small value like 0.2, whereas $w^b_{b,c} = 0.8$, and $w^c_{b,c} = 0.6$. With the SIWO's current method for computing the edge strength, the strongest edge incident to node $v_B$ is $e_{ab}$. However, the connection to node $v_C$ is much stronger, considering the weights from both perspectives. These weights can be larger because of more shared neighbors among $v_B$ and $v_C$ or larger primary weights on their corresponding edges.

We proposed a new way to assign the strength values, which represents the edges more accurately. In this new way, for any edge $e_{i,j}$, after computing $S^{max}_i$ and $S^{max}_j$ the new weight can be calculated with the given formula:

$$w_{i,j} = \frac{S_{i,j}}{2} \times \left(\frac{1}{S^{max}_i} + \frac{1}{S^{max}_j}\right) \tag{3.4}$$

We call this new approach $\mathbf{RSIWO_1}$. RSIWO stands for Reciprocal SIWO, as it enables us to calculate the strength of edges based on the local information obtained by the end-nodes of both sides of each edge. The subscript 1 at the end of the name denotes the fact that this is only the first improvement in three improvements that ultimately lead to the final RSIWO. It also solves the problem of negative values, which makes it a good fit for any community detection algorithm that can handle only positive edge weights.

## 3.3 Smart Local Move

Most greedy community detection approaches have two significant steps. In the first step, nodes move toward neighboring communities, and this process repeats until no further move can increase the quality of the partition. There are plenty of algorithms, including Louvain [13] and SIWO [1], that investigate all nodes in each iteration. However, we only need to move the nodes whose neighbors' communities have been changed in the last iteration. If none of the neighbors of node $v_i$ transfer to new communities, moving node $v_i$ to any new community cannot improve the partitioning quality. The smart local move is an approach to investigate and move nodes to new communities, considering the idea as mentioned earlier to reduce the number of inquiries on nodes of the network, mostly in prior iterations. Leiden uses this approach in its optimization step.

The Leiden algorithm initially uses a queue of all nodes of the network, then every time the first node in the queue pops out to be analyzed. If moving $v_i$ to a new community causes an improvement in the quality, all of the neighboring nodes of $v_i$ from its former community, which are not in the queue at that moment, would be appended to the end of the queue. By doing this, we will avoid many unnecessary nodes' moves.

Experiments show that although this idea works and significantly reduced the run time of the community detection process, using a hash table instead of a queue can improve the performance even more. The hash table consists of True/False values, which allows/prevents the algorithm from investigating a node. This enhanced approach does not need a queue with an ever-changing length, making the whole process faster with the same functionality. The improvement is more sensible on denser networks, where any node has many neighbors that can potentially be inserted into the queue. The smart local move is also applicable for any other greedy optimization-based algorithm, such as Louvain and SIWO, which are called accelerated Louvain and RSIWO$_2$. The Algorithm 1 shows how the optimization step of RSIWO$_2$ works using this new enhanced smart local move.

**Algorithm 1:** Smart Local Move in RSIWO$_2$

```
Input: Network $G$
$modified \leftarrow True$
$TraceNodes_1 \leftarrow dict$(x : True for x in G.nodes())
while $modified = True$ do
    $modified \leftarrow False$
    $TraceNodes_2 \leftarrow dict$(x : False for x in G.nodes())
    for each node $n \in$ G.nodes() do
        if $TraceNodes_1[n] = False$ then
            continue
        $OldComm \leftarrow Find\_Comm(n)$
        N $\leftarrow G.neighbors(n)$
        $NewComm \leftarrow Find\_Best\_Comm(n)$
        if $ComOld \neq NewComm$ then
            $modified \leftarrow True$
            $TraceNodes_2.update(dict$(x : True for x in N))
        $TraceNodes_1.update(TraceNodes_2)$
```

## 3.4 Guaranteeing Connectedness

Communities have different definitions; however, these definitions are based on a densely connected group of nodes inside a network. Although Louvain is a successful approach toward detecting communities in the network, and its results are frequently considered to be qualified and adequate, there is not even the simplest guarantee for the communities' connectedness. Traag *et. al.* [14] demonstrated this issue using a worst-case scenario example shown in Figure 3.2. Since, in each iteration, any node may leave its community to merge into a new community to improve the quality function, it may divide a connected community into two or more disjoint subgraphs. Because there is no amendment step in Louvain, disconnected communities' problem may only worsen after each iteration (because the same accident may occur).

To overcome this issue, a refinement step seems useful. The Leiden algorithm [14] proposed an intermediary refinement step after moving nodes locally and aggregating the graph. However, this step has some drawbacks. First of all, this refinement step is not deterministic. Even though Leiden takes advantage of this feature for more exploration to find better communities for each node, it jeopardizes the method's stability. There are large complex networks that need too many iterations to reach a stable condition because of the stochasticity of Leiden's suggested refinement step. Moreover, a new parameter for randomness has been used that requires fine-tuning as well. Leiden chooses a community that improves the quality value in the refinement step when a node needs to move. This community is not necessarily the one with the most significant improvement. This choice leads to another drawback. Finally, no guarantee has not been provided for Leiden's refinement step that it can ultimately converge. Since there is a limitation for the number of iterations (which is two times) in its implementation and mentioned in [14] as well, the produced results may be different from the stable, high-quality communities that Leiden is supposed to make in the first place.

To avoid the issues that Leiden algorithm causes, we suggest a new simple way to handle disconnectedness in the detected communities by Louvain. Every time all nodes are investigated, we need to check all the merged and built communities to ensure the problem of disconnected communities does not propagate through multiple iterations. This can be done using a DFS or BFS search in $O(n+m)$ time complexity where $n$ is the number of nodes of a community, and $m$ is the number of edges in that community. Since each node belongs to exactly one community, they are visited once in the whole graph every time we check the connectivity, so the overall time complexity is not over $O(\|V\| + \|E\|)$. It is easy to prove that this is the lowest time complexity in which the connectivity investigation can be done. We need to check all nodes and edges at least once to ensure their connectivity status to the rest of the network.

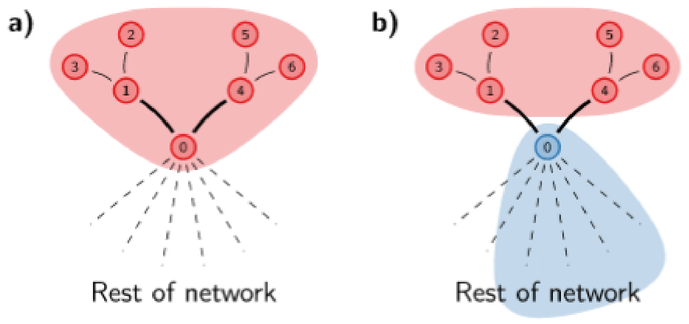


Figure 3.2: Connectivity Problem of Louvain, When Node 0 Leaves, There Will Be Two Disconnected Parts in the Red Community [14]

Even though RSIWO$_1$ has much less disconnected cases of detected communities, we propose our refinement step. Any greedy-based method that works based on an iterative merging of nodes can take advantage of to make sure no disconnected community exists in the final result. We call this improved version of SIWO that guarantees connected communities RSIWO$_3$.

## 3.5 Experiments

### 3.5.1 Experiment on the Strengths of Edges

The first experiment is to test the impact of the new weighting scheme (given in Equation 3.4) using the networks from Table 3.1. Three pairs of algorithms are being tested, SIWO and RSIWO$_1$, original and preprocessing Louvain, and original and preprocessing Fast-Greedy. The preprocessing version of Louvain and Fast-Greedy have an extra preprocessing step that adds strength values to the edges based on the RSIWO$_1$'s new weighting scheme before the step of moving nodes. We hypothesize that by using the modified version of each algorithm, those methods would be able to detect communities with smaller sizes, which ultimately improves the existing resolution problem.

The results from Tables 3.2 to 3.9 show that our hypothesis was valid and using the new weighting scheme, in almost every case that the main algorithm suffers from resolution limit, results in finding better communities with higher $Q$ and NMI measures. Although we know the Dolphins Network consists of

| | $\|V\|$ | $\|E\|$ | $\|C\|$ | $Q$ (ground-truth) |
|---|---|---|---|---|
| Football Network [9] | 115 | 613 | 12 | 0.553 |
| Zachary's Karate Club [59] | 34 | 78 | 2 | 0.358 |
| Political Blogs Network [60] | 1222 | 16717 | 2 | 0.405 |
| Political Books Network [60] | 105 | 441 | 3 | 0.414 |
| Dolphins Network [61] | 62 | 159 | 2 | - |
| Small LFR Generated Network | 50 | 332 | 3 | 0.501 |
| Medium LFR Generated Network | 150 | 979 | 12 | 0.731 |
| Large LFR Generated Network | 250 | 1569 | 20 | 0.760 |

Table 3.1: Networks to test the edge strength effect on community detection

| | $C/C_r$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 1.08 | 0.581 | 0.911 | 0.049 |
| RSIWO$_1$ (new preprocessing) | 0.91 | 0.603 | 0.909 | 0.053 |
| Louvain | 0.83 | 0.604 | 0.884 | 0.069 |
| Louvain + preprocessing | 1.0 | 0.601 | 0.926 | 0.072 |
| Fast-Greedy | 0.5 | 0.568 | 0.743 | 0.051 |
| Fast-Greedy + preprocessing | 0.75 | 0.578 | 0.824 | 0.074 |

Table 3.2: Experiments on Football Network for Edge Strength

| | $C/C_r$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 1.0 | 0.371 | 0.837 | 0.010 |
| RSIWO$_1$ (new preprocessing) | 1.0 | 0.371 | 0.837 | 0.009 |
| Louvain | 2.0 | 0.383 | 0.529 | 0.018 |
| Louvain + preprocessing | 3.0 | 0.404 | 0.545 | 0.018 |
| Fast-Greedy | 1.5 | 0.380 | 0.564 | 0.009 |
| Fast-Greedy + preprocessing | 2.5 | 0.380 | 0.420 | 0.011 |

Table 3.3: Experiments on Karate Network for Edge Strength

| | $C/C_r$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 2.0 | 0.416 | 0.695 | 2.034 |
| RSIWO$_1$ (new preprocessing) | 2.0 | 0.425 | 0.701 | 2.054 |
| Louvain | 6.0 | 0.427 | 0.631 | 1.871 |
| Louvain + preprocessing | 116 | 0.416 | 0.408 | 3.281 |
| Fast-Greedy | 5.5 | 0.426 | 0.650 | 6.012 |
| Fast-Greedy + preprocessing | 16 | 0.211 | 0.270 | 0.420 |

Table 3.4: Experiments on PolBlogs Network for Edge Strength

|  | $C/C_r$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 1.33 | 0.519 | 0.562 | 0.036 |
| $\text{RSIWO}_1$ (new preprocessing) | 1.33 | 0.523 | 0.553 | 0.040 |
| Louvain | 1.33 | 0.526 | 0.536 | 0.083 |
| Louvain + preprocessing | 2.33 | 0.516 | 0.454 | 0.095 |
| Fast-Greedy | 1.33 | 0.501 | 0.530 | 0.044 |
| Fast-Greedy + preprocessing | 2.0 | 0.497 | 0.441 | 0.052 |

Table 3.5: Experiments on PolBooks Network for Edge Strength

|  | $C/C_r$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 2.0 | 0.472 | - | 0.015 |
| $\text{RSIWO}_1$ (new preprocessing) | 2.0 | 0.526 | - | 0.018 |
| Louvain | 2.0 | 0.522 | - | 0.035 |
| Louvain + preprocessing | 10 | 0.437 | - | 0.038 |
| Fast-Greedy | 2.0 | 0.495 | - | 0.014 |
| Fast-Greedy + preprocessing | 2.5 | 0.442 | - | 0.172 |

Table 3.6: Experiments on Dolphins Network for Edge Strength

|  | $C/C_r$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 1.0 | 0.501 | 1.0 | 0.026 |
| $\text{RSIWO}_1$ (new preprocessing) | 1.0 | 0.501 | 1.0 | 0.024 |
| Louvain | 1.0 | 0.501 | 1.0 | 0.027 |
| Louvain + preprocessing | 1.0 | 0.501 | 1.0 | 0.036 |
| Fast-Greedy | 1.0 | 0.489 | 0.931 | 0.021 |
| Fast-Greedy + preprocessing | 1.0 | 0.489 | 0.931 | 0.097 |

Table 3.7: Experiments on Small Synthetic Network for Edge Strength

|  | $C/C_r$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 1.0 | 0.731 | 1.0 | 0.067 |
| $\text{RSIWO}_1$ (new preprocessing) | 1.0 | 0.731 | 1.0 | 0.081 |
| Louvain | 0.91 | 0.732 | 0.984 | 0.068 |
| Louvain+ + preprocessing | 1.0 | 0.731 | 1.0 | 0.115 |
| Fast-Greedy | 0.75 | 0.704 | 0.926 | 0.061 |
| Fast-Greedy + preprocessing | 1.0 | 0.718 | 0.990 | 0.097 |

Table 3.8: Experiments on Medium Synthetic Network for Edge Strength

| | $^C/_{C_r}$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 1.0 | 0.760 | 1.0 | 0.099 |
| $\text{RSIWO}_1$ (new preprocessing) | 1.0 | 0.760 | 1.0 | 0.162 |
| Louvain | 0.85 | 0.761 | 0.976 | 0.133 |
| Louvain + preprocessing | 1.0 | 0.760 | 1.0 | 0.190 |
| Fast-Greedy | 0.7 | 0.747 | 0.926 | 0.109 |
| Fast-Greedy + preprocessing | 0.85 | 0.749 | 0.959 | 0.189 |

Table 3.9: Experiments on Large Synthetic Network for Edge Strength

| | $^C/_{C_r}$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 1.08 | 0.581 | 0.911 | 0.049 |
| $\text{RSIWO}_2$ (with SLM) | 1.08 | 0.581 | 0.911 | 0.044 |
| $\text{RSIWO}_1$ | 0.91 | 0.603 | 0.909 | 0.053 |
| $\text{RSIWO}_{1+2}$ (with SLM) | 0.91 | 0.603 | 0.909 | 0.052 |
| Louvain | 0.83 | 0.604 | 0.884 | 0.069 |
| Louvain + SLM | 0.83 | 0.604 | 0.884 | 0.049 |

Table 3.10: Experiments on Football Network for Smart Local Move

two communities, no $Q$ or NMI measure is reported for this network as we could not find the ground-truth communities (we are not sure which entity belongs to which community).

## 3.5.2 Experiment on the Smart Local Move

The second experiment is designed to test if the smart local move could have been used in different algorithms that exploit different objective functions and still produce the same result in a shorter period. To test this hypothesis, we used the same networks; however, we only use original SIWO, modified SIWO, original Louvain, and accelerated versions to compare the run times.

The results from Tables 3.10 to 3.17 show that our hypothesis was valid

| | $^C/_{C_r}$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 1.0 | 0.371 | 0.837 | 0.010 |
| $\text{RSIWO}_2$ (with SLM) | 1.0 | 0.371 | 0.837 | 0.007 |
| $\text{RSIWO}_1$ | 1.0 | 0.371 | 0.837 | 0.009 |
| $\text{RSIWO}_{1+2}$ (with SLM) | 1.0 | 0.371 | 0.837 | 0.008 |
| Louvain | 2.0 | 0.383 | 0.529 | 0.018 |
| Louvain + SLM | 2.0 | 0.383 | 0.529 | 0.015 |

Table 3.11: Experiments on Karate Network for Smart Local Move

| | $^C/_{C_r}$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 2.0 | 0.416 | 0.695 | 2.034 |
| RSIWO$_2$ (with SLM) | 2.0 | 0.416 | 0.695 | 1.897 |
| RSIWO$_1$ | 2.0 | 0.425 | 0.701 | 1.928 |
| RSIWO$_{1+2}$ (with SLM) | 2.0 | 0.425 | 0.701 | 1.911 |
| Louvain | 6.0 | 0.427 | 0.631 | 1.871 |
| Louvain + SLM | 6.0 | 0.427 | 0.631 | 1.459 |

Table 3.12: Experiments on PolBlogs Network for Smart Local Move

| | $^C/_{C_r}$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 1.33 | 0.519 | 0.562 | 0.036 |
| RSIWO$_2$ (with SLM) | 1.33 | 0.519 | 0.562 | 0.036 |
| RSIWO$_1$ | 1.33 | 0.523 | 0.553 | 0.040 |
| RSIWO$_{1+2}$ (with SLM) | 1.33 | 0.523 | 0.553 | 0.038 |
| Louvain | 1.33 | 0.526 | 0.536 | 0.083 |
| Louvain + SLM | 1.33 | 0.526 | 0536 | 0.046 |

Table 3.13: Experiments on PolBooks Network for Smart Local Move

| | $^C/_{C_r}$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 2.0 | 0.472 | - | 0.015 |
| RSIWO$_2$ (with SLM) | 2.0 | 0.472 | - | 0.015 |
| RSIWO$_1$ | 2.0 | 0.526 | - | 0.018 |
| RSIWO$_{1+2}$ (with SLM) | 2.0 | 0.526 | - | 0.014 |
| Louvain | 2.0 | 0.522 | - | 0.035 |
| Louvain + SLM | 2.0 | 0.522 | - | 0.023 |

Table 3.14: Experiments on Dolphins Network for Smart Local Move

| | $^C/_{C_r}$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 1.0 | 0.501 | 1.0 | 0.026 |
| RSIWO$_2$ (with SLM) | 1.0 | 0.501 | 1.0 | 0.023 |
| RSIWO$_1$ | 1.0 | 0.501 | 1.0 | 0.024 |
| RSIWO$_{1+2}$ (with SLM) | 1.0 | 0.501 | 1.0 | 0.022 |
| Louvain | 1.0 | 0.501 | 1.0 | 0.027 |
| Louvain + SLM | 1.0 | 0.501 | 1.0 | 0.022 |

Table 3.15: Experiments on Small Synthetic Network for Smart Local Move

|  | $^{C}/_{C_r}$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 1.0 | 0.731 | 1.0 | 0.067 |
| RSIWO$_2$ (with SLM) | 1.0 | 0.731 | 1.0 | 0.061 |
| RSIWO$_1$ | 1.0 | 0.731 | 1.0 | 0.081 |
| RSIWO$_{1+2}$ (with SLM) | 1.0 | 0.731 | 1.0 | 0.073 |
| Louvain | 0.91 | 0.732 | 0.984 | 0.068 |
| Louvain + SLM | 0.91 | 0.732 | 0.984 | 0.062 |

Table 3.16: Experiments on Medium Synthetic Network for Smart Local Move

|  | $^{C}/_{C_r}$ | Q | NMI | Run Time |
|---|---|---|---|---|
| SIWO | 1.0 | 0.760 | 1.0 | 0.099 |
| RSIWO$_2$ (with SLM) | 1.0 | 0.760 | 1.0 | 0.093 |
| RSIWO$_1$ | 1.0 | 0.760 | 1.0 | 0.162 |
| RSIWO$_{1+2}$ (with SLM) | 1.0 | 0.760 | 1.0 | 0.140 |
| Louvain | 0.85 | 0.761 | 0.976 | 0.133 |
| Louvain + SLM | 0.85 | 0.761 | 0.976 | 0.102 |

Table 3.17: Experiments on Large Synthetic Network for Smart Local Move

and using the new enhanced smart local move speeds up the process without jeopardizing the quality of the final results. It improves SIWO and RSIWO$_1$ up to 10% and improves Louvain up to 50% with no damaging effects on the detected communities. We conclude that using this scheme in greedy-based methods that iteratively investigate nodes of the network has advantages.

### 3.5.3 Experiment on the Guaranteed Connectedness

The third experiment is designed to test how vulnerable Louvain and SIWO are to detecting disconnected communities, and if our proposed refinement step can solve this problem. For this experiment, we used the AMAZON network [62] and the DBLP network [62], which are relatively large networks with hundreds of thousands of nodes and almost one million edges. We decided to do this experiment using extensive networks as it is more probable an algorithm finds disconnected communities in them rather than smaller networks. Leiden and Louvain are the main algorithms we want to investigate; the guaranteed version of Louvain is also considered. We already claimed that SIWO could handle this issue better than Louvain. However, we include SIWO, RSIWO$_1$, and RSIWO$_3$, its guaranteed version, to support our hypothesis and show the

|  | ${}^{C}/{}_{C_r}$ | % Disconnected |
|---|---|---|
| Leiden | 0.005 | 0 |
| Louvain | 0.003 | 2.02 |
| Louvain + Guarantee | 0.003 | 0 |
| SIWO | 0.303 | 0.04 |
| RSIWO$_1$ | 0.306 | 0.00 |
| RSIWO$_{1+3}$ (with Guarantee) | 0.306 | 0 |

Table 3.18: Experiments on Amazon Network for Guaranteed Connectedness

|  | ${}^{C}/{}_{C_r}$ | % Disconnected |
|---|---|---|
| Leiden | 0.023 | 0 |
| Louvain | 0.016 | 4.12 |
| Louvain + Guarantee | 0.017 | 0 |
| SIWO | 1.141 | 0.94 |
| RSIWO$_1$ | 1.294 | 0.12 |
| RSIWO$_{1+3}$ (with Guarantee) | 1.295 | 0 |

Table 3.19: Experiments on DBLP Network for Guaranteed Connectedness

final method is capable of finding the best communities in terms of accuracy, run time, and connectivity.

The results from Tables 3.18 and 3.19 show that our hypothesis was valid and using the new method to guarantee the connectedness of the detected communities works. Considering the low ${}^{C}/{}_{C_r}$ measure of Leiden and Louvain, we might infer that they address the connectedness issue by merging the communities too much. We also observed that SIWO, using its unique quality function, handles this issue better. However, if we modify the weighting scheme to the aforementioned positive weights, it gets better. By exploiting the guaranteed version, we can claim that no disconnected community would be found in the final result.

## 3.6 Conclusion

We realized all three novel modifications to the SIWO approach: the new weighting scheme, the smart local move, and the connectivity guaranteeing scheme work as expected and can improve the performance of the former algorithm in different ways.

RSIWO provides a better representation of edge strength that results in better partitioning and more immunity to resolution limit and the inherent connectivity issue of such methods. Comparing to Louvain, it is faster, able to find communities of smaller sizes, and can guarantee that the detected communities are entirely connected. Comparing to Leiden, it is more robust, does not need any fine-tuned parameter to perform correctly. The algorithm is guaranteed to converge and converges to the same final answer every time, which means RSIWO is a deterministic method.

# Chapter 4

# Local Community Detection Methods

## 4.1 Motivation

There have been many types of research done on community search in complex networks, in which the goal is to find all nodes that belong to the same community as the given node does. Placing a source node in the community and expanding it is one of the most successful methods for community search and local community detection. Such methods show their value in different cases, particularly when the network's global structure is not accessible or too large to be used altogether.

In the context of local community detection, the observed part of the network is the neighborhood around certain nodes in the community ($D$). The community itself consists of two parts, the set of core nodes ($C$) and the set of boundary nodes ($B$). The core nodes are the ones that have no direct neighbors outside of the community, whereas boundary nodes have at least one neighbor outside of set $D$. The set of all nodes outside of $D$ that are directly connected to nodes in $B$ is called the shell nodes ($S$), and the rest of the network is unknown. Figure 4.1 shows these different areas. In local community detection approaches, with expanding the community further, more nodes will be exposed and may be considered for merging to the community in the next steps.

Numerous local community detection algorithms have been proposed as we

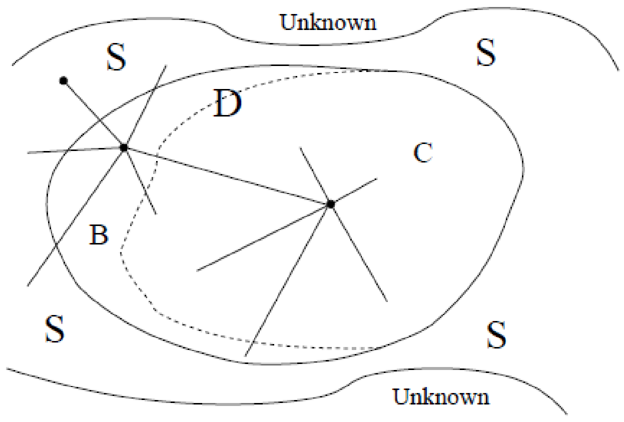


Figure 4.1: A Network Regarding Local Community Approaches [28]

discussed in Section 2.2.2, many of which can be grouped as greedy community expansion methods. They start by placing an initial node in the community, merging more nodes from the neighborhood to expand the community based on maximizing a modularity or quality function. Some of these functions can be used to evaluate local communities, as well.

Clauset [33] proposed local modularity $R$ to find the best local community. The modularity $R$ is defined as:

$$R = \frac{B_{in\_edge}}{B_{in\_edge} + B_{out\_edge}} \tag{4.1}$$

where $B_{in\_edge}$ is the number of edges that connect boundary nodes and other nodes in $D$, while $B_{out\_edge}$ is the number of edges that connect boundary nodes and nodes in $S$. Intuitively, a good community should have a sharp boundary with fewer connections from the boundary to the unknown portion of the graph, while having a higher number of links from the boundary nodes back into the local community. Thus, $R$ measures the fraction of those inside-community edges to all edges with one or more endpoints in $B$. So, community $D$ is measured by the sharpness of the boundary given by $B$. The fact that modularity R does not take the edges with both endpoints in $C$ into account is a limit that ultimately results in poor communities.

Similarly, Luo *et. al.* [34] 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. Internal means two endpoints are both in $D$, whereas external means only one of the endpoints belongs to $D$. The modularity $M$ is defined as:

$$M = \frac{\text{number of internal edges}}{\text{number of external edges}} \tag{4.2}$$

Another proposed metric is called modularity $L$ (or L-metric) presented by Chen *et. al.* [28], which aims at reducing outliers that both $R$ and $M$ suffer from and improving detection accuracy. The definition of the modularity $L$ is:

$$L = \frac{\frac{\sum_{i \in D} IK_i}{|D|}}{\frac{\sum_{j \in B} EK_j}{|B|}} \tag{4.3}$$

Where $IK_i$ is the number of edges between node $V_i$ and nodes in $D$, and $EK_j$ is the number of connections between node $V_j$ and nodes in $S$. Thus, the numerator is the average internal degree of nodes in $D$, and the denominator is the average external degree of nodes in $B$.

Regardless of which modularity function is selected to evaluate and build local communities in a network, the algorithm firstly places the start node in the community and its neighbors in the shell set. At each step, the algorithm adds the neighbor node, which gives the most significant increase of modularity to the community. Then updates the community set, the boundary set, the shell set, and finally, the modularity value. This process will finish when no candidate node could increase the modularity measure.

Although each of these algorithms addressed an issue of a previous method (for example, modularity $R$ ignores edges with both endpoints in $C$, whereas modularity M addresses this limit by considering such edges), they are almost similar in practice. They often suffer from poor outlier detection and the discovery of incorrect communities in simple ground-truth networks. To address this problem, Fagnan *et. al.* [63] proposed their algorithm, Metric $T$, based on triads (cliques of size three) to identify local community structure in a complex network. They presented the $T$ measure as:

$$T = T_{in} \times T_{diff} \tag{4.4}$$

where

$$T_{diff} = \begin{cases} T_{in} - T_{ex}, & \text{if } T_{in} > T_{ex} \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

$$T_{in} = \frac{1}{6} \times \sum_{i \in D, j \in D, k \in D} A_{i,j} A_{j,k} A_{i,k} \tag{4.6}$$

$$T_{ex} = \frac{1}{2} \times \sum_{i \in D, j \in S, k \in S} A_{i,j} A_{j,k} A_{i,k} \tag{4.7}$$

Where $D$ is the set of nodes in the community, $S$ is the set of nodes in the shell set, and $A$ is the adjacency matrix such that $A_{i,j}$ is 1 if nodes $i$ and $j$ share an edge. They divide the $T_{in}$ score by 6 and $T_{ex}$ by 2 to prevent multiple-counting all permutations of the same triad.

We expect this method to perform better than the previous modularity based approaches because the $T$ score intuitively favors nodes that form many triads with nodes within the community and few triads with nodes outside of the community. This method finds a given node's community by placing it (or a direct neighbor of it with the highest degree) in the community. Then, it finds the best node from $S$ regarding the highest increase in $T$ score (breaking ties randomly). After a new node joins the community, the algorithm updates the shell set. Many different experiments have shown that starting from a neighbor with the highest degree, results in a better community rather than expanding with the given node itself.

Although the Metric $T$ has shown better performance than Modularity $R$, Modularity $M$, and Modularity $L$ in many cases, it suffers from a unique problem. This problem occurs while expanding the community such that starting from a neighbor node with the highest degree may result in a group of nodes that does not include the given node itself. In such cases, the algorithm declares the node does not belong to any community by returning an empty set as the output.

Despite the advantages of the approaches mentioned earlier, they share two significant issues. They are highly sensitive to their early expansion steps, such that starting with an inadequate initial node or merging bad primary nodes will result in a community with low accuracy. They are also extremely slow when applied to large dense social networks. To address these issues, we will introduce two different methods. We first present a modification to the Metric $T$ to solve its problem with its early steps. Then, by following a different path, we propose a novel method that can locally discover the communities very fast. It will be desirable and applicable to large networks, which is the main reason we prefer local approaches over global ones.

In Section 4.2, we try to find a solution to see how we should select the initial nodes to start and then continue expanding when a node is given to discover its community. In Section 4.3, we introduce a novel approach based on the weighting scheme of the *RSIWO* presented in Chapter 3. We show that we can use its weighting scheme and quality function to discover communities with only local information in a one-node-expansion model. In Section 4.4, we conduct some experiments to support our hypotheses, and finally, in Section 4.5, we make conclusions regarding local community detection methods, based on discussions in this chapter.

## 4.2 Initial Steps of Community Expansion

According to [63], Metric $T$ shows significant improvement compared to its predecessors in both terms of community search for a given node and community detection using only local information while covering the whole network. However, the Metric $T$ is not empty of flaws, so we decided to work on them to address some issues in this method and hopefully develop a better algorithm to discover communities in a network.

The first problem emerges when a node is given to search the network to find other community nodes. The experiments show that starting with a neighbor with the highest degree presumably results in a better community than the given node itself. However, there is no proof of this, and it is not always the case. A random neighbor often outperforms both, but, starting with an arbitrary node raises other concerns such as uncertainty and inaccuracy of the final results. Moreover, expansion in the early steps (after selecting an

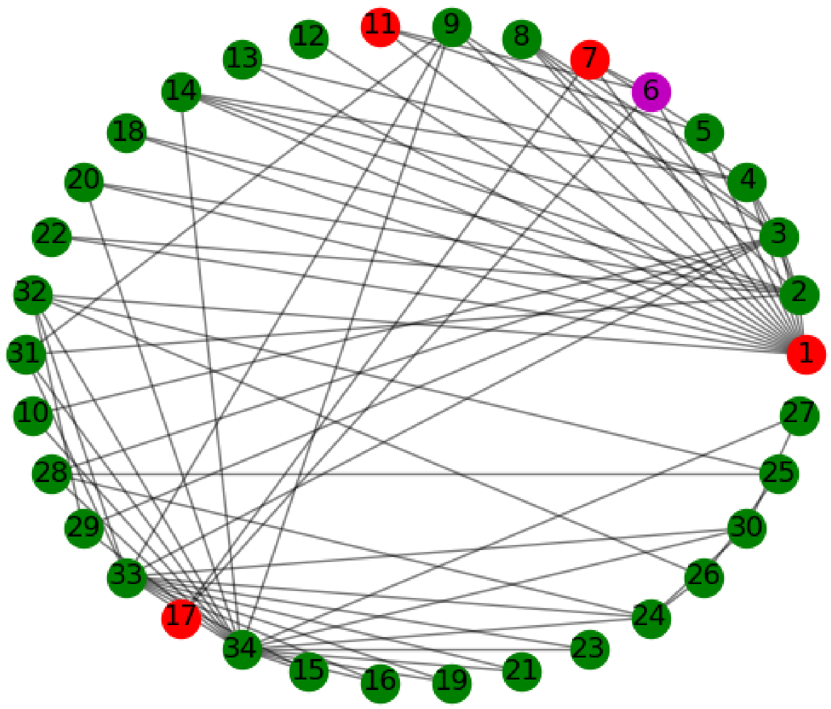


Figure 4.2: Community Search for Node 6 in Karate Club Network; the red nodes are the neighbors of the starting node.

initial node) is another critical element of discovering proper communities.

To address this problem, we propose to look more comprehensively to realize better how the community should expand. To do this, we use a new paradigm similar to the famous **n-step return** in Reinforcement Learning [64]. In this new approach, instead of looking one step ahead at a time, we look forward to $n$ steps. We select one immediate neighbor candidate leading to the highest quality after all $n$ nodes merge into the community.

We call this new improved method *the n-step method*, which uses the same quality function as Metric $T$. Metric $T$ computes the improvement for one immediate neighbor of the community. The n-step method first finds all trajectories of size $n$ (which is the algorithm's parameter) regarding the community's expansion. Then the improvements are computed for the cases when all next $n$ nodes join the community. If trajectory $T_i$ has the biggest positive improvement, the first node in that trajectory, $T_i[0]$, merges to the community. Figure 4.2 shows this procedure starting from node 6. If $n = 4$, there are many trajectories of size 4 including $T_1 = (1, 2, 3, 4)$, $T_2 = (1, 2, 4, 8)$, and $T_3 = (11, 5, 7, 17)$. Since $T_3$ makes the biggest improvement concerning the quality function, $T_3[0] = 11$ will be the node to join the community (which currently has node 6 inside it).

The main issue is finding the best initial nodes of the community in the first few steps, so we only use the n-step trajectory searching scheme in a limited number of times to avoid extra computations. We define another parameter $K$, which shows that the n-step approach should only be used before $|D| < K$. Considering that increasing $n$ and $K$ adds too much to the run-time, and they both should be at least 3, we choose $n = K = 3$ in our experiments. Other results show that bigger values improve the final result only a little. After that, the conventional Metric $T$ method would be applied by neglecting expansion's future trajectories. The rest of the algorithm is similar to Metric $T$. The Algorithm 2 shows how n-step method works.

**Algorithm 2:** The $n$-step Community Discovery Method

```
Input: Network $G$, Depth $n$, Repetition $K$, and Given Node $N_0$
Output: Community D
$D \leftarrow \{N_0\}$
$i \leftarrow 1$
$Finish \leftarrow False$
while $i \leq K$ do
    $T \leftarrow \{\}$
    for each node $N_j \in D$ do
        add all paths starting after node $N_j$ of size $n$ to $T$
    remove paths that pass any node in $D$
    for each trajectory $T_j \in T$ do
        calculate final T score if nodes in $T_j$ join D
    $T_h, T_{h,s} \leftarrow$ trajectory with highest T score and its score
    if $T_{h,s} > 0$ then
        $D \leftarrow T_h[0]$
        $i \leftarrow i + 1$
    else
        $Finish \leftarrow True$
if $Finish = False$ then
    $D \leftarrow Triads(G, D)$
return $D$
```

There are two aspects regarding this new method; 1- why the n-step method finds better communities than its predecessors. 2- why the n-step method is not the ideal approach to discover communities. We cover both sides, then suggest the algorithm formally at the end of this section.

**The *n*-step method may work better than its predecessors, because** in the simplest case (with $n = 1$ or $K = 1$), it would produce the same communities as the original Metric $T$ does. There are two user-defined parameters in this algorithm, $n$, which determines how many steps in the future should be considered, and $K$, which determines after merging how many nodes the algorithm should stop this approach and continue with Metric $T$.

By increasing either $n$ or $K$, the method looks at the network and how the community grows more extensively and profoundly, but in different ways. If we select a larger $n$, there will be more exploration in the network to add one single node to the community, whereas by choosing a larger $K$, more nodes will be added to the community in this way. So, we consequently expect to have more meaningful communities that consist of nodes with qualified connections if larger values are utilized. If we set both parameters to the minimum values ($n = 1$ and $K = 1$), it would perform similar to the original Metric $T$, which can be considered the purest form.

Considering the karate club dataset, Table 4.1 shows the results achieved by each method when they are applied to discover the nodes in the same community as node 6. We expect each method to find as many nodes as possible from the true community shown in that table. However, modularity-based methods and Metric $T$ fail to discover more than five nodes of the real community. On the other hand, the n-step method could find a set of nodes very close to the actual community because of a better initial expansion (adding node 11, then node 1, and so on, instead of node 7).

**The *n*-step method may not be the ideal approach, because** there are some crucial problems which it cannot solve. These problems are not new ones caused by the n-step method, but the issues with Metric $T$ algorithm that have never been noticed before. For example, the $T$ measure is calculated based on the internal and external cliques of size three, then before adding the

| | Discovered Community | Community Size |
|---|---|---|
| Modularity $R$ | {6, 7, 17} | 3 |
| Modularity $M$ | {6, 17, 7, 11, 5} | 5 |
| Modularity $L$ | {6, 7, 17} | 3 |
| Metric $T$ | {6, 7, 5, 11, 17} | 5 |
| $n$-step Method | {6, 11, 1, 2, 4, 3, 8, 14, 7, 5, 13, 17, 18, 20, 22, 10, 12} | 17 |
| True Community (without any order) | {1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 17, 18, 20, 22} | 16 |

Table 4.1: Performance of Different Methods for Community Search on Karate Dataset for node 6

second and third node to the community, there are no internal cliques, so we have to pick $n \geq 2$, but this is not enough. With three nodes in the community, there is only one possibility, and that is containing exactly one triangle. If we increase $n$, we will have more possibilities, making the comparison and the node selection process meaningful. On the other hand, by increasing $n$, the number of external triangles does not increase as much, which is good because we want the two figures, $n_{internal}$ and $n_{external}$ to be comparable (in the original method, $n_{external}$ exceeds the $n_{internal}$ with a high margin in the first steps).

Considering the notes mentioned above, one may think it is possible to increase $n$ to have better results. However, this is not practical for three reasons: 1- It increases the required computations much more than linearly. 2- There may not be $n$ nodes between the starting node and the border of the network, which is practically similar to using a smaller $n$ value or ignoring such trajectories. 3- Instead of considering exactly $n$ steps in the future, we can assess up to $n$ steps in the future, but it does not necessarily make sense. Because $n_{internal}$ and $n_{external}$ are not meaningfully comparable (it would be like comparing a bigger initial community and a smaller one, their tendency to merge nodes from their neighborhood would be different).

Metric $T$'s second problem is its lack of detection for nodes that do not contribute to any triangles. Many links are connected to one other node (dangling nodes as introduced by SIWO [1]) or connected to two nodes (works as a bridge between two parts of the network), or even to many nodes (works as a core but with no triangles). The Metric $T$ will have difficulty recognizing them (green nodes in Figure 4.3) because of the whole procedure is based on the number of triangles a node participates in. Such cases increase the impact of randomness on the early steps, where no internal cliques could exist.

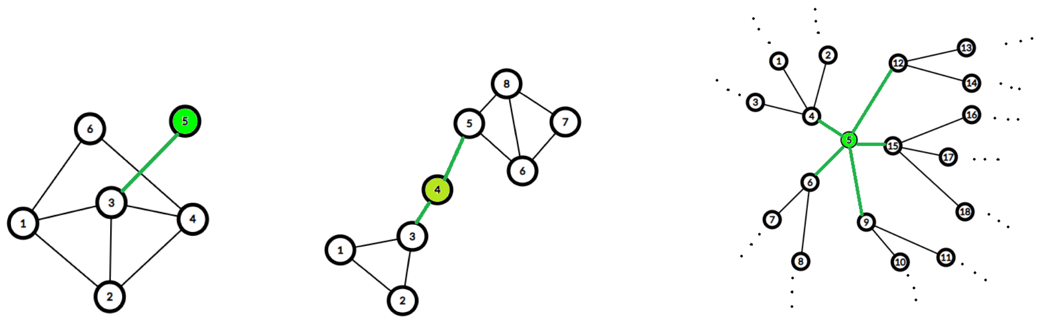


Figure 4.3: Different Cases Showing Drawbacks of Metric $T$

The third problem is these algorithms (both Metric $T$ and $n$-step method) operate very slowly if applied on large networks. Although increasing $n$ and $K$ theoretically improves the performance by producing more accurate communities, considering the required time and the amount of improvement, there is no point in using it because the main reason to prefer local methods over global methods is to save time. It needs to be mentioned that the results are not as accurate as of the results of global approaches.

## 4.3 Local SIWO

This new method to discover the community of a given node or all communities with only using the local information is called Local SIWO. This novel method is designed to address the problems mentioned earlier of the well-known modularity-based methods and the issues of the Metric $T$ and the $n$-step method. To do this, we propose to use the weighting scheme of the RSIWO$_1$ (given in Equation 3.4) to find strong and weak edges in the network as the community grows, then investigate it to discover the communities. All strength values are initially zero, and they will be updated during the community detection process. Algorithm 3 shows how the local weights are assigned to the edges of the network.

**Algorithm 3:** Assigning Strength for Edges in Local SIWO

```
Input: Network $G$, and Given Node $N_0$
Output: Network $G$ with strength attribute for edges of node $N_0$
$A \leftarrow$ neighbors of node $N_0$
for each node $N_j \in A$ do
    $w_{0,j} \leftarrow$ number of mutual neighbors of nodes $N_0$ and $N_j$
    $B \leftarrow$ neighbors of node $N_j$
    for each node $N_k \in B$ do
        $w_{j,k} \leftarrow$ number of mutual neighbors of nodes $N_j$ and $N_k$
    $w_{j,max} \leftarrow \max_k\{w_{j,k}\}$
$w_{0,max} \leftarrow \max_j\{w_{0,j}\}$
for each node $N_j \in A$ do
    $s_{0,j} \leftarrow \frac{w_{0,j}}{2} \times (\frac{1}{w_{0,max}} + \frac{1}{w_{j,max}})$
    assign strength $s_{0,j}$ to edge $e_{0,j}$
return $G$
```

We initially place node $v_X$ in the community, then all edges that are incident to node $v_X$ should update their strength. After that, the next candidates are determined (which are in the community's immediate neighborhood). Then, the strengths values of all edges that are connected to the candidate nodes will be updated. In each iteration, a node from the candidate set joins the community if it maximally increases the sum of strengths of the edges incident to the nodes inside the community. The algorithm stops adding nodes to the community if the most significant improvement is a negative value.

There are three main reasons why we use Local SIWO:

1- We want to know if Local SIWO can work adequately like how RSIWO outperforms the existing global community detection approaches. We expect its novel weighting scheme and new quality function can be used for local methods as well.

2- Since this method does not count the edges many times, theoretically, it should be much faster than any other previous way. We need to visit each edge once to measure the parameters required for computing the strength, and we need to visit and calculate each edge only once for an update. So we need to do computations on every edge two times.

3- Size of the discovered community does not change the tendency of nodes to merge into the community as much as it does for the other approaches. For example, in the Metric $T$, no internal triangle exists before a particular time during the process. The number of external triangles significantly decreases after the community becomes large enough, which both affect the T score and the tendency of the community to merge new nodes to it.

## 4.4 Experiments

Most methods in this field usually search a network only once to find all nodes with the same community index as the given node. However, in our experiments, it is also essential to measure the performance of different methods regarding a community detection task. We need to find all communities of the network with only local access to the network's information. Thus, we should adjust these algorithms according to our questions. However, there are some gray areas in which we tried different settings to achieve the best results. For example, although removing the nodes in the discovered communities presumably leads to better results, the experiments show that it is not always the case. The rare inadequate results are probably because of selecting the initial nodes or the deleted edges for the next round of community search. Considering this, to have a consistent setting in all experiments, we remove the discovered communities' nodes from the set of candidates any time we need to update it. This plan works for Modularity $R$, Modularity $M$, and Modularity $L$. However, either approach may work better depending on the given network for Metric $T$. So we conduct two experiments, first with only ignoring the nodes in the discovered communities while looking for the best next candidate and second with removing such nodes entirely from the network.

There are different experiments on real-world and synthetic networks to validate and support our new method and hypotheses. The first experiment is done on three small real-world networks described in Table 4.2. These small networks are well-known, and we only use them to show the credibility of the methods. The results of these experiments are shown in Tables 4.3 to 4.5. We also experimented on synthetic networks generated by LFR. The parameters are initially set to $n = 1000$, $\tau_1 = 20$, $\tau_2 = 10$, $\mu = 0.1$, and $min\_deg = 10$,

|  | $\|V\|$ | $\|E\|$ | $\|C\|$ | $Q$ |
|---|---|---|---|---|
| Football Network [9] | 115 | 613 | 12 | 0.553 |
| Zachary's Karate Club [59] | 34 | 78 | 2 | 0.358 |
| Political Books Network [60] | 105 | 441 | 3 | 0.414 |

Table 4.2: Real Networks in the Local Community Detection Experiments

|  | $C/C_r$ | Q | NMI | Run Time (s) |
|---|---|---|---|---|
| Modularity $R$ | 0.92 | 0.575 | 0.892 | 0.044 |
| Modularity $M$ | 0.92 | 0.575 | 0.892 | 0.034 |
| Modularity $L$ | 0.92 | 0.575 | 0.892 | 0.063 |
| Metric $T$ | 0.83 | 0.544 | 0.876 | 0.113 |
| $n$-step | 1.08 | 0.510 | 0.845 | 3.424 |
| **Local SIWO** | **1.0** | **0.601** | **0.926** | **0.034** |

Table 4.3: Experiments on Football Network for Community Detection

unless told otherwise in each experiment set up. We wanted to know how different methods perform on different types of large synthetic networks. We conducted two sets of experiments considering both community search for one given node and community detection for the whole network.

### 4.4.1 Evaluate Different Network Sizes

Some results for the $n$-step method are missing because it takes so much time that it is practically infeasible to be performed as a local community detection method. So although it may improve the Metric $T$ in some cases, it adds so much to the run time that it does not make sense anymore to be used. In this experiment, we set the parameters of $n$-step method $n = 3$ and $K = 3$, which are relatively small values selected in favor of the run time.

First, we show the synthetic networks in Table 4.6. As can be seen, we gen-

|  | $C/C_r$ | Q | NMI | Run Time (s) |
|---|---|---|---|---|
| Modularity $R$ | 1.5 | 0.354 | 0.631 | 0.007 |
| Modularity $M$ | 1.5 | 0.374 | 0.608 | 0.005 |
| Modularity $L$ | 2.5 | 0.375 | 0.558 | 0.006 |
| Metric $T$ | 1.0 | 0.371 | 0.837 | 0.008 |
| $n$-step | 1.0 | 0.371 | 0.837 | 0.120 |
| **Local SIWO** | **1.0** | **0.371** | **0.837** | **0.006** |

Table 4.4: Experiments on Karate Network for Community Detection

|  | $^{C}/_{C_r}$ | Q | NMI | Run Time (s) |
|---|---|---|---|---|
| Modularity $R$ | 2.0 | 0.443 | 0.454 | 0.073 |
| Modularity $M$ | 1.0 | 0.445 | 0.534 | 0.204 |
| Modularity $L$ | 2.33 | 0.440 | 0.435 | 0.108 |
| Metric $T$ | 1.0 | 0.471 | 0.451 | 0.135 |
| $n$-step | 1.0 | 0.463 | 0.435 | 3.346 |
| **Local SIWO** | **1.33** | **0.497** | **0.554** | **0.038** |

Table 4.5: Experiments on Political Books Network for Community Detection

|  | $\|V\|$ | $\|E\|$ | $\|C\|$ | avg. deg. | $Q$ |
|---|---|---|---|---|---|
| Synthetic Network A1 | 100 | 585 | 9 | 11.700 | 0.744 |
| Synthetic Network A2 | 500 | 2898 | 46 | 11.592 | 0.830 |
| Synthetic Network A3 | 1000 | 5804 | 93 | 11.608 | 0.838 |
| Synthetic Network A4 | 5000 | 28855 | 467 | 11.542 | 0.853 |
| Synthetic Network A5 | 10000 | 57767 | 928 | 11.553 | 0.851 |

Table 4.6: Synthetic Networks in the Local Community Detection Experiments

erated networks with similar average degree. The size of the networks increases gradually while keeping good community structures as the $Q$-modularity values are high. We expect to see similar results for modularity based approaches, whereas the Metric $T$ should be better than them. We first show how they performed in a community detection task by comparing their run-time, and other measures obtained from the detected communities, such as $^{C}/_{C_r}$, $Q$ modularity value, and NMI score.

As the experiments show, the novel Local SIWO outperforms other methods in every test. We picked the winner method in each trial based on a realistic number of detected communities, the highest $Q$ and NMI values, and the shortest run-time. In a case that $Q$ and NMI contradict each other, we prefer the algorithm with higher NMI as it is directly calculated using the

|  | $^{C}/_{C_r}$ | Q | NMI | Run Time |
|---|---|---|---|---|
| Modularity $R$ | 1.0 | 0.812 | 0.984 | 0.035 |
| Modularity $M$ | 1.0 | 0.812 | 0.984 | 0.035 |
| Modularity $L$ | 1.0 | 0.812 | 0.984 | 0.081 |
| Metric $T$ | 1.0 | 0.743 | 1.0 | 0.069 |
| **Local SIWO** | **1.0** | **0.743** | **1.0** | **0.028** |

Table 4.7: Experiments on Synthetic Network A1 for Community Detection

| | ${}^{C}/{}_{C_r}$ | Q | NMI | Run Time |
|---|---|---|---|---|
| Modularity $R$ | 1.0 | 0.829 | 0.978 | 0.292 |
| Modularity $M$ | 0.98 | 0.831 | 0.977 | 0.233 |
| Modularity $L$ | 1.0 | 0.829 | 0.978 | 0.354 |
| Metric $T$ | 1.0 | 0.829 | 1.0 | 0.455 |
| **Local SIWO** | **1.0** | **0.829** | **1.0** | **0.194** |

Table 4.8: Experiments on Synthetic Network A2 for Community Detection

| | ${}^{C}/{}_{C_r}$ | Q | NMI | Run Time |
|---|---|---|---|---|
| Modularity $R$ | 0.99 | 0.847 | 0.988 | 0.978 |
| Modularity $M$ | 0.98 | 0.858 | 0.988 | 0.900 |
| Modularity $L$ | 0.99 | 0.847 | 0.988 | 0.1.106 |
| Metric $T$ | 1.0 | 0.838 | 1.0 | 1.389 |
| **Local SIWO** | **1.0** | **0.838** | **1.0** | **0.645** |

Table 4.9: Experiments on Synthetic Network A3 for Community Detection

| | ${}^{C}/{}_{C_r}$ | Q | NMI | Run Time |
|---|---|---|---|---|
| Modularity $R$ | 0.98 | 0.855 | 0.992 | 71.262 |
| Modularity $M$ | 0.98 | 0.861 | 0.992 | 75.181 |
| Modularity $L$ | 0.98 | 0.854 | 0.992 | 68.863 |
| Metric $T$ | 1.0 | 0.851 | 0.999 | 65.899 |
| **Local SIWO** | **1.0** | **0.853** | **1.0** | **41.748** |

Table 4.10: Experiments on Synthetic Network A4 for Community Detection

| | ${}^{C}/{}_{C_r}$ | Q | NMI | Run Time |
|---|---|---|---|---|
| Modularity $R$ | 0.98 | 0.851 | 0.991 | 555.473 |
| Modularity $M$ | 0.97 | 0.860 | 0.991 | 511.239 |
| Modularity $L$ | 0.98 | 0.852 | 0.991 | 515.252 |
| Metric $T$ | 0.99 | 0.849 | 0.999 | 503.428 |
| **Local SIWO** | **1.0** | **0.851** | **1.0** | **287.674** |

Table 4.11: Experiments on Synthetic Network A5 for Community Detection

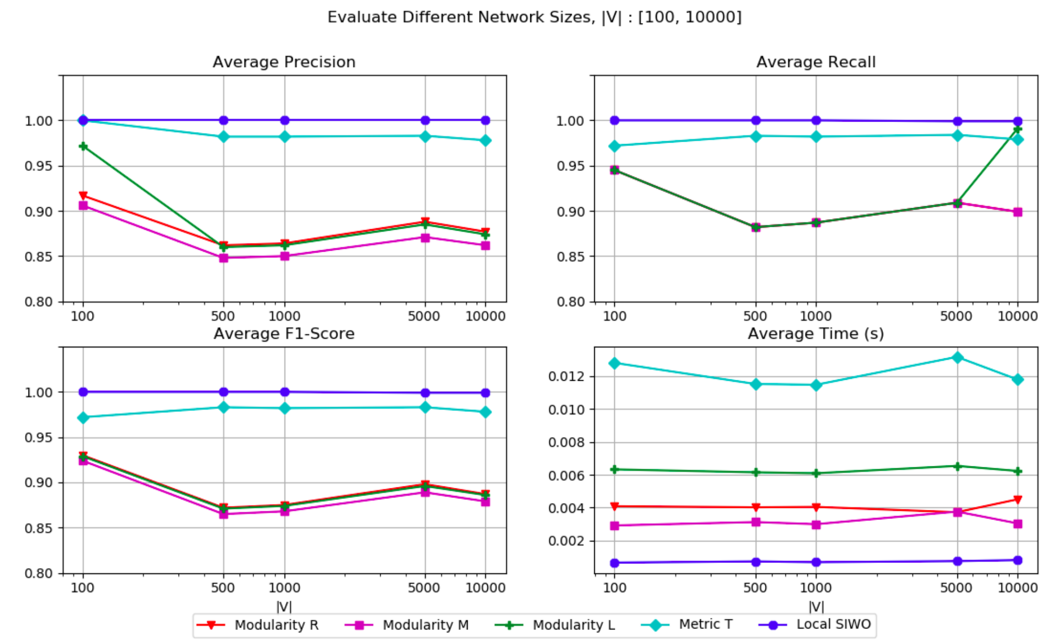


Figure 4.4: Accuracy Results of Experiment 1: Community Discovery

ground-truth community. One important observation is to notice how well Local SIWO performed when the networks' size increases in different experiments. This vital feature of our new method makes it a good fit for cases when the network's size is too large to be analyzed using global approaches.

Figure 4.4 also shows the accuracy of different methods when used to discover the community of a single node of the network. In each test, nodes of the graph are given to the methods separately. Then, we calculated the precision, recall, and F1-score for them. The reported measures are the average accuracy values over all nodes of each network. As you can see, as the network's size increases, most of the algorithms could not maintain their excellent results. On the other hand, the run-times considerably increase when networks got bigger. Among all methods, Local SIWO shines, as its accuracy measures are almost perfect regardless of the given network's size. Its run-time is another bright point in this comparison, as it has not been affected as much as the other methods did.

| | $\|V\|$ | $\|E\|$ | $\|C\|$ | avg. deg. | $Q$ |
|---|---|---|---|---|---|
| Synthetic Network B1 | 1000 | 5776 | 95 | 11.552 | 0.843 |
| Synthetic Network B2 | 1000 | 11730 | 46 | 23.460 | 0.813 |
| Synthetic Network B3 | 1000 | 17800 | 30 | 35.600 | 0.799 |
| Synthetic Network B4 | 1000 | 23641 | 23 | 47.282 | 0.784 |
| Synthetic Network B5 | 1000 | 29231 | 19 | 58.462 | 0.774 |
| Synthetic Network B6 | 1000 | 35651 | 15 | 71.302 | 0.757 |

Table 4.12: Synthetic Networks of the Second Experiments

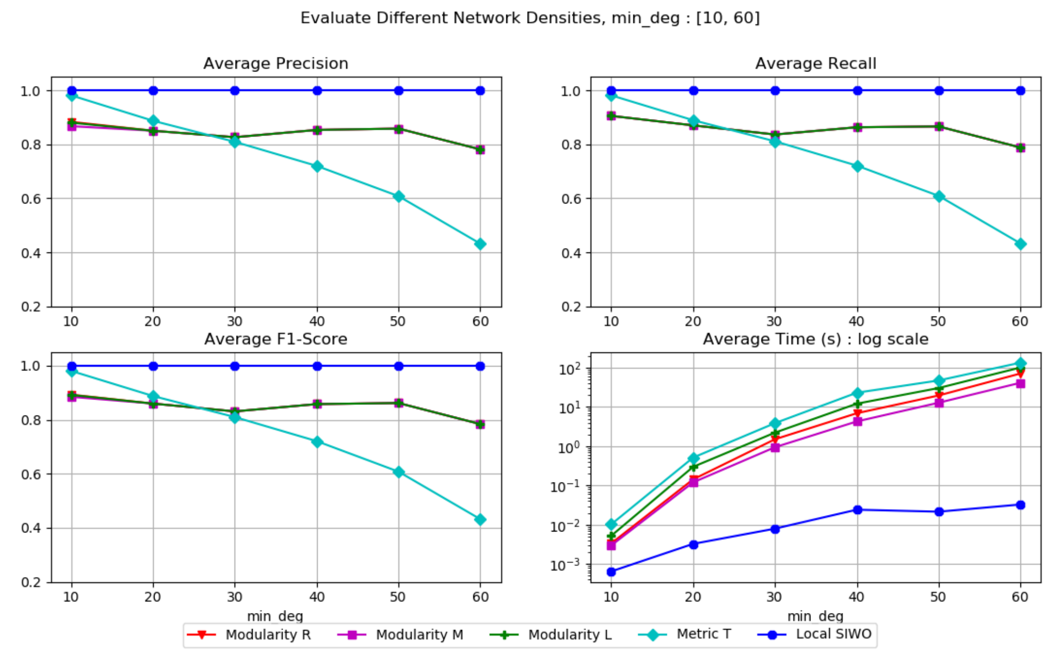


Figure 4.5: Accuracy Results of Experiment 2: Community Discovery

## 4.4.2 Evaluate Different Network densities

In this experiment, we evaluate the performance of different methods using networks with the given average degrees in Table 4.12. For this purpose, we generated six different networks with the same number of nodes using LFR benchmark. However, by changing the parameter that determines the minimum degree of a node in the network, we increase the number of edges and, consequently, the network's density. Table 4.12 describes the networks used in this experiment.

In the community discovery task, as Figure 4.5 shows, modularity based methods are practically the same. However, the Metric $T$ performs better than

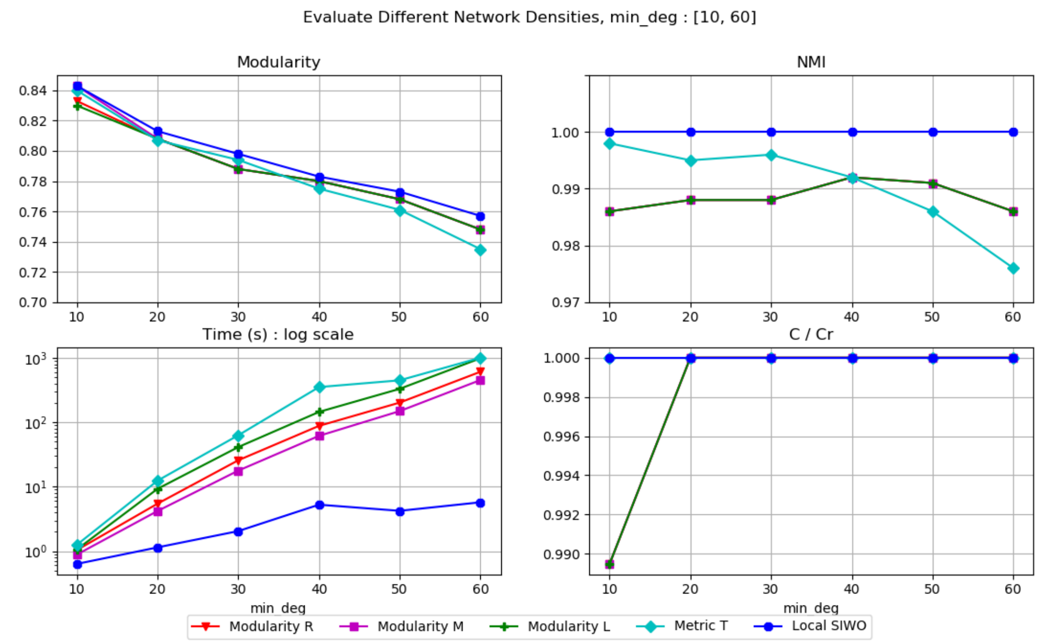


Figure 4.6: Accuracy Results of Experiment 2: Community Detection

them when the network is more sparse and performs poorly when it is denser. In all cases, Local SIWO outperforms the other approaches and finishes the task in a much shorter time. Considering these experiments, we can claim that Local SIWO is a promising method in community search applications, even on large and dense networks.

In the community detection task, as Figure 4.6 shows, due to the excellent community structure of the networks, any method could find consistent results. However, Local SIWO can find communities with perfect matches compared to the ground truth, which results in higher Q-modularity value. The run-time is still by far in favor of Local SIWO. This method is able to find all communities of a network with tens of thousands of edges in a few seconds, whereas others need more than some minutes to do the same job.

### 4.4.3 Evaluate Different Community Structure Quality

In this experiment, we evaluate the performances of different methods when networks with different mixing parameters $\mu$ are given. The mixing parameter determines the average ratio of the edges between communities to all edges for

|  | $\|V\|$ | $\|E\|$ | $\|C\|$ | avg. deg. | $Q$ | $\mu$ |
|---|---|---|---|---|---|---|
| Synthetic Network C1 | 1000 | 5783 | 93 | 11.566 | 0.843 | 0.10 |
| Synthetic Network C2 | 1000 | 6070 | 92 | 12.140 | 0.682 | 0.15 |
| Synthetic Network C3 | 1000 | 6063 | 95 | 12.126 | 0.679 | 0.20 |
| Synthetic Network C4 | 1000 | 6053 | 95 | 12.106 | 0.656 | 0.25 |
| Synthetic Network C5 | 1000 | 6325 | 93 | 12.650 | 0.534 | 0.30 |
| Synthetic Network C6 | 1000 | 6425 | 95 | 12.850 | 0.402 | 0.35 |
| Synthetic Network C7 | 1000 | 6455 | 94 | 12.910 | 0.396 | 0.40 |

Table 4.13: Synthetic Networks of the Third Experiments

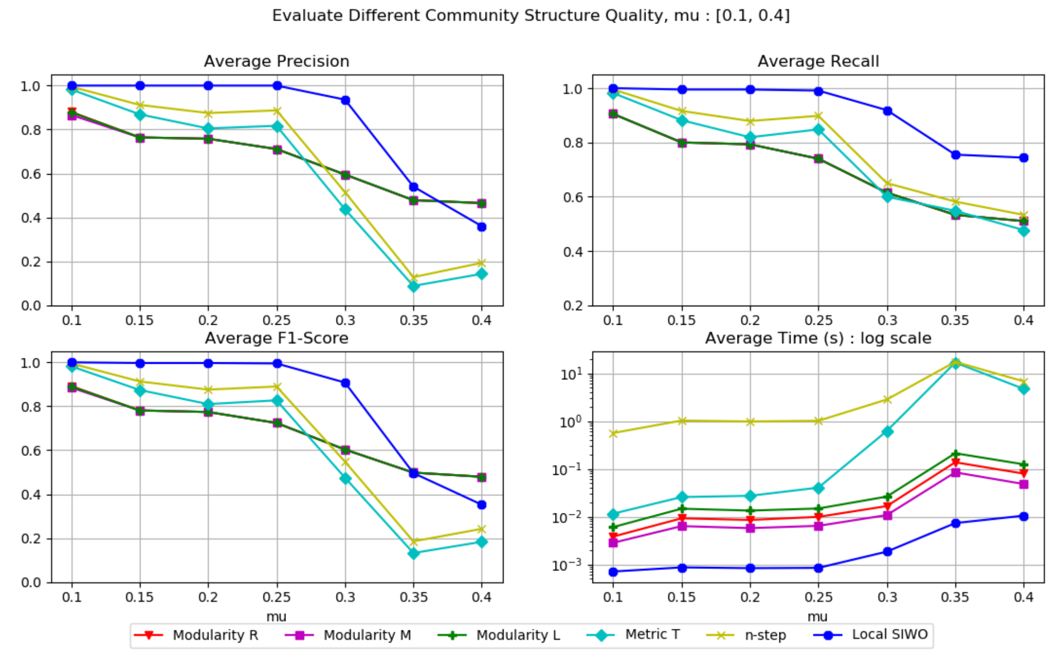


Figure 4.7: Accuracy Results of Experiment 3: Community Discovery

each node. We expect to have networks with higher quality communities when $\mu$ has smaller values. We generated seven different networks with the same number of nodes and a minimum degree. However, by changing the $\mu$ in each network, we increased the number of edges that lie between communities, and consequently, the quality of community structure drops. Table 4.13 describes the networks used in this experiment.

In the community discovery task, as Figure 4.7 shows, modularity based methods are practically the same. However, Metric $T$ performs better than them when $\mu < 0.3$ and performs poorly when $\mu$ is larger than that. In this experiment, we can see the $n$-step method; it is better than the Metric $T$

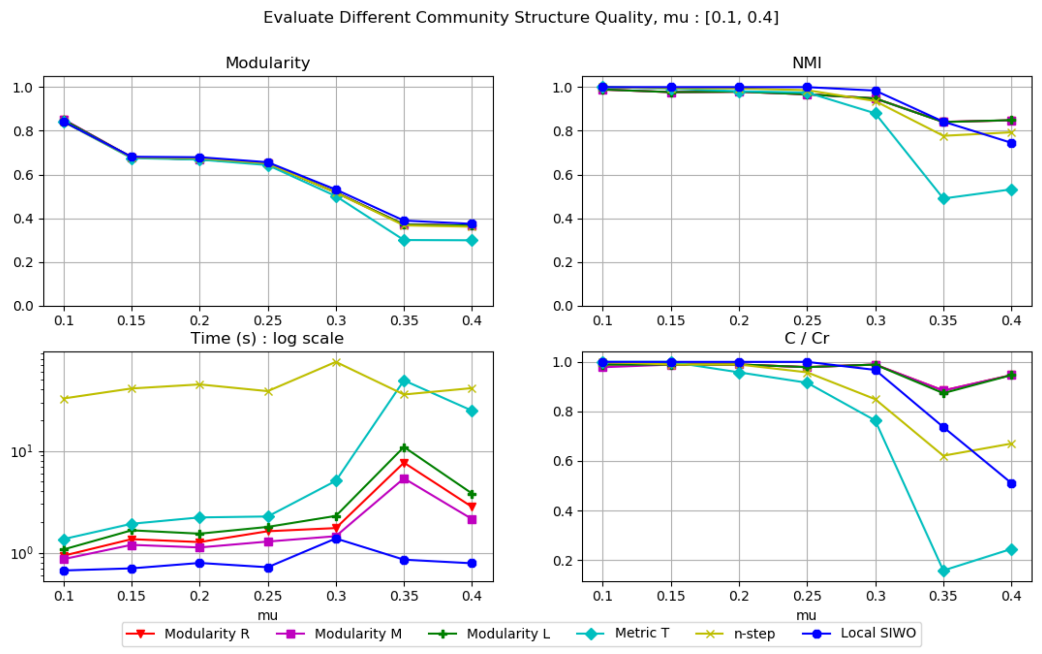


Figure 4.8: Accuracy Results of Experiment 3: Community Detection

in all cases. However, this method's average required time is much higher, so there is a trade-off when we want to determine the better approach. In most cases, Local SIWO outperforms the others regarding the performance measures and finishes the task in a much shorter time. However, it could not perform better than modularity based approaches when $\mu > 0.35$, which means there are fewer meaningful communities in the network. Considering these experiments, we can claim that Local SIWO is a promising method in community search applications, even on networks that do not have suitable community structures.

Figure 4.8 shows in the community detection task, even though modularity based methods performed similarly, modularity $M$ is the fastest. On the other hand, Local SIWO can find communities that result in the highest modularity, and they almost perfectly match the ground truth information. Local SIWO's run-time is better than the others, which is a significant advantage.

## 4.5 Conclusion

In local community discovery and detection, we cannot access the whole network altogether. This previously raised many problems, including finding communities that suffer from low accuracy, merging outliers to the communities, etc. Local SIWO does not seem to suffer from such limitations based on the results in this chapter. The accuracy measures are almost perfect for different test-case scenarios, regardless of size, density, or even the given network's community structure quality. This new local method proves itself reliable when we need to analyze huge communities and cannot use the whole graph simultaneously.

In the previous chapter, we showed that RSIWO, our novel global method, outperforms the best-known and state-of-the-art methods in community detection tasks. In this chapter, we demonstrated that the same weighting scheme and quality function (with small alterations) could be applied to find communities using only local information in a short time with reliable accuracy measures.

The $n$-step method is theoretically better than Metric $T$, and different experiments prove that as well. However, an essential feature of a local approach is its speed. The $n$-step process suffers from a long computation time. When comparing modularity-based methods, we can claim that modularity $M$ is better than the others. Because although the communities it detects are almost the same as modularity $R$ and modularity $L$, the run time is in favor of modularity $M$.

Comparing the run-times, we can see the novel Local SIWO is extremely fast, and there are three reasons for that:

- **Fewer Edge Visiting:** While other methods have to revisit each edge that is out of the communities multiple times; Local SIWO only needs to visit each edge twice to update the strength value. Since we consider this a new attribute for edges of the graph, once it is processed, we do not need to recalculate again.

- **Different Calculation Process:** Modularity based methods need to consider edges inside the communities and edges that connect the community to the rest of the network. In contrast, the local SIWO only needs to sum up the edges' strength values that connect the shell set to the community's boundary nodes. This difference is another reason why local SIWO is so much faster. On the other hand, the Metric $T$ needs to count the number of triangles every time it looks for a new node to merge into the community, which is computationally much more expensive. The n-step method not only does that but also has some brute-force search mechanism in its first $K$ steps, which puts it on the last ranking regarding the shortest run-time.

- **Smarter Optimization Techniques:** As the network's information is exposed gradually, any method needs to recalculate the same values repeatedly. Different methods can be implemented by some tweaks and optimization techniques to reduce such repetitions. For example, in the Metric $T$'s paper, the authors suggested an incremental counting system that significantly reduces the run-time. We also added another technique in the triangle-counting process that reduced the required time three times. Nonetheless, it cannot perform as fast as Local SIWO.

Although we demonstrated that Local SIWO is by far the fastest, there may be better implementations of each algorithm that makes them faster. However, speed has never been our main concern in these experiments. The most important achievement of this chapter is to propose a novel approach with a completely different viewpoint to the problem of local community detection that can find correct outcomes.

# Chapter 5

# Community Detection in Uncertain Networks

## 5.1 Motivation

In the last chapter, we saw how we could use local community discovery algorithms to find all the nodes that belong to the same community as a given node in a network with deterministic edges. A network with deterministic edges means that we are entirely sure whether a link between two nodes exists. It is also possible to detect all of the communities of such a network using only local information. This chapter discusses how we can discover and detect communities for a network with uncertain edges. This means that each edge of the network, i.e., between nodes $v_i$ and $v_j$, is assigned a probability value $p_{i,j}$ where $0 \leq p_{i,j} \leq 1$.

The uncertainty in the edges' probabilities may occur because of the observations, measurement errors, report mistakes, etc. For these reasons, reported networks might include edges with probabilities mostly close to the extremes of the spectrum of the probability values $[0, 1]$. However, probability values may be around 0.5. A probability value close to 0 shows that the edge does not exist in the network, and the errors cause a low probability. For the high probability edges, we can infer the opposite.

Local SIWO already demonstrated promising results in cases of community discovery and detection using only local information of the known part of a network. We develop an uncertain version of SIWO such that it would be

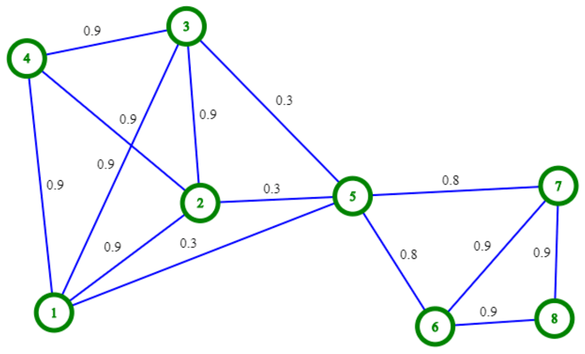


Figure 5.1: An Example to Show the Problem of Using Local SIWO for Uncertain Networks

capable of taking edge probabilities into account. However, first, we show why Local SIWO is not enough to handle the probabilities. Figure 5.1 depicts an uncertain network with the likelihood of each edge on top of it, consisting of two cliques of size 3 and 4, where they are connected to node 5 with 3 and 2 edges, respectively. Local SIWO would find nodes $\{v_1, v_2, v_3, v_4, v_5\}$ in one community and nodes $\{v_6, v_7, v_8\}$ in another community, whereas we expect two communities of size 4, such that node 5 belongs to the other community. This is because we are evidently more confident about the edges $e_{5,6}$ and $e_{5,7}$ rather than edges $e_{1,5}$ $e_{2,5}$, and $e_{3,5}$. In the following sections, we introduce a metric to systematically measure the strength of found communities and show why two communities of size 4 are more meaningful than what Local SIWO offers as the network's detected communities.

In Section 5.2, we first review Modularity *UR* and *UR+K*, two previous methods for community detection on uncertain networks, then propose *Uncertain SIWO*, our new uncertain community detection algorithm, based on Local SIWO introduced in 4.3. In Section 5.3, we go over an old uncertain network generator and discuss its drawbacks, then suggest a new approach to synthesize networks with probabilistic edges. In Section 5.4, we conduct some experiments to evaluate our proposed algorithm. Finally, in Section 5.5, we make conclusions regarding the local community detection when the edges of the network are uncertain, based on the discussions in this chapter.

## 5.2 Local Community Discovery for Networks with Uncertain Edges

### 5.2.1 Review of Previous Methods

Based on some local community detection algorithms for deterministic networks, proper methods are devised to find communities in uncertain networks. In Chapter 4, we review some deterministic approaches, including local modularity $R$. Inspired by this, Zhang *et. al.* [41] introduced two methods to discover communities in networks with edge uncertainty locally; local modularity *UR* and its revised version, *UR+K*.

The main idea of modularity *UR* is to convert the uncertain edges into a deterministic scenario using probability values for the edges. Then, after partitioning the network to the detected community $D$ (consisting of $C$ and $B$, refer to Section 4.1), the neighborhood $S$ and the unknown part of the network, the uncertain modularity formula can be defined as follows:

$$UR = \frac{\mathbb{E}(B_{in\_edge})}{\mathbb{E}(B_{in\_edge}) + E(B_{out\_edge})} \tag{5.1}$$

where $E(B_{in\_edge})$ is the expected number of edges that connect boundary nodes and the nodes in $D$ and $E(B_{out\_edge})$ is the expected number of edges that connect boundary nodes and the nodes in $S$. These two terms can be expressed as follows:

$$\mathbb{E}(B_{in\_edge}) = \frac{1}{2} \sum_{v_i, v_j \in B, i \neq j} p_{i,j} + \sum_{v_i \in B, v_j \in C} p_{i,j} \tag{5.2}$$

$$\mathbb{E}(B_{out\_edge}) = \sum_{v_i \in B, v_j \in S} p_{i,j} \tag{5.3}$$

In each step to add a new node to the community, after the modularity values are calculated using this new formula, the method is similar to the local modularity $R$. The algorithm uses the uncertain modularity measure *UR* instead of the original $R$ in Clauset [33] method. However, this new uncertain method has the same problem that the deterministic modularity $R$ suffers from, which is the early steps in expanding and discovering outliers

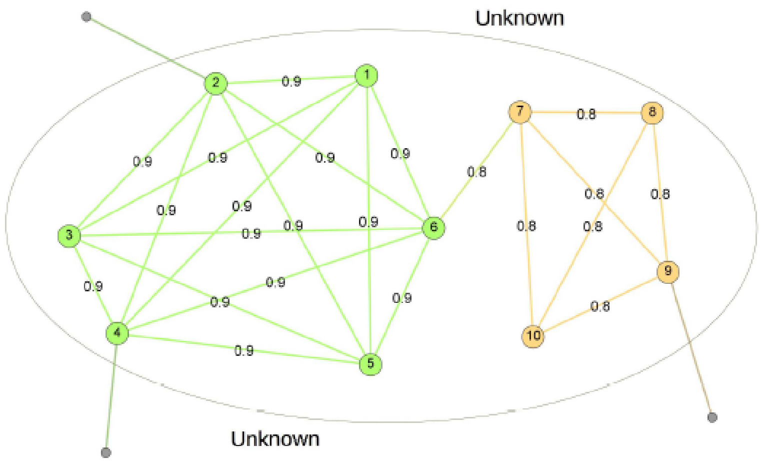


Figure 5.2: An example showing the problem when only using $UR$ to find the local community [41]

into the community. Similar to that method, $UR$ may expand the community in the wrong direction and ultimately declares a wrong group of nodes as the discovered community. Zhang exposed this issue with an example shown in Figure 5.2 in which starting from node $v_6$ leads to discovering false communities $\{v_1, v_2, v_3, v_4, v_5\}$ and $\{v_6, v_7, v_8, v_9, v_{10}\}$. These are not the correct communities because nodes $v_1$ to $v_6$ compose a 6-vertex clique and nodes $v_7$ to $v_{10}$ compose another clique of size 4. Although the community structure is quite simple, modularity $UR$ is unable to discover the actual communities correctly. There are two ways to handle this issue; the first is to regard such start nodes as periphery nodes and report no community for them. This ambiguity is not acceptable as there may exist more periphery nodes in more complex networks. The need to effectively address this issue is inevitable so that the real community members could be detected for any node in the network. Zhang *et. al.* [41] suggested an amendment to modularity $UR$ and called it $UR$+$K$ to overcome the problem of periphery nodes.

The new measure $K$ takes the nodes in $S$ into account while paying attention to nodes in $B$. It aims to measure the relationship between a candidate node $v_i$ and the discovered community $D$, which can be defined as follows:

$$K_i = \mathbb{E}(N_{i,in\_edge}) + \mathbb{E}(N_{i,shell\_edge}) \tag{5.4}$$

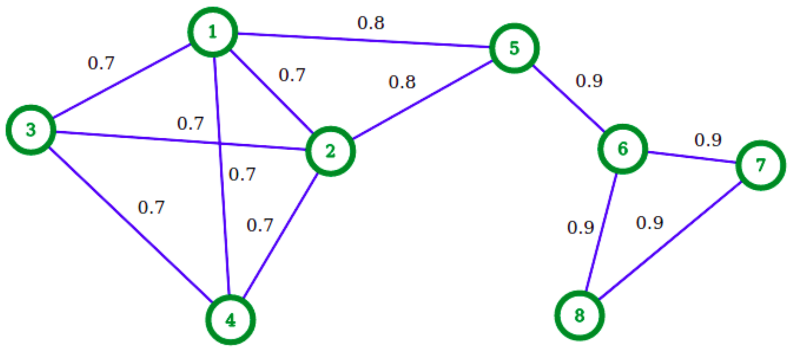


Figure 5.3: Am Example to Show the Problem of Using $R$ and $UR$ for Uncertain Networks

where $\mathbb{E}(N_{i,in\_edge})$ is the expected number of edges that connect node $v_i$ and the nodes in $D$ and $\mathbb{E}(N_{i,shell\_edge})$ is the expected number of edges that connects node $v_i$ and the other nodes in $S$. They can be represented as follows:

$$\mathbb{E}(N_{i,in\_edge}) = \sum_{v_j \in D} p_{i,j} \tag{5.5}$$

$$\mathbb{E}(N_{i,shell\_edge}) = \sum_{v_j \in S} p_{i,j}[1 - \prod_{v_k \in D}(1 - p_{j,k})] \tag{5.6}$$

This new parameter $K$ plays two critical roles; to determine which node from the shell set to join the discovered community $D$ in the first few steps and break ties when modularity $UR$ ties in future phases. The $UR$+$K$ algorithm works by placing the start node in the community $D$, and then, it sorts all candidate nodes (nodes in $S$) based on their $K$ or $UR$ measures for the first few steps and the later steps respectively. The algorithm picks the first node and adds it to the community until no node in $S$ can increase its modularity. The number of steps that $K$ supersedes $UR$ in ranking the candidate nodes is a hyper-parameter demonstrated by $\lambda$, which can be set to $\lambda = 3$ based on the experiments they conducted.

Although this new method addresses a crucial drawback of the former $UR$ algorithm and can handle the edges' probability values, it still cannot perform well on simple networks such as the example shown in Figure 5.3. We expect two important features; a good community discovery algorithm should ideally

| Start Node | Modularity *UR* | Modularity *UR+K* | Uncertain SIWO |
|---|---|---|---|
| $v_1$ | {1, 2, 3, 4, 5} | {1, 2, 3, 4} | {1, 2, 3, 4, 5} |
| $v_2$ | {1, 2, 3, 4, 5} | {1, 2, 3, 4} | {1, 2, 3, 4, 5} |
| $v_3$ | {1, 2, 3, 4} | {1, 2, 3, 4} | {1, 2, 3, 4, 5} |
| $v_4$ | {1, 2, 3, 4} | {1, 2, 3, 4} | {1, 2, 3, 4, 5} |
| $v_5$ | {1, 2, 3, 4, 5, 6, 7, 8} | {1, 2, 3, 4, 5} | {1, 2, 3, 4, 5} |
| $v_6$ | {6, 7, 8} | {6, 7, 8} | {6, 7, 8} |
| $v_7$ | {6, 7, 8} | {6, 7, 8} | {6, 7, 8} |
| $v_8$ | {6, 7, 8} | {6, 7, 8} | {6, 7, 8} |

Table 5.1: Performance of Different Methods for Community Search on the Example Network in Figure 5.3

be able to find the desired group of nodes, which connect to each other more densely, correctly and entirely, regardless of which node in that particular community starts the expansion.

Table 5.1 shows the results of the previous methods, modularity *UR*, modularity *UR+K*. The third column depicts the communities discovered by *Uncertain SIWO*, which is our proposed method that we introduce in the next section. The first two could not find the true communities when nodes $v_1$ to $v_4$ started community expansion. Modularity *UR* also fails to find the true community when node $v_5$ is the start node and put all nodes in a single community. This problem originates from the inability to initialize the expansion in the right direction, leading to different community members when other nodes are used for the start. So, paying attention to the shell set alongside the sharpness of the boundary set (which *UR+K* claims to take into account) is not enough, as this simple network and the discovered communities in Table 5.1 illustrate. We have already demonstrated that similar approaches would fail in a deterministic scenario in Chapter 4 (refer to different modularity-based methods discussed in Section 4.1). So, we propose a new algorithm based on *Local SIWO* to find communities in the uncertain networks and address the problem derived from the deterministic local approaches.

### 5.2.2 Introducing the Uncertain SIWO

We have already shown that Local SIWO outperforms many other local community detection algorithms in Chapter 4. It could effectively address the problem of initial steps while the community is expanding and dropping the outliers to not join the community in a considerably less amount of time. Inspired by that novelty, we propose a new method to detect and discover communities in large uncertain networks using only local information.

The major transformation in the *Uncertain SIWO*, which we call *USIWO* after this, is how to compute the strength values. Studying the deterministic networks taught us exploiting cliques would lead to more meaningful local communities than when an algorithm only uses edges separately (like how modularity $R$ and $L$ find communities). Thus, we used the idea of counting the number of shared neighbors of each two nodes, i.e., $v_i$ and $v_j$, to calculate the strength value of the edge $e_{i,j}$ that connects them. This quantity is the number of size three cliques that both nodes $v_i$ and $v_j$ belong to. In a probabilistic scenario, for any node $v_k$ to be the shared neighbor of both nodes $v_i$ and $v_j$, both edges $e_{i,k}$ and $e_{j,k}$ are important. Thus, instead of counting the number of shared neighbors, we need to calculate their expected number, equal to half of the expected number of the edges that connect any shared neighbor $v_k$ to the end-nodes of the edge $e_{i,j}$. This can be defined as follows:

$$s_{i,j} = \mathbb{E}(N_{i,j}) = \frac{1}{2} \sum_{\substack{k \neq i, k \neq j, \\ v_k \in V}} (p_{i,k} + p_{j,k}) \tag{5.7}$$

Algorithm 4 shows how *USIWO* uses the edges' probabilities to calculate the strength values for the edges. *USIWO* starts with placing a given node into the community $D$, and all its neighbors, which are any edges with probability greater than zero, into the shell set $S$. Each step aims to add only the best node from $S$ to $D$ if such a node exists that improves the quality function. To do this, the strength values of the edges incident to the nodes in $S$ needs to be calculated based on the expected number of neighbors that a candidate shares with another node in its neighborhood. Algorithm 4 describes how *USIWO* assigns strengths to the edges.

**Algorithm 4:** Assigning Strength for Edges in USIWO

```
Input: Network $G$, and Given Node $N_0$
Output: Network $G$ with strength attribute for edges incident to $N_0$
for each node $N_j \in G.neighbors(N_0)$ do
    $M \leftarrow$ shared neighbors of $N_0$ and $N_j$
    $s_{0,j} \leftarrow 0$
    for each node $N_k \in M$ do
        $s_{0,j} \leftarrow s_{0,j} + \frac{1}{2}[p_{0,k} + p_{k,j}]$
    for each node $N_k \in G.neighbors(N_j)$ do
        $M' \leftarrow$ neighbors of node $N_k$
        $s_{j,k} \leftarrow 0$
        for each node $N_l \in M'$ do
            $s_{j,k} \leftarrow s_{j,k} + \frac{1}{2}[p_{j,l} + p_{l,k}]$
    $s_j^{max} \leftarrow \max_k\{s_{j,k}\}$
$s_0^{max} \leftarrow \max_j\{s_{0,j}\}$
for each node $N_j \in G.neighbors(N_0)$ do
    $S_{0,j} \leftarrow \frac{s_{0,j}}{2} \times (\frac{1}{s_0^{max}} + \frac{1}{s_j^{max}})$
    assign strength $S_{0,j}$ to edge $e_{0,j}$
return $G$
```

From all candidate nodes in $S$, the one that increases the total strength of the edges inside the community the most joins $D$. Then, we need to update $S$ accordingly and try to find the next best node among nodes in the new $S$. This procedure stops when there are no more nodes in the shell set to increase the discovered community's total strength.

*USIWO* has three more important advantages over modularity *UR* and modularity *UR+K*:

1. This algorithm does not need to calculate two different modularity measures for each node when that node is considered a candidate to join the community $D$. The strength value of an edge requires to be computed only once, and it remains fixed until the algorithm stops so that we can use it multiple times without extra computational costs.

2. *USIWO* is free of hyper-parameters. Tuning any hyper-parameter, such as *UR+K*'s $\lambda$, usually requires a significant amount of data. Otherwise, an untrained value may lead the algorithm to fail in unusual cases.

3. Modularity *UR* and *UR+K* discover communities similar to how modularity *R* finds communities. They need extensive and repetitive edge counting whenever a node is about to join the community. *USIWO* follows the instructions introduced by *Local SIWO* and needs significantly less computation. We have already demonstrated in the deterministic scenarios how fast *Local SIWO* is, compared to other methods such as modularity *R*. Thus, we can anticipate *USIWO* would be faster than *UR* and *UR+K*.

We also can measure the goodness of a detected community $C$ or the whole partition $P$ in an uncertain network using the following metrics. The *USIWO* measure can be in the range of $[0, 1]$. A value close to 0 shows a weak community/partitioning, meaning there is the most uncertainty about the nodes' inter-relations in each community. However, a value close to 1 shows a strong community/partitioning for the opposite reason.

$$USIWO_C = \frac{\sum_{v_i, v_j \in C} p_{i,j}}{|\{e_{i,j} \in E : v_i, v_j \in C\}|} \tag{5.8}$$

$$USIWO_P = \frac{1}{|P|} \sum_{c_i \in P} USIWO(c_i) \tag{5.9}$$

Looking back at Figure 5.1 at the beginning of this chapter, we can use this goodness measure to support our claim why two communities of size four are better than a community of size five and another of size three. Using *USIWO* measure, the evaluations for both partitions are as follows, when $P_1 = \{\{v_1, v_2, v_3, v_4, v_5\}, \{v_6, v_7, v_8\}\}$ and $P_2 = \{\{v_1, v_2, v_3, v_4\}, \{v_5, v_6, v_7, v_8\}\}$:

$$\begin{gathered} USIWO_{P_1} = \frac{1}{2}[USIWO_{\{v_1:v_5\}} + USIWO\{v_6 : v_8\}] = \\ \frac{1}{2}[\frac{6 \times 0.9 + 3 \times 0.3}{\binom{4}{2} + 3} + \frac{3 \times 0.9}{\binom{3}{2}}] = \frac{1}{2} \times [0.7 + 0.9] = 0.8 \end{gathered} \tag{5.10}$$

$$\begin{gathered} USIWO_{P_2} = \frac{1}{2}[USIWO_{\{v_1:v_4\}} + USIWO\{v_5 : v_8\}] = \\ \frac{1}{2}[\frac{6 \times 0.9}{\binom{4}{2}} + \frac{3 \times 0.9 + 2 \times 0.8}{\binom{3}{2} + 2}] = \frac{1}{2} \times [0.9 + 0.86] = 0.88 \end{gathered} \tag{5.11}$$

Based on evaluations in Equation 5.10 and Equation 5.11 we conclude that $P_2$ is a better partitioning for the example network in Figure 5.1. Following the same evaluation, for the example network in Figure 5.3, the quality of $P_1 = \{\{v_1, v_2, v_3, v_4, v_5\}, \{v_6, v_7, v_8\}\}$ is $\frac{1}{2} \times [0.725+0.9] = 0.8125$ is higher than the quality of $P_2 = \{\{v_1, v_2, v_3, v_4\}, \{v_5, v_6, v_7, v_8\}\}$ which is $\frac{1}{2} \times [0.7+0.9] = 0.8$. So, our assumption for the true community structure was correct.

## 5.3 Uncertain Network Generator

Since there are not many publicly available uncertain network datasets, we need to generate such networks synthetically to conduct our experiments and evaluate our hypotheses. In this section, we first review a network generator suggested by Zhang *et. al.* [41], then after mentioning how we can improve on that, we propose our new uncertain network generator, which produces more realistic networks.

### 5.3.1 Review of A Previous Method

Zhang *et. al.* [41] proposed a method to generate networks with edge uncertainty derived from a deterministic network, based on three main assumptions:

1. If an edge exists between nodes $v_i$ and $v_j$ in the deterministic network, the probability of this edge in the uncertain network should be high.

2. If nodes $v_i$ and $v_j$ are not connected in the deterministic network, a non-existential edge between them is still possible, which should have a low probability in the uncertain network. These edges would be new to the uncertain network compared to the original deterministic network.

3. Nodes with a higher degree in the deterministic network are more likely to have new added edges in the uncertain network.

The ratio of the extra non-existential edges to the number of edges in the deterministic network is adjustable in this method. It is also possible to set mean values for the normal probability distribution function that generates high and low probabilities.

Although this network generator produces uncertain networks with probabilistic edges, the second assumption leads to adding new edges. These edges may change the network's community structure even though the probabilities are low, so it would be challenging to evaluate a community detection algorithm's output.

## 5.3.2 The New Uncertain Network Generator

To address the critical problem that the former network generator suffers from, we propose a new approach to synthesize uncertain networks. This algorithm takes a deterministic network $G$ with the ground-truth community $C$, $R_p$, and $R_s$. $R_p$ is the ratio of the number of probabilistic edges to the number of all edges in the network. $R_s$ is the swap ratio, which is the proportion of probabilistic edges inside the communities to all probabilistic edges in the network. Since we initially consider uncertainty only for the edges between the communities (where swap ratio is zero), we can increase the swap ratio to replace some of such inter-community edges with the edges inside the communities. However, if the swap ratio is higher than a certain amount, it would spoil the community structure's guarantee.

The output of this generator is a network with only some uncertain edges, and for the rest of them, there is no doubt about their existence, i.e., $p_{i,j} = 1.0$ for the edge $e_{i,j}$. This feature makes more sense as not all data is unclear in real datasets. The input deterministic network can be created using LFR benchmark [43], DANCer [65], etc.

After a deterministic network and its ground-truth community structure are available, the number of uncertain edges is determined using the ratio of probabilistic edges. The algorithm then generates the same number of positive values between $[0, 1]$ as the probabilities. We suggest a normal distribution; however, users can apply other probability distribution functions to generate the random probability values. Since we need to assign the higher probabilities to the edges inside the communities first and then lower probabilities to the between-community edges, we need to sort these values in descending order. We swap as many probability values as required by the swap ratio in the input parameters. The algorithm provides the possibility of swapping not to limit the network's uncertainty to the inter-community edges; it is still possible that edges inside a community have this issue. Finally, we assign the probability values to the edges inside and between the communities, respectively. The Algorithm 5 demonstrates how the uncertain network generator works.

**Algorithm 5:** Generating Uncertain Network with Ground-truth Communities

```
Input: Deterministic Network $G$, Ground-truth Communities $C$,
  Probabilistic Edges Ratio $R_P$, and Swapping Ratio $R_S$
Output: Network $G'$ with Uncertain Edges
$G' \leftarrow$ Create a new graph
$m \leftarrow$ Number of Edges of G
$m_1 \leftarrow$ Number of Edges Inside Communities of G
$m_2 \leftarrow$ Number of Edges Between Communities of G
$m' \leftarrow m \times R_P$                    ▷ number of all probabilistic edges
$m'' \leftarrow m' \times R_S$               ▷ number of swapped probabilistic edges
$P \leftarrow$ Generate $m'$ Random Numbers $\in [0,1]$
$P \leftarrow$ Sort($P$)
$i \leftarrow 0$
while $i < m''$ do
    $j \leftarrow$ Random Number $\in [0, m')$
    $P[j], P[m'-j] \leftarrow P[m'-j], P[j]$
    $i \leftarrow i + 1$
$P_1 \leftarrow P[0 : \lfloor \frac{m_1}{m} \rfloor]$
$P_2 \leftarrow P[\lfloor \frac{m_1}{m} \rfloor : \text{end}]$
for each edge $e_{i,j} \in G.edges$ do
    if $C(i) = C(j)$ & $len(P_1) > 0$ then
        $p \leftarrow P_1.pop()$
        Add Probabilistic Edge $e_{i,j}$ with $p_{i,j} = p$ to $G'$
    else if $C(i) \neq C(j)$ & $len(P_2) > 0$ then
        $p \leftarrow P_2.pop()$
        Add Probabilistic Edge $e_{i,j}$ with $p_{i,j} = p$ to $G'$
    else
        Add Probabilistic Edge $e_{i,j}$ with $p_{i,j} = 1.0$ to $G'$
return $G', C$
```

| Variable | Description | Value |
|---|---|---|
| $n$ | number of nodes | 100 |
| $\tau_1$ | parameter of degree distribution | 20 |
| $\tau_2$ | parameter of community size distribution | 10 |
| $\mu$ | mixing parameter | 0.2 |
| $min\_degree$ | minimum degree of nodes | 30 |

Table 5.2: Parameters for Generating Synthetic Networks using LFR

## 5.4 Experiments

In this section, we evaluate our proposed algorithm, *USIWO*, against the previous methods, modularity *UR* and modularity *UR+K* in terms of accuracy and run time. We use our uncertain network generator, which needs a deterministic network with known ground-truth community information, to prepare the evaluation's required datasets. We use both real and synthetic networks with different sizes and community structures for the initial deterministic network. We consider the Zachary's Karate Club [59], the Football network [9], and a synthetic network generated according to Table 5.2.

We conduct two sets of experiments for each network dataset; in the first one, we increase the probabilistic edges' ratio in a network and record each method's performance. In the second set of experiments, we keep the percentage of the probabilistic edges constant on 30% and increase the swap ratio. The higher the swap ratio, the more inside community edges are assigned with smaller probability values. This ultimately leads to failing to keep the community structure.

We generate with a normal distribution with parameters $\mu = 0.5$ and $\sigma = 0.1$ and assign them to 0% up to 35% of the edges of the network with 5% step size. In the next experiment, 0% up to 70% of the edges are swapped as described before. To get more reliable results, we repeat the random probability assignments 20 times, and the values shown in the following figures are the average over those 20 runs. For the run time bar plot, we measure each algorithm's cumulative required time to start with every node in the network and discover its corresponding community. So, both the network's size and the density of the connections inside it have an impact on the final run time.

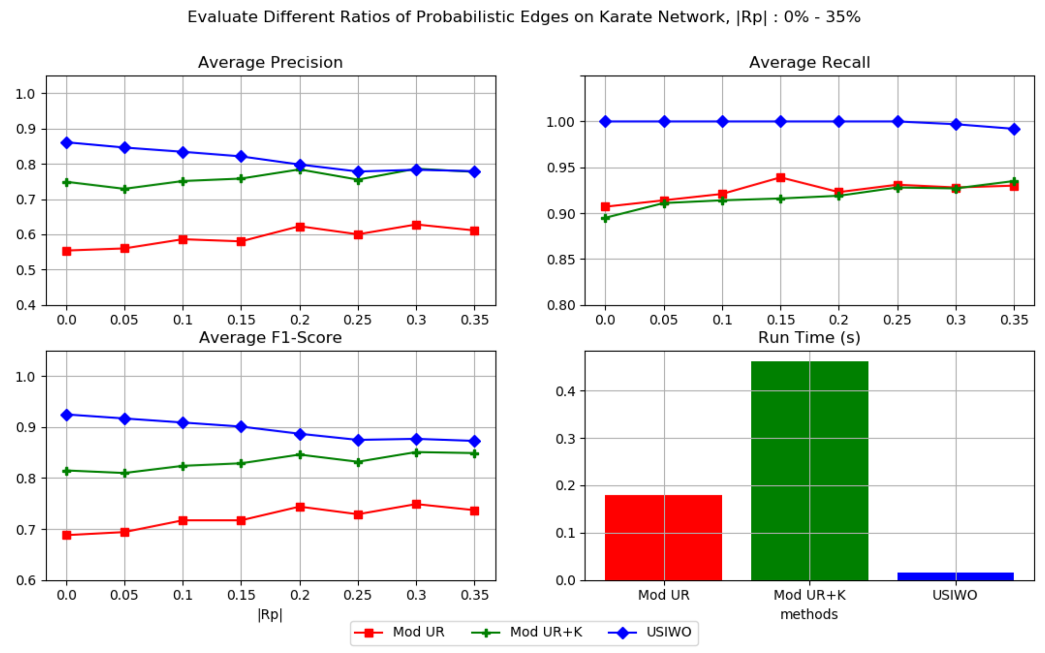


Figure 5.4: Experiment on Percentage of Probabilistic Edges on Karate Dataset. We assign probabilities to up to 35 % of network's edges.

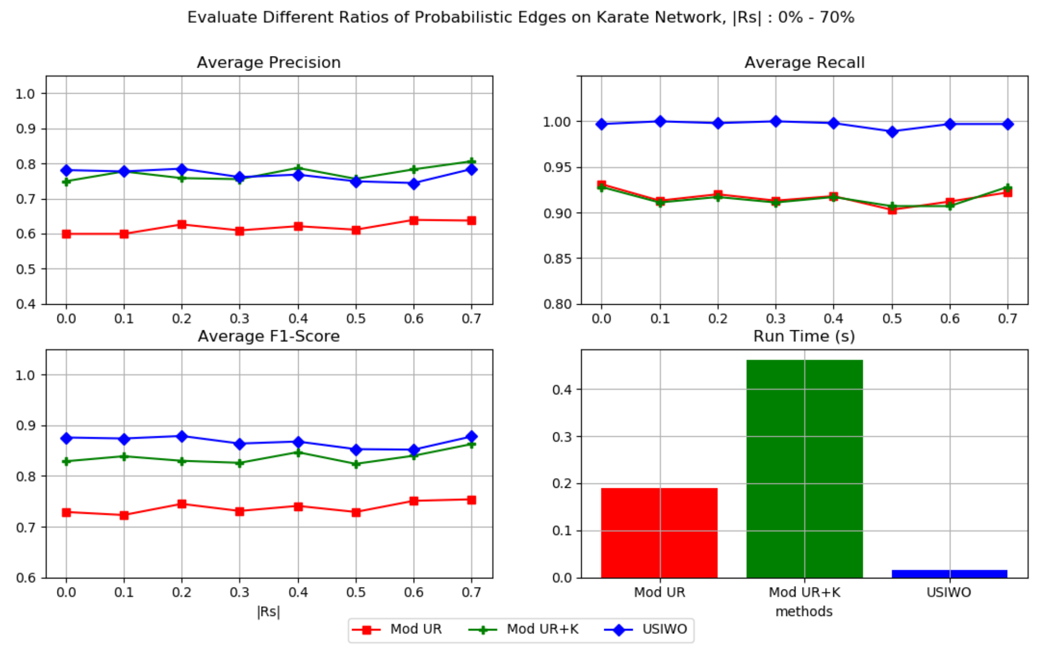


Figure 5.5: Experiment on Swapping Probabilistic Edges on Karate Dataset. We assign up to 70 % of the generated probabilities to the edges inside the communities.

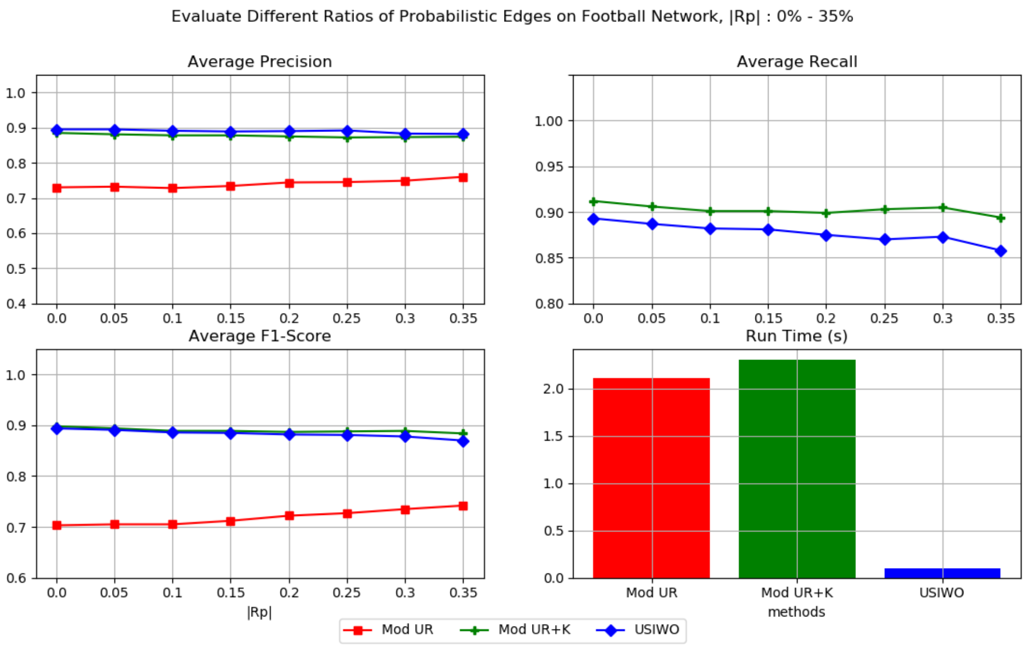


Figure 5.6: Experiment on Percentage of Probabilistic Edges on Football Dataset. We assign probabilities to up to 35 % of network's edges.

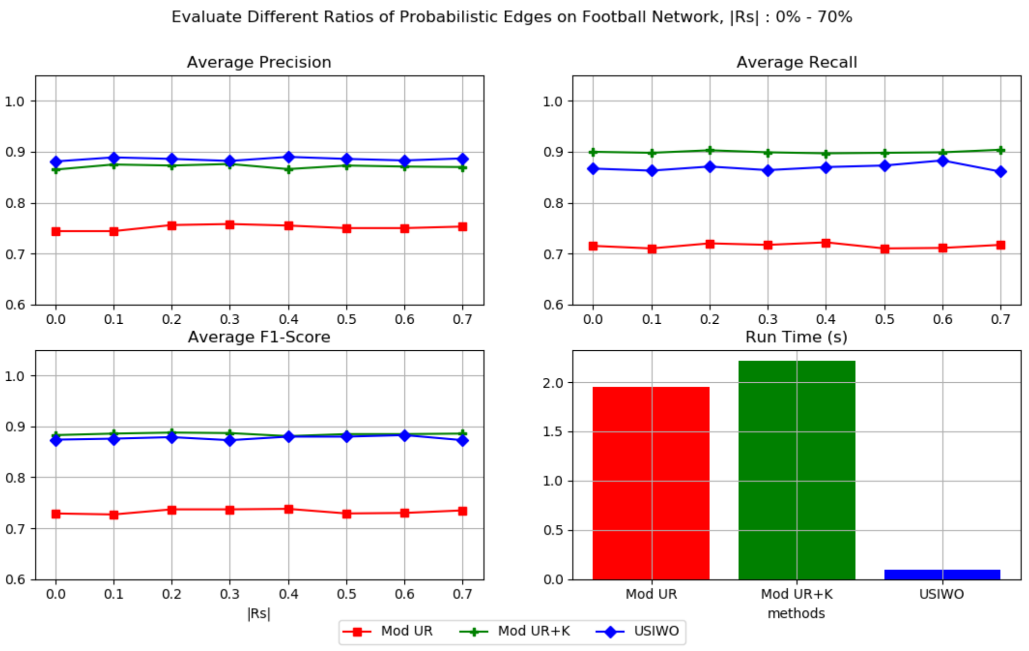


Figure 5.7: Experiment on Swapping Probabilistic Edges on Football Dataset. We assign up to 70 % of the generated probabilities to the edges inside the communities.

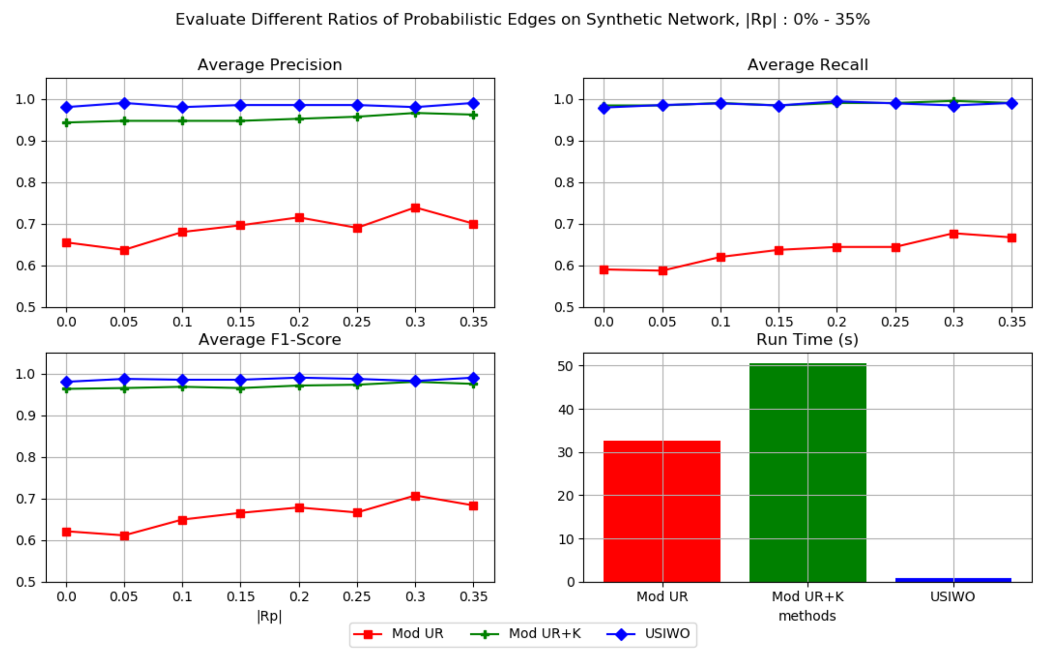


Figure 5.8: Experiment on Percentage of Probabilistic Edges on Synthetic Dataset.We assign probabilities to up to 35 % of network's edges.

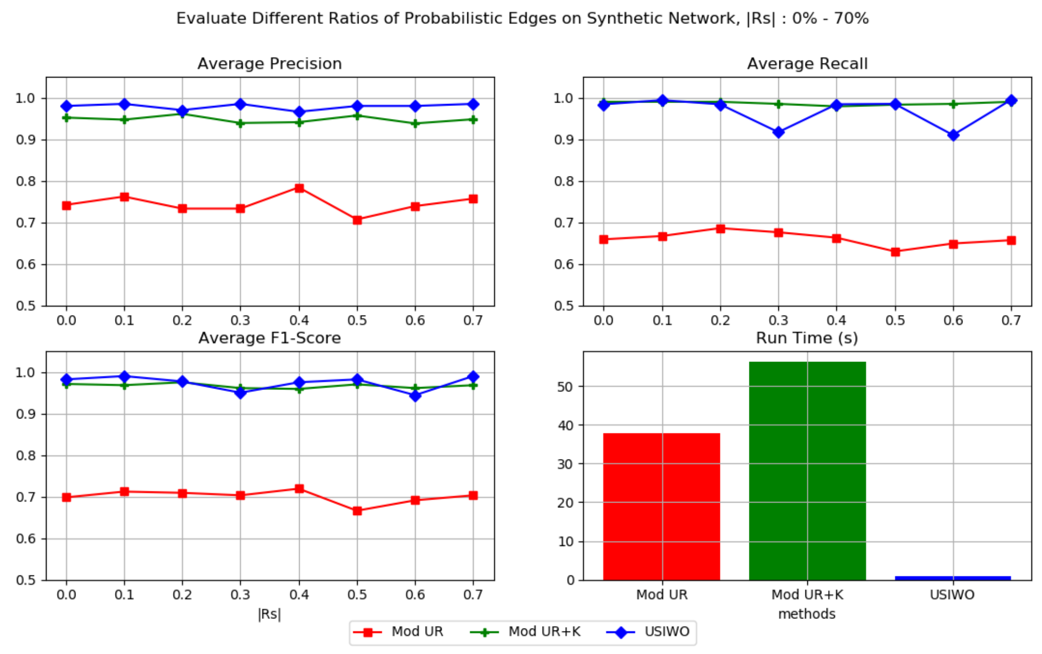


Figure 5.9: Experiment on Swapping Probabilistic Edges on Synthetic Dataset. We assign up to 70 % of the generated probabilities to the edges inside the communities.

Figure 5.4 to Figure 5.9 show that *USIWO* could outperform *UR* and *UR+K* in most cases considering the accuracy measures (precision, recall, and F1-score). *USIWO*'s performance drops by assigning probabilities to more edges in some cases and remains relatively constant in others. However, the performance of *UR* and *UR+K* improved by adding these uncertainties, which is counter-intuitive. Zhang had the same observation in [41]. The other significant improvement of *USIWO* is its run time compared to the other two methods, which confirms our anticipation in 5.2.2.

## 5.5 Conclusion

In this chapter, we propose a novel method that is able to locally discover the community of a given node in an uncertain network. By converting the uncertainty to edge probabilities and exploiting the notion of local strength of the edges, we could avoid wrong expansion in the first steps that former modularity *R*-based approaches suffer from. Our algorithm can discover the communities even for the periphery nodes, so any node in an uncertain network can be grouped into their actual community.

We also provide a principled approach to evaluate the quality of a discovered community or the whole detected partition of an uncertain network such that the nodes with the highest level of certain connections should form a community. Based on this definition of quality, our proposed method could handle special networks that *UR* and *UR+K* could not process correctly. The experiments on different types of uncertain networks show that our method not only performs better than the former approaches in terms of accuracy but also is much faster (up to 70 times). This advantage is more evident on larger and denser networks.

The new *USIWO* method is hyper-parameter-free, which means it does not need any prior training to be ready to detect and discover communities of the uncertain networks. It also avoids extra computation, which *UR+K* needs to perform.

# Chapter 6

# Conclusion

## 6.1 Contributions

In this thesis, we first study existing deterministic community detection algorithms, both global and local, and then develop our novel approaches to detect and discover communities globally and locally. We call them *RSIWO* (which stands for Reciprocal SIWO) and *Local SIWO* respectively. After analyzing deterministic local approaches, we find the essential features that enable our proposed method to perform well. Then, we transfer them into an uncertain scenario and develop *USIWO*, which is a new method to detect and discover communities in networks with probabilistic edges. We finally propose a goodness function which can be used to measure the quality of a community or an entire partitioning done by a community detection algorithm over a network with uncertain edges.

We build *RSIWO* by merging a few beneficial features of other existing methods, combined with some novelty in both algorithm and implementation. *RSIWO*'s goal is to solve the resolution problem, guarantee the connectedness for the detected communities, and perform quickly.

*Local SIWO* shows that discovering local communities with a combination of nodes and their inter-connection, i.e., their shared neighbors, leads to more accurate results. It aims to solve the problem of wrong initial expansion that many existing methods suffer from and provide a mechanism that finds the same group of nodes regardless of which node of the community starts the expansion.

*USIWO* demonstrated that we can use local deterministic community discovery algorithms and adjust them to perform well on the uncertain networks. It also could remedy some drawbacks of existing methods in this field.

Since there is a shortage of publicly available real-world uncertain networks, using a generator that produces synthetic networks is essential to conduct experiments. Because of this, we provide a network generator that produces realistic uncertain datasets. We also introduce, review, and use different evaluation metrics, each suitable for different cases and viewpoints.

## 6.2 Future Work

We were able to propose community discovery and detection methods that can find communities using the network's information locally or globally. Although these methods are accurate, reliable, and fast, there is more room to improve them. We may continue this research path to enable our next method capable of dealing with weights on the edges. More generally, we also may consider networks with attributes. In addition to the topological structure of the network, we may need to take another source of data about the network's entities into account.

The new algorithms suggested in this thesis assume each node only belong to one community. They also consider the edges have no direction, meaning there is no difference between $e_{i,j}$ an $e_{j,i}$. Related work in the future may consider networks with overlapping communities or networks with directed edges. These scenarios may be considered for both deterministic and uncertain networks, as well as global community detector algorithms and local ones. For example, in local community detection, we currently remove the nodes in the discovered communities or ignore them among the candidates to find the next community. However, we can give such nodes the chance to be picked for the next communities by not removing or ignoring them. It would also be possible to consider a parallel implementation when the overlap is desired.

We mentioned in Section 1.2.3 that there are various types of uncertainty in a social network, such as uncertainty in the existence of nodes, the presence of edges, or their attributes. We may follow this line of research to propose methods to handle other types of uncertainty or even be capable of processing networks with multiple types of uncertainty.

# References

[1] S. Gharaghooshi, O. Zaïane, C. Largeron, M. Zafarmand, and C. Liu, “Addressing the resolution limit and the field of view limit in community mining,” in. Apr. 2020, pp. 210–222, ISBN: 978-3-030-44583-6. DOI: 10.1007/978-3-030-44584-3_17.

[2] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” *Science*, vol. 286, no. 5439, pp. 509–512, Oct. 1999. DOI: 10.1126/science.286.5439.509. [Online]. Available: https://doi.org/10.1126/science.286.5439.509.

[3] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’ networks,” *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998. DOI: 10.1038/30918. [Online]. Available: https://doi.org/10.1038/30918.

[4] M. E. J. Newman, “Modularity and community structure in networks,” *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, May 2006. DOI: 10.1073/pnas.0601602103. [Online]. Available: https://doi.org/10.1073/pnas.0601602103.

[5] N. Humphries, N. Queiroz, J. Dyer, N. Pade, M. Musyl, K. Schaefer, D. Fuller, J. Brunnschweiler, T. Doyle, J. Houghton, G. Hays, C. Jones, L. Noble, V. Wearmouth, E. Southall, and D. Sims, “Environmental context explains lévy and brownian movement patterns of marine predators,” *Nature*, vol. 465, pp. 1066–9, Jun. 2010. DOI: 10.1038/nature09116.

[6] A. Klaus, S. Yu, and D. Plenz, “Statistical analyses support power law distributions found in neuronal avalanches,” *PloS one*, vol. 6, e19779, May 2011. DOI: 10.1371/journal.pone.0019779.

[7] J. Leskovec, J. Kleinberg, and C. Faloutsos, “Graph evolution: Densification and shrinking diameters,” en, *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, p. 2, Mar. 2007, ISSN: 1556-4681, 1556-472X. DOI: 10.1145/1217299.1217301. [Online]. Available: https://dl.acm.org/doi/10.1145/1217299.1217301.

[8] R. D. Luce and A. D. Perry, “A method of matrix analysis of group structure,” *Psychometrika*, vol. 14, no. 2, pp. 95–116, Jun. 1949, ISSN: 1860-0980. DOI: 10.1007/BF02289146. [Online]. Available: https://doi.org/10.1007/BF02289146.

[9] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, Jun. 2002. DOI: 10.1073/pnas.122653799. [Online]. Available: https://doi.org/10.1073/pnas.122653799.

[10] T. Silva and L. Zhao, "Semi-supervised learning guided by the modularity measure in complex networks," *Neurocomputing*, vol. 78, pp. 30–37, Feb. 2012. DOI: 10.1016/j.neucom.2011.04.042.

[11] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, Jun. 2009. DOI: 10.1016/j.physrep.2009.11.002.

[12] M. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 69, p. 026 113, Mar. 2004. DOI: 10.1103/PhysRevE.69.026113.

[13] V. 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, Apr. 2008. DOI: 10.1088/1742-5468/2008/10/P10008.

[14] V. A. Traag, L. Waltman, and N. J. van Eck, "From louvain to leiden: Guaranteeing well-connected communities," *Scientific Reports*, vol. 9, no. 1, Mar. 2019. DOI: 10.1038/s41598-019-41695-z. [Online]. Available: https://doi.org/10.1038/s41598-019-41695-z.

[15] S. Wasserman and K. Faust, "Social network analysis in the social and behavioral sciences," in *Social Network Analysis: Methods and Applications*, ser. Structural Analysis in the Social Sciences. Cambridge University Press, 1994, pp. 3–27. DOI: 10.1017/CBO9780511815478.002.

[16] E. Otte and R. Rousseau, "Social network analysis: A powerful strategy, also for the information sciences," *Journal of Information Science*, vol. 28, no. 6, pp. 441–453, 2002. DOI: 10.1177/016555150202800601.

[17] R. Alsamadani, M. Hallowell, and A. N. Javernick-Will, "Measuring and modelling safety communication in small work crews in the us using social network analysis," *Construction Management and Economics*, vol. 31, no. 6, pp. 568–579, 2013. DOI: 10.1080/01446193.2012.685486.

[18] M. Keeling and K. Eames, "Networks and epidemic models," *Journal of the Royal Society, Interface / the Royal Society*, vol. 2, pp. 295–307, Oct. 2005. DOI: 10.1098/rsif.2005.0051.

[19] M. D. Ward, K. Stovel, and A. Sacks, "Network analysis and political science," *Annual Review of Political Science*, vol. 14, no. 1, pp. 245–264, 2011. DOI: 10.1146/annurev.polisci.12.040907.115949.

[20] J. Bouttier, P. Francesco, and E. Guitter, "Geodesic distance in planar graphs," *Nuclear Physics B*, vol. 663, pp. 535–567, Jul. 2003. DOI: 10.1016/S0550-3213(03)00355-9.

[21] R. D. Alba, "A graph-theoretic definition of a sociometric clique," *The Journal of Mathematical Sociology*, vol. 3, no. 1, pp. 113–126, 1973. DOI: 10.1080/0022250X.1973.9989826.

[22] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Mixing local and global information for community detection in large networks," *Journal of Computer and System Sciences*, vol. 80, Mar. 2013. DOI: 10.1016/j.jcss.2013.03.012.

[23] M. Newman, "Fast algorithm for detecting community structure in networks. phys. rev. e stat. nonlin. soft. matter. phys. 69(6 pt 2), 066133," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 69, p. 066 133, Jul. 2004. DOI: 10.1103/PhysRevE.69.066133.

[24] S. Rahiminejad, M. Maurya, and S. Subramaniam, "Topological and functional comparison of community detection algorithms in biological networks," *BMC Bioinformatics*, vol. 20, Dec. 2019. DOI: 10.1186/s12859-019-2746-0.

[25] A. Clauset, M. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 70, p. 066 111, Jan. 2005. DOI: 10.1103/PhysRevE.70.066111.

[26] V. Traag, P. Van Dooren, and Y. Nesterov, "Narrow scope for resolution-limit-free community detection," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 84, p. 016 114, Jul. 2011. DOI: 10.1103/PhysRevE.84.016114.

[27] L. Waltman and N. J. van Eck, "A smart local moving algorithm for large-scale modularity-based community detection," *European Physical Journal B*, vol. 86, Aug. 2013. DOI: 10.1140/epjb/e2013-40829-0.

[28] J. Chen, O. Zaïane, and R. Goebel, "Detecting communities in social networks using local information," *From Sociology to Computing in Social Networks, ISBN 978-3-7091-0293-0. Springer-Verlag Wien, 2010, p. 197*, Jan. 2010. DOI: 10.1007/978-3-7091-0294-7_11.

[29] L. Branting, "Context-sensitive detection of local community structure," *Soc. Netw. Anal. Min.*, vol. 2, pp. 1–11, Sep. 2011. DOI: 10.1007/s13278-011-0035-7.

[30] A. Zakrzewska and D. Bader, "A dynamic algorithm for local community detection in graphs," Aug. 2015, pp. 559–564. DOI: 10.1145/2808797.2809375.

[31] Y. Li, K. He, D. Bindel, and J. Hopcroft, "Uncovering the small community structure in large networks: A local spectral approach," Sep. 2015, pp. 658–668. DOI: 10.1145/2736277.2741676.

[32] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," vol. 2, Nov. 2001.

[33] A. Clauset, “Finding local community structure in networks,” en, *Physical Review E*, vol. 72, no. 2, p. 026 132, Aug. 2005, ISSN: 1539-3755, 1550-2376. DOI: 10.1103/PhysRevE.72.026132. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.72.026132.

[34] F. Luo, J. Wang, and E. Promislow, “Exploring local community structures in large networks,” *Web Intelligence and Agent Systems*, vol. 6, pp. 387–400, Jan. 2008. DOI: 10.1109/WI.2006.72.

[35] J. Whang, I. Dhillon, and D. Gleich, “Non-exhaustive, overlapping k -means,” in. Jun. 2015, pp. 936–944, ISBN: 978-1-61197-401-0. DOI: 10.1137/1.9781611974010.105.

[36] Z. Ding, X. Zhang, D. Sun, and L. Bin, “Overlapping community detection based on network decomposition,” *Scientific Reports*, vol. 6, Apr. 2016. DOI: 10.1038/srep24115.

[37] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, “Uncovering the overlapping community structure of complex networks in nature and society,” *Nature*, vol. 435, pp. 814–818, Jul. 2005.

[38] J. Whang, D. Gleich, and I. Dhillon, “Overlapping community detection using seed set expansion,” Oct. 2013, pp. 2099–2108. DOI: 10.1145/2505515.2505535.

[39] Z. Yakoubi and R. Kanawati, “Applying leaders driven community detection algorithms to data clustering,” Aug. 2012.

[40] M. Elyasi, M. Meybodi, A. Rezvanian, and M. Amir Haeri, “A fast algorithm for overlapping community detection,” Sep. 2016.

[41] C. Zhang and O. R. Zaiane, “Detecting local communities in networks with edge uncertainty,” in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, IEEE, Aug. 2018. DOI: 10.1109/asonam.2018.8508543. [Online]. Available: https://doi.org/10.1109/asonam.2018.8508543.

[42] W. W. Zachary, “An information flow model for conflict and fission in small groups,” *Journal of anthropological research*, pp. 452–473, 1977.

[43] A. Lancichinetti, S. Fortunato, and F. Radicchi, “Benchmark graphs for testing community detection algorithms,” *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 78, p. 046 110, Nov. 2008. DOI: 10.1103/PhysRevE.78.046110.

[44] M. Niewiadomska-Bugaj and D. Mihalko, “On similarity indices and correction for chance agreement,” *Journal of Classification*, vol. 23, pp. 301–313, Feb. 2006. DOI: 10.1007/s00357-006-0017-z.

[45] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008, ISBN: 978-0-521-86571-5. [Online]. Available: http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

[46] S. Ahajjam, E. H. Mohamed, and B. Hassan, "A new scalable leader-community detection approach for community detection in social networks," *Social Networks*, vol. 54, pp. 41–49, Jul. 2018. DOI: 10.1016/j.socnet.2017.11.004.

[47] L. Danon, J. Duch, A. Diaz-Guilera, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, Jun. 2005. DOI: 10.1088/1742-5468/2005/09/P09008.

[48] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: Good, bad and spectral," *Journal of the ACM*, vol. 51, Aug. 2001. DOI: 10.1145/990308.990313.

[49] U. Brandes, M. Gaertler, and D. Wagner, "Engineering graph clustering : Models and experimental evaluation," *First publ. in: ACM Journal of Experimental Algorithmics 12 (2007), Article 1.1*, vol. 12, Jan. 2007. DOI: 10.1145/1227161.1227162.

[50] S. Dongen, "Graph clustering by flow simulation," *PhD thesis, Center for Math and Computer Science (CWI)*, May 2000.

[51] J. Dunn, "Well-separated clusters and optimal fuzzy partitions," *Cybernetics and Systems*, vol. 4, pp. 95–104, Apr. 2008. DOI: 10.1080/01969727408546059.

[52] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," en, *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, Nov. 1987, ISSN: 03770427. DOI: 10.1016/0377-0427(87)90125-7. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0377042787901257.

[53] E. C. Dalrymple-Alford, "Measurement of clustering in free recall.," *Psychological Bulletin*, vol. 74, no. 1, pp. 32–34, 1970. DOI: 10.1037/h0029393. [Online]. Available: https://doi.org/10.1037/h0029393.

[54] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, ser. Structural Analysis in the Social Sciences. Cambridge University Press, 1994. DOI: 10.1017/CBO9780511815478.

[55] R. Rabbany, M. Takaffoli, J. Fagnan, O. R. Zaïane, and R. Campello, "Relative validity criteria for community mining algorithms," in *Encyclopedia of Social Network Analysis and Mining*, Springer New York, 2014, pp. 1562–1576. DOI: 10.1007/978-1-4614-6170-8_356. [Online]. Available: https://doi.org/10.1007/978-1-4614-6170-8_356.

[56] S. Fortunato and M. Barthelemy, "Resolution limit in community detection," *Proc. Nat. Acad. Sci.*, vol. 104, pp. 36–41, Jan. 2006.

[57] M. Schaub, J.-C. Delvenne, S. Yaliraki, and M. Barahona, "Markov dynamics as a zooming lens for multiscale community detection: Non clique-like communities and the field-of-view limit," *PloS one*, vol. 7, e32210, Feb. 2012. DOI: `10.1371/journal.pone.0032210`.

[58] M. Rosvall and C. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, pp. 1118–23, Feb. 2008. DOI: `10.1073/pnas.0706851105`.

[59] W. Zachary, "An information flow model for conflict and fission in small groups1," *Journal of anthropological research*, vol. 33, Nov. 1976. DOI: `10.1086/jar.33.4.3629752`.

[60] L. Adamic, "The political blogosphere and the 2004 u.s. election: Divided they blog," *Proceedings of the 3rd International Workshop on Link Discovery*, Apr. 2005. DOI: `10.1145/1134271.1134277`.

[61] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015. [Online]. Available: `http://networkrepository.com`.

[62] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, May 2012. DOI: `10.1145/2350190.2350193`.

[63] J. Fagnan, O. Zaiane, and D. Barbosa, "Using triads to identify local community structure in social networks," in *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, China: IEEE, Aug. 2014, pp. 108–112, ISBN: 9781479958771. DOI: `10.1109/ASONAM.2014.6921568`. [Online]. Available: `http://ieeexplore.ieee.org/document/6921568/`.

[64] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Second. The MIT Press, 2018, ch. 7. [Online]. Available: `http://incompleteideas.net/book/the-book-2nd.html`.

[65] C. Largeron, P.-N. Mougel, O. Benyahia, and O. Zaïane, "Dancer: Dynamic attributed networks with community structure generation," *Knowledge and Information Systems*, vol. 53, Mar. 2017. DOI: `10.1007/s10115-017-1028-2`.