THE UNIVERSITY
of ADELAIDE

Community Detection in Complex Networks

by

Ba Dung Le

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

in the

Faculty of Engineering, Computer and Mathematical Sciences
School of Computer Science

November 2018

Abstract

Complex networks such as social networks and biological networks represent complex systems in the real world. These networks usually consist of communities which are groups of nodes with dense connections among nodes in the same group and sparse connections between nodes in different groups. Identifying communities in complex networks is useful for many real-world applications.

Numerous community detection approaches have been investigated over the past decades. Modularity is a well-known function to measure the quality of a network division into communities. The most popular detection approach is modularity optimization that identifies communities by finding the community division with highest modularity over all possible community divisions of the network. Current state-of-the-art algorithms for maximizing modularity perform well on networks of strong communities, which have more intra-community connections than inter-community connections. However, these algorithms tend to get trapped in a poor local maximum on networks with weak communities, which have more inter-community connections than intra-community connections.

In the first part of this thesis, we develop a new algorithm for maximizing modularity in networks with weak communities. Our proposed algorithm extends the state-of-the-art algorithm LPAm+ by introducing a method to escape local maximum. Our algorithm follows a guided search strategy inspired by the record-to-record travel algorithm for a trade-off between performance and complexity. Experimental results show that our proposed algorithm, named meta-LPAm+, outperforms state-of-the-art algorithms, in terms of modularity, on networks with weak communities while retaining a comparable performance on networks of strong communities.

In the second part of this thesis, we study the problem of evaluating community detection algorithms. Evaluating the detection algorithms on networks with known communities is important to estimate the accuracy of the algorithms and

to compare different algorithms. Since there are currently only a small number of real networks with known communities available, the detection algorithms are most dominantly tested on synthetic networks with built-in community structure. Current benchmarks, that generate networks with built-in community structure, assign the same fraction of inter-community connections, referred to as the mixing fraction, for every community in the same network and ignore the presence of noise, or outliers. These existing benchmarks, therefore, cannot capture properties of nodes and communities in real networks.

We address this issue by proposing a new benchmark that accounts for the heterogeneity in community mixing fractions and the presence of outliers. Our proposed benchmark extends the state-of-the-art benchmark LFR by incorporating heterogeneous community mixing fractions and outliers. We use our new benchmark to evaluate the performances of existing community detection algorithms. The results show that the variation in community mixing fractions and outliers change the performances of the detection algorithms in ways that lead to different evaluation results. Therefore, community detection algorithms need to be evaluated with heterogeneous community mixing fractions and outliers for a comprehensive view of the performances of the algorithms on real networks. The new benchmark is appropriate for implementing this evaluation as it provides parameters to control the heterogeneity among community mixing fractions and the number of outliers. Furthermore, we show that the evaluation of the detection algorithms with heterogeneous community mixing fractions and outliers gives more accurate estimates of the performances of the algorithms on several commonly studied real networks than the evaluation of the algorithms with homogeneous community mixing fractions and without outliers.

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I acknowledge that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Ba Dung Le

October 2018

Publications

This thesis is based on the following research papers that have been published in peer-reviewed conferences or submitted to quality international journals for publication:

- **Ba-Dung Le**, Hung Nguyen, and Hong Shen. Community Detection in Networks with Less Significant Community Structure. In International Conference on Advanced Data Mining and Applications (CORE Rank: B), pp. 65-80. Springer, Cham, 2016.

  DOI: https://doi.org/10.1007/978-3-319-49586-6_5

- **Ba-Dung Le**, Hung X. Nguyen, Hong Shen, and Nickolas Falkner. GLFR: A Generalized LFR Benchmark for Testing Community Detection Algorithms. In Computer Communication and Networks (ICCCN, CORE Rank: A), 2017 26th International Conference on, pp. 1-9. IEEE, 2017.

  DOI: https://doi.org/10.1109/ICCCN.2017.8038442

- **Ba-Dung Le**, Hong Shen, Hung X. Nguyen, and Nickolas Falkner. Improved Network Community Detection Using Meta-heuristic Based Label Propagation. (Accepted for publication at the Applied Intelligence journal).

- **Ba-Dung Le**, Hong Shen, Hung X. Nguyen, and Nickolas Falkner. GLFR: A Generalized LFR Benchmark with Community Mixing Heterogeneity and Outliers. (Under revision for the ACM Transactions on Knowledge Discovery from Data).

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgements

Undertaking this Ph.D. has been a truly life-changing experience for me. This thesis could not be possible without the support of many people.

I would first like to thank my principal supervisor, Professor Hong Shen, for his continuous guidance and assistance in this study. I am so grateful for the experience and knowledge I have gained from working with him during my entire Ph.D. candidature. I am also thankful to Associate Professor Nickolas Falkner and Dr. Hung Nguyen for their cordial supervisions. Their constructive suggestions and useful comments helped me to significantly improve the content of this thesis.

During my time at the School of Computer Science in the University of Adelaide, I have received constant support from colleagues, staffs, and friends. In particular, I would like to mention Trung T. Pham, Alvaro Parra Bustos, Zhigang Lu, Ron Seidel for many useful discussions, and Sharyn Liersch, for supporting me in conference traveling.

I would like to thank the anonymous reviewers for their valuable comments that led to many improvements in our published papers arising from this thesis.

Finally, I would like to thank the Vietnam International Education Development (VIED) under the Ministry of Education and Training (MoET, Project 911), the University of Adelaide, and the Australian Research Council Discovery Projects funding DP150104871 that financially support me during my Ph.D. study.

This thesis is dedicated to my parents, my wife, and my daughter for their great encouragement and sacrifice during my Ph.D. journey.

Chapter 1

Introduction

Complex networks represent the network structure of complex systems, which are not regular such as lattices nor purely random, in the real-world [87]. Examples of complex networks include social networks [40], biological networks [40], scientific collaboration networks [86], citation networks [105] and the World Wide Web [67]. Nodes in a complex network correspond to entities of a complex system and the links correspond to the relationships among the entities [3]. For instance, in an online social network such as Facebook, the most popular online social network with more than 2.2 billion people as of the fourth quarter of 2017 [1], nodes are people and links are virtual acquaintanceships among the people [118]. In a scientific collaboration network, nodes represent scientists associated with an organization and a link between two scientists indicates that they have been co-authors in a research article [86]. In a network of the World Wide Web, nodes illustrate documents or web pages and links illustrate hyperlinks, or internal page links, that point from one web page to another [67].

Many complex networks exhibit community structure where nodes tend to form communities [40]. Communities in complex networks are informally defined as groups of nodes where interactions between nodes in the same group are more frequent than interactions between nodes in different groups [40]. Nodes in the same communities share common properties or have similar roles in the network. For example, in a social network, communities may be circles of friends [118]. In a scientific collaboration network, communities may be groups of scientists who regularly collaborate with other scientists in the same group [89]. In a network of the World Wide Web, communities may correspond to groups of web pages

[1] https://www.statista.com/statistics/264810.

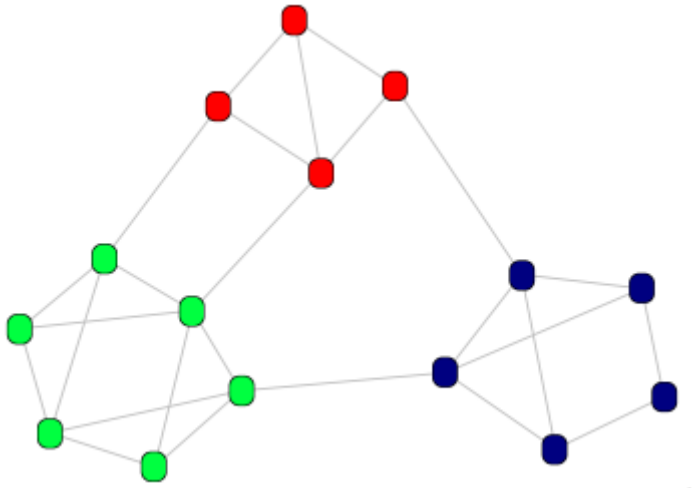


FIGURE 1.1: A sample network consists of three communities (in different colors) with nodes in the same community densely connected and nodes in different communities sparely connected.

sharing the same or relevant topic [35]. Figure 1.1 demonstrates a sample network consisting of three communities.

Identifying communities in complex networks has many concrete applications. Some examples of these applications include extracting knowledge about the functions of network components [44], information transfer in networks [27], epidemic prevention [57], viral marketing [123], water loss detection for water distribution networks [113], and Web services optimization [59]. More specifically, identifying the community structure of a network allows classifying nodes into different roles based on their pattern of within-community and between-community connections [44]. Packet transmission in a network can be significantly improved by providing information about the community structure of the network to the packet routing algorithm [27]. Grouping people in a social network, representing a population, into communities gives a better understanding of the effects of communities on the disease dynamics over the population and promotes more community-focused disease outbreak prevention [57]. Identifying communities in a social network also helps predict knowledge about the viral spreading of information over the network, as the more communities a unit of transmissible information participates in, the more viral it is [123]. The incoming and outgoing flows of communities in a water distribution network allow quantifying the water losses in the network [113]. Identifying communities of the clients of a web application helps to move the content of the web application closer to its clients to minimize the latency experienced by the clients and the load on the Web servers [59].

Community detection has recently become an active research topic in various disciplines such as computer science, physics and sociology [99, 36]. The revolution of computer technology in recent years has significantly increased the availability of network data. The large volume of network data available encourages researchers to investigate efficient computational tools to determine the structural organization, particularly the community structure, of the networks.

Community detection, also referred to as network clustering, is the task of grouping nodes of a network into a priorly unknown number of communities, by only using the information of the network topology [36]. The partitioning of a network into communities aims at maximizing the intra-community interactions and minimizing the inter-community interactions [94]. Communities fundamentally can be overlapped with nodes joining in more than one community at a time. However, the scope of this thesis is limited to non-overlapping community detection, which aims at finding disjoint communities in networks.

Researchers from different disciplines have proposed a large number of community detection algorithms using different approaches [36, 90]. Although existing detection approaches share similar intuitive notions of community, each detection approach generally uses a different quantitative criterion to determine communities [38]. Some community detection algorithms classify nodes into communities that are not covered by any quantitative community definition [38, 40].

One of the earliest algorithms which started the field of community detection is the divisive algorithm proposed by Girvan and Newman [40]. This divisive algorithm and other divisive algorithms [94, 39, 101] are based on the idea that a network can be divided into communities by iteratively removing edges between communities. Each edge in the network is, therefore, quantified by a metric called betweenness which measures, for example, the frequency that the edge participates in the shortest paths between all pairs of nodes [40]. The divisive algorithms then repeatedly remove the edge with highest betweenness until no edges remain. The resulting community division with highest modularity, which is a measure to evaluate the quality of a community division [94], gives the community structure of the network. As modularity is a core concept used in this thesis, it will be explained in detail in Chapter 2.

Since modularity [94] measures the quality of a network division into communities, the best community division of a network should be the one corresponding to the maximum modularity value. Modularity optimization algorithms identify communities by finding the community division with maximum modularity [94] over

all possible community divisions of the network. Unfortunately, performing an exhaustive search over all possible community divisions of a network is practically impossible due to the large number of candidate solutions in the search space [19]. Modularity optimization algorithms [15, 45, 91, 22, 111, 33, 92, 12, 75, 71, 117], therefore, employ a wide variety of heuristic search strategies [1] to find good approximations of the maximum modularity value. We will discuss these algorithms in more details in the next section.

Random walk based algorithms [98, 108] identify communities by considering a random walk process on the network. These algorithms follow the idea that a random walker on a network is more likely to travel within communities and is less likely to travel across communities as the communities are more well-separated. One of the popular random walk based algorithms is the Walktrap algorithm [98] which iteratively groups nodes into communities to minimize the distance between each node and other nodes in its community. The distance between two nodes is defined based on the probabilities that a random walker in the network moves from a node to another in a given number of steps. The quality of the resulting community divisions is measured based on modularity. Another popular random walk based algorithm is the Infomap algorithm [108] which identifies communities by finding the optimal community division of the network with respect to the map equation [108, 107], an alternative measure to evaluate the quality of a community division. The map equation will be explained later in Chapter 2 of this thesis.

Local community detection algorithms [10, 96, 21, 9, 64, 66, 49, 25, 51] identify communities by expanding each community from a single node or an initial group of nodes. These algorithms expand a community by iteratively adding neighboring nodes, which satisfy a given condition, to the community. One of such conditions is to maximize a function that measures the quality of the community. The community expansion process is continued until a stopping criterion is reached.

Some other community detection algorithms [102, 53] are also available in the literature. The label propagation algorithm (LPA) [102] is a time-efficient detection algorithm that iteratively propagates labels of nodes over the network to identify communities. In each iteration, each node updates its label to the one that the maximum number of its neighbors hold. After the convergence of the algorithm, communities are identified as groups of nodes sharing the same label. The statistical inference based algorithm proposed in [53] identifies communities by maximizing the likelihood that the network is generated by the degree-corrected stochastic blockmodel [53], a generative model for networks with community structure.

Among existing community detection approaches, modularity optimization has received considerable attention of researchers in the literature [36, 38]. Furthermore, as different detection approaches generally use different quantitative measures to evaluate the goodness of the detected community structure, evaluating the detection algorithms on networks with know communities, or simply benchmark networks, is important to estimate the accuracy of the detection algorithms and to compare different detection algorithms. Below, we identify two motivational issues toward modularity optimization for community detection and benchmarks for evaluating community detection algorithms.

1.1 Motivations

1.1.1 Modularity optimization for community detection

As briefly mentioned above, modularity optimization algorithms aim at finding the community division with maximum modularity over all possible community divisions of the network to identify communities [91]. However, since performing an exhaustive search for the optimal community division is practically impossible [19], modularity optimization algorithms employ heuristic search strategies to find fairly good approximations of the maximum modularity value within a reasonable execution time [91].

The first algorithm that maximizes modularity to identify communities was the greedy optimization algorithm proposed by Newman [91]. This algorithm starts from an initial community division where each node is in its own community. The algorithm then iteratively merges a pair of communities that lead to the greatest increase in modularity. The iterations stop when all communities are merged into one community. The best community division is the resulting one with the maximum modularity value. The Fastgreedy algorithm [22] improves the running time of the greedy optimization of Newman [91] by using an efficient data structure and removing unnecessary calculations. The extremal optimization algorithm proposed in [33] starts by randomly partitioning the network into two equal-size communities. This algorithm then repeatedly moves nodes from one community to another to improve modularity until the modularity value cannot be further improved. Each resulting community is then regarded as a new network and the algorithm is recursively applied on the new networks. The extremal optimization algorithm [33] outperforms the greedy optimization algorithm [91] but has a

higher computational complexity. The spectral optimization algorithm proposed in [92] uses the spectral bi-partitioning method [13] that repeatedly divides the network into two communities to maximize modularity. This algorithm performs better than the extremal optimization algorithm [33] on large networks with a shorter execution time. The MSG-VM (Multistep Greedy- Vertex Mover) algorithm [111, 112] improves the Fastgreedy algorithm [22] by merging multiple pairs of communities at a time and then refining the resulting communities to further increase modularity. The MSG-VM algorithm outperforms previous algorithms and has an efficient running time [111].

The Louvain method [15] maximizes modularity by iteratively performing two steps. The first step initially assigns every node into its own community and then iteratively moves each node to one of its neighboring communities that yields the largest positive increase in modularity. The first step stops when modularity cannot be further increased. The second step is to build a new network whose nodes are the resulting communities from the first step. The two steps are then repeatedly applied to the new network until no further improvement in modularity can be achieved. The Louvain method is a state-of-the-art algorithm for maximizing modularity as it outperforms previous algorithms in term of computational time while resulting in a comparable modularity value [15].

The LPAm algorithm [12] is equivalent to the first step of the Louvain method but is expressed in an information diffusion perspective. This algorithm initially assigns each node a unique label and then iteratively propagates labels of nodes over the network to maximize modularity. Communities are identified as groups of nodes with the same labels. The LPAm+ algorithm [75] improves LPAm by iteratively combining LPAm with merging pairs of communities to avoid local maxima. Currently, LPAm+ is considered as one of the state-of-the-art algorithms for modularity optimization as it detects higher modularity values than existing algorithms [75].

Other modularity optimization algorithms such as the Simulated annealing algorithm [45], the Mean field algorithm [71], the Column generation algorithm [4], and the Conformational space annealing algorithm [70] uses exhaustive search strategies to search for the maximum modularity value. These algorithms, therefore, have a high computational complexity and cannot be applied to large networks. The approximation algorithms for maximizing modularity introduced in [32, 31] can theoretically guarantee the quality of the detected community structure but hardly find good near optimal solutions.

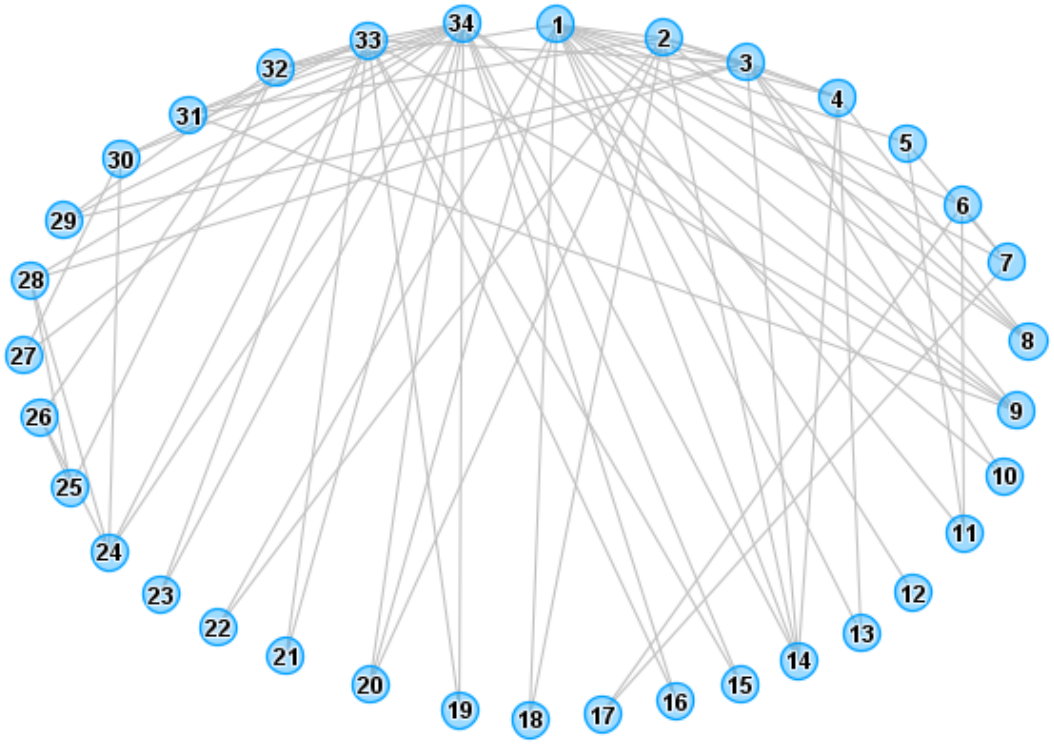


FIGURE 1.2: The Karate Club network.

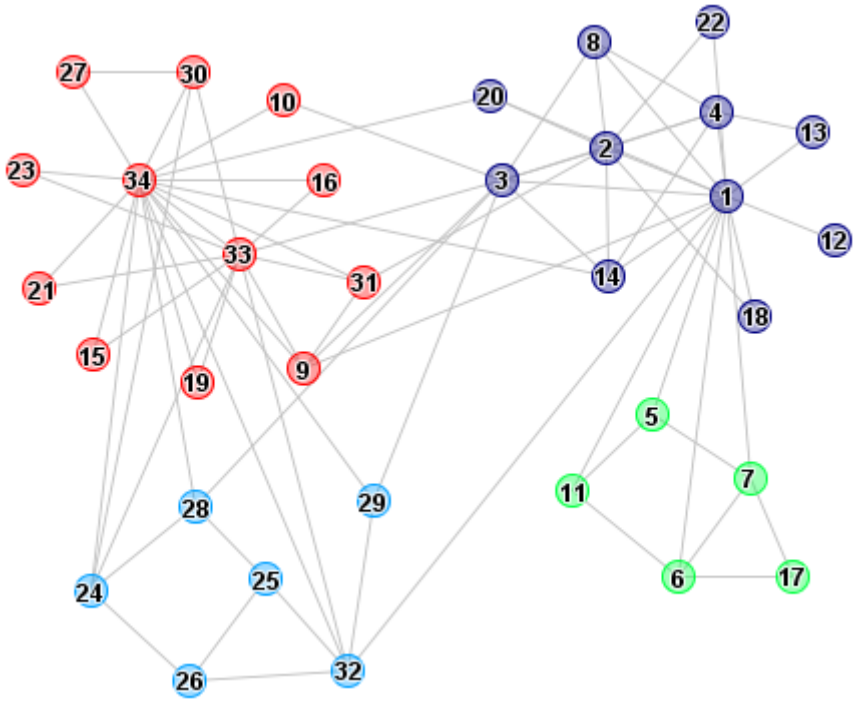


FIGURE 1.3: Communities (in different colors) in the Karate Club network. The community division corresponds to the highest modularity value found by the algorithm proposed in [75]

We depict here a real network with communities discovered by modularity optimization algorithms. Figure 1.2 plots the Karate Club network studied in [125]. The network consists of 34 nodes representing the members of a karate club in a university in the United States. The edges in the network represent interactions of the members outside the activities of the club. Due to a conflict between the club president (node 34) and the club instructor (node 1), members in the club are observed to be separated into two groups, supporting the president and the instructor respectively. Figure 1.3 plots the community division of the Karate Club network corresponding to the highest modularity value found by the algorithm proposed in [75]. This community division correctly distinguishes the two observed groups of members with further subdivisions in each group. The members of the group supporting the president lie in the two communities on the left of the figure and the members of the group supporting the instructor lie in the two communities on the right.

Current state-of-the-art modularity optimization algorithms [15, 75] employ greedy heuristic search strategies to search for the largest modularity value. These greedy optimization algorithms can only find good approximations of the maximum modularity value when there are few local maxima, with respect to modularity, but easily get trapped in a poor local maximum when there are many local maxima in the search space. The state-of-the-art modularity optimization algorithms, therefore, perform well on networks of strong communities, which have more intra-community connections than inter-community connections [62]. However, these algorithms perform poorly on networks with weak communities, which have more inter-community connections than intra-community connections.

In this thesis, we also refer to networks of strong communities as networks with strong community structure and refer to networks with weak communities as networks with weak community structure.

1.1.2 Benchmarks for evaluating community detection algorithms

Evaluating a detection algorithm essentially means applying the detection algorithm to networks with known communities, and then comparing the identified community division with the known community division [94, 62]. The more similar between the identified community division and the known community division, the better the detection algorithm performs.

Real networks with known communities [125, 79] would be ideally used for evaluating community detection algorithms. A reliable detection algorithm should realize the communities observed in these networks. Unfortunately, there are only a few real networks with known communities available and the network sizes are small [94]. Therefore, the evaluations of the detection algorithms are mostly performed on synthetic networks with built-in community structure [62], or simply benchmark networks.

Benchmarks [47, 23, 121, 40, 28, 53, 9, 61], for generating networks with planted community structure, are mainly differ in the generative models they use to generate the networks.

The stochastic block model [47] is a general class of random graph models with nodes divided into communities. Links are placed between pairs of nodes with probabilities depending only on community memberships of the nodes. The planted partition model [23] is a realization of the stochastic block model with communities of the same size. Nodes in the same community have a probability of connection p_{in}, and nodes in different communities have a probability of connection p_{out}, with $p_{out} < p_{in}$. The Relaxed Caveman (RC) graph model [121] can be seen as a special realization of the planted partition model with communities formed by connecting a set of disconnected cliques. The connection probabilities p_{in} and p_{out} are proportional to each other and dependent on community sizes. The Girvan-Newman (GN) benchmark [40] is a specific realization of the planted partition model with four communities of size 32 and average node degree 16. These benchmarks, which are based on the stochastic block model, have essentially communities of the same size and nodes of the same degree. These features are unrealistic as real networks are observed to have power-law distributions in node degree and community size [11].

The degree-corrected stochastic block model [53] extends the stochastic block model by defining the connection probabilities of nodes based on community memberships and degrees of the nodes. This allows variation in node degrees within communities. However, the degree-corrected stochastic block model fails to resemble real networks because the link generation process of the model often produces a large number of nodes with degree zero [53].

The Lancichinetti-Fortunato-Radicchi (LFR) benchmark [61] can be seen as a special version of the degree-corrected stochastic block model. This benchmark determines node degrees and communities sizes from network parameters and constructs the networks to meet the properties of nodes and communities. Therefore,

the LFR benchmark accounts for the heterogeneity of node degrees and community sizes. The LFR benchmark is regarded as a state-of-the-art benchmark for evaluating community detection algorithms [38]. Figure 1.4 illustrates a network realization of the LFR benchmark.

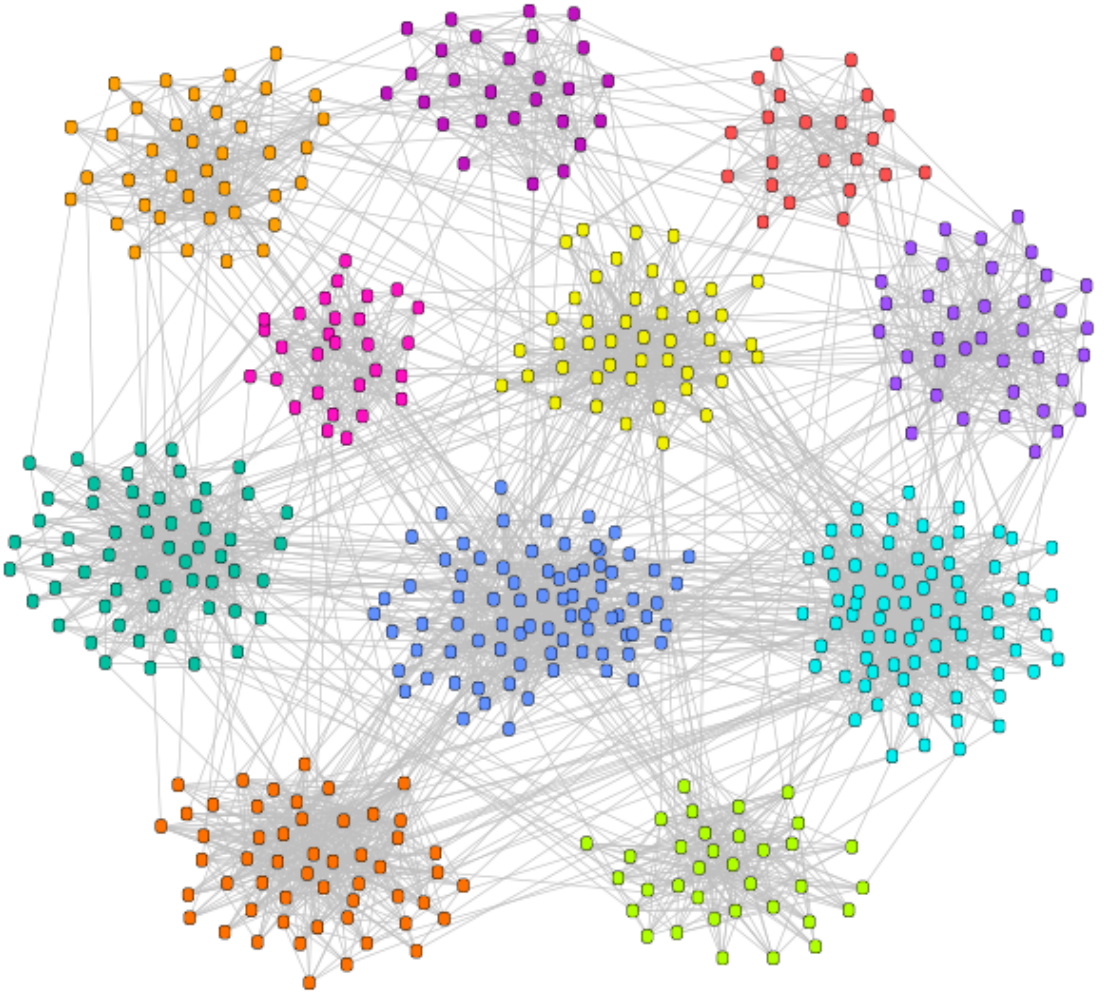


FIGURE 1.4: A network with community structure generated by the LFR benchmark. Communities in the network are in different colors.

The above benchmarks [40, 61] for evaluating community detection algorithms assign a fixed fraction of inter-community links, referred to as the *mixing fraction*, for every community in the same network. Additionally, the current benchmarks eliminate the presence of noise, or outliers. Outliers are defined as nodes that randomly connect to other nodes in the network and, therefore, are not associated with any community. Real world networks exhibit the heterogeneity in community mixing fractions [73, 74] and the mixture of communities and outliers [66]. These existing benchmarks, therefore, cannot capture properties of nodes and communities in real networks.

1.2 Aims of the thesis

This thesis consists of two separate but related aims to address the above issues.

Modularity optimization in networks with weak community structure

Our first aim is to develop a new algorithm for maximizing modularity in networks with weak community structure. To achieve this research aim, we first review the state-of-the-art algorithm LPAm+ [75] and demonstrate its drawback of getting trapped in a poor local maximum on networks with weak community structure. We then introduce a method, that offers a balance between performance and running time, to overcome the local maxima issue of the LPAm+ algorithm. We next perform extensive experiments to evaluate the performances, in term of modularity, of our proposed algorithm and existing algorithms on synthetic and real networks. We finally compare the performances, in term of detection accuracy, of the proposed algorithm and existing algorithms on the synthetic networks.

Benchmarks for evaluating community detection algorithms with heterogeneous community mixing fractions and outliers

Our second aim is to develop a realistic benchmark for evaluating community detection algorithms with the heterogeneous community mixing fractions and outliers. To accomplish this aim, we first review the state-of-the-art benchmark LFR [65, 61]. We then introduce a method to incorporate the heterogeneity in community mixing fractions and outliers into the LFR benchmark. We next perform extensive experiments to quantify the effects of the variation in community mixing fractions and outliers on the performances of community detection algorithms. We finally compare the evaluation of the detection algorithms with heterogeneous community mixing fractions and outliers and the evaluation of the detection algorithms with homogeneous community mixing fractions and without outliers, with respect to reflecting the performances of the detection algorithms on real networks.

1.3 Contributions

The contributions of the research conducted in this thesis are highlighted as follows:

- A new algorithm (presented in Chapter 3), named meta-LPAm+, for maximizing modularity in networks with weak community structure

- Extensive experiments (presented in Chapter 4) to evaluate the performances of meta-LPAm+ and existing community detection algorithms on synthetic and real networks
- A new benchmark (presented in Chapter 5), named GLFR, for evaluating community detection algorithms with heterogeneous community mixing fractions and outliers
- Extensive experiments (presented in Chapter 6) to quantify the effects of the variation in community mixing fractions and outliers on the performances of the detection algorithms

1.4 Thesis structure

This thesis is based on the content of our research papers that have been published in peer-reviewed conferences or submitted to international journals for publication. The remaining chapters of this thesis are organized as follows:

- **Chapter 2** provides background of the research related to this thesis. This chapter contains a summary of the existing definitions relating to community. This chapter also discusses existing community detection algorithms that will be used in our experiments in this thesis. Furthermore, this chapter gives an overview of the state-of-the-art benchmarks for evaluating community detection algorithms and measures for comparing community divisions of networks.
- **Chapter 3** presents our proposed algorithm for maximizing modularity in networks with weak community structure. We first review the state-of-the art algorithm LPAm+ and illustrate the local maxima problem of this algorithm. We then improve the LPAm+ algorithm by introducing a method to escape local maxima. We next extend our proposed algorithm to detect communities in directed networks. As our proposed algorithm has parameters to be specified, we perform an empirical analysis to study the dependence of our algorithm on its parameters. This chapter also includes our analysis on the computational complexity of the proposed algorithm.
- **Chapter 4** gives extensive experiments to evaluate the performances of our proposed algorithm and existing community detection algorithms on synthetic and real networks.

- **Chapter 5** presents our proposed benchmark for evaluating community detection algorithms with heterogeneous community mixing fractions and outliers. We first review the state-of-the-art benchmark LFR as it is the foundation of our proposed benchmark. We then present our method to incorporate the heterogeneity in community mixing fractions and the presence of outliers into the LFR benchmark.

- **Chapter 6** gives extensive tests to quantify the effects of the variation in community mixing fractions and outliers on the performance of community detection algorithms. This chapter also gives our comparative analysis on the evaluation of the detection algorithms with heterogeneous community mixing fractions and outliers and the evaluation of the detection algorithms with homogeneous community mixing fractions and without outliers.

- **Chapter 7** concludes this thesis with possible directions for future work.

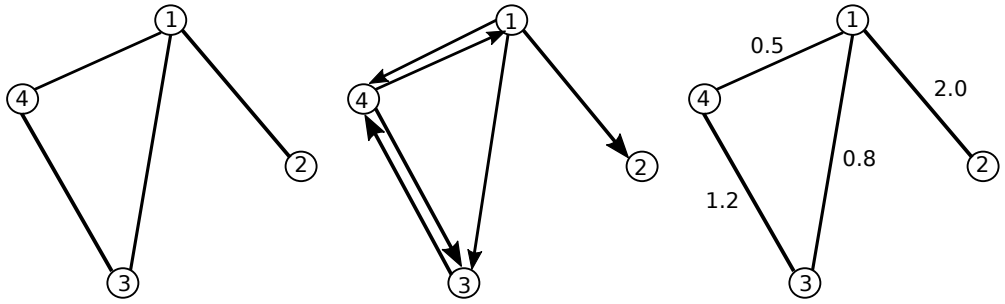
Chapter 2

Background

This chapter provides the background of the research related to this thesis. We first introduce general concepts of networks. We then give various quantitative definitions of community and community structure existing in the literature. We next describe local search strategies that are generally used by community detection algorithms to search for the best community solution over a large number of candidate solutions. Afterward, we discuss existing community detection algorithms that will be compared with our proposed algorithm (presented in Chapter 3). We finally discuss the state-of-the-art benchmarks for evaluating community detection algorithms and measures of the similarity between community divisions of a network.

2.1 Basic network concepts

A network, or a graph, G is a pair of sets (V, E), where V is a set of nodes and E is a set of edges with each edge being a pair of nodes in V. Nodes are also called vertices. Edges are also called links or connections. If each edge is an unordered pair of nodes, the edge is undirected and the network is an undirected network. Otherwise, if each edge is an ordered pair of nodes, the edge is directed from one node to the other and the network is a directed network. In this case, an ordered pair of nodes (u,v) is an edge directed from node u to node v. If each edge has an associated numeric value called weight, the edge is weighted and the network is a weighted network. Figure 2.1 demonstrates three examples of networks including an undirected network, a directed network, and a weighted (undirected) network.



(A) Undirected network (B) Directed network (C) Weighted (undirected) network

FIGURE 2.1: Examples of networks

This thesis mainly focuses on undirected unweighted networks. If other types of networks are considered, they will be explicitly stated.

In undirected networks, two nodes are neighbors, or adjacent to each other, if there is an edge connecting the nodes. The degree of a node is the number of edges incident on the node. The degree sequence of a network is the list of the degrees of nodes in the network. In directed networks, a node has two types of degree, in-degree and out-degree. The in-degree of a node is the number of directed edges pointing to the node, called incoming edges, and the out-degree of a node is the number of directed edges pointing from the node, called outgoing edges.

2.2 Community-related definitions

Community detection researchers commonly agree on the general concept of community, that communities are groups of nodes with dense connections within the groups and sparse connections between the groups [40]. However, there is no commonly accepted quantitative definition of community [36]. Different quantitative definitions of community have been proposed for different applications of the community definitions [38]. In many cases, communities are the final outcome of an algorithm aiming to divide the network into groups of cohesively connected nodes without a rigorous definition of community [40].

2.2.1 Community

As viewed from a local perspective, communities are defined as groups of nodes in a network with their own characteristics that possibly relate to some intermediate neighbors but are independent from the rest of the network [9]. Definitions of community are closely linked with the concepts of cohesive subgroups in social network analysis [120]. These two terms, communities and cohesive subgroups, are used to describe groups of cohesively connected nodes.

The earliest form of a cohesive subgroup is perhaps clique [78], a maximal subset of nodes in which each node connects to all other nodes. However, the condition of the existence of cliques in networks is extremely strict and, therefore, cliques do not frequently appear in real networks. The concept of clique is generalized to the notion of n-clique [77] that is a maximal subset of nodes with the distance between any two nodes not larger than n. When n=1, the n-clique becomes a clique because all nodes in the n-clique are adjacent to each other. It is noted that a clique of k nodes is also called a k-clique [30, 96]. Other relaxed forms of cliques including n-club, n-plan, k-plex and k-core can be found in [120]. The notions of cohesive subgroups mentioned above are inadequate to describe communities since these concepts take into account only the density of connections within the groups but neglect the density of connections between the groups.

Let $G = (V, E)$ be a network, where V is a set of nodes and E is the set of the edges connecting nodes in V. The task of community detection is to divide the set V of nodes into n_c communities, $c_1, c_2, ..., c_{n_c}$, where n_c is not known priorly and each node must belong to at least one community. Two communities are overlapped if they share at least one common node. This thesis mainly focuses on the case of non-overlapping community detection, that is $c_i \cap c_j = \emptyset$ for $i \neq j$ and $1 \leq i, j \leq n_c$. If overlapping communities are considered, they will be explicitly stated.

A community division $C = \{c_1, c_2, ..., c_{n_c}\}$ of a network is also called a partition of the network. Let A be the adjacency matrix of the network where $A_{ij} = 1$ if there is a connection between node i and node j, and $A_{ij} = 0$ otherwise. Let k_i^{int} be the number of links from node i to nodes in its community, or the internal degree of node i with respect to its community, and k_i^{ext} be the number of links from node i to the rest of the network, or the external degree of node i. For node $i \in c$,

$$k_i^{int} = \sum_{j \in c} A_{ij}$$

and

$$k_i^{ext} = \sum_{j \notin c} A_{ij}.$$

Let k_i be the total number of links connected to node i, or the degree of node i, that is

$$k_i = k_i^{int} + k_i^{ext}.$$

The first quantitative notion of cohesive subgroup that takes into account both internal connection and external connection densities of the subgroup is LS-Set [76]. A LS-Set is a subset of nodes such that the internal degree of each node, with respect to the subset, is greater than the external degree of the node. That is, for any node i in a LS-Set,

$$k_i^{int} > k_i^{ext}.$$

The notion of LS-Set is similar to the definition of community in [101].

The notion of community used in [102, 48] implies that a community is a group of nodes where each node in the community has more links with nodes in its community than with nodes in any other community. That is

$$k_i^{int} > \max_c \{k_i(c)\}$$

where $k_i(c) = \sum_{j \in c} A_{ij}$.

A community can also be defined based on the sharpness of the boundary between the community and the rest of the network. In [10], a community, called a L-shell, initially consists of only a single node and then iteratively expanded by adding neighboring nodes to the community. The sharpness of the boundary of the community is measured based on a metric called total emerging degree. Let c_0 be the initial community and c_l be the community formed at step l. The total emerging degree of the community formed at step l is

$$K_l = \sum_{i \in c_l - c_{l-1}} k_i^{ext}$$

where $c_l - c_{l-1}$ denotes the group of nodes that belong to c_l but do not belong to c_{l-1} and k_i^{ext} is the external degree of node i, with respect to c_l. The sharpness of the boundary of the community formed at step l is the ratio of the total emerging degree of the community to the total emerging degree of the community formed

at step $l-1$, which is

$$\Delta_{K_l} = \frac{K_l}{K_{l-1}}.$$

A community is iteratively expanded until the condition $\Delta_{K_l} \geq \alpha$, with α being a predetermined threshold, is not satisfied. The value of α is flexibly chosen to allow for various scales of communities. However, it is not clear which value of α should be best taken to define communities.

In [21], the sharpness of the boundary of a community is measured based on a metric called local modularity. Let C be the currently formed community. Let B be the boundary of C which is the group of nodes in C having at least one connection outside C. The local modularity of C, which is proportional to the sharpness of the boundary B, is defined as the ratio between the number of edges in C having at least one endpoint in B and the number of edges in the network having at least one endpoint in B [21]. The local modularity of community C is mathematically expressed as

$$R = \frac{\sum_{i \in C} \sum_{j \in B} A_{ij}}{\sum_{i} \sum_{j \in B} A_{ij}}.$$

A community is iteratively expanded by adding one neighboring node at a time to the community until its local modularity is maximized.

In [64], the authors define a community based on a fitness function that evaluates the quality of the community. Let κ_C^{int} and κ_C^{ext} are respectively the internal and external degrees of the currently formed community C, where

$$\kappa_C^{int} = \sum_{i \in C} k_i^{int}$$

and

$$\kappa_C^{ext} = \sum_{i \in C} k_i^{ext}.$$

The Local Fitness of community C is defined as [64]

$$f_C = \frac{\kappa_C^{int}}{(\kappa_C^{int} + \kappa_C^{ext})^{\alpha}},$$

where α is a positive-real value parameter. The value of α defines the resolution at which communities are resolved but it is not clear which value of α should be best taken.

2.2.2 Community structure

From a global viewpoint, communities are defined as interrelated parts of the whole network [36]. The community structure of a network is regarded as a network division into communities that satisfies a certain criterion. The two most popular criteria to define the community structure of a network are modularity [94] and the map equation [107].

2.2.2.1 Modularity

Modularity [94], or also called Newman-Girvan modularity, is a function that measures the goodness of a network division into communities. The basic idea of modularity is that a random network, where nodes are randomly connected, is not expected to have a community structure. Therefore, the quality of a community division of a network can be asserted by the difference between the actual fraction of edges within communities and that expected fraction of edges in an equivalent random network.

In Newman-Girvan modularity, the equivalent random network is generated by the random network model called the configuration model [85]. In this random network model, each node is attached with a number of stubs, or half-edges, equal to the degree of the node. The stubs of nodes are randomly paired with each other to connect nodes in the network. Given an undirected unweighted network with n nodes and m edges, the expected number of edges between node i, with degree k_i, and node j, with degree k_j, in the configuration model is

$$P_{ij} = \frac{k_i k_j}{2m}.$$

Let A be the adjacency matrix of the network, where $A_{ij} = 1$ if there is an edge between node i and node j and, otherwise, $A_{ij} = 0$. The definition of modularity is given as follows.

Definition 1. [94] The modularity of a community division of an undirected and unweighted network is defined as

$$Q = \frac{1}{2m} \sum_{g} \sum_{i,j \in g} \left(A_{ij} - \frac{k_i k_j}{2m} \right), \tag{2.1}$$

where g is a community in the network, k_i and k_j are the degrees of node i and node j respectively, and m is the number of edges in the network.

The value of modularity lies in the range from 0 to 1. If the network has a community structure, the value of modularity is higher than zero. The larger the modularity value, the more the number of edges within communities compared with the expected number of the edges in an equivalent random network, and therefore, the better the quality, or the strength, of the community structure. The best community division of a network is, therefore, regarded as the one that gives the maximum modularity value.

Modularity has been widely adopted as the objective function of many algorithms to identify communities [36]. Modularity optimization is currently a state-of-the-art approach for detecting communities [62]. However, modularity has a resolution limit [37] that impedes the detection of small size-scale communities in networks.

2.2.2.2 The map equation

The map equation [107, 108] is another function to evaluate the quality of a community division. The fundamental of the map equation is that given a network with nodes divided into communities, the path of a random walk through the network can be represented by a sequence of codewords decoding the entered communities, the visited nodes and the exits from communities of the random walker. The codewords of communities are uniquely defined while the codewords of nodes and the exits from communities are reused among different communities. If connections within communities are dense and connections between communities are spare, the random walk infrequently crosses different communities. Therefore, the codewords of communities are rarely used that saves the length of the codeword sequence describing the path of the random walk. The shorter the length of the codeword sequence, the better the quality of the community division.

The description length of a random walk across the network is calculated from the two terms, the Shannon information [114] of the codewords representing communities and the Shannon information of the codewords representing nodes in communities and the exits from communities of the random walker. Let c be a community in an undirected unweighted network visited by a random walker. Let κ_c^{ext} be the external degree of community c. The probability that the random walker enters or exits community c is [54]

$$p_c^{enter} = p_c^{exit} = \frac{\kappa_c^{ext}}{2m}.$$

The probability that the random walker enters or exits any community in the network is

$$p^{enter} = p^{exit} = \sum_c p_c^{enter} = \sum_c p_c^{exit}.$$

The Shannon information [114] of the codewords decoding communities is [107, 108]

$$H(C) = -\sum_c \frac{p_c^{enter}}{p^{enter}} log\left(\frac{p_c^{enter}}{p^{enter}}\right). \tag{2.2}$$

Let k_i be the degree of node i and m be the number of edges in the network. The probability that the random walker visits node i is [54]

$$p_i = \frac{k_i}{2m}.$$

The probability that the random walker moves within and exits community c is

$$p_c^{travel} = \sum_{i \in c} p_i + p_c^{exit}.$$

The Shannon information of the codewords decoding the visited nodes in community c and the exist from community c of the random walker is [107, 108]

$$H(c) = -\sum_{i \in c} \frac{p_i}{p_c^{travel}} log\left(\frac{p_i}{p_c^{travel}}\right) - \frac{p_c^{exit}}{p_c^{travel}} log\left(\frac{p_c^{exit}}{p_c^{travel}}\right). \tag{2.3}$$

The map equation which measures the description length of a random walk across the network is given as follows.

Definition 2. [108] The map equation of a community division in an undirected and unweighted network is defined as

$$L(M) = p^{enter} H(C) + \sum_c p_c^{travel} H(c) \tag{2.4}$$

The smaller the value of the map equation, the less frequently the random walker moves between communities and, therefore, the better the quality of the community structure. The best community division of a network is regarded as the one that gives the minimum value of the map equation [108].

The map equation has been employed as the objective function of the Infomap algorithm [108] to identify communities. This algorithm is considered as a state-of-the-art algorithm for detecting communities [62]. However, the map equation suffers the resolution limit [54, 55] which implies that the optimal community division may not recognize communities with a small ratio between the numbers of internal links and external links.

2.2.3 Definitions in generative models

Generative models [47, 23, 65, 53] are used to generate networks with community structure. In generative models, the community structure of a network is defined based on the probabilities of connections between nodes in the network. Different generative models give different definitions of community depending on the assumption of the models on the connection probabilities of nodes.

The most popular model for networks with communities is the stochastic block model (SMB) [47]. In this model, the connection probability of two nodes is a function of their community memberships. Nodes in a network are divided into K communities. Nodes in community r and nodes in community s are connected with a probability ω_{rs}. The network is considered to have a community structure, or called an assortative structure, if the connection probabilities between nodes in the same community are higher than the connection probabilities between nodes in different communities [5, 38], which is expressed by the following condition:

$$\min_{r}\{\omega_{rr}\} > \max_{l,k}\{\omega_{lk}\},$$

for $\forall$ r, l, m = 1,2,..,K and l $\neq$ k.

The planted partition model [23] is a special version of the stochastic block model. In the planted partition model, nodes in the same community are connected with a probability p_{in} and nodes in different communities are connected with a probability p_{out}. Therefore, the community structure of a network is well defined for $p_{in} > p_{out}$.

The degree-corrected stochastic block model (DCSBM) [53] is an extension of the stochastic block model [47] that defines the connection probability of two nodes as a function of their community memberships and degrees. The connection probability between node i and nodes in its community is p_i^{in}. The connection probability between node i and nodes in community c, with $i \notin c$, is $p_i^{out}(c)$. This model implies that the community structure of a network is recovered if every

node has a higher connection probability with nodes in the same community than with nodes in any other community [53]. That is

$$p_i^{in} > \max_c \{p_i^{out}(c)\},$$

for every node i.

The generative model which is implicitly employed in [65] can be seen as a simplified version of the degree-corrected stochastic block model. In this model, node i is connected with nodes in its community with a probability p_i^{in} and connected with nodes outside its community with a probability p_i^{out}. The community structure is constructed to satisfy the condition $p_i^{in} > p_i^{out}$, for every node i.

2.3 Local search strategies

Local search strategies [1] are heuristic algorithms to search for an optimal solution with respect to an objective function among a large number of candidate solutions. These algorithms [1] typically start with an initial solution and then locally perturbs the current solution to generate neighbor solutions. The changes that may be applied to perturb a solution are defined by a neighborhood structure. The search algorithm evaluates the neighbor solutions and chooses the one that meets a given condition to replace the current solution. The search process is repeated until a stopping criterion is satisfied.

Let S be a solution space of all candidate solutions. Let $f : S \to \mathbb{R}$ be an objective function to maximize. The objective function $f(s)$ gives a value indicating the quality of a solution $s \in S$. The aim of the search is to find a solution $s^* \in S$ with the maximal value of the objective function.

Definition 3. [116] A solution s^* is a global maximum if its value given by the objective function is higher than the given value of all solutions in the solution space, that is, $f(s^*) \geq f(s)$ for $\forall s \in S$.

Definition 4. [16] A neighborhood structure is a function that assigns to every solution $s \in S$ a set of neighborhood solutions $N(s) \in S$.

Definition 5. [16] A solution s^* is a local maximum with respect to a neighborhood solution space $N(s^*)$ if for $\forall s \in N(s^*) : f(s^*) \geq f(s)$.

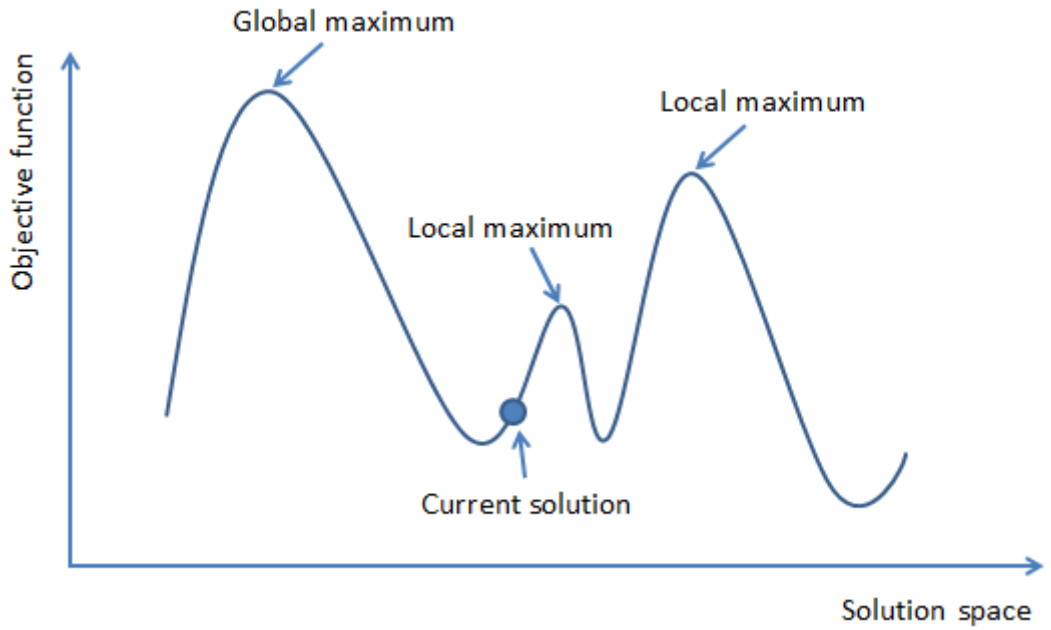


FIGURE 2.2: An example of a solution space defined by an objective function

Figure 2.2 demonstrates a one-dimension solution space landscape with the quality of a solution defined by the objective function. The solution space consists of two local maximums and a global maximum.

2.3.1 Hill-climbing

The simple hill-climbing algorithm is a typical local search strategy that replaces the current solution by the first evaluated neighbor solution with a higher value in the objective function. The algorithm terminates when it reaches a local maximum where no neighbor solution has a higher quality. The steepest ascend hill-climbing algorithm, also known as greedy local search strategy [109], is a variant of the simple hill-climbing algorithm. The steepest ascend hill-climbing algorithm examines all neighbors of the current solution and then selects the neighbor with the highest quality to replace the current solution. The stochastic hill-climbing algorithm [109] is similar to the steepest ascend hill-climbing algorithm but does not examine all neighbors of the current solution. The stochastic version randomly examines a neighbor solution of the current solution and retains the neighbor with higher quality than the current one. The pseudo codes of the simple hill-climbing algorithm, the steepest ascend hill-climbing algorithm and the stochastic hill-climbing algorithm are respectively given in Algorithms 1, 2 and 3.

Algorithm 1 Simple hill-climbing [109]

Input: an initial solution s_0, a neighborhood structure and an objective function f

Output: a solution s^* that is a local optimum

```
1: s* = s0
2: while (true) do
3:    s = a neighbor of s* that has not been evaluated
4:    if f(s) > f(s*) then
5:       s* = s
6:    end if
7:    if all neighbors of s* are evaluated then
8:       return s*
9:    end if
10: end while
```

Algorithm 2 Steepest ascend hill-climbing (Greedy local search) [109]

Input: an initial solution s_0, a neighborhood structure and an objective function f

Output: a solution s^* that is a local optimum

```
1: s* = s0
2: while (true) do
3:    s = the neighbor of s* with the highest value of f(s)
4:    if f(s) > f(s*) then
5:       s* = s
6:    else
7:       return s*
8:    end if
9: end while
```

Algorithm 3 Stochastic hill-climbing [109]

Input: an initial solution s_0, a neighborhood structure and an objective function f

Output: a solution s^* that is a local optimum

```
1: s* = s0
2: for a predefined number of iterations do
3:    s = a randomly selected neighbor of s*
4:    if f(s) > f(s*) then
5:       s* = s
6:    end if
7: end for
8: return s*
```

The hill-climbing algorithms fall into the class of heuristic algorithms [109] that have no mechanism to escape a local maximum. The hill-climbing algorithms can quickly converge to a local maximum because the search continually improves the

current solution. However, the success of the hill-climbing algorithms significantly depends on the distribution of local maxima in the solution space. If there are many local maxima, the hill-climbing algorithms generally get trapped in a local maximum.

2.3.2 Simulated annealing

The simulated annealing (SA) algorithm [56] is similar to the stochastic hill-climbing algorithm since the SA algorithm accepts a randomly evaluated neighbor if it has a higher quality than the current one. However, the SA algorithm also accepts the neighbor with a worse quality with a probability that is exponentially proportional to a temperature parameter and the decrease in the quality between the neighbor and the current solution. The temperature parameter is decreased as the running time of the algorithm increases. Over time, the SA algorithm is less likely to accept worse neighbors. If the temperature is decreased slowly enough, the SA algorithm is almost certain to find the global optimum [109]. The pseudo code of the SA algorithm is given in Algorithm 4.

Algorithm 4 Simulated annealing [56]

Input: an initial solution s_0, a neighborhood structure, an objective function f, a mapping function from time to temperature *schedule*
Output: a solution s^*

1: $s^* = s_0$
2: **for** t=1 **to** ∞ **do**
3: T = $schedule(t)$
4: **if** T=0 **then**
5: **return** s^*
6: **end if**
7: s = a randomly chosen neighbor of s^*
8: $\Delta E = f(s) - f(s^*)$
9: **if** $\Delta E > 0$ **then**
10: $s^* = s$
11: **else**
12: $s^* = s$ with a probability $\epsilon^{\Delta E/T}$
13: **end if**
14: **end for**

The SA algorithm falls into the class of meta-heuristic algorithms [106, 116] that guide the search process to avoid being trapped in a local maximum. A meta-heuristic algorithm escapes a local maximum by temporarily accepting neighbor solutions with lower quality. This allows the algorithm to examine more neighbor

solutions to find a better solution. One of the major drawbacks of the SA algorithm is that it is considerably slow when the search space is large [16].

2.3.3 Record-to-record travel

The record-to-record travel (RRT) algorithm [34] is a variant of the simulated annealing algorithm [56] with a different mechanism to accept worse neighbor solutions. The RRT algorithm accepts a worse neighbor solution if the difference between the quality of the neighbor solution and the quality of the best solution found is less than a specific threshold. A small value of the threshold generally gives a poor result with a short running time while a large one gives a good result with a long running time. If the threshold is equal to zero, the RRT algorithm will become a stochastic hill climbing algorithm which accepts only better neighbor solutions. The algorithm stops after a predefined number of iterations without improvement in the objective function. The pseudo code of the RRT algorithm is given in Algorithm 5.

Algorithm 5 Record-to-Record Travel [34]

Input: an initial solution s_0, a neighborhood structure, an objective function f, a positive deviation value DEV and the maximum number of iterations without improvement in the objective function max_{no}
Output: a solution s^*

$s = s_0$
$s^* = s$
while $t < max_{no}$ **do**
    s' = a randomly chosen neighbor of s
    if $f(s') \geq f(s^*) - DEV$ **then**
        $s = s'$
        t = t+1
    end if
    if $f(s') > f(s^*)$ **then**
        s* = s'
        t = 0
    end if
end while

The RRT algorithm is simpler and requires less computational cost than the SA algorithm. The RRT algorithm is also reported to outperform the SA algorithm with much lower running time for some optimization problems [34].

2.4 Community detection algorithms

Numerous community detection algorithms have been proposed in the literature [36, 38]. Among the existing algorithms, the Louvain method [15] and Infomap [107], are regarded as state-of-the-art algorithms [62]. Walktrap [98], Fastgreedy [22], and MSG-VM [111] are also a popular methods [38]. In this section, we review these algorithms which will be used in our experiments in Chapter 3.

2.4.1 Fastgreedy

The Fastgreedy algorithm [22], also referred to as the Clauset-Newman-Moore (CNM) algorithm, is an efficient implementation of the greedy modularity optimization algorithm proposed in [91]. The Fastgreedy algorithm initially assigns each node in a unique community and then repeatedly joins communities in pairs to improve modularity. The pair of communities joined at each step is selected to lead to the largest increase (or the smallest decrease) in modularity. When the last pair of communities is joined, the resulting community division with maximum modularity gives the community structure. The Fastgreedy algorithm follows an agglomerative hierarchical approach that results in a dendrogram of the network. The dendrogram is a tree structure representing the hierarchical structure of the joined communities. The complexity of the Fastgreedy algorithm is $O(mdlog(n))$ where n is the number of nodes in the network, m is the number of edges, and d is the height of the dendrogram. An improvement of the Fastgreedy algorithm in term of execution efficiency and scalability is proposed in [119]. The pseudocode of the Fastgreedy algorithm is presented in Algorithm 6.

Algorithm 6 Fastgreedy [22, 91]

1: Assign each node in its own community
2: Calculate the modularity change matrix ΔQ with ΔQ_{ij} being the change in modularity if communities i and j are merged.
3: **while** there are more than one community in the network **do**
4: Select the pair of communities i and j with the largest ΔQ_{ij}
5: Merge communities i and j
6: Store the current partition of the network
7: Update the modularity change matrix ΔQ
8: **end while**
9: **return** The stored network partition with the greatest modularity value

2.4.2 MSG-VM

The MSG-VM (Multistep Greedy- Vertex Mover) [111, 112] is an enhanced version of the Fastgreedy algorithm [22]. The MSG-VM algorithm consists of a community join procedure (MSG) that joins multiple pairs of communities simultaneously and a community refinement procedure (VM) that reassigns nodes to neighboring communities to maximize modularity. At each iteration, the community join procedure merges all pairs of communities such that each pairwise community merging leads to a positive change in modularity. The iterations stop when all pairwise joins of communities yield a negative modularity change. After the convergence of the community merging procedure, a node refinement procedure is applied to reassign nodes to one of their neighboring communities if the node reassignment leads to the greatest modularity improvement. The MSG-VM algorithm results in more balancing community sizes and has a shorter execution time than the greedy modularity optimization algorithm [111]. The pseudo codes of the MSG procedure and the VM procedure are presented in Algorithm 7 and Algorithm 8.

Algorithm 7 Multistep Greedy (MSG) [111]

1: Assign each node in its own community
2: Calculate the modularity change matrix ΔQ with ΔQ_{ij} being the change in modularity if communities i and j are merged.
3: **while** there exists at least a pair of communities i and j with $\Delta Q_{ij} > 0$ **do**
4: **for** all pairs of communities u and v with $\Delta Q_{uv} > 0$ in the decreasing order of ΔQ_{uv} **do**
5: **if** communities u and v are unchanged in the current iteration **then**
6: Merge communities u and v
7: **end if**
8: **end for**
9: **end while**

2.4.3 The Louvain method

The Louvain method [15] is a two-step iterative algorithm to maximize modularity. The first step of the Louvain method is similar to the node refinement procedure (VM) in the MSG-VM algorithm. The first step starts with every node in its own community and then reassigns each node, in a sequence order, to the neighboring community with the maximal (positive) modularity improvement. The node reassignment procedure is repeated until every node stays in their own community after an iteration. The second step of the Louvain method is to build a new

Algorithm 8 Vertex Mover (VM) [111]

Input: An network partition
Output: The network partition with the maximal improvement in modularity

```
while true do
  for each node u in the order of increasing degree do
    v = the neighbor of u with the greatest change in modularity ΔQ when
    reassigning u to the community of v
    if ΔQ > 0 then
      Move node u into the community of v
    end if
  end for
  if no improvement in modularity can be achieved then
    return the current network partition
  end if
end while
```

network from the resulting network of the first step. Nodes in the new network are the resulting communities of the first step and edges in the new network are weighted by the sum of the weights of edges between the nodes in the corresponding communities. The first and the second steps are then re-applied to the new network until there is no further improvement in modularity. The Louvain method executes in a linear time with a computational complexity of $O(m)$ where m is the number of edges in the original network.

2.4.4 Walktrap

The Walktrap algorithm [98] defines a distance metric between two nodes in a network based on a random walk of a fixed length on the network. The node distance metric is expected to be larger for two nodes in different communities and smaller for two nodes in the same community. The node distance metric is then generalized to a distance metric between a node and its community. The Walktrap algorithm starts by assigning each node to a unique community. The algorithm then repeatedly merges a pair of communities that minimizes the average of the squared distances between each node and its community. When the last pair of communities is merged, the algorithm results in a dendrogram representing the hierarchical structure of the joined communities. The community division at the hierarchical level that gives the maximum modularity value is chosen as the community structure. The computational complexity of the Walktrap algorithm is $O(n^2 log(n))$.

2.4.5 Infomap

The Infomap algorithm [107] identifies communities by minimizing the map equation of the community division. This algorithm applies a similar heuristic to the Louvain method to search for the community division with the minimum value of the map equation. The resulting community division is then refined by moving nodes between communities to further decrease the map equation. The Infomap algorithm has an execution time of a few seconds for networks of the size about 10,000 nodes [107].

2.5 Evaluating community detection algorithms

Evaluating community detection algorithms basically means applying the detection algorithms to networks with known communities and quantifying their detection accuracies on these networks. The networks are generally computer-generated by generative models, or simply benchmarks.

2.5.1 Benchmarks

The two most popular benchmarks, that generate networks with built-in community structure, for evaluating the detection algorithms are the Girvan-Newman benchmark [40] and the Lancichinetti-Fortunato-Radicchi benchmark [65, 61].

2.5.1.1 The Girvan-Newman benchmark

The Girvan-Newman (GN) benchmark [40] generates a network with community structure as follows. The network is given 128 nodes divided into four communities of the same size. Edges are independently placed between nodes in the same community at random with probability p_{in} and placed between nodes in different communities at random with probability p_{out}. The values of p_{in} and p_{out} are chosen so that $p_{in} > p_{out}$ and the average degree of each node is 16. This implies that p_{in} and p_{out} are proportional to each other and $p_{in} + 3p_{out} = \frac{1}{2}$. The values of p_{in} and p_{out} can also be indirectly defined from the average number z_{in} of links from each node to nodes in the same community and the average number z_{out} of links from each node to nodes in other communities, with $z_{in} + z_{out} = 16$. Figure 2.3 demonstrates a realization of the GN benchmark.

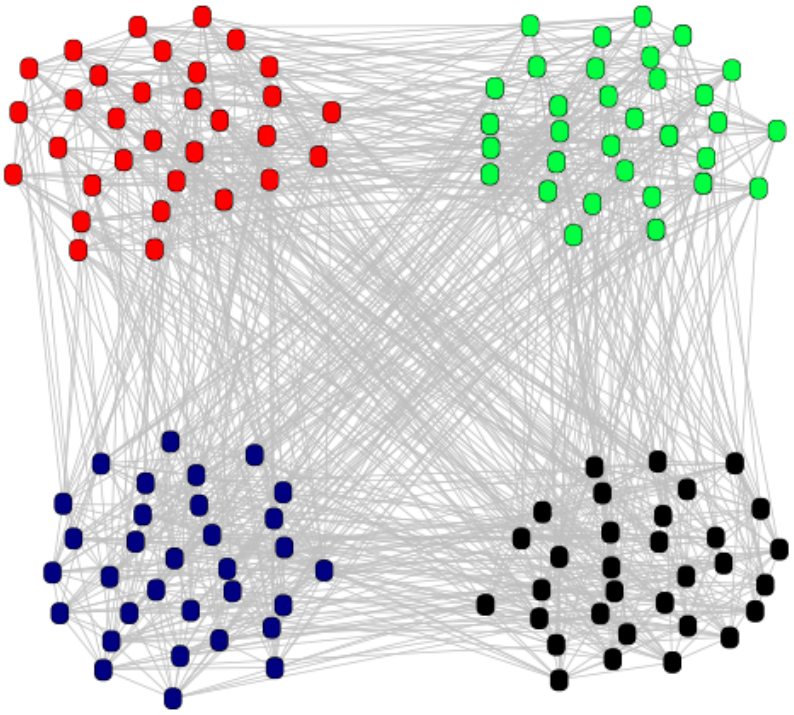


FIGURE 2.3: A realization of the GN benchmark.

The GN benchmark is widely used as a standard benchmark in the literature [36]. However, nodes in this benchmark have similar degrees and communities have similar sizes while real networks are observed to have heterogeneous distributions in node degree and community size [3]. Therefore, the GN benchmark is not a cohort representative for the general class of real networks with community structure.

2.5.1.2 The Lancichinetti-Fortunato-Radicchi benchmark

The Lancichinetti-Fortunato-Radicchi (LFR) benchmark [61] is an extension of the GN benchmark that accounts for the heterogeneity in node degree and community size. In the LFR benchmark, the properties of nodes and communities are given by network parameters. Links in the networks are generated by a construction algorithm to obtain the desired properties of nodes and communities. We will detail the construction process of the LFR benchmark in Chapter 5 as this benchmark is the foundation of our work in this thesis.

The LFR benchmark is the most widely used benchmark for evaluating community detection algorithms [36, 38]. However, one of the main limitations of this

benchmark is that communities in the benchmark are assigned the same fraction of external links, or mixing fraction, while real networks exhibit communities with heterogeneous mixing fractions [73, 74]. Moreover, each node in the LFR benchmark must belong to at least a community while real networks are observed to contain outliers which are considered as random noise and do not associate with any community [66].

2.5.2 Partition comparison metrics

The accuracy of a community detection algorithm on a synthetic network with community structure is measured by the similarity between the recovered network partition and the planted network partition corresponding to the community structure [83]. Let $X = (X_1, X_2, ..., X_{n_X})$ be the recovered network partition and $Y = (Y_1, Y_2, ..., Y_{n_Y})$ be the planted network partition, with n_X and n_Y denoting the numbers of communities in network partitions X and Y respectively. The similarity between network partitions X and Y can be measured using different approaches such as pair counting [103], [14] and information theory [82], [6].

2.5.2.1 Pair counting based metrics

The pair counting approach measures the similarity between two network partitions based on the number of pairs of nodes that are classified in the same or different communities in the two network partitions. Let N_{11} be the number of pairs of nodes which are in the same community in both network partitions and N_{00} be the number of pairs of nodes which are in the different communities in both network partitions. Let N_{10} be the number of pairs of nodes which are in the same community in network partition X and in different communities in network partition Y. Let N_{01} be the number of pairs of nodes which are in different communities in network partition X and in the same community in network partition Y.

The Rand index [103] that measures the similarity between X and Y is defined as

$$R(X, Y) = \frac{N_{11} + N_{00}}{N_{11} + N_{01} + N_{10} + N_{00}},$$

which is the ratio of the number of node pairs that are in the same or in different communities in both network partitions to the total number of node pairs of the

network. The Rand index is typically limited to a narrow range of values slightly below 1 due to the dominance of N_{00} in $R(X,Y)$ when N_{00} is large.

The Jaccard index [14] is similar to the Rand index but eliminates the contribution of N_{00} in the similarity measure. The Jaccard index that measures the similarity between X and Y is defined as follows:

$$J(X,Y) = \frac{N_{11}}{N_{11} + N_{01} + N_{10}},$$

which is the ratio between the number of node pairs in the same community in both network partitions and the number of node pairs in the same community in at least one network partition.

Pair counting based measures are often undesirable because these measures are sensitive to certain characteristics of communities such as the number of communities and community sizes [115].

2.5.2.2 Information theory based metrics

The information theory approach measures the similarity between two network partitions based on the Shannon information content [122] of the network partitions. Let n_i^X and n_j^Y be the numbers of nodes in community $X_i \in X$ and community $Y_j \in Y$ respectively. The probability that a randomly chosen node is in community $X_i \in X$ is

$$P(X_i) = \frac{n_i^X}{N}$$

and the probability that a randomly chosen node is in community $Y_j \in Y$ is

$$P(Y_j) = \frac{n_j^Y}{N},$$

where N is the number of nodes in the network.

The Shannon information of X is

$$\begin{aligned} H(X) &= -\sum_{i=1}^{n_X} P(X_i) log(P(X_i)) \\ &= -\frac{1}{N}\sum_{i=1}^{n_X} n_i^X \log(\frac{n_i^X}{N}) \end{aligned} \tag{2.5}$$

The Shannon information of Y is

$$\begin{aligned} H(Y) &= -\sum_{j=1}^{n_Y} P(Y_j) log(P(Y_j)) \\ &= -\frac{1}{N}\sum_{i=1}^{n_Y} n_i^Y \log(\frac{n_i^Y}{N}) \end{aligned} \tag{2.6}$$

The probability that a randomly chosen node is in community X_i and in community Y_j is

$$P(X_i, Y_j) = \frac{n_{ij}^{XY}}{N}$$

The mutual information [80, 115] of X and Y that measures the agreement between the two network partitions is defined as

$$I(X,Y) = \sum_{i=1}^{n_X}\sum_{j=1}^{n_Y} P(X_i, Y_j) log\left(\frac{P(X_i, Y_j)}{P(X_i)P(Y_j)}\right)$$

which, after simplification, is written as

$$I(X,Y) = \frac{1}{N}\sum_{i=1}^{n_X}\sum_{j=1}^{n_Y} n_{ij}^{XY} \log\left(\frac{n_{ij}^{XY}.N}{n_i^X.n_j^Y}\right) \tag{2.7}$$

Figure 2.4 demonstrates the mutual information I(X,Y) between two network partitions X and Y. Since the mutual information is not bounded by a constant value, this metric is not often a good way to measure the similarity between two partitions.

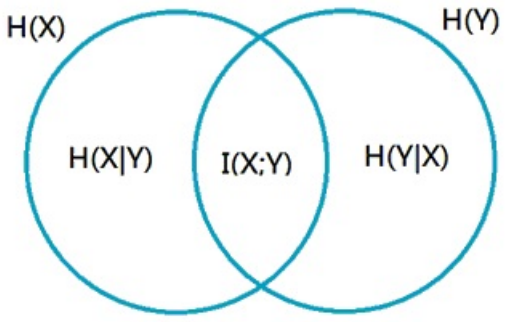


FIGURE 2.4: An information diagram illustrates the mutual information of X and Y

The normalized mutual information [6] between network partitions X and Y of a network is defined as

$$\begin{aligned} NMI(X,Y) &= \frac{2I(X,Y)}{H(X)+H(Y)} \\ &= -\frac{2\sum_{i=1}^{n_X}\sum_{j=1}^{n_Y} n_{ij}^{XY}\log(\frac{n_{ij}^{XY}.N}{n_i^X.n_j^Y})}{\sum_{i=1}^{n_X} n_i^X\log(\frac{n_i^X}{N})+\sum_{j=1}^{n_Y} n_j^Y\log(\frac{n_j^Y}{N})} \end{aligned} \tag{2.8}$$

The normalized mutual information has a range from 0 to 1. The value of 0 indicates that the two compared network partitions are totally dissimilar while the value of 1 indicates that the two network partitions are identical. The normalized mutual information is intuitively adopted as a reliable metric to compare network partitions [62].

2.6 Summary

Communities in a network are conceptually defined as groups of nodes with dense connections within each group and spare connections between different groups. Several quantitative definitions of community exist in the literature but none of them is commonly accepted yet. The most successful criteria to define communities, as interrelated parts of the whole network, include modularity [94] and the map equation [107].

The state-of-the-art algorithms for detecting communities include the Louvain method [15], which maximizes modularity, and Infomap [108], which minimizes the map equation. Community detection algorithms are typically evaluated on artificial networks with built-in community structure, or simply benchmark networks. The state-of-the-art benchmarks, that generate benchmark networks, for evaluating community detection algorithms include the GN benchmark [40] and the LFR benchmark [61]. The accuracy of a detection algorithm on a benchmark network is measured based on the similarity between the detected network partition and the planted network partition. The normalized mutual information [6] is currently often used for comparing the network partitions.

For further details of research in community detection, readers are referred to [99, 36, 24, 81, 124, 110, 38].

Chapter 3

Modularity Optimization in Networks with Weak Communities

This chapter presents our approach to develop an algorithm for maximizing modularity in networks with weak community structure. It is noted that, in this thesis, we refer networks with strong community structure to networks of communities which have more intra-community connections than inter-community connections and refer networks with weak community structure to networks with communities which have more inter-community connections than intra-community connections. First, we review the three existing algorithms, LPA [102], LPAm [12], and the state-of-the-art algorithm LPAm+ [75], which are the foundations of our approach. Second, we give an example demonstrating the local maxima problem of LPAm+ on networks with weak community structure. We then propose an algorithm which extends the LPAm+ algorithm by introducing a method to overcome local maxima. Afterward, we extend our proposed algorithm to identify communities in directed networks. Since the proposed algorithm has two parameters to set, we perform an empirical analysis to investigate the dependence of the algorithm on its parameters. Finally, we analyze the computational complexity of the proposed algorithm to evaluate its running time. Parts of the content of this chapter have been published in [68].

3.1 Preliminaries

3.1.1 The LPA algorithm

The label propagation algorithm (LPA) [102] is known as a time-efficient method for detecting communities. LPA identifies communities by propagating labels of nodes over the network. LPA [102] works by initially assigning a unique label to every node in the network. The algorithm then iteratively updates the labels of nodes in a random sequential order. At each iteration, each node updates its label by the label that the maximum number of its neighbors hold. If a node has many candidate labels to update, one of the labels is chosen randomly. The label updating rule for a node u in an undirected and unweighted network of n nodes can be mathematically specified as

$$l_u^{new} = \underset{l'_u}{\operatorname{argmax}} \sum_{v=1}^{n} A_{uv}\delta(l'_u, l_v),$$

where l_u^{new} is the new label to be assigned to node u, l'_u is a label of the neighbors of u, v is a node in the network, l_v is the label of node v, A_{uv} is the element of the adjacency matrix of the network representing the connection between node u and node v, and δ is the Kronecker delta function which is 1 if l_v and l'_u are the same, and 0 otherwise.

The label updating process stops when node labels remain unchanged after an iteration. Communities are identified as groups of nodes holding the same labels. To ensure the convergence of the algorithm, node labels are updated asynchronously which means that each node updates its label based on the labels in the previous iteration of some of its neighbors and the labels in the current iteration of the other neighbors. LPA has a near linear time complexity of $O(m)$, where m is the number of edges in the network [102].

The objective function of LPA can be understood as finding the community division of the network that maximizes the number of edges falling within communities [12]. The trivial solution of LPA is to assign the same label to every node in the network. This illustrates a potential drawback of LPA that the detected community structure does not necessarily have any meaningful interpretation [12]. The pseudo code of the LPA algorithm is presented in Algorithm 9.

Algorithm 9 LPA [102]

```
Initialize each node with a unique label
while true do
    for each node u with label l in a sequential order of nodes do
        l' = the neighboring label of u shared by the maximum number of the
        neighbors of u
        if l' ≠ l then
            Assign label l' to node u
        end if
    end for
    if labels of every node remain unchanged after an iteration then
        return the current node label assignment
    end if
end while
/* Communities are groups of nodes with the same label */
```

3.1.2 The LPAm algorithm

The modularity-specialized label propagation algorithm (LPAm) [12] is an enhanced version of LPA that improves the quality of the detected community structure by propagating labels of nodes to maximize network modularity. LPAm [12] modifies the label updating rule in LPA to drive the solution toward the maximum modularity value [94]. Starting from an initial community division, LPAm performs a label propagation step at each iteration. The label propagation step updates the label of each node by a label of its neighbors that leads to the maximal increase in network modularity. Let l_u be the current label of a node u, and l'_u be the new label of node u after updating. Let g_u and g'_u denote the communities of nodes holding label l_u and l'_u respectively.

Lemma 3.1. *The gain in network modularity when updating label l'_u for node u is*

$$\Delta Q = \frac{1}{m}\sum_{j\in g'_u}\left(A_{uj} - \frac{k_u k_j}{2m}\right) - \frac{1}{m}\sum_{j\in g_u - u}\left(A_{uj} - \frac{k_u k_j}{2m}\right)$$

Proof. The gain in network modularity when updating label l'_u for node u is

$$\begin{aligned}\Delta Q \;=\; & \frac{1}{2m}\left(\sum_{i,j\in g'_u + u}\left(A_{ij} - \frac{k_i k_j}{2m}\right) + \sum_{i,j\in g_u - u}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\right) \\ & -\frac{1}{2m}\left(\sum_{i,j\in g'_u}\left(A_{ij} - \frac{k_i k_j}{2m}\right) + \sum_{i,j\in g_u}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\right), \end{aligned} \tag{3.1}$$

where $g_u - u$ denotes the community formed by removing node u from its community, and $g'_u + u$ denotes the community formed by adding node u into community g'_u.

We can rewrite Eq. 3.1 as

$$\begin{aligned}\Delta Q &= \frac{1}{2m}\left(\sum_{i,j\in g'_u+u}\left(A_{ij}-\frac{k_i k_j}{2m}\right)-\sum_{i,j\in g'_u}\left(A_{ij}-\frac{k_i k_j}{2m}\right)\right)\\ &\quad -\frac{1}{2m}\left(\sum_{i,j\in g_u}\left(A_{ij}-\frac{k_i k_j}{2m}\right)-\sum_{i,j\in g_u-u}\left(A_{ij}-\frac{k_i k_j}{2m}\right)\right)\end{aligned}$$

or equivalently,

$$\Delta Q = \frac{1}{m}\sum_{j\in g'_u}\left(A_{uj}-\frac{k_u k_j}{2m}\right)-\frac{1}{m}\sum_{j\in g_u-u}\left(A_{uj}-\frac{k_u k_j}{2m}\right) \tag{3.2}$$

□

Since the second term of Eq. 3.2 remains unchanged for every choice of label l'_u, choosing label l'_u that maximizes ΔQ is equivalent to choosing label l'_u that maximizes the sum

$$\sum_{j\in g'_u}\left(A_{uj}-\frac{k_u k_j}{2m}\right)$$

Therefore, the label updating rule of LPAm can be expressed as

$$l_u^{new} = \underset{l'_u}{\operatorname{argmax}}\sum_{v=1}^{n}\left(A_{uv}-\frac{k_u k_v}{2m}\right)\delta(l'_u, l_v)$$

As LPAm employs a greedy search strategy for finding the optimal community division regarding modularity, this algorithm easily get trapped in a local maximum of the modularity search space [75]. The pseudo code of the LPAm algorithm is presented in Algorithm 10.

3.1.3 The LPAm+ algorithm

The advanced modularity-specialized label propagation algorithm (LPAm+) [75] is a further improvement of LPAm. LPAm+ [75] drives LPAm out of local maxima

Algorithm 10 LPAm [12]

```
Initialize each node with a unique label
while true do
    for each node u in a sequential order of nodes do
        l' = the neighboring label of u with the greatest change in modularity ΔQ
        when updating label l' for node u
        if ΔQ > 0 then
            Assign label l' to node u
        end if
    end for
    if labels of every node remain unchanged after an iteration then
        return the current node label assignment
    end if
end while
/* Communities are groups of nodes with the same label */
```

by iteratively combining LPAm with merging pairs of communities that improve network modularity the most. LPAm+ utilizes the multi-step greedy algorithm (MSG) in [111] to merge multiple pairs of communities simultaneously at a time [75]. After performing iterations of LPAm, two communities having labels l_1 and l_2 are merged if

$$(\Delta Q_{l_1 \to l_2} > 0) \wedge [!\exists l : (\Delta Q_{l_1 \to l} > \Delta Q_{l_1 \to l_2}) \vee (\Delta Q_{l_2 \to l} > \Delta Q_{l_1 \to l_2})]$$

with $\Delta Q_{l_1 \to l_2}$ denoting the gain in network modularity when community having label l_1 is merged with community having label l_2. The algorithm stops when no pair of communities can be merged to increase network modularity. Let g be the community labeled l_1 and g' be the community labeled l_2.

Lemma 3.2. *The gain in network modularity when merging community labeled l_1 and community labeled l_2 is*

$$\Delta Q_{l_1 \to l_2} = \frac{1}{m}\left(I_{l_1 \to l_2} - \frac{D_{l_1} D_{l_2}}{2m}\right), \tag{3.3}$$

where D_{l_1} and D_{l_2} are the total degree of nodes in community labeled l_1 and the total degree of nodes in community labeled l_2 respectively, and $I_{l_1 \to l_2}$ is the total number of edges between these two communities.

Proof. The gain in network modularity when merging community g labeled l_1 and community g' labeled l_2 is

$$\Delta Q_{l_1 \to l_2} = \frac{1}{2m} \sum_{i,j \in g' \cup g} \left(A_{ij} - \frac{k_i k_j}{2m} \right) - \frac{1}{2m} \left(\sum_{i,j \in g'} \left(A_{ij} - \frac{k_i k_j}{2m} \right) + \sum_{i,j \in g} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \right),$$

where $g' \cup g$ denotes the community formed by merging community labeled l_1 and community labeled l_2.

After simplifying, we have

$$\Delta Q_{l_1 \to l_2} = \frac{1}{m} \sum_{i \in g} \sum_{j \in g'} \left(A_{ij} - \frac{k_i k_j}{2m} \right) = \frac{1}{m} \left(I_{l_1 \to l_2} - \frac{D_{l_1} D_{l_2}}{2m} \right)$$

□

We demonstrate the pseudo code of the LPAm+ algorithm in Algorithm 11.

Algorithm 11 LPAm+ [75]

```
Assign every node into its own community
Maximize network modularity by LPAm
while ∃ a pair of communities (l1, l2) with ΔQ_{l1,l2} > 0 do
    for every community pair(l1, l2) that (ΔQ_{l1→l2} > 0) ∧ [!∃l : (ΔQ_{l1→l} > ΔQ_{l1→l2}) ∨ (ΔQ_{l2→l} > ΔQ_{l1→l2})] do
        Merge communities l1 and l2
    end for
    Maximize network modularity by LPAm
end while
/* ΔQ_{l1,l2} is the change in network modularity when merging community labeled 'l1' and community labeled 'l2'
The embedded LPAm procedure do not reinitialize the community assignment */
```

3.2 The local maximum problem of LPAm+

We demonstrate below an example in which LPAm+ gets trapped in a poor local maximum on networks with weak community structure. Figure 3.1a illustrates a toy network with two intuitively divided communities (one community with nodes in white color and the other community with nodes in dark color) and the modularity Q is 0.2117. Applying LPAm+ to this toy network, the first round of LPAm in LPAm+ results in a local maximum with modularity $Q = 0.1097$

(Figure 3.1b), for the initial community division of nodes in their own community and the sequence of node orders $\{6, 0, 9, 5, 7, 11, 10, 1, 8, 2, 3, 4\}$ to update the node labels. The first round of merging communities in LPAm+ further improves the network modularity by merging community labeled 'a' and community labeled 'd', and merging community labeled 'c' and community labeled 'e' with the new modularity value $Q = 0.1607$ (Figure 3.1c). Note that this modularity value is

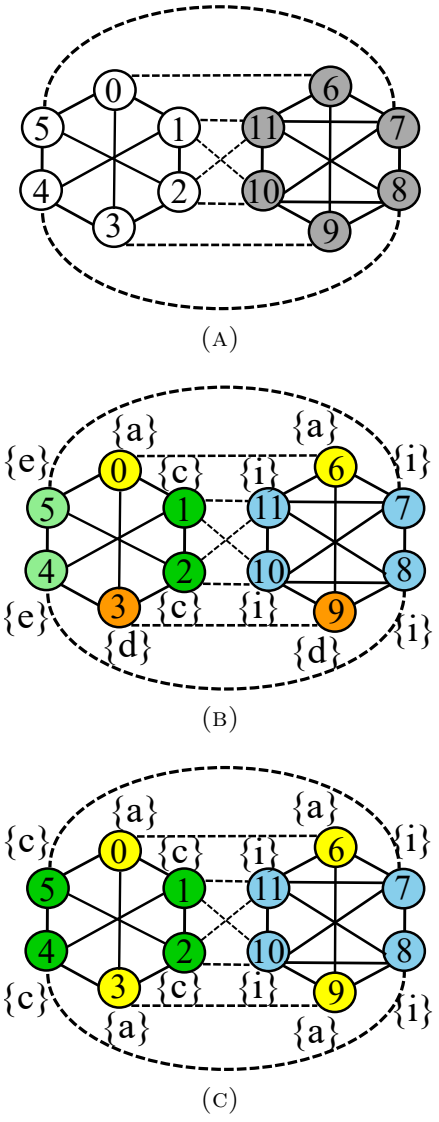


FIGURE 3.1: A toy network consists of two intuitively divided communities with modularity $Q = 0.2117$ (a). The first round of LPAm in LPAm+ results in a local maximum with modularity $Q = 0.1097$ (b). The first round of merging communities in LPAm+ merges community labeled 'a' and community labeled 'd', and merges community labeled 'c' and community labeled 'e' with the new modularity value $Q = 0.1607$ (c). LPAm+ gets trapped in this local maximum because the misplaced nodes 0, 3, 6 and 9 cannot be adjusted by any other round of LPAm or merging communities.

still significantly below the modularity value of 0.2117 achieving in Figure 3.1a. LPAm+ gets trapped in the local maximum $Q = 0.1607$ mainly due to the initially misplaced nodes 0, 3, 6 and 9, in Figure 3.1b. These nodes are assigned to the same community with the nodes which actually belong to different communities with respect to the global optimal community solution. LPAm+ does not have any mechanism to correct this misplacement of node labels and cannot adjust the labels in any other round of LPAm or merging communities.

3.3 Meta-LPAm+: an extension of LPAm+ to networks with weak community structure

We propose a method to overcome the drawback of LPAm+ demonstrated in the previous section as follows. We introduce a label propagation procedure that follows a guided search strategy [16] to avoid local maxima before merging communities. The new label propagation procedure is to search for a better local maximum than that given by LPAm to correct the misplaced nodes in the intermediate solutions given by LPAm. Guided search strategies, also called meta-heuristics, can better avoid local maxima than greedy search strategies, also called heuristics, by temporarily accepting worse solutions to explore more thoroughly the solution space.

The meta-heuristic employed by the new label propagation procedure is inspired by the record-to-record travel (RRT) algorithm [34] for a balance between performance and computational complexity. The RRT algorithm is a variant of the simulated annealing (SA) algorithm [56] with a different mechanism to accept worse solutions. The RRT algorithm accepts a worse solution if the difference between the accepted solution and the best solution found is less than a specific threshold. The RRT algorithm is reported to outperform the SA algorithm with much lower running time for some optimization problems [34]. To the best of our knowledge, the RRT algorithm has never been applied to the task of community detection.

To escape local maxima, we first perform a label propagation step that accepts a decrease in network modularity. In this step, the label of each node is updated sequentially by a label of its neighbors if the decrease in network modularity is less than a predetermined threshold DEV. The decrease in network modularity is calculated as the difference between the current modularity value and the highest

modularity value found. After accepting a worse solution, the new label propagation procedure performs a round of LPAm to quickly improve network modularity. The process is repeated until the number of iterations without improvement in the highest network modularity value found exceeds a predefined number $\max_{no}$ of iterations.

The improved LPAm+ algorithm, named meta-LPAm+, is basically an iterative combination of the modularity-specialized label propagation algorithm LPAm in [12], the meta-heuristic based label propagation algorithm, named meta-LPAm, and the community merging algorithm based on the MSG algorithm in [111]. The pseudo code of meta-LPAm and meta-LPAm+ are presented in **Algorithm 12**

Algorithm 12 Meta-LPAm

Input: An initial community division S, the threshold DEV and the maximum number of iterations without improvement in modularity max_{no}
Output: The best found community division RECORD

Set RECORD = S and $n_{no} = 0$
While $n_{no} < \max_{no}$
    Update the label of each node sequentially by a label of its neighbors if
    the new network modularity $Q_{S'} \geq Q_{RECORD} - DEV$
    Maximize network modularity by LPAm
    Set $n_{no} = n_{no} + 1$
    If $Q_S > Q_{RECORD}$ Then
        Set RECORD = S and $n_{no} = 0$
    End if
End While

/* S' is the new community division to assign to S if the node label is updated */

Algorithm 13 Meta-LPAm+

Input: The threshold DEV and the maximum number of iterations without improvement in modularity $\max_{no}$ for the embedded meta-LPAm algorithm
Output: The best found community division

Assign every node into its own community
Maximize network modularity by LPAm
Find a better local maximum by meta-LPAm
While $\exists$ a pair of communities (l_1, l_2) with $\Delta Q_{l_1,l_2} > 0$
    Merging pairs of communities by MSG
    Maximize network modularity by LPAm
    Find a better local maximum by meta-LPAm
End While

/* $\Delta Q_{l1,l2}$ is the change in network modularity when merging community labeled 'l1' and community labeled 'l2'
The LPAm and MSG algorithms do not reinitialize the community assignment */

and **Algorithm 13**.

Figure 3.2 illustrates how meta-LPAm+ converges to the global maximum for the toy network in Figure 3.1a. Meta-LPAm+ starts by applying the first round of LPAm to produce the local maximum in Figure 3.1b. Meta-LPAm+ then applies the first round of meta-LPAm with the input parameters DEV = 0.01

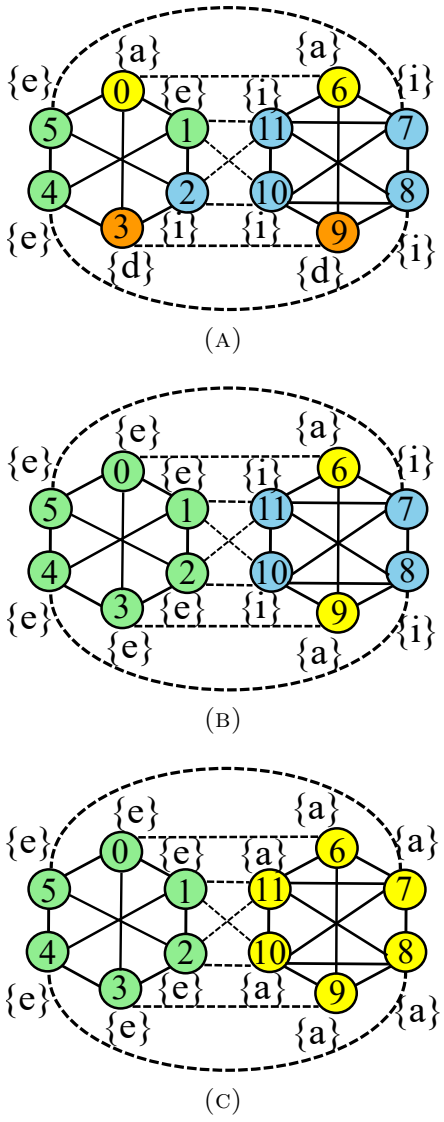


FIGURE 3.2: A pathway to converge to the global maximum of meta-LPAm+ for the toy network in Figure 3.1a. Giving the community solution in Figure 3.1b as the local maximum returned by the first round of LPAm in meta-LPAm+, the first round of meta-LPAm in meta-LPAm+ improves this local maximum by: first performing a label propagation step that decreases modularity by 0.0083 and the modularity is 0.1014 (a), and then performing a round of LPAm to reach to another local maximum with modularity $Q = 0.1811$ (b). The first round of merging communities in meta-LPAm+ results in the global maximum with modularity $Q = 0.2117$ (c).

and $\max_{no} = 50$ to escape this local maximum by the following steps. First, the first round of meta-LPAm performs a label propagation step that assigns label 'e' for node 1 and label 'i' for node 2 with modularity decreased by 0.0083 and the modularity is 0.1014. The next round of LPAm embedded in meta-LPAm then reaches to a better local maximum with modularity $Q = 0.1811$. The first round of merging communities in meta-LPAm+ merges community labeled 'a' and community labeled 'i' to increase modularity to 0.2117. The algorithm stops at the global maximum solution because the network modularity cannot be improved by any further round of LPAm, meta-LPAm, or merging communities.

3.4 Extending meta-LPAm+ to directed networks

Many networks in the real world, such as networks of the World Wide Web [67], are directed. For detecting communities in directed networks, our approach follows the principal idea that the quality of the community structure of a directed network is measured by the difference between the total fraction of directional edges within communities in the directed network and that expected fraction of directional edges in an equivalent directed random network [72]. In directed networks, edge directions affect the chance to have directed edges among nodes [81]. Therefore, the original definition of modularity in Eq. 2.1, which ignores the directionality of edges, is not a proper measure of the quality of community structure in directed networks. In other words, maximizing modularity would not find the meaningful community structure of directed networks [72].

Directed modularity [7, 72] is an extension of modularity to directed networks that takes into account edge directions.

Definition 6. [7, 72] The directed modularity of a community division of a directed network is defined as

$$Q^d = \frac{1}{m} \sum_g \sum_{i,j \in g} \left(A_{ij} - \frac{k_i^{out} k_j^{in}}{m} \right) \tag{3.4}$$

where g is a community in the network, k_i^{out} is the total number of outgoing edges from node i, k_j^{in} is the total number of incoming edges to node j, and m is the total number of directional edges in the network.

Therefore, we extend our proposed algorithm meta-LPAm+ to the directed case by employing directed modularity as the objective function to maximize. This

requires only slight modifications to the changes in the objective function when updating node labels and merging communities. For updating node labels, the gain in directed modularity when updating label l'_u for node u with label l_u is

$$\begin{aligned}\Delta Q^d &= \frac{1}{m}\sum_{j\in g'_u}\left(A_{uj}-\frac{k_u^{out}k_j^{in}}{m}\right)+\frac{1}{m}\sum_{j\in g'_u}\left(A_{ju}-\frac{k_j^{out}k_u^{in}}{m}\right)\\ &\quad -\frac{1}{m}\sum_{j\in g_u-u}\left(A_{uj}-\frac{k_u^{out}k_j^{in}}{m}\right)-\frac{1}{m}\sum_{j\in g_u-u}\left(A_{ju}-\frac{k_j^{out}k_u^{in}}{m}\right),\end{aligned} \tag{3.5}$$

with g_u and g'_u denoting the communities of nodes holding label l_u and l'_u respectively,

As the third term and the last term in Eq. 3.5 are unchanged for any choice of g'_u, the rule for updating node labels in the directed network that leads to the maximal increase in directed modularity can be written as

$$l_u^{new}=\underset{l'_u}{\operatorname{argmax}}\sum_{v=1}^{n}\left(\left(A_{uv}-\frac{k_u^{out}k_v^{in}}{m}\right)+\left(A_{vu}-\frac{k_v^{out}k_u^{in}}{m}\right)\right)\delta(l'_u,l_v),$$

where l'_u is a label of neighbors of u and l_v is the label of node v.

For merging communities, the gain in directed modularity when merging community g labeled l_1 and community g' labeled l_2 is

$$\Delta Q^d_{l_1\to l_2} = \frac{1}{m}\sum_{i\in g}\sum_{j\in g'}\left(A_{ij}-\frac{k_i^{out}k_j^{in}}{2m}\right)+\frac{1}{m}\sum_{i\in g}\sum_{j\in g'}\left(A_{ji}-\frac{k_i^{in}k_j^{out}}{2m}\right)$$

or, equivalently,

$$\Delta Q^d_{l_1\to l_2}=\frac{1}{m}\left(I_{l_1\to l_2}-\frac{D_{l_1}^{out}D_{l_2}^{in}}{m}\right)+\frac{1}{m}\left(I_{l_2\to l_1}-\frac{D_{l_2}^{out}D_{l_1}^{in}}{m}\right),$$

with $I_{l_1\to l_2}$ denoting the total number of directional edges from community labeled l_1 to community labeled l_2, and $D_{l_1}^{out}$ and $D_{l_2}^{in}$ are the total numbers of outgoing edges from nodes in community labeled l_1 and incoming edges to nodes in community labeled l_2 respectively.

3.5 Setting parameters

As with many other meta-heuristics [116], our proposed algorithm meta-LPAm+ has parameters to guide the search process. The values of the parameters DEV and $\max_{no}$ of meta-LPAm+ determine the trade-off between performance, in term of modularity, and running time of the algorithm. For example, while LPAm+ finds the global optimal solution for the toy network in Figure 3.1a about 50 times out of 100 runs with randomized initial solutions, the number of times that meta-LPAm+ finds the global maximum depends on the setting of the parameters. Meta-LPAm+ finds the global maximum about 70 times over 100 runs for $DEV = 0.01$ and $\max_{no} = 10$. However, for $DEV = 0.1$ and $\max_{no} = 10$, meta-LPAm+ finds the global maximum in almost all of the 100 runs of the algorithm. Therefore, determining the dependence of meta-LPAm+ on its parameters DEV and $\max_{no}$ is important to locate near optimal values of the parameters for use in the algorithm.

Meta-LPAm+ drives the LPAm+ algorithm [75] out of a local maximum by performing the meta-LPAm algorithm. In each iteration, the meta-LPAm algorithm first randomly perturbs the current local maximum solution and then greedily improves the perturbed solution to find a better local maximum than the previous one. The perturbation amount is mainly defined by the parameter DEV while the iteration time is mainly defined by the parameter max_{no}. For a given a value of DEV, the larger the value of max_{no}, the longer the iteration time is. The longer the iteration time, the more likely it is that meta-LPAm+ finds a better solution since the algorithm has more iterations to escape local maxima. The parameter max_{no} should, therefore, be set to the value that leads to the maximum allowed running time of the algorithm. For a given value of max_{no}, choosing a too small value for DEV could cause the algorithm to explore only a small number of neighbor solutions and, therefore, cannot find the best possible neighbor solution. On the other hand, choosing a too large value for DEV could cause the algorithm to explore too many neighborhood solutions and, therefore, cannot converge to the best possible neighbor solution within the allowed running time.

To validate the dependence of our proposed algorithm meta-LPAm+ on the parameters DEV and max_{no}, we evaluate the proposed algorithm on artificial networks for many combinations of these parameters. The artificial networks are generated by the Lancichinetti-Fortunato-Radicchi (LFR) benchmark [65], which is a commonly-used benchmark to generate networks with community structure [36]. The network parameters of the LFR benchmark are given as follows: the number of nodes in a network $N = 1000$; the average node degree $\overline{k} = 20$; the maximum

TABLE 3.1: Average modularity obtained by meta-LPAm+ on the artificial networks for DEV from 0.01 to 0.01 and max_{no} from 10 to 100.

Parameter values		max_{no}									
		10	20	30	40	50	60	70	80	90	100
DEV	0.01	0.248	0.249	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.251
	0.02	0.249	0.25	0.25	0.251	0.251	0.251	0.251	0.251	0.251	0.251
	0.03	0.25	0.25	0.251	0.251	0.251	0.251	0.252	0.251	0.252	0.252
	0.04	0.249	0.25	0.251	0.251	0.251	0.251	0.251	0.251	0.252	0.252
	0.05	0.249	0.251	0.251	0.251	0.251	0.251	0.252	0.252	0.252	0.252
	0.06	0.25	0.25	0.251	0.251	0.251	0.251	0.251	0.252	0.252	0.252
	0.07	0.249	0.25	0.25	0.251	0.251	0.251	0.251	0.251	0.252	0.251
	0.08	0.249	0.25	0.25	0.25	0.251	0.25	0.25	0.251	0.251	0.251
	0.09	0.248	0.249	0.249	0.25	0.25	0.25	0.25	0.25	0.25	0.251
	0.1	0.247	0.248	0.248	0.248	0.249	0.249	0.249	0.249	0.249	0.249

TABLE 3.2: Running time (in seconds) of meta-LPAm+ on the artificial networks for DEV from 0.01 to 0.1 and max_{no} from 10 to 100.

Parameter values		max_{no}									
		10	20	30	40	50	60	70	80	90	100
DEV	0.01	0.71	1.28	1.68	2.02	2.47	2.93	3.39	3.89	3.93	4.55
	0.02	0.66	1.14	1.44	1.82	2.39	2.6	3.02	3.66	3.69	4.15
	0.03	0.62	1.05	1.53	1.75	2.43	2.56	3.01	3.07	3.6	3.83
	0.04	0.58	0.99	1.5	1.82	2.18	2.57	3.03	3.09	3.5	3.95
	0.05	0.59	1	1.4	1.8	2.02	2.47	2.95	3.34	3.74	4.08
	0.06	0.62	1.1	1.43	1.74	2.35	2.35	2.96	3.54	3.81	4.09
	0.07	0.72	1	1.43	1.82	2.34	2.67	2.86	3.14	3.68	4.05
	0.08	0.69	1.05	1.32	1.73	2.37	2.66	2.91	3.37	3.77	4.03
	0.09	0.65	0.99	1.42	1.79	2.17	2.77	3.12	3.44	3.69	4.02
	0.1	0.62	1.01	1.37	1.72	2.26	2.56	3.08	3.37	3.75	3.87

node degree $k_{max} = 50$; the minimum community size $s_{min} = 20$; the maximum community size $s_{max} = 100$; the power-law exponent for the node degree sequence $\gamma = -2$; the power-law exponent for the community size sequence $\beta = -2$, and the fraction of external links of each node, with respect to its community, $\mu = 0.7$.

Table 3.1 and Table 3.2 respectively show the average of network modularity obtained by meta-LPAm+ and its running time on the artificial networks for the value of max_{no} varying from 10 to 100 and the value of DEV varying from 0.01 to 0.1. As can be seen from the tables, for a given value of DEV, setting a larger value for $\max_{no}$ generally allows the algorithm to find a higher modularity value with additional running time. For a given value of max_{no}, setting a too small or too large value for DEV generally leads to a decrease in the modularity value found by the algorithm with respect to the modularity value found for a moderate value of DEV. For example, given $max_{no} = 80$, the modularity value found by the algorithm for $DEV = 0.01$ and $DEV = 0.1$ are lower than the modularity value found for $DEV = 0.05$.

The results confirm that the performance of our proposed algorithm is dependent on two parameters DEV and max_{no}. For the algorithm to achieve high performance, the parameter max_{no} should be set to the value corresponding to the largest allowed running time of the algorithm while the parameter DEV should be set to an appropriate value that is not too small and not too large. We leave the problem of systematically estimating near-optimal values for these parameters for future work. In our experiments in this thesis, the values of DEV and max_{no} are manually chosen based on experience.

3.6 Computational complexity

The computational complexity of meta-LPAm+ is mainly from the complexity of the three components embedded in the algorithm: (1) the modularity-specialized label propagation algorithm LPAm, (2) the meta-heuristic based label propagation algorithm meta-LPAm, and (3) the community merging algorithm MSG.

One round of LPAm in meta-LPAm+ has a computational complexity of $rO(m)$, with r being the average number of label propagation steps required for the round of LPAm to converge [12]. One round of meta-LPAm in meta-LPAm+ takes a computational cost of $l(O(m) + rO(m))$, with l being the average number of times that the round of meta-LPAm accepts a worse solution. The combination of LPAm

and meta-LPAm in meta-LPAm+ thus has a computational complexity of $sO(m)$ with $s = r + l(1 + r)$ being the average number of label propagation steps needed for the round of LPAm and meta-LPAm to stop iterations. The computational cost of one round of MSG in meta-LPAm+ is $O(m\log N)$ [111]. Therefore, the total computational cost of meta-LPAm+ is

$$s_1 O(m) + h(O(m\log N) + s_2 O(m))$$

where s_1 is the average number of label propagation steps of the round of LPAm and meta-LPAm before the while loop in meta-LPAm+, s_2 is the average number of label propagation steps of the round of LPAm and meta-LPAm in the while loop, and h is the average number of iterations for the while loop.

Table 3.3 shows the values of s_1, s_2 and h when applying meta-LPAm+ on the LFR benchmark networks. According to the tables, the values of s_1 and s_2 are almost stable for $\mu \leq 0.5$, where communities generally have more intra-community links than inter-community links, whereas the values of s_1 and s_2 are much larger for $\mu \geq 0.6$, where communities have more inter-community links than intra-community links. This evidently indicates that LPAm results in community solutions with more misplaced nodes on networks with weak community structure leading to the subsequent increase in the number of iterations needed for the round of LPAm and meta-LPAm to converge.

TABLE 3.3: The values of s_1, s_2 and h when applying meta-LPAm+ on the LFR benchmark networks.

LFR benchmark networks	N=1000			N=5000		
	s_1	s_2	h	s_1	s_2	h
$\mu = 0.1$	154.60	0	0	157.60	152.0	1.60
$\mu = 0.2$	156.10	0	0	158.60	152.0	2.10
$\mu = 0.3$	156.70	45.60	0.30	159.10	152.0	2.20
$\mu = 0.4$	159.20	76.0	0.50	166.30	152.0	2.50
$\mu = 0.5$	162.90	106.40	0.70	170.900	152.0	2.90
$\mu = 0.6$	212.60	107.95	0.80	262.20	152.0	3.40
$\mu = 0.7$	1134.60	492.0	1.20	498.60	267.44	3.80
$\mu = 0.8$	1605.90	185.0	0.40	2859.50	444.50	0.70

Table 3.4 reports the values of s_1, s_2, and h when applying meta-LPAm+ on the LFR benchmark networks of different sizes for $\mu = 0.7$ and $\mu = 0.8$, where s_1 and s_2 are expected to be maximum on the networks of the same size with different μ. As can be seen from Table 3.4, s_1 and s_2 are essentially upper-bounded by N and a small fraction (about $\frac{1}{10}$) of N respectively. Therefore, we can safely set $s_1 = O(N)$

TABLE 3.4: The values of s_1, s_2 and h when applying meta-LPAm+ on the LFR benchmark networks of different sizes for $\mu = 0.7$ and $\mu = 0.8$.

LFR benchmark networks	$\mu = 0.7$			$\mu = 0.8$		
	s_1	s_2	h	s_1	s_2	h
N=1000	1208.32	395.67	1.05	1535.29	114.35	0.18
N=2000	998.69	412.10	2.58	2069.66	98.35	0.11
N=3000	775.20	324.07	3.04	2313.99	280.67	0.34
N=4000	654.94	280.86	3.42	2694.37	364.51	0.52
N=5000	543.48	266.60	3.74	3317.09	464.60	0.77
N=6000	503.91	241.29	4.09	3599.37	650.20	1.23
N=7000	503.76	230.22	4.24	3674.05	828.16	1.54
N=8000	512.91	224.34	4.39	4106.22	874.32	1.99
N=9000	503.21	219.22	4.68	4479.50	773.13	2.20
N=10000	499.88	218.94	4.84	4810.25	793.28	2.69

and $s_2 = O(N)$. The values of h are almost bounded by a small constant and therefore h can be estimated to be $\log(n)$, the depth of the dendrogram describing the hierarchical decomposition of a network with balanced hierarchical community structure into communities [75]. Therefore, the total computational complexity of meta-LPAm+ is

$$O(mN) + \log N(O(m\log N) + O(mN))$$

or equivalently, $O(mN\log N)$.

3.7 Summary

In this chapter, we proposed a new algorithm for maximizing modularity in networks with weak community structure, where communities have more inter-community connections than intra-community connections. Our proposed algorithm, called meta-LPAm+, extends the state-of-the-art algorithm LPAm+ [75] by introducing a method to escape local maxima.

The LPAm+ algorithm maximizes modularity by iteratively performing two procedures: the first procedure is to propagate labels of nodes over the network to greedily maximize modularity and the second procedure is to merge communities to further improve modularity. We identified that LPAm+ easily results in a poor local maximum on networks with weak community structure because the community merging procedure cannot efficiently drive the algorithm out of the local maximum reached by the label propagation procedure.

Our proposed algorithm overcomes this drawback of LPAm+ by introducing a method to escape local maxima before merging communities. Our method follows a guided search strategy, or meta-heuristic, inspired by the record-to-record travel algorithm [34] for a balance between performance and running time. To apply our algorithm to directed networks, we employed directed modularity [7, 72] as the objective function to maximize in the algorithm. Our algorithm is dependent on two parameters, DEV and max_{no}. For the algorithm to yield high performance, max_{no} is suggested to set to the value that leads to the maximum allowed running time of the algorithm while DEV needs to be chosen carefully, not too small or too large. Given certain parameter values, our algorithm achieves a computational complexity of $O(mN\log N)$ where m is the number of edges and N is the number of nodes in the network.

It is noted that our proposed algorithm has a limitation, called the resolution limit [37], which is induced by using modularity to evaluate the quality of community divisions. Modularity is basically the sum of individual components, with each component being the difference between the fraction of edges in a community and that expected fraction of edges in an equivalent random network [94]. The optimal modularity value, therefore, depends on the choice of the best trade-off between the number of communities and the contribution values of the individual components. When several existing smaller communities could form a larger community to give a higher contribution to modularity, modularity optimization favors the community division with the larger one. This implies that actual communities of small sizes may not be resolved, especially in large networks with heterogeneous community sizes [62].

To solve the resolution limit of modularity optimization, one of the possible ways is to rescale the network topology to increase the contributions of small size-scale communities to modularity. The authors in [8] propose a rescaling method that assigns to each node a self-edge with the edge weights given by a parameter. This rescaling method increases the total weight of edges in a community more significantly than that expected value in the random case [8]. The weighted network topology is then used to determine the community structure by optimizing the weighted version of modularity [88]. Small size-scale communities are more likely to be identified as they have larger contributions to the weighted modularity value. In [60], the authors rescale a network by assigning more weight to the edges in the same community than to the edges across different communities. The edges are iteratively re-weighted based on random walk processes over the network with the walk length defined by a parameter. In [127], the authors suggest a rescaling

method that adds new edges within communities and removing edges across different communities. The edges are added or removed based on logical inferences from the existing connections between nodes. Rescaling network topology helps avoiding the resolution limit of modularity optimization as it rewards the community division with smaller communities. However, the question that arises is that which resolution scale should be best representative for the community structure of a network. Furthermore, when a resolution scale is chosen to depict the community structure of a network, it should introduce a new scale limit beyond which communities cannot be resolved. This suggests that more solid solutions than these network rescaling methods needs to be investigated to better resolve the resolution limit of modularity.

In addition to the resolution limit, our algorithm often results in different modularity values for different runs since it makes many random choices during execution. The algorithm also may not always achieve the best possible performance since its near optimal parameter values need to be manually located.

Chapter 4

Performance Evaluation

This chapter describes the evaluation of the algorithm presented in the previous chapter. We first introduce the experiment setting. We then compare the performances of the proposed algorithm and existing community detection algorithms, in term of modularity, on synthetic and real networks. Finally, we further compare the performances of the proposed algorithm and the other algorithms, in term of accuracy on synthetic networks, where the community structure is known. Parts of the results in this chapter have been published in [68].

4.1 Experiment setting

The setting of the experiments in this chapter is as follows. We use the two popular benchmarks, the Girvan-Newman (GN) benchmark [40] and the Lancichinetti-Fortunato-Radicchi (LFR) benchmark [65, 61], to generate synthetic networks with community structure. The GN benchmark has 128 nodes divided into four equal-sized communities with average node degree of 16. The LFR benchmark allows generating networks of larger sizes and power-law distributions of node degree and community size. The LFR benchmark networks are generated for two different network sizes, $N = 1000$ and $N = 5000$. The modularity of the generated networks is controlled by varying the mixing parameter μ, which defines the fraction of links from each node to nodes outside its community. μ is varied from 0.1 to 0.6 for the GN benchmark and from 0.1 to 0.8 for the LFR benchmark. The other network parameters of the LFR benchmark include the average degree of nodes, the maximum degree of nodes, the minimum community size, and the maximum

community size are set to 20, 50, 20, and 100 respectively. The exponents of the power-law distributions of node degree and community size are both set to -2.

As for real-world networks, we use the commonly studied real networks including the Zachary's karate club network [125], the Lusseau's dolphins' network [79], the Political Books network [58], the American college football network [40], the Jazz musicians network [41], the *C.elegans* metabolic network [50], the University email network [46], and the PGP network [91]. The network sizes (numbers of nodes and edges) are listed in Table 4.1.

TABLE 4.1: The number of nodes, denoted by n, and the number of edges, denoted by m, of the real networks

Network	n	m
Karate club	34	78
Dolphins	62	159
Political books	105	441
College football	115	613
Jazz	198	2,742
C.elegans	453	2,025
E-mail	1,133	5,451
PGP	10,680	24,316

To evaluate the effectiveness of the proposed algorithm in maximizing modularity, we compare the proposed algorithms with existing modularity optimization algorithms including LPAm [12], LPAm+ [75], Fastgreedy [22, 119], MSG-VM [111], and the Louvain method [15]. To evaluate the accuracy of the proposed algorithm in detecting the known community structure of the synthetic networks, we compare the proposed algorithms with the above algorithms and two other popular community detection algorithms, Walktrap [98] and Infomap [108]. Our implementation of MSG-VM follows [111] closely. The implementations of Fastgreedy, the Louvain method, Walktrap, and Infomap are provided in the public network software library igraph [26]. The proposed algorithm uses the parameter value settings which are $DEV = 0.02$ and $max_{no} = 100$ for networks of up to a thousand nodes and $DEV = 0.01$ and $max_{no} = 50$ for networks of more than a thousand nodes. All the other algorithms are executed with default parameter values. The experiments are performed on a desktop PC with Intel ® Core™ i7 @ 3.40GHz CPU and 8GB RAM.

4.2 Modularity

4.2.1 LPAm, LPAm+, and meta-LPAm+

To demonstrate the effectiveness of our modification on LPAm+, we first compare the performances, in term of modularity, of LPAm, LPAm+, and meta-LPAm+ on the synthetic and real networks.

Table 4.2 shows the average of network modularity, denoted by Q_{avg}, the maximum of network modularity, denoted by Q_{max}, and the running time (in seconds), denoted by t, obtained by LPAm, LPAm+, and meta-LPAm+ on the GN benchmark networks. As can be seen from the table, meta-LPAm+ obtains a higher average modularity value than LPAm+ on the GN benchmark networks with $\mu \geq 0.5$, where the network modularity is below 0.3. For example, meta-LPAm+ finds the average modularity value of 0.238 on the GN benchmark networks with $\mu = 0.5$ while LPAm+ finds the average modularity value of 0.220 only. The improvement in the average of network modularity of meta-LPAm+ on LPAm+ is as significant as the improvement of LPAm+ on LPAm, with an extra running time of less than a second. Meta-LPAm+ and LPAm+ find similar average modularity values on the GN benchmark networks with $\mu \leq 0.4$, where the network modularity is above 0.3.

TABLE 4.2: Comparison on the modularity values obtained by LPAm, LPAm+, and meta-LPAm+ on the GN benchmark networks.

Network	LPAm			LPAm+			meta-LPAm+		
	Q_{avg}	Q_{max}	t(s)	Q_{avg}	Q_{max}	t(s)	Q_{avg}	Q_{max}	t(s)
$\mu = 0.1$	0.647	0.657	n/a	0.650	0.657	n/a	0.650	0.657	0.001
$\mu = 0.2$	0.537	0.557	n/a	0.550	0.557	n/a	0.550	0.557	0.014
$\mu = 0.3$	0.423	0.457	0.001	0.450	0.457	0.001	0.450	0.457	0.016
$\mu = 0.4$	0.303	0.349	0.001	0.350	0.357	0.001	0.350	0.357	0.022
$\mu = 0.5$	0.193	0.223	0.001	0.220	0.250	0.002	0.238	0.250	0.072
$\mu = 0.6$	0.188	0.215	0.001	0.209	0.222	0.002	0.221	0.232	0.073

Table 4.3 and Table 4.4 show the average modularity value, the maximum modularity value, and the running time achieved by LPAm, LPAm+, and meta-LPAm+ on the LFR benchmark networks with $N = 1000$ and $N = 5000$ respectively. According to the tables, meta-LPAm+ finds a higher average modularity value than LPAm+ on the LFR benchmark networks with $\mu \geq 0.7$, corresponding to the benchmark networks with modularity below 0.3. However, the improvement in

TABLE 4.3: Comparison on the modularity values obtained by LPAm, LPAm+, and meta-LPAm+ on the LFR benchmark networks with N=1000.

Networks	LPAm			LPAm+			meta-LPAm+		
	Q_{avg}	Q_{max}	t(s)	Q_{avg}	Q_{max}	t(s)	Q_{avg}	Q_{max}	t(s)
$\mu = 0.1$	0.844	0.860	0.005	0.844	0.860	0.005	0.844	0.860	0.171
$\mu = 0.2$	0.746	0.762	0.005	0.747	0.762	0.007	0.747	0.762	0.204
$\mu = 0.3$	0.645	0.663	0.010	0.647	0.663	0.009	0.647	0.663	0.255
$\mu = 0.4$	0.542	0.559	0.012	0.547	0.559	0.014	0.547	0.561	0.310
$\mu = 0.5$	0.441	0.462	0.013	0.449	0.462	0.018	0.449	0.462	0.478
$\mu = 0.6$	0.336	0.360	0.023	0.346	0.364	0.033	0.348	0.364	0.677
$\mu = 0.7$	0.224	0.241	0.034	0.237	0.258	0.062	0.250	0.265	3.325
$\mu = 0.8$	0.190	0.204	0.034	0.205	0.212	0.071	0.219	0.226	2.698

TABLE 4.4: Comparison on the modularity values obtained by LPAm, LPAm+, and meta-LPAm+ on the LFR benchmark networks with N=5000.

Network	LPAm			LPAm+			meta-LPAm+		
	Q_{avg}	Q_{max}	t(s)	Q_{avg}	Q_{max}	t(s)	Q_{avg}	Q_{max}	t(s)
$\mu = 0.1$	0.888	0.890	0.037	0.889	0.890	0.054	0.889	0.890	2.337
$\mu = 0.2$	0.788	0.791	0.044	0.790	0.791	0.071	0.790	0.791	2.985
$\mu = 0.3$	0.687	0.691	0.054	0.690	0.692	0.088	0.690	0.692	3.731
$\mu = 0.4$	0.584	0.591	0.071	0.591	0.593	0.111	0.591	0.593	4.449
$\mu = 0.5$	0.480	0.489	0.087	0.493	0.495	0.151	0.493	0.495	5.303
$\mu = 0.6$	0.375	0.385	0.111	0.394	0.396	0.204	0.394	0.396	6.610
$\mu = 0.7$	0.273	0.281	0.201	0.293	0.298	0.453	0.294	0.298	14.250
$\mu = 0.8$	0.201	0.207	0.229	0.211	0.215	0.633	0.228	0.232	33.388

the average of network modularity of meta-LPAm+ over LPAm+ on the LFR benchmark networks with $N = 5000$ is less significant than the improvement on the LFR benchmark networks with $N = 1000$. The running time of meta-LPAm+ on the LFR benchmark networks with $N = 5000$ is notably longer than that of meta-LPAm+ on the LFR benchmark networks with $N = 1000$ but less than a minute. Meta-LPAm+ and LPAm+ result in similar average modularity values on the LFR benchmark networks with $\mu \leq 0.6$, where the network modularity is above 0.3.

Table 4.5 shows the performances of LPAm, LPAm+, and meta-LPAm+ on real networks. Meta-LPAm+ and LPAm+ achieve comparable modularity values except that meta-LPAm+ has a slight improvement in modularity over LPAm+ on the Dolphins network and the Email network. It can be inferred that, the Dolphins

TABLE 4.5: Comparison on the modularity values obtained by LPAm, LPAm+, and meta-LPAm+ on the real networks.

Network	LPAm			LPAm+			meta-LPAm+		
	Q_{avg}	Q_{max}	t(s)	Q_{avg}	Q_{max}	t(s)	Q_{avg}	Q_{max}	t(s)
Karate	0.351	0.399	na	0.418	0.420	na	0.418	0.420	0.005
Dolphins	0.495	0.502	na	0.522	0.528	na	0.526	0.529	0.012
Political books	0.494	0.516	na	0.527	0.527	na	0.527	0.527	0.033
Football	0.582	0.603	na	0.604	0.605	na	0.604	0.605	0.030
Jazz	0.435	0.445	0.003	0.444	0.445	0.005	0.445	0.445	0.075
C.elegans	0.378	0.404	0.005	0.443	0.451	0.016	0.444	0.451	0.630
Email	0.493	0.537	0.011	0.576	0.581	0.035	0.578	0.582	1.897
PGP	0.703	0.718	0.071	0.883	0.885	0.980	0.883	0.885	67.701

network and the Email network have weak communities. Therefore, LPAm+ results in a poor local maximum on these networks and our modification on LPAm+ is able to escape the local maximum to improve the network modularity.

The results confirm that, by employing the meta-heuristic based label propagation procedure meta-LPAm before merging communities, our proposed algorithm meta-LPAm+ can adjust misplaced nodes in community solutions leading to the increase in network modularity on networks with weak community structure, where communities have more inter-community links than intra-community links. Meta-LPAm+ and LPAm+ achieve similar modularity values on networks with strong community structure, where communities have more intra-community links than inter-community links, since the greedy label propagation procedure LPAm is less likely to result in a community solution with many misplaced nodes on these networks. This property of LPAm has been observed in previous works [12, 75]. The meta-heuristic based label propagation procedure is, therefore, more or less unnecessary on networks with strong community structure. However, it is worth applying meta-LPAm+ on these networks without prior knowledge of the strength of the community structure because the meta-heuristic based label propagation procedure generally requires an extra running time of only a few seconds. In our experiments, meta-LPAm+ converges in less than a minute on networks of ten thousands nodes. This running time is adequate for practical applications.

4.2.2 Meta-LPAm+ and other modularity optimization algorithms

To show the effectiveness of our proposed algorithm meta-LPAm+ in maximizing modularity, we further compare the performances of meta-LPAm+ to other modularity optimization algorithms, including Fastgreedy, MSG-VM, and the Louvain method.

Table 4.6 shows the average network modularity and the maximum network modularity found by meta-LPAm+, Fastgreedy, MSG-VM, and the Louvain method on the GN benchmark networks. As can be seen from the table, meta-LPAm+ outperforms the other algorithms on the GN benchmark networks with $\mu \geq 0.4$. For example, meta-LPAm+ has the average modularity value of 0.238 on the GN benchmark networks with $\mu = 0.5$ while Fastgreedy, MSG-VM, and the Louvain method have the average modularity values of 0.199, 0.221, and 0.207 respectively. Meta-LPAm+ obtains a significantly higher modularity value than Fastgreedy and

TABLE 4.6: Comparison on the modularity values obtained by meta-LPAm+, Fastgreedy, MSG-VM, and the Louvain method on the GN benchmark networks.

Network	meta-LPAm+		Fastgreedy		MSG-VM		Louvain	
	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}
$\mu = 0.1$	0.650	0.657	0.647	0.656	0.650	0.657	0.650	0.657
$\mu = 0.2$	0.550	0.557	0.544	0.557	0.550	0.557	0.550	0.557
$\mu = 0.3$	0.450	0.457	0.436	0.453	0.450	0.457	0.450	0.457
$\mu = 0.4$	0.350	0.357	0.307	0.348	0.344	0.356	0.345	0.357
$\mu = 0.5$	0.238	0.250	0.199	0.221	0.212	0.235	0.207	0.225
$\mu = 0.6$	0.221	0.232	0.190	0.205	0.203	0.215	0.200	0.218

a comparable modularity value with MSG-VM and the Louvain method on the GN-benchmark networks with $\mu \leq 0.3$.

Table 4.7 and Table 4.8 show the average network modularity and the maximum network modularity found by meta-LPAm+, Fastgreedy, MSG-VM, and the Louvain method on the LFR benchmark networks with $N = 1000$ and $N = 5000$ respectively. As shown in the tables, meta-LPAm+ finds a notably higher modularity value than Fastgreedy, MSG-VM, and the Louvain method on the LFR benchmark networks with $\mu \geq 0.7$. Meta-LPAm+ achieves a comparable modularity value with the Louvain method on the LFR benchmark networks with $\mu \leq 0.6$.

TABLE 4.7: Comparison on the modularity values obtained by meta-LPAm+, Fastgreedy, MSG-VM, and the Louvain method on the LFR benchmark networks with N=1000.

Network	meta-LPAm+		Fastgreedy		MSG-VM		Louvain	
	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}
$\mu = 0.1$	0.844	0.860	0.833	0.845	0.831	0.846	0.844	0.860
$\mu = 0.2$	0.747	0.762	0.707	0.726	0.732	0.748	0.747	0.762
$\mu = 0.3$	0.647	0.663	0.584	0.602	0.631	0.646	0.647	0.663
$\mu = 0.4$	0.547	0.561	0.467	0.490	0.530	0.543	0.547	0.561
$\mu = 0.5$	0.449	0.462	0.356	0.381	0.431	0.445	0.449	0.462
$\mu = 0.6$	0.348	0.364	0.260	0.277	0.331	0.346	0.347	0.364
$\mu = 0.7$	0.250	0.265	0.202	0.218	0.222	0.238	0.231	0.250
$\mu = 0.8$	0.219	0.226	0.189	0.196	0.195	0.203	0.198	0.208

TABLE 4.8: Comparison on the modularity values obtained by meta-LPAm+, Fastgreedy, MSG-VM, and the Louvain method on the LFR benchmark networks with N=5000.

Network	meta-LPAm+		Fastgreedy		MSG-VM		Louvain	
	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}
$\mu = 0.1$	0.889	0.890	0.875	0.879	0.877	0.879	0.889	0.890
$\mu = 0.2$	0.790	0.791	0.745	0.757	0.776	0.780	0.790	0.791
$\mu = 0.3$	0.690	0.692	0.626	0.644	0.676	0.680	0.690	0.692
$\mu = 0.4$	0.591	0.593	0.514	0.530	0.577	0.581	0.591	0.593
$\mu = 0.5$	0.493	0.495	0.403	0.418	0.478	0.481	0.493	0.495
$\mu = 0.6$	0.394	0.396	0.297	0.312	0.379	0.383	0.394	0.396
$\mu = 0.7$	0.294	0.298	0.220	0.233	0.268	0.277	0.291	0.297
$\mu = 0.8$	0.228	0.232	0.188	0.197	0.192	0.197	0.206	0.211

Table 4.9 shows the performances of meta-LPAm+ and the other modularity optimization algorithms on the real networks. As for the real networks, meta-LPAm+ performs better than Fastgreedy and MSG-VM on all the real networks. Meta-LPAm+ performs notably better than the Louvain method on the Dolphins network, the Political books network, the *C.elegans* network, and the Email network while has a comparable modularity value with the Louvain method on the other real networks. It seems that, after performing the first step that greedily improves network modularity at node level, meta-LPAm+ and the Louvain method produce more misplaced nodes in the Dolphins network, the Political books network, the *C.elegans* network, and the Email network than in the other networks. However, meta-LPAm+ can further adjust the misplaced nodes in the community solutions to improve network modularity while the Louvain method cannot.

TABLE 4.9: Comparison on the modularity values obtained by meta-LPAm+, Fastgreedy, MSG-VM, and the Louvain method on the real networks.

Network	meta-LPAm+		Fastgreedy		MSG-VM		Louvain	
	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}
Karate	0.418	0.420	0.381	0.381	0.401	0.420	0.419	0.419
Dolphins	0.526	0.529	0.495	0.495	0.522	0.522	0.519	0.519
Political books	0.527	0.527	0.502	0.502	0.520	0.521	0.520	0.520
Football	0.604	0.605	0.550	0.550	0.596	0.596	0.605	0.605
Jazz	0.445	0.445	0.439	0.439	0.444	0.445	0.443	0.443
C.elegans	0.444	0.451	0.404	0.404	0.439	0.440	0.441	0.441
Email	0.578	0.582	0.500	0.500	0.565	0.556	0.543	0.543
PGP	0.883	0.885	0.853	0.583	0.875	0.875	0.883	0.883

The results show that our proposed algorithm meta-LPAm+ obtains the highest modularity values among the compared algorithms on networks with weak community structure and obtains similar or higher modularity values than the compared algorithms on networks with strong community structure.

4.3 Accuracy on benchmarks

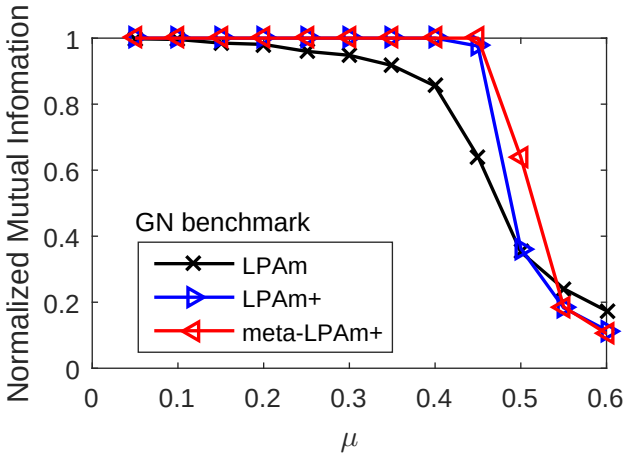
In this section, we compare meta-LPAm+ and the other community detection algorithms based on their accuracies in recovering the planted community structure in the GN benchmark networks and the LFR benchmark networks.

The accuracy of an algorithm on a synthetic network is measured based on the similarity between the detected network partition and the planted network partition of the synthetic network. To compare the network partitions, we adopt the normalized mutual information (NMI) [29], a commonly used metric of the similarity between network partitions. The normalized mutual information has the value in the range [0,1]. A value of 0 indicates that the two compared community divisions are totally dissimilar while the value of 1 indicates that the two network partitions are identical. The larger the normalized mutual information, the more similar the two compared network partitions.

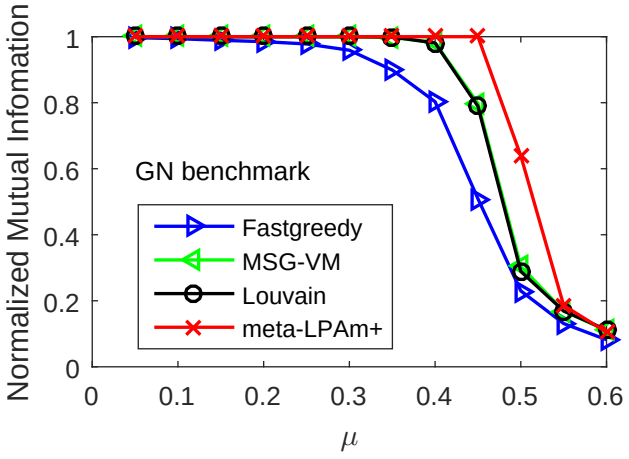
4.3.1 The GN benchmark

Figure 4.1 plots the normalized mutual information scores achieved by meta-LPAm+ and the compared algorithms on the GN benchmark networks as functions of μ. It is noted that communities in the GN benchmark networks are well-defined for $\mu < 0.75$ [62]. As μ increases, the community structure of the GN benchmark networks is harder to detect. The NMI scores of meta-LPAm+ and the compared algorithms are divided into three different sub-figures. In each sub-figure, the NMI score of meta-LPAm+ is re-plotted to make the comparison easier. As can be seen from the sub-figures, meta-LPAm+ performs significantly better than LPAm, Fast-greedy, and Infomap on the GN benchmark networks. Meta-LPAm+ has a similar NMI score with LPAm+, MSG-VM, the Louvain method, and Walktrap on the GN benchmark networks with μ up to 0.4. However, meta-LPAm+ obtains a higher NMI score than LPAm+, MSG-VM, the Louvain method, and Walktrap on the GN benchmark networks with $\mu = 0.45$ and $\mu = 0.5$. All the algorithms almost

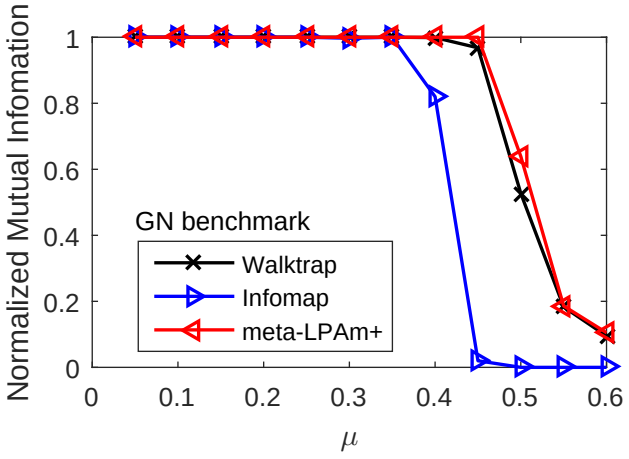
fail to recover the planted community structure of the GN benchmark networks with $\mu \geq 0.55$.



(A)



(B)



(C)

FIGURE 4.1: Normalized mutual information scores achieved by meta-LPAm+ and the compared algorithms on the GN benchmark networks.

The results show that meta-LPAm+ is the best performing algorithm among the evaluated algorithms on the GN benchmark networks.

4.3.2 The LFR benchmark

Figure 4.2 plots the NMI scores achieved by meta-LPAm+ and the compared algorithms on the LFR benchmark networks with $N = 1000$ as functions of μ. It is noted that communities in the LFR benchmark networks with $N = 1000$ are well-defined for $\mu < (N - n_c^{max})/N$ with n_c^{max} being the largest size of a community in the networks [62]. Since we set $n_c^{max} = 100$ in our experiments, communities in the LFR benchmark networks are well defined for $\mu < 0.9$. As can be seen from the figure, meta-LPAm+ gives a similar NMI score with LPAm, LPAm+, the Louvain method, Walktrap, and Infomap for μ up to 0.60. However, meta-LPAm+ has a higher NMI score than these algorithms on the LFR benchmark networks with $N = 1000$ for $\mu = 0.65$ and $\mu = 0.70$.

Figure 4.3 shows the NMI scores given by meta-LPAm+ and the compared algorithms on the LFR benchmark networks with $N = 5000$ as functions of μ. As shown in the figure, meta-LPAm+ has a similar NMI score with LPAm+ and the Louvain method on these benchmark networks but generally has a lower NMI score than LPAm, Walktrap, and Infomap. The disadvantage of meta-LPAm+ with respect to LPAm, Walktrap, and Infomap on the LFR benchmark networks with $N = 5000$ is likely due to the resolution limit [37] of modularity. This limitation of modularity implies that the community division with maximum modularity may not recover communities whose sizes are smaller than a specific scale compared with the network size. As for the LFR benchmark networks with $N = 5000$, communities generally have small size-scales. Since meta-LPAm+ finds a better approximation of the maximum modularity value than LPAm on the LFR benchmark with $N = 5000$ (see Table 4.4), meta-LPAm+ results in a lower NMI score than LPAm. As for Walktrap and Infomap, these algorithms do not maximize modularity to identify communities and, therefore, the algorithms avoid the limitation of modularity. Therefore, Walktrap and Infomap perform better than meta-LPAm+ the LFR benchmark networks with $N = 5000$.

The results show that meta-LPAm+ outperforms the other algorithms, in term of detection accuracy, on the LFR benchmark networks with $N = 1000$. However, meta-LPAm+ performs worse than LPAm, Walktrap, and Infomap on the LFR benchmark networks with $N = 5000$.

Up to this point, we have mainly focused on the performance of meta-LPAm+

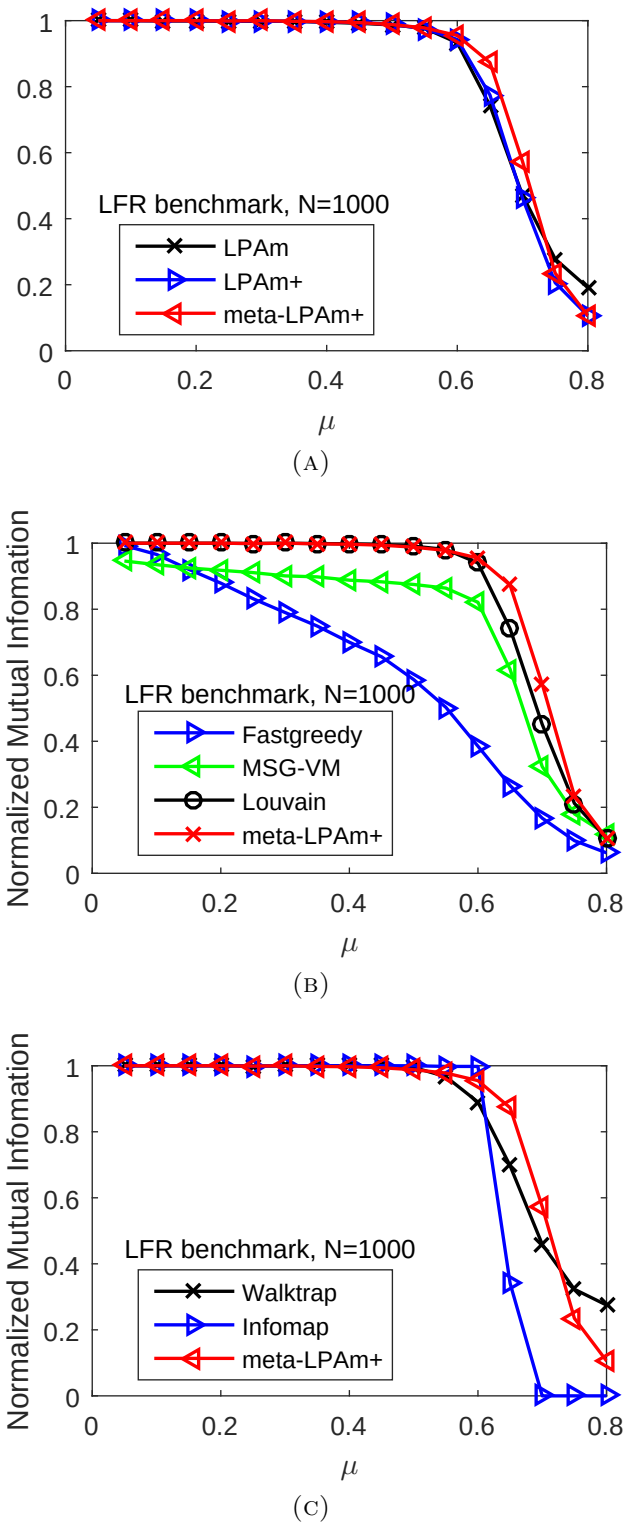
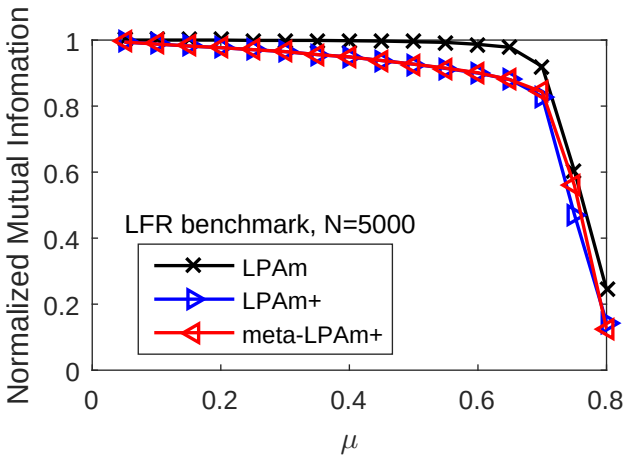
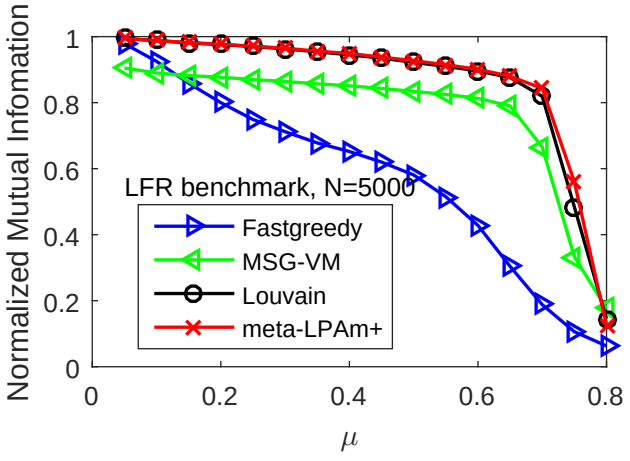


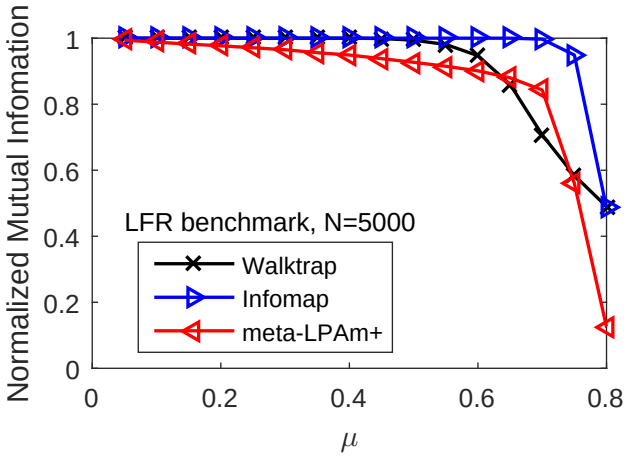
FIGURE 4.2: Normalized mutual information scores achieved by meta-LPAm+ and the compared algorithms on the LFR benchmark networks with $N = 1000$.



(A)



(B)



(C)

FIGURE 4.3: Normalized mutual information scores achieved by meta-LPAm+ and the compared algorithms on the LFR benchmark networks with $N = 5000$.

on undirected networks. As meta-LPAm+ can handle directed networks, we further evaluate the performance of meta-LPAm+ for the directed case. Among the

compared algorithms, only Infomap works with directed networks while the other algorithms can be extended to the directed case but have not been explicitly implemented. Moreover, Infomap is regarded as the state-of-the-art algorithm for detecting communities in directed networks [62]. We therefore compare the performances of meta-LPAm+ and Infomap on directed LFR benchmark networks [61].

Figure 4.4 plots the NMI scores obtained by meta-LPAm+ and Infomap on the directed LFR benchmark networks with $N = 1000$ and $N = 5000$ as functions of μ. In the directed LFR benchmark networks, the parameter μ defines the fraction of incoming links which point to each node from other nodes outside its community [61]. The value of μ is also the fraction of outgoing links which point

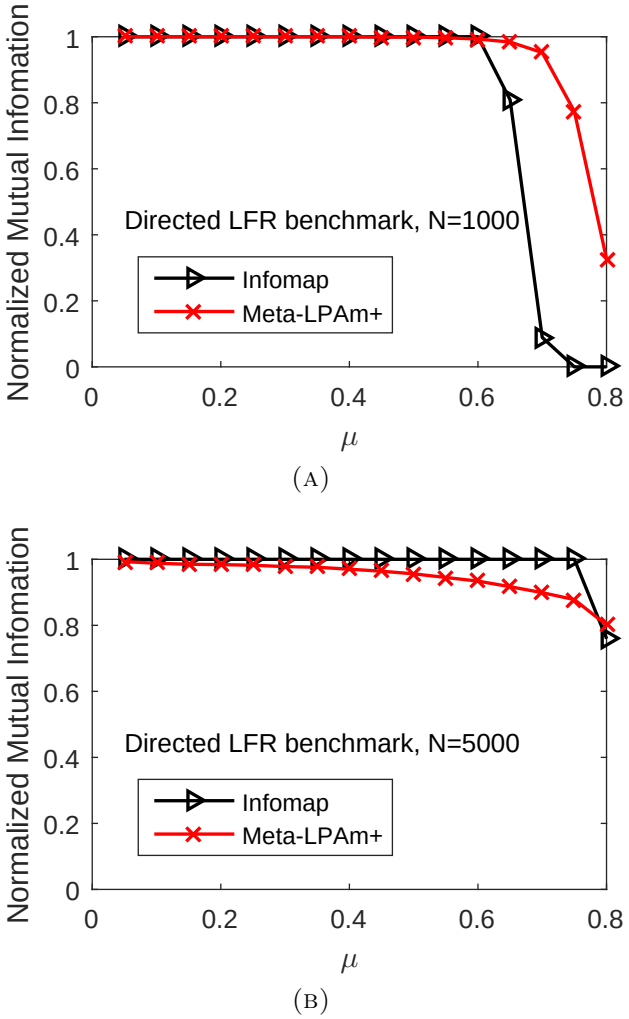


FIGURE 4.4: Normalized mutual information scores achieved by meta-LPAm+ and Infomap on the directed LFR benchmark networks with $N = 1000$ (a) and $N = 5000$ (b).

from each node to other nodes outside its community [61]. As can be seen from the figure, the performances of meta-LPAm+ and Infomap on the directed LFR benchmark networks are similar to the performances of these algorithms on the undirected ones. Meta-LPAm+ performs better than Infomap on the directed LFR benchmark networks with $N = 1000$ but has a lower performance than Infomap on the directed LFR benchmark networks with $N = 5000$.

4.4 Summary

In this chapter, we evaluated our proposed algorithm meta-LPAm+ and existing community detection algorithms on synthetic and real networks.

The experimental results show that meta-LPAm+ finds community divisions with higher modularity than LPAm+ on networks with weak community structure, where communities have more inter-community connections than intra-community connections, while resulting in a comparable modularity value with LPAm+ on networks with strong community structure, where communities have more intra-community connections than inter-community connections. Meta-LPAm+ also outperforms the other modularity optimization methods including Fastgreedy, MSG-VM, and the Louvain method, in term of modularity, on networks with weak community structure.

The results also show that meta-LPAm+ is the best performing algorithm, in term of detection accuracy, on the synthetic networks of small size. However, as the network size increases, meta-LPAm+ performs worse compared with LPAm, Walktrap, and Infomap.

Chapter 5

A Benchmark with Community Mixing Heterogeneity and Outliers

This chapter presents our approach to develop a realistic benchmark for evaluating community detection algorithms with heterogeneous community mixing fractions and outliers. We first give an overview of the state-of-the-art benchmark LFR [61] which is the foundation of our proposed benchmark. We then present our extensions to the LFR benchmark that incorporate the heterogeneity in community mixing fractions and the presence of outliers. Afterward, we extend our proposed benchmark for generating overlapping communities. Finally, we analyze the computational complexity of the construction process of our proposed benchmark. Parts of the content of this chapter have been published in [69].

5.1 Construction of the LFR benchmark

The LFR benchmark, introduced by Lancichinetti, Fortunato, and Radicchi in [65, 61], is regarded as the state-of-the-art benchmark for evaluating community detection algorithms [38]. This benchmark accounts for the heterogeneity of node degree and community size. A network realization of the LFR benchmark is constructed as follows [65, 61]:

1. The network is given a number N of nodes. The degrees of the nodes are taken from a power-law distribution with exponent γ. The power-law distribution of node degree is randomly generated such that the average degree of nodes is $\overline{k}$ and the maximum degree of nodes is k_{max}.

2. The sizes of communities are taken from a power-law distribution with exponent β. The power-law distribution of community size is randomly generated such that the minimum community size is s_{min}, the maximum community size is s_{max}, and the sum of all community sizes equals the number of nodes in the network.

3. Each node shares a fraction μ of links, with respect to the total number of links of the node, with nodes in its community and shares a fraction $1-\mu$ of links with nodes outside its community. Nodes are then randomly assigned to communities under the condition that the number of internal links of each node, with respect to its community, must not exceed the size of the community.

4. Links among nodes in the same communities are created. This is equivalent to generating links for a number of sub-networks with one sub-network for each community. The number of nodes in each subnetwork is the number of nodes in the corresponding community and the degrees of nodes in the sub-network are the internal degrees of nodes in the community. Nodes in each sub-network are connected by the stub-pairing algorithm which is also referred to as the configuration model [85]. This stub-pairing algorithm can be described as follows. Each node in the sub-network is attached with a stub, which can be seen as a "half" edge. All the stubs of nodes are then randomly paired together. A link is created between two nodes if two subs attached to the two nodes are paired. After all stubs of nodes in the sub-network are paired, a link rewiring procedure is performed to avoid multiple links between nodes in the sub-network [61].

5. Links between nodes in different communities are created. This is equivalent to generating links for a new network of N nodes with the degrees of the nodes are the external degrees of nodes in the generating network. Nodes in the new network are connected by the configuration model [85]. A link rewiring procedure is then performed to avoid links between nodes in the same community.

TABLE 5.1: Network parameters of the LFR benchmark.

Parameter	Description
N	Number of nodes
$\overline{k}$	Average degree of nodes
k_{max}	Maximum degree of nodes
s_{min}	Minimum size of communities
s_{max}	Maximum size of communities
γ	Power-law exponent for node degree sequence
β	Power-law exponent for community size sequence
μ	Mixing parameter

Algorithm 14 Steps in the LFR algorithm [61] to generate benchmark networks, with homogeneous community mixing fractions and without outliers

1: Generate N nodes and, for each node, determine the node degree from the three parameters $\overline{k}$, k_{max} and γ.
2: Generate the communities using s_{min}, s_{max} and β.
3: For each node, determine the numbers of internal links and external links from μ.
4: Generate links connecting nodes in the same communities to match the numbers of internal links of nodes determined in step 3.
5: Generate links connecting nodes in different communities to match the numbers of external links of nodes determined in step 3.

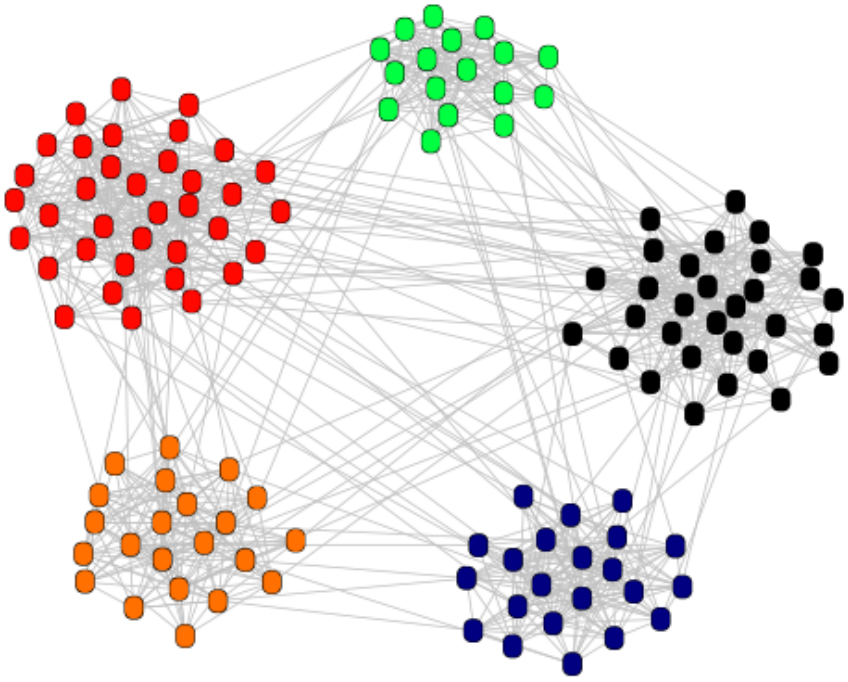


FIGURE 5.1: A realization of the LFR benchmark.

We summarize the network parameters of the LFR benchmark in Table 5.1 and the steps of the LFR algorithm in Algorithm 14 as these details are crucial to understanding our extensions to the LFR benchmark. Figure 5.1 illustrates a realization of the LFR benchmark with the network size of 128 nodes.

5.2 GLFR: a generalization of LFR with heterogeneous community mixing fractions and outliers

5.2.1 Incorporating community mixing heterogeneity

In this section, we extend the LFR benchmark by incorporating the heterogeneity in community mixing fractions. Our main extension of the LFR benchmark is to replace the single mixing parameters μ in Table 5.1 with a set of different mixing parameters $\{\mu_1, \ldots, \mu_m\}$ for community $1, \ldots, m$ in the benchmark network. For community c with mixing fraction μ_c, nodes in community c have the same fraction μ_c of external links and the same fraction $1 - \mu_c$ of internal links. Therefore, the mixing fraction of a community is also the fraction of external links for every node in the community. There are several conditions that $\mu_1, \ldots, \mu_m$ need to meet for them to be valid mixing fractions in our generalized LFR model with community mixing heterogeneity. We list and discuss them below.

Condition 1. **(Average Mixing Fraction)** We would like the average mixing fraction to remain as μ so that our model reduces to the LFR benchmark as a special case. For this purpose, the mixing fractions of communities are taken from a uniform distribution in the range of $\mu - \Delta_\mu$ to $\mu + \Delta_\mu$.

Condition 2. **(Well-defined Community)** The mixing fraction of a community makes sense only if the community is well defined. The condition for the existence of community c is [62]

$$p_i^{in} > p_i^{out}$$

for every node i in the community, where p_i^{in} is the connection probability between node i and other nodes in community c, and p_i^{out} is the connection probability between node i and nodes outside community c. Let k_i, k_i^{int} and k_i^{ext} be the total degree, the internal degree and the external degree of node i respectively. Let κ_c be the total degree of nodes in community c, and κ is the total degree of nodes in

the whole network. The following lemma gives the condition for μ_c to be a valid mixing parameter.

Lemma 5.1. *For a community c with a mixing parameter μ_c, total node degree κ and total internal node degree κ_c to be well-defined, the condition on μ_c is*

$$\mu_c < \frac{\kappa - \kappa_c}{\kappa}$$

Proof. We start with the standard definitions for p_i^{in} and p_i^{out} in [62]

$$p_i^{in} = \frac{k_i^{int}}{\kappa_c}$$

and

$$p_i^{out} = \frac{k_i^{ext}}{\kappa - \kappa_c},$$

The condition $p_i^{in} > p_i^{out}$ becomes

$$\frac{k_i^{int}}{\kappa_c} > \frac{k_i^{ext}}{\kappa - \kappa_c}. \tag{5.1}$$

As μ_c is the fraction of external links of node i, we have

$$k_i^{int} = (1 - \mu_c)k_i \tag{5.2}$$

and

$$k_i^{ext} = \mu_c k_i \tag{5.3}$$

Substituting (5.2) and (5.3) into (5.1), the condition $p_i^{in} > p_i^{out}$ is equivalent to

$$\frac{(1 - \mu_c)}{\kappa_c} > \frac{\mu_c}{\kappa - \kappa_c} \Rightarrow \mu_c < \frac{\kappa - \kappa_c}{\kappa}$$

□

As shown in Lemma 5.1, a community c exists only if the mixing fraction μ_c is smaller than the ratio between the total degree of nodes outside the community and the total degree of nodes in the network. This ratio is also the maximum value that can be assigned to the mixing fraction μ_c. As different communities may have different κ_c, they will have different upper-bound on μ_c. For the sake of simplicity, in our generalized benchmark, we use the same maximum valid value μ_{max} for the mixing fractions of every community in the network, which is defined

as

$$\mu_{max} = \frac{\kappa - \kappa_{max}}{\kappa},$$

where κ_{max} is the largest total degree of nodes in a community in the network

Condition 3. **(Non-negative Mixing Fraction)** The mixing fractions $\mu_c > 0$ for all communities c is to ensure the connectivity of the communities. This is a trivial condition but is crucial for generating a valid network. In our generalized benchmark, we use a common lower bound $\mu_{min} = 0.025$ for all communities.

Combining conditions 1, 2, and 3, the mixing fractions of communities in the network are generated from a uniform distribution in the range $[\max(\mu_{min}, \mu - \Delta_\mu), \min(\mu_{max}, \mu + \Delta_\mu)]$ with the average μ of the mixing fractions. Our generalized benchmark takes an additional parameter Δ_μ, named *mixing heterogeneity parameter*, in addition to those in Table 5.1.

The steps to generate benchmark networks with heterogeneous community mixing fractions are summarized in Algorithm 15. The main differences between Algorithm 14 and Algorithm 15 are in steps 3 and 4 where different community mixing fractions are introduced.

Algorithm 15 Steps in our GLFR algorithm to generate benchmark networks with heterogeneous community mixing fractions

1: Generate N nodes and, for each node, determine the node degree from the three parameters $\overline{k}$, k_{max} and γ.
2: Generate the communities using s_{min}, s_{max} and β.
3: For each community c, determine the mixing parameter μ_c from a uniform distribution on the support $[\max(\mu_{min}, \mu - \Delta_\mu), \min(\mu_{max}, \mu + \Delta_\mu)]$ with the average μ of the mixing parameters.
4: For each node in community c, determine the numbers of internal links and external links from μ_c.
5: Generate links connecting nodes in the same communities to match the numbers of internal links of nodes determined in step 4.
6: Generate links connecting nodes in different communities to match the numbers of external links of nodes determined in step 4.

The proposed benchmark GLFR allows us to control the heterogeneity of the mixing fractions of communities by the parameter Δ_μ. For $\Delta_\mu = 0$, all the mixing fractions of communities are fixed by μ and the generated graphs are the same as the ones produced by the LFR benchmark. The greater Δ_μ, the more diverse the mixing fractions of communities. The ambiguity of community structure is controlled mainly by the parameter μ. A network with a larger μ generally has more links, on average, between communities than with a smaller μ, and therefore,

the built-in communities with a larger μ are often harder to detect than with a smaller μ.

Figure 5.2 illustrates the distribution of the mixing fractions of communities in networks with $N = 1000$ and $\mu = 0.5$ for different $\Delta\mu$. We observe that our extension to the LFR benchmark generates a rich heterogeneity among the community mixing fractions. The fraction of external links of every node in a community is expected to be assigned the mixing fraction of the community. However, since the number of the possible values of the fraction of external links of a node is limited and dependent on the node degree, the fraction of external links of a node

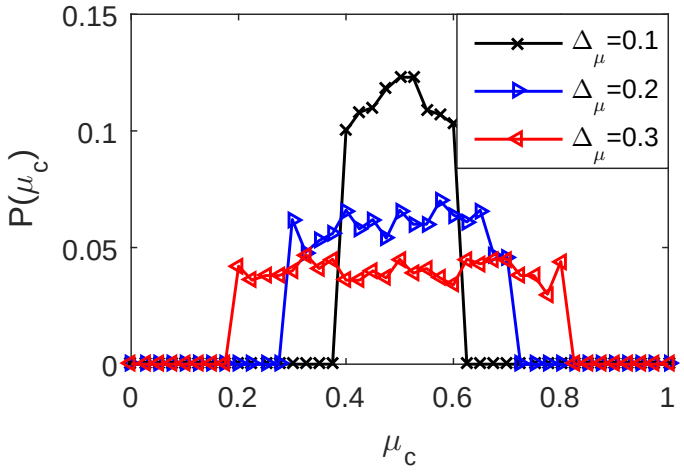


FIGURE 5.2: Distribution of the mixing fractions of communities in networks with $N = 1000$ and $\mu = 0.5$.

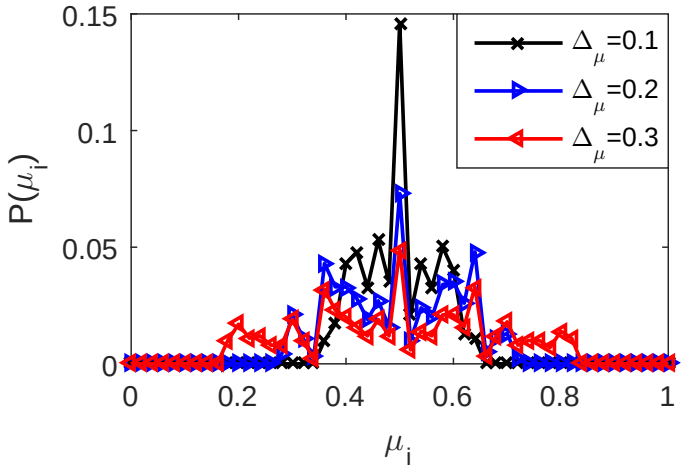


FIGURE 5.3: Distribution of the fractions of external links of nodes in networks with $N = 1000$ and $\mu = 0.5$.

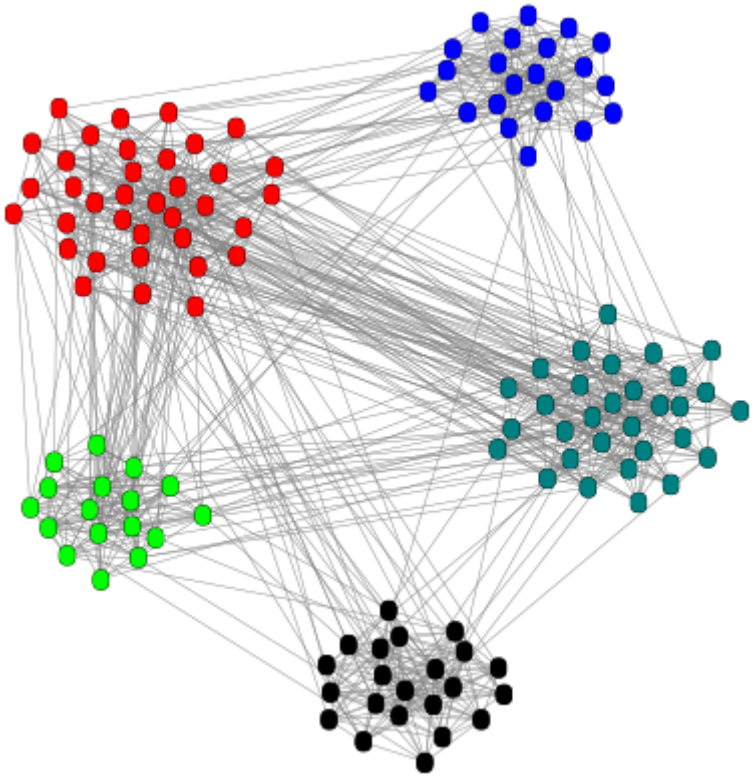


FIGURE 5.4: (Color online) A realization of the GLFR benchmark, with parameters $N = 128$, $\overline{k} = 16$, $k_{max} = 32$, $\mu = 0.2$, and $\Delta_\mu = 0.2$. The network consists of five communities represented by different colors. The community mixing fractions are heterogeneously distributed.

is assigned the possible value closest to the mixing fraction of its associated community. The distribution of the fractions of external links of nodes in networks with $N = 1000$ and $\mu = 0.5$ is demonstrated in Figure 5.3. Again we observe a large variety of fractions of external links among the nodes. Note that the original LFR benchmark provides only a single mixing value and corresponds to the case where $\Delta\mu = 0$ in our generalized LFR model. Figure 5.4 shows a realization of the GLFR benchmark with heterogeneous community mixing fractions.

5.2.2 Incorporating outliers

In this section, we extend our proposed benchmark GLFR to incorporate the presence of outliers. We introduce a new network parameter n_s to control the number of outliers in the network. The network size, in term of the total number of nodes in the network, is, therefore, the sum of the number N of nodes in communities and the number n_s of outliers. Each outlier has a fraction μ_s of links to nodes in communities and a fraction $1 - \mu_s$ of links to other outliers.

Edges in the networks are randomly created by the stub-pairing method, or the configuration model [85], to match the numbers of internal links and external links of nodes in communities and the numbers of internal links and external links outliers. Internal links and external links of outliers are defined with respect to the group of the outliers. The following requirements need to be satisfied to ensure random connections among outliers and between outliers and the rest of the network.

Condition 4. **(Random connections)** The connection probability, p_i^{in}, between outlier i and other outliers is equal to the connection probability, p_i^{out}, between the outlier and nodes in communities. Let κ_s be the total degree of outliers in the network and κ be sum of the total degree of nodes in communities and the total degree of outliers in the network. The condition $p_i^{in} = p_i^{out}$ is satisfied for

$$\mu_s = \frac{\kappa - \kappa_s}{\kappa}. \tag{5.4}$$

The proof that $p_i^{in} = p_i^{out}$ for the value of μ_s given by Equation 5.4 is similar to the proof of Lemma 5.1.

Condition 5. **(Internal and external degrees of outliers)** The total external degree of nodes in communities must not be smaller than the total external degree of outliers, that is

$$\sum_c \mu_c \kappa_c \geq \mu_s \kappa_s,$$

where c is a community in the network, μ_c is the mixing fraction, or the fractions of external links, of community c and κ_c is the total degree of nodes in community c.

This condition is to guarantee that condition 4 can be satisfied so that all external links of outliers are connected to nodes in communities.

The proposed benchmark GLFR has an additional parameter n_s, to those existing in the benchmark, which controls the number of outliers in the network. The GLFR benchmark, therefore, allows evaluating the performance of community detection algorithm as a function of the number of outliers. When n_s is set to 0, the generated networks consist of only nodes in communities.

The steps to generate networks with heterogeneous community mixing fractions and outliers are summarized in Algorithm 16. The main difference between Algorithm 16 and Algorithm 15 is in steps 1, 5, and 7 where outliers are introduced. Figure 5.5 shows a realization of the GLFR benchmark with outliers.

Algorithm 16 Steps in our GLFR algorithm to generate benchmark networks with community mixing heterogeneity and outliers

1: Generate $N + n_s$ nodes and, for each node, determine the node degree from the three parameters $\overline{k}$, k_{max} and γ.
2: Generate the communities for N nodes (randomly selected) using s_{min}, s_{max} and β.
3: For each community c, determine the mixing parameter μ_c from a uniform distribution on the support $[\max(\mu_{min}, \mu - \Delta_\mu), \min(\mu_{max}, \mu + \Delta_\mu)]$ with the average μ of the mixing parameters.
4: For each node in community c, determine the numbers of internal links and external links from μ_c.
5: For each node that has not been assigned into any community, determine the numbers of internal links and external links, with respect to the group of these nodes, using the equation (5.4).
6: Generate links connecting nodes in the same communities to match the numbers of internal links of the nodes determined in step 4.
7: Generate links connecting nodes without community membership to match the numbers of internal links of the nodes determined in step 5.
8: Generate links connecting nodes in the network to match the numbers of external links of nodes determined in steps 4 and 5.

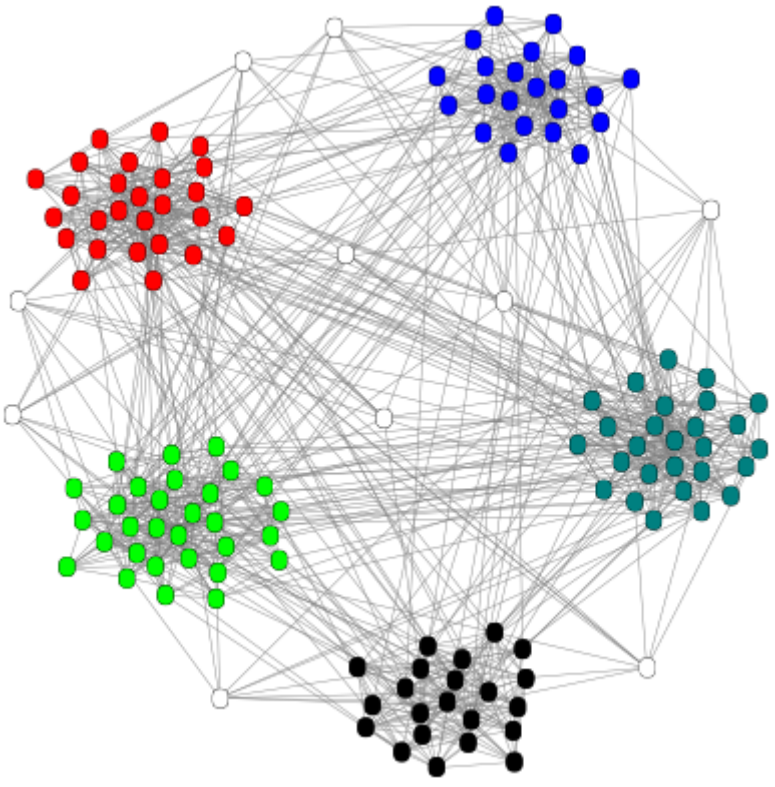


FIGURE 5.5: (Color online) A realization of the GLFR benchmark, with $N = 128$, $\overline{k} = 16$, $k_{max} = 32$, $\mu = 0.1$, $\Delta_\mu = 0$, and $n_s = 10$. The network consists of five communities (in different colors) and ten outliers (in white color).

5.3 Extending GLFR to overlapping communities

Real networks may have overlapping communities [96], with nodes belonging to more than one community at a time. This feature of real networks has been introduced in the LFR benchmark [61]. However, so far, we have not explained how to allow nodes to have multiple community memberships in our GLFR benchmark. In this section, we extend our GLFR benchmark to overlapping communities as follows.

We generate communities with overlapping nodes using a procedure as in the LFR benchmark [61]. We first assign the numbers of community memberships for N nodes. The number of overlapping nodes is determined by parameter o_n, with $o_n \leq N$. For simplicity, we assume that overlapping nodes have the same number, which is determined by parameter o_m, of community memberships. Community sizes are randomly generated to satisfy the constraint that the sum of the community sizes must equal the sum of the numbers of community memberships of nodes, that is

$$\sum_{i=1}^{n_c} s_{c_i} = o_n * o_m + N - o_n,$$

where n_c is the number of communities to be generated and s_{c_i} is the size of community c_i. Nodes are then assigned to communities using the configuration model [85] as follows. Each non-overlapping node is attached with a stub and each overlapping node is attached with a number o_m of stubs. Each community is attached with a number s_{c_i} of stubs. Nodes are connected to communities by randomly pairing stubs of nodes and stubs of communities. A node is assigned into a community if a stub of the node and a stub of the community are paired together. After all stubs of nodes and communities are paired, a link rewriting procedure is performed to avoid multiple links between a node and a community [61].

Since overlapping nodes are assigned into multiple communities, we need to determine the numbers of internal links and external links of these nodes. The numbers of internal links and external links of each overlapping node are defined from all the mixing fractions of its associated communities. Let i be an overlapping node and $\{c_1, c_2, .., c_{m_i}\}$ be the set of communities associated with node i, where m_i is the number of the associated communities. Let k_i be the degree of node i and κ_{c_j} be the total degree of community c_j. The number of links (including internal links

and external links) that node i shares with community c_i is chosen to be

$$k_i(c_j) = \frac{\kappa_{c_j}}{\sum_{l=1}^{m_i} \kappa_{c_l}} k_i.$$

Since each community has a different mixing fraction, the number of external links that node i shares with community c_j is

$$k_i^{ext}(c_j) = \mu_{c_j} k_i(c_j),$$

where μ_{c_j} is the mixing fraction of community c_j.

We can rewrite $k_i^{ext}(c_j)$ as

$$k_i^{ext}(c_j) = \frac{\mu_{c_j}\kappa_{c_j}}{\sum_{l=1}^{m_i} \kappa_{c_l}} k_i.$$

Therefore, the total number of external links of node i is

$$k_i^{ext} = \frac{\sum_{l=1}^{m_i} \mu_{c_l}\kappa_{c_l}}{\sum_{l=1}^{m_i} \kappa_{c_l}} k_i. \tag{5.5}$$

The number of internal links that node i shares with community c_j is

$$k_i^{int}(c_j) = k_i(c_j) - k_i^{ext}(c_j),$$

, or equivalently,

$$k_i^{int}(c_j) = \frac{(1-\mu_{c_j})\kappa_{c_j}}{\sum_{l=1}^{m_i} \kappa_{c_l}} k_i. \tag{5.6}$$

As now we have determined the number of external links of each overlapping node and the numbers of internal links that each node shares with its associated communities, we need to modify the construction process of GLFR to match the numbers of internal links and external links of the nodes. This involves steps 4 and 5 of the construction process of the LFR benchmark. Step 4 is to generate links among nodes in the same communities. This step is equivalent to generating links for sub-networks with one sub-network for each community. The degree of nodes in each sub-network are the internal degrees of the nodes that are assigned for the corresponding community, calculated from Equation (5.6). Step 5 is to generate

links among nodes in different communities. This step is equivalent to generating a new network of N nodes with the degrees of the nodes being the external degrees of nodes in the generating networks, calculated from Equation (5.5).

TABLE 5.2: Network parameters of the GLFR benchmark.

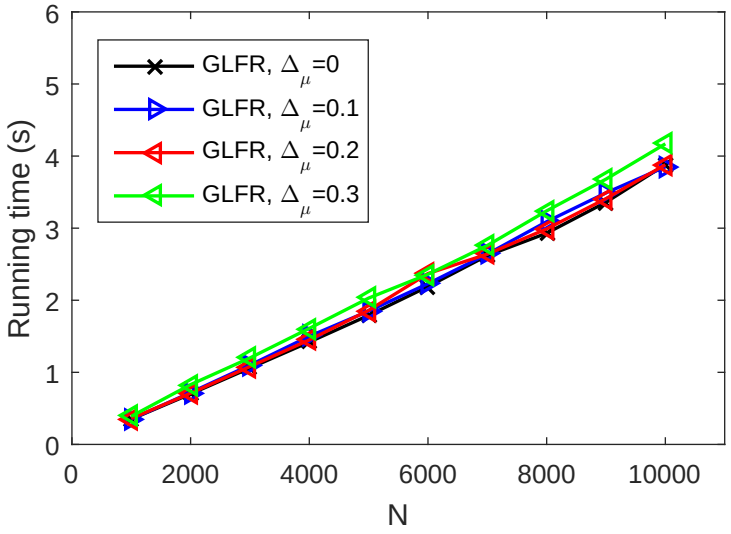
Parameter	Description
N	Number of nodes
$\overline{k}$	Average degree of nodes
k_{max}	Maximum degree of nodes
s_{min}	Minimum size of communities
s_{max}	Maximum size of communities
γ	Power-law exponent for node degree sequence
β	Power-law exponent for community size sequence
μ	Average mixing parameter
Δ_μ	Community mixing heterogeneity parameter
n_s	Number of outliers
o_n	Number of overlapping nodes
o_m	Number of communities associated with each overlapping node

Algorithm 17 Steps in our GLFR algorithm to generate overlapping benchmark networks with heterogeneous community mixing fractions and outliers

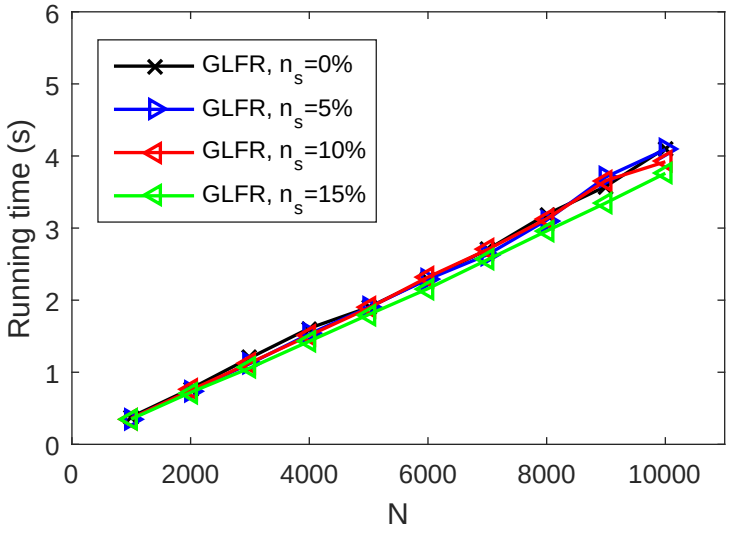
1: Generate $N + n_s$ nodes and, for each node, determine the node degree from the three parameters $\overline{k}$, k_{max}, γ.
2: Determine the numbers of community memberships for N nodes from the two parameters o_n and o_m.
3: Generate the communities for N nodes (randomly selected) to match the number of community memberships of each node using s_{min}, s_{max}, β.
4: For each community c, determine the mixing parameter μ_c from a uniform distribution on the support $[\max(\mu_{min}, \mu - \Delta_\mu), \min(\mu_{max}, \mu + \Delta_\mu)]$ with the average μ of the mixing parameters.
5: For each node in community c, determine the numbers of internal links, with respect to the community, using the equation (5.6) and external links using the equation (5.5).
6: For each node that has not been assigned into any community, determine the numbers of internal links and external links, with respect to the group of these nodes, using the equation (5.4).
7: Generate links connecting nodes in the same communities to match the numbers of internal links of the nodes determined in step 5.
8: Generate links connecting nodes without community membership to match the numbers of internal links of the nodes determined in step 6.
9: Generate links connecting nodes in the network to match the numbers of external links of nodes determined in steps 5 and 6.

The steps of the GLFR algorithm to generate networks with heterogeneous community mixing fractions, outliers, and overlapping communities are summarized in Algorithm 17. The main differences between Algorithm 17 and Algorithm 16 are in steps 2, 3, and 5 where overlapping nodes are involved. The network parameters of the GLFR benchmark are summarized in Table 5.2.

5.4 Computational complexity



(A)



(B)

FIGURE 5.6: (Color online) Running time of the GLFR algorithm for different Δ_μ (a) and for different n_s (b) as a function of the number of nodes N. The mixing parameter μ is set equal to 0.5.

The computational complexity of the GLFR algorithm is mainly calculated from steps 7, 8, and 9 in Algorithm 17 where links between nodes are created. In these steps, the links are created using the configuration model which takes a linear running time as in the LFR benchmark [61]. The steps of generating heterogeneous community mixing fractions and outliers in the GLFR algorithm does not introduce more complexity into the algorithm. The complexity of the GLFR algorithm is, therefore, $O(m)$ with m being the number of links in the networks [61]. Figure 5.6 demonstrates the running time of the GLFR algorithm which scales linearly with the number of nodes in the network, or interchangeably the number of links for sparse networks.

5.5 Summary

In this chapter, we proposed a new benchmark for evaluating community detection algorithms with heterogeneous community mixing fractions and outliers. Our proposed benchmark extends the state-of-the-art benchmark LFR [65, 61] by incorporating the heterogeneity in community mixing fractions and the presence of outliers.

Our proposed benchmark generates a network by performing a sequence of nine construction steps. The first six steps of the construction algorithm are to determine the properties of nodes and communities from network parameters. While the LFR benchmark assigns the same mixing fraction to every community in the network, our proposed benchmark assigns different mixing fractions to different communities. Furthermore, while the LFR benchmark assigns each node to at least a community, our proposed benchmark allows generating outliers which have no community membership. We determined five conditions on community mixing fractions and outliers to guarantee the validity of the community structure. The last three steps of the construction algorithm are to generate links between nodes in the same communities, links between outliers, and links between nodes in different communities and outliers by the configuration model [85]. The proposed benchmark has two additional network parameters, to those existing in the LFR benchmark, to control the variation in community mixing fractions and the number of outliers. The new benchmark therefore allows evaluating the performance of community detection algorithms as a function of the heterogeneity among community mixing fractions and the number of outliers.

It should be noted, however, our model basically follows the Stochastic Block Model (SBM) [47] which involves dividing nodes into blocks, or groups, and using an independent stochastic link formation process to construct the network. The network construction process, therefore, cannot closely resemble the complex generative process of real networks and the synthetic networks lacks many features of the real ones. Some of these missing features include the presence of motifs, or link patterns that frequently occur in networks [84], high network transitivity [17], the correlations of node degrees [11], and the hierarchical structure of (possible) overlapping communities [2]. Furthermore, our proposed benchmark assumes that the mixing fractions of communities follow a uniform distribution which may not always hold for a wide range of real networks. Our proposed benchmark also assumes that communities in a network to be static over time while real networks change over time through the addition or deletion of nodes and links [11].

The lack of the realistic characteristics in our model may neglect the challenges that community detection algorithms need to overcome on real networks. The detection algorithms, therefore, can easily recover communities in synthetic networks but hardly find communities in real ones. This is partially demonstrated by the strong disagreements between the communities identified by the detection algorithms and the communities observed by node information, or meta-data, in many real networks [97]. However, note that these strong disagreements may also be due to incorrect data handling [97]. As evaluating the reliability of observed communities, or ground-truths, in real networks is difficult, incorporating more realistic properties of real networks into benchmarks could make the tests of community detection algorithms more reliable. A possible way for doing this is to replace the employed SBM model by other more realistic models such as the hierarchical network model [104], the growing network model [52], or the BA model [95].

Chapter 6

Experimental Analysis

This chapter presents our experimental analysis to quantify the effects of the variation in community mixing fractions and outliers on the performances of community detection algorithms. This chapter also provides our quantitative analysis to compare the evaluation of the detection algorithms on the proposed benchmark GLFR presented in the previous chapter, with heterogeneous community mixing fractions and outliers, and the evaluation of the algorithms on the state-of-the-art benchmark LFR [65, 61], with homogeneous community mixing fractions and without outliers, with respect to reflecting the performances of the algorithms on real networks. Parts of the results in this chapter have been published in [69].

6.1 Experiment setting

The experiment setting is as follows. We evaluate the following community detection algorithms on the LFR benchmark and our GLFR benchmark: the label propagation algorithm (LPA) [102], the modularity-specialized label propagation algorithm (LPAm) [12], Fastgreedy [22, 119], the Louvain method [15], Infomap [108], and Walktrap [98]. We use two different network sizes, $N = 1000$ and $N = 5000$, to generate synthetic networks of small and large sizes respectively. The other network parameters include the average degree of nodes $\overline{k}$, the maximum degree of nodes k_{max}, the minimum community size s_{min}, the maximum community size s_{max}, the exponents γ and β are set as follows: $\overline{k} = 20$, $k_{max} = 50$, $s_{min} = 20$, $s_{max} = 100$, $\gamma = -2$ and $\beta = -2$. The performance, in term of detection accuracy, of the tested algorithms are measured by the Normalized Mutual

Information (NMI) [29], a measure of the similarity between two partitions of a network based on information theory.

6.2 Effects of the variation in community mixing fractions and outliers on community detection methods

6.2.1 Variation in community mixing fractions

To quantify the impact of the variation in community mixing fractions on the performances of the tested algorithms, we set the parameter Δ_μ of the GLFR benchmark to 0.3.

Figure 6.1 and Figure 6.2 shows the performances of the tested methods on the LFR benchmark and the GLFR benchmark with $\Delta_\mu = 0.3$ for $N = 1000$ and $N = 5000$ respectively, as functions of μ. The performances of the original Fastgreedy algorithm [22], the Louvain method, and Infomap on the LFR benchmark have previously been reported in [62]. As can be seen from the figures, the performances of the tested methods change in different ways as the mixing fraction varies among communities. For example, as detailed in Table 6.1, the performance of LPAm on networks with $N = 1000$ and $\mu = 0.6$ decreases significantly from 0.94 on the LFR benchmark to 0.65 on the GLFR benchmark while the performance of Walktrap decreases less significantly from 0.88 to 0.74. The performance of the Louvain method on networks with $N = 5000$ and $\mu = 0.5$ declines from 0.92 on LFR to 0.84 on GLFR while the performance of LPA drops more rapidly from 1.0 to 0.69, as detailed in Table 6.2.

LPAm and Walktrap exchange their relative orders in which the methods are ranked based on their performances between the LFR benchmark and the GLFR benchmark for $N = 1000$ and $\mu = 0.6$. The Louvain method and LPA swap their relative performance ranking orders between the benchmarks for $N = 5000$ and $\mu = 0.5$. The exchange in the relative performance ranking orders of the methods could affect the decision to select a particular detecting algorithm because an algorithm with higher performance is more likely to be selected to detect communities than an algorithm with lower performance.

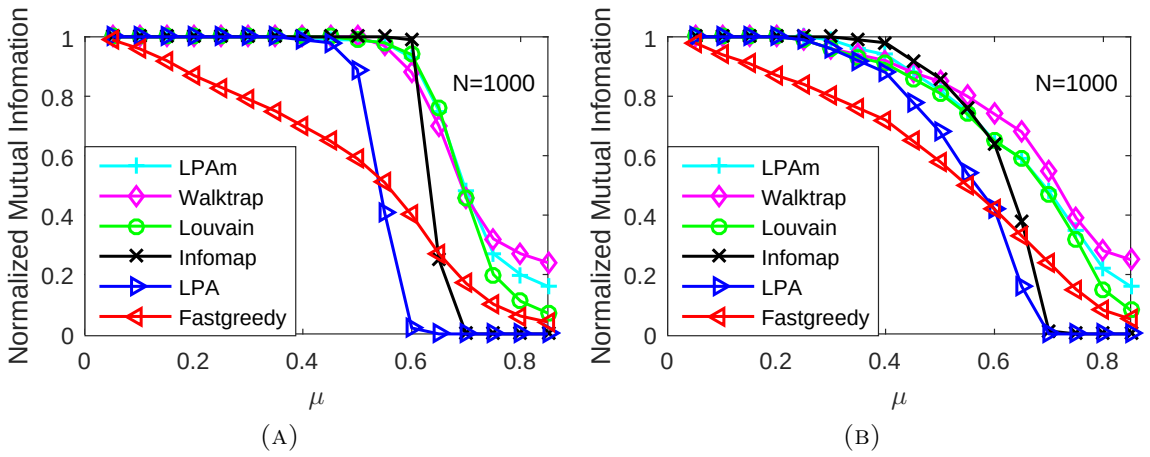


FIGURE 6.1: (Color online) Performances of the tested methods on the LFR benchmark (A) and the GLFR benchamrk with $\Delta_\mu = 0.3$ (B) for $N = 1000$. Each data point is the average over 100 network realizations.

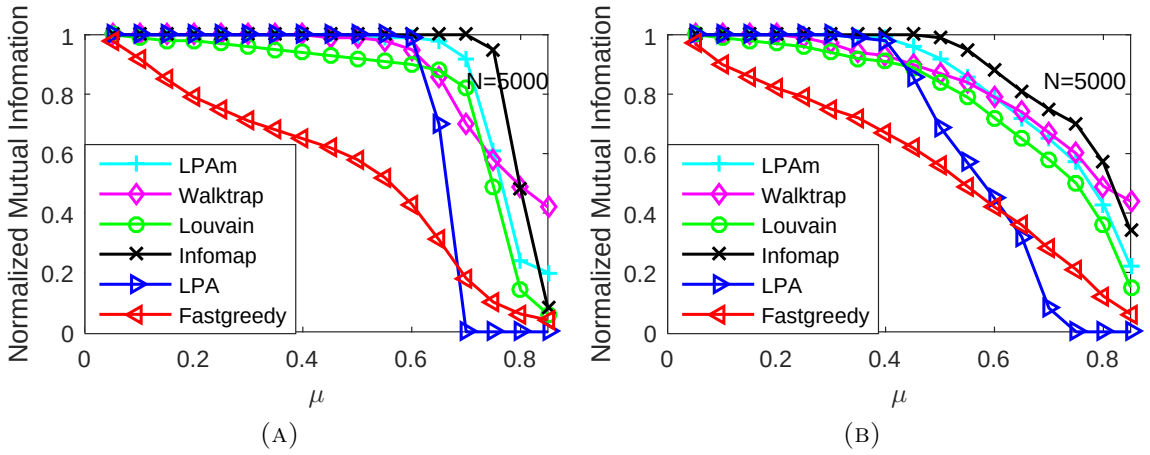


FIGURE 6.2: (Color online) Performances of the tested methods on the LFR benchmark (A) and the the GLFR benchmark with $\Delta_\mu = 0.3$ (B) for $N = 5000$. Each data point is the average over 100 network realizations.

To explain the change in the relative performance ranking orders of the tested methods, we plot in Figure 6.3 the performances of LPAm and Walktrap on the GLFR benchmark with $N = 1000$ as functions of μ and Δ_μ. As can be seen from the figure, the performance of LPAm on the networks with $N = 1000$ and $\mu = 0.6$, where the dashed line is plotted, falls more rapidly than the performance of Walktrap does as Δ_μ increases. This is due to the reason that LPAm is more sensitive to community mixing fractions greater than 0.6 than Walktrap. Since LPAm merges nodes into communities to greedily improve network modularity, this algorithm easily gets stuck in a poor local maximum for the large values of

TABLE 6.1: Performances (NMI) of the tested methods on the LFR benchmark and the GLFR benchmark with $\Delta_\mu = 0.3$ for $N = 1000$, $\mu = 0.5$ and $\mu = 0.6$

Method	$N = 1000, \mu = 0.5$		$N = 1000, \mu = 0.6$	
	LFR	GLFR, $\Delta_\mu = 0.3$	LFR	GLFR, $\Delta_\mu = 0.3$
LPAm	0.99	0.82	**0.94**	**0.65**
Walktrap	1.00	0.85	**0.88**	**0.74**
Louvain	0.99	0.81	0.94	0.65
Infomap	1.00	0.86	1.00	0.64
LPA	0.87	0.68	0.02	0.43
Fastgreedy	0.59	0.58	0.40	0.42

TABLE 6.2: Performances (NMI) of the tested methods on the LFR benchmark and the GLFR benchmark with $\Delta_\mu = 0.3$ for $N = 5000$, $\mu = 0.5$ and $\mu = 0.6$

Method	$N = 5000, \mu = 0.5$		$N = 5000, \mu = 0.6$	
	LFR	GLFR, $\Delta_\mu = 0.3$	LFR	GLFR, $\Delta_\mu = 0.3$
LPAm	0.99	0.92	0.99	0.79
Walktrap	0.99	0.87	0.95	0.79
Louvain	**0.92**	**0.84**	0.90	0.72
Infomap	1.00	0.99	1.00	0.88
LPA	**1.00**	**0.69**	0.99	0.45
Fastgreedy	0.58	0.56	0.43	0.42

community mixing fractions. This is the local maxima problem of LPAm as shown in [75, 68]. Whereas, Walktrap is less affected by the local maxima problem since this algorithm iteratively merges nodes into communities based on the structural similarities among nodes and chooses the network partition with the highest modularity value obtained during the community merging process as the community structure. Recall that LFR is a specific realization of GLFR with $\Delta\mu = 0$. This explains why LPAm and Walktrap exchange their relative performance ranking orders between the benchmarks for $N = 1000$ and $\mu = 0.6$. The exchange of the relative performance ranking orders of the Louvain method and LPA between the benchmarks appears to result from the different sensitivities of the methods on community mixing fractions with large values.

The experimental results show that the performances of many existing community detection algorithms respond asymmetrically to the variation in community mixing fractions. The heterogeneity of community mixing fractions can also re-rank

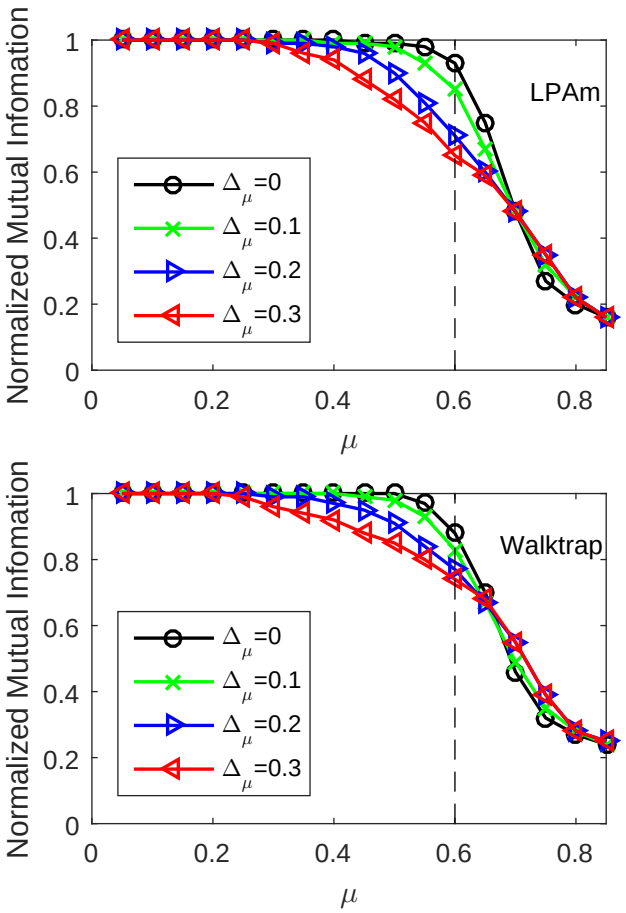


FIGURE 6.3: (Color online) Performances of LPAm and Walktrap on the GLFR benchmark with $N = 1000$ as functions of μ and Δ_μ.

the relative performance orders of the methods. As the mixing fractions of communities in real networks are likely more heterogeneous than homogeneous, it is important to perform the heterogeneous community mixing tests on benchmarks in order to comprehensively compare community detection methods.

6.2.2 Outliers

To quantify the impacts of outliers on the performances of the detection algorithms, we set the number n_s of outliers in the GLFR benchmark to 30% of the number N of nodes in communities, which means $n_s = 300$ for $N = 1000$ and $n_s = 1500$ for $N = 5000$. Since the tested algorithms have no feature to separate outliers from nodes in communities, the detected community division should ideally have each outlier in its own community or have all outliers grouped into a single community. Here we assume that the best community division found by the tested algorithms contains a single community of all outliers in the network. The

performances of the tested algorithms on the benchmarks are, therefore, evaluated based on the similarity between the detected community division and the planted community division in which the group of outliers forms as a pseudo-community.

Figure 6.4 and Figure 6.5 plot the performances of the evaluated algorithms on the LFR benchmark and the GLFR benchmark for $N = 1000$ and $N = 5000$ respectively, as functions of μ. It is noted that the performances of the tested methods on the GLFR benchmark are not available for $\mu = 0.05$ since the validity of the community structure is not preserved for this value of μ. As can be seen from the figures, the evaluation of the algorithms on the LFR benchmark, without outliers, cannot discriminate the performances of the algorithms for $\mu < 0.5$ since all the algorithms, except Fastgreedy, obtain comparable performances. Furthermore, for $\mu < 0.5$, all the algorithms almost achieve the highest possible performances and, therefore, there is no room for performance improvement. Whereas, the evaluation of the algorithms on the GLFR benchmark, with outliers, visibly separate the performances of the algorithms into different groups. Walktrap achieves the highest performance among the algorithms for $\mu < 0.5$. Infomap has the second highest performance while LPAm and the Louvain method perform worse than Infomap, as detailed in Table 6.3.

The results show that the presence of outliers degrades the performances of some algorithms, such as LPAm, more significantly than others, such as Walktrap. The effects of outliers on the performances of LPAm and Walktrap are due to the same reasons as the effects of the variation in community mixing fractions on the performances. When outliers are presented in the network, communities are highly connected to each other through outliers. Since LPAm greedily maximizes modularity to identify communities, this algorithm tends to result in a poor local maximum modularity value, as observed in [75, 68]. However, as Walktrap groups nodes into communities based on their structural similarities and outliers are highly structural similar, this algorithm tends to place outliers into a single pseudo community. Therefore, the performance of LPAm is more sensitive to outliers than the performance of Walktrap.

Furthermore, the results show that outliers reveal a potential gap, which was not found in previous work [100, 62], between the achieved performance and the highest possible performance of each algorithm. This suggests a room for potential performance improvement of the algorithms which could be implemented by taking into account the presence of outliers.

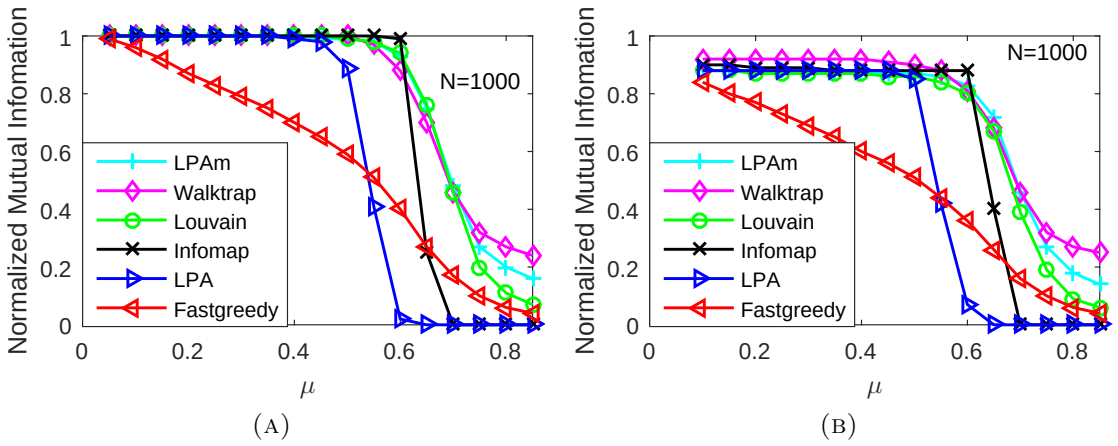


FIGURE 6.4: (Color online) Performances of the tested methods on the LFR benchmark (a) and the GLFR benchmark with $n_s = 300$ (b) for $N = 1000$. Each data point is the average over 100 network realizations.

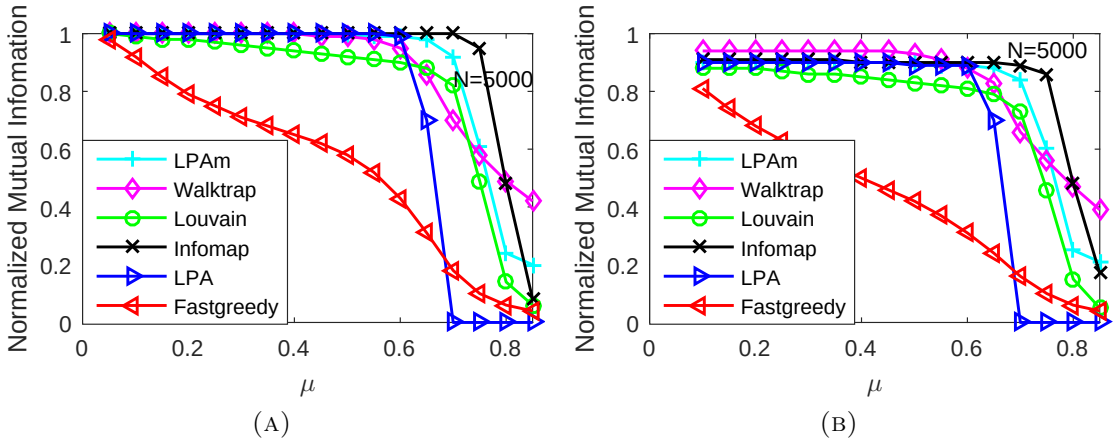


FIGURE 6.5: (Color online) Performances of the tested methods on the LFR benchmark (a) and the GLFR benchmark with $n_s = 1500$ (b) for $N = 5000$. Each data point is the average over 100 network realizations.

As the evaluations of the algorithm with outliers and without outliers lead to different evaluation results, it is important to evaluate the algorithms with outliers to get a comprehensive view of the performances of the algorithms on real networks.

TABLE 6.3: Performances (NMI) of the tested methods on the LFR benchmark and the GLFR benchmark with $\mu = 0.3$ for $N = 1000$ and $N = 5000$.

Method	N=1000, $\mu = 0.3$		N=5000, $\mu = 0.3$	
	LFR	GLFR, $n_s = 150$	LFR	GLFR, $n_s = 750$
LPAm	1.0	0.88	1.0	0.90
Walktrap	1.0	0.92	1.0	0.94
Louvain	1.0	0.87	0.96	0.86
Infomap	1.0	0.89	1.0	0.91
LPA	1.0	0.88	1.0	0.90
Fastgreedy	0.78	0.69	0.60	0.59

6.3 Comparisons between tests using GLFR and using LFR

In this section, we compare the goodness of the test of the detection algorithms with heterogeneous community mixing fractions and outliers, conducted on the GLFR benchmark, and the test of the algorithms with homogeneous community mixing fractions and without outliers, conducted on the LFR benchmark, with respect to reflecting the performances of the algorithms on real networks.

As there is no available metric to measure the difference between the performances of community detection algorithms on benchmarks and the performances that would be expected on real networks, we present our approach to derive a metric in order to compare the evaluation of the detection algorithms using GLFR and the evaluation of the detection algorithms using LFR. The performance of an algorithm running on a network with known communities is measured by the normalized mutual information (NMI). For synthetic network a and real network b with ground truth, the distance between the performance of algorithm CD_i can be measured by calculating the absolute difference in the NMI scores:

$$d_{a,b}(CD_i) = |NMI(A_i, A) - NMI(B_i, B)|$$

where A_i and B_i are the network partitions detected by method CD_i on synthetic network a and real network b respectively, A is the planted community structure in synthetic network a and B is the ground truth in real network b.

In real networks, we do not know exactly the ground truth data to compute the $NMI(B_i, B)$ as required in $d_{a,b}$. Therefore, we use the approximate ground truth

in real networks as the reference to determine the approximate distance between the performances. The approximate ground truth in a real network with strong community structure [86], where communities are well separated, can be safely obtained by finding the optimal network partition with respect to a well-known community structure quality function such as modularity [92] or the map equation [108]. Since modularity is widely used as the objective function of many algorithms to detect communities [36], we define the approximate ground truth in a real network as the network partition with the highest modularity value found by existing community detection algorithms when applied on the real network. Therefore, the approximate distance between the performance of algorithm CD_i on synthetic network a and real network b with approximate ground truth $\tilde{B}$ can be measured by

$$\tilde{d}_{a,b}(CD_i) = |NMI(A_i, A) - NMI(B_i, \tilde{B})|$$

A good evaluation of algorithm CD_i on a synthetic network should produce a similar performance for the algorithm on the real network resembled by the synthetic network. The evaluation of algorithm CD_i, therefore, should have low $\tilde{d}_{a,b}(CD_i)$. As $\tilde{d}_{a,b}(CD_i)$ gets close to 0, the performance of algorithm CD_i on the synthetic network gets approximately close to the performance on the real network. We can generalize the approximate distance for an algorithm to be the average of the approximate distances of n algorithms as follows.

Definition 7. Let CD_1, CD_2, .., CD_n be n community detection methods. The approximate distance between their performance on synthetic network a and the performance that would be expected on real network b is

$$\tilde{d}_{a,b}(CD_1, .., CD_n) = \frac{\sum_{i=1}^{n} |NMI(A_i, A) - NMI(B_i, \tilde{B})|}{n},$$

where A_i and B_i are the network partitions detected by method CD_i on synthetic network a and real network b respectively, A is the planted community structure in synthetic network a and $\tilde{B}$ is the approximate ground truth in real network b.

Using our proposed metric $\tilde{d}_{a,b}$, we compare the evaluation of the detection algorithms on the LFR benchmark and the evaluation of the detection algorithms on the GLFR benchmark. The performances of these algorithms on several commonly studied real networks are used as the references to calculate $\tilde{d}_{a,b}$. The real networks include the *C.elegans* metabolic network (C.elegans) [50], the University email network (E-mail) [46], the Jazz musicians network (Jazz) [41] and the Pretty

Good Privacy web of trust social network network (PGP) [18]. For the *C.elegans* metabolic network, the University email network, and the Jazz musicians network, we calculate $\tilde{d}_{a,b}$ using the performances of the tested algorithms on the synthetic networks with $N = 1000$ since these real networks are of small sizes with respect to the number of nodes (approximately 1000 nodes or less). For the PGP network, we calculate $\tilde{d}_{a,b}$ using the performances of the tested algorithms on the synthetic networks with $N = 5000$ because this real network is of large size with regard to the number of nodes (approximately 5000 nodes or more). The approximate ground truth community structure in the real networks we choose to be the network partition with the highest modularity value detected by the advanced modularity-specialized label propagation algorithm (LPAm+) [75] since this algorithm has the best performance with respect to network modularity among existing community detection algorithms as reported in [75, 68]. The network sizes and the highest modularity found by LPAm+ in the real networks are listed in Table 6.4.

TABLE 6.4: The number n of nodes, the number m of links, and the highest modularity Q_{max} found in the real networks

Network	n	m	Q_{max}
Jazz	198	2742	0.445
C.elegans	453	2025	0.451
E-mail	1133	5451	0.582
PGP	10680	24316	0.885

6.3.1 Heterogeneous community mixing versus homogeneous community mixing tests

Figure 6.6 shows the values of $\tilde{d}_{a,b}$ of the evaluation of the detection algorithms on the LFR benchmark and the evaluation on the GLFR benchmark with Δ_μ ranging from 0.1 to 0.3 as functions of μ. According to the figure, the more the variation in community mixing fractions, the smaller the value of $\tilde{d}_{a,b}$ obtained by the evaluation on the GLFR benchmark, for μ from about 0.3 to 0.7. The smallest value of $\tilde{d}_{a,b}$ of the evaluation on the GLFR benchmark is lower than the smallest value of $\tilde{d}_{a,b}$ of the evaluation on the LFR benchmark for all the real networks, as detailed in Table 6.5. For example, the smallest value of $\tilde{d}_{a,b}$ of the evaluation on the GLFR benchmark for the Jazz network is approximately 0.12 for $\mu = 0.50$ and $\Delta_\mu = 0.3$ while the smallest value of $\tilde{d}_{a,b}$ of the evaluation on the LFR benchmark is approximately 0.21 for $\mu = 0.3$. The smallest value of $\tilde{d}_{a,b}$ of the evaluation on

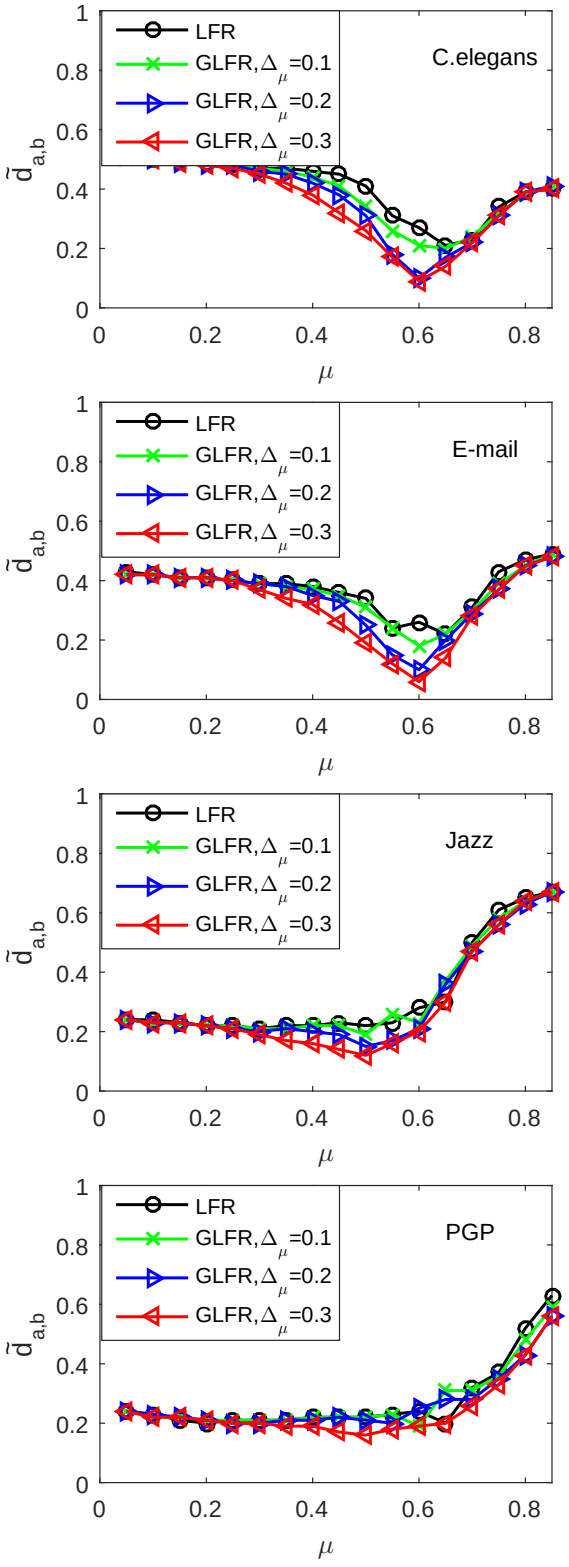


FIGURE 6.6: (Color online) Values of $\tilde{d}_{a,b}$ of the evaluation on the LFR benchmark and the evaluation on the GLFR benchmark with heterogeneous community mixing fractions for four commonly studied real networks.

TABLE 6.5: The smallest values of $\tilde{d}_{a,b}$ of the evaluation on the LFR benchmark and the evaluation on the GLFR benchmark with corresponding values of μ and Δ_μ.

Network	LFR		*GLFR*		
	smallest $\tilde{d}_{a,b}$	μ	smallest $\tilde{d}_{a,b}$	μ	Δ_μ
Jazz	0.21	0.30	0.12	0.50	0.3
C.elegans	0.20	0.65	0.09	0.60	0.2
E-mail	0.21	0.65	0.06	0.60	0.3
PGP	0.20	0.65	0.16	0.50	0.3

the GLFR benchmark for the *C.elegans* network is about 0.09 for $\mu = 0.6$ and $\Delta_\mu = 0.2$ while the smallest value of $\tilde{d}_{a,b}$ of the evaluation on the LFR benchmark is about 0.20 for $\mu = 0.65$.

The results clearly show that the evaluation of the detection methods on the GLFR benchmark, with heterogeneous community mixing fractions, better reflects the performances that would be expected on real networks than the evaluation of the algorithms on the LFR benchmark, with homogeneous community mixing fractions.

6.3.2 Tests with outliers versus without outliers

Figure 6.7 shows the values of $\tilde{d}_{a,b}$ of the evaluation of the detection algorithms on the LFR benchmark and the evaluation of the detection algorithms on the GLFR benchmark with n_s ranging from 5% to 15% of N as functions of μ. As can be seen from the figure, the evaluation on the GLFR benchmark networks generally produces a smaller value of $\tilde{d}_{a,b}$ than the evaluation on the LFR benchmark networks, with the same network parameter values. The larger the number of outliers, the smaller the value of $\tilde{d}_{a,b}$ that is given by the evaluation on the GLFR benchmark networks, except for $\mu > 0.6$. The smallest value of $\tilde{d}_{a,b}$ of the test on the GLFR benchmark is lower than the smallest value of $\tilde{d}_{a,b}$ of the test on the LFR benchmark for all the real networks, as detailed in Table 6.6. For instance, the test on the GLFR benchmark for the Jazz network produces the smallest value of $\tilde{d}_{a,b}$ of approximately 0.13 for $\mu = 0.15$ and $n_s = 150$ while the test on the LFR benchmark produces the smallest value of $\tilde{d}_{a,b}$ of approximately 0.21 for $\mu = 0.30$. The smallest value of $\tilde{d}_{a,b}$ achieved by the test on the GLFR benchmark for the *C.elegans* network is approximately 0.16 for $\mu = 0.65$ and $n_s = 150$ while the

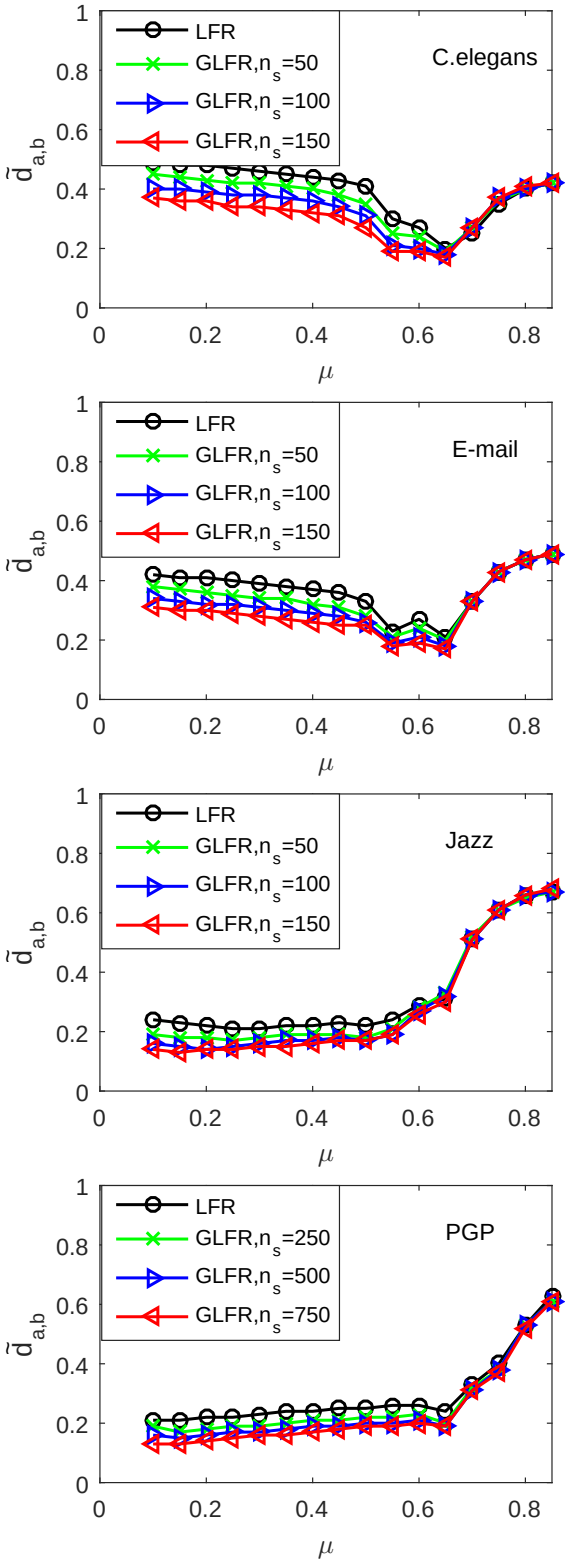


FIGURE 6.7: (Color online) Values of $\tilde{d}_{a,b}$ of the evaluation on the LFR benchmark and the evaluation on the GLFR benchmark with outliers for four commonly studied real networks.

TABLE 6.6: The smallest values of $\tilde{d}_{a,b}$ of the evaluation on the LFR benchmark and the evaluation on the GLFR benchmark with corresponding values of μ and n_s.

Network	LFR		*GLFR*		
	smallest $\tilde{d}_{a,b}$	μ	smallest $\tilde{d}_{a,b}$	μ	n_s
Jazz	0.21	0.30	0.13	0.15	150
C.elegans	0.20	0.65	0.16	0.65	150
E-mail	0.21	0.65	0.17	0.65	150
PGP	0.20	0.65	0.13	0.15	750

smallest value of $\tilde{d}_{a,b}$ achieved by the test on the LFR benchmark is about 0.20 for $\mu = 0.65$.

The results indicate that the tests of the detection algorithms on the GLFR benchmark, with outliers, more closely resemble the performances that would be achieved on real networks than the tests on the LFR benchmark, without outliers.

6.4 Summary

In this chapter, we first evaluated existing community detection algorithms on our proposed benchmark to quantify the effects of the variation in community mixing fractions and outliers on the performances of the detection algorithms. The evaluated algorithms include LPA [102], LPAm [12], Fastgreedy [22, 119], the Louvain method [15], Infomap [108], and Walktrap [98]. We found that that the variation in community mixing fractions changes the performances of different detection algorithms in different ways that exchange the performance ranking orders of some detection algorithms. We also found that the evaluation of the detection algorithms with outliers better discriminates the performances of the detection algorithms than the evaluation of the algorithms without outliers. The presence of outliers in networks poses more difficult performance tests to the detection algorithms that reveals a gap between the achieved performance and the highest possible performance of the detection algorithms.

We then compare the evaluation of the detection algorithms on our proposed benchmark and the evaluation of the algorithms on the LFR benchmark, with respect to reflecting the performances of the algorithms on real networks. We proposed a metric, named $\tilde{d}_{a,b}$, that measures the approximate distance between the

performances of the detection algorithms on a benchmark network and the performances that would be expected a real network. The closer to 0 the value of $\tilde{d}_{a,b}$, the closer the performances of the detection algorithms on the synthetic network and the performances of the algorithms on the real network. In our comparisons, the value of $\tilde{d}_{a,b}$ is calculated for four commonly studied real networks, including the *C.elegans* network [50], the E-mail network [46], the Jazz [41] and the PGP network [18]. The results show that the smallest value of $\tilde{d}_{a,b}$ of the evaluation of the detection algorithms with heterogeneous community mixing fractions is lower than that of the evaluation of the detection algorithms with homogeneous community mixing fractions for all the real networks. The results also show that the smallest value of $\tilde{d}_{a,b}$ of the evaluation of the detection algorithms with outliers is lower than that of the evaluation of the detection algorithms without outliers for all the real networks.

Chapter 7

Conclusions

In this thesis, we presented two major contributions towards community detection, one for detecting communities using modularity optimization and one for evaluating community detection algorithms. Firstly, we proposed a new algorithm for maximizing modularity to identify communities in networks with weak community structure. Secondly, we proposed a new benchmark for evaluating community detection algorithms with heterogeneous community mixing fractions and outliers. The results from this thesis have been published in two peer-reviewed conference papers [68, 69]. We also currently have one manuscript accepted for publication and another one under revision for quality international journals. Below, we summarize our main contributions and suggest directions for future research in community detection.

7.1 Summary of the contributions

Modularity optimization in networks with weak community structure

We proposed a new modularity optimization algorithm, called meta-LPAm+, which is based on the state-of-the-art algorithm LPAm+ [75]. Our proposed algorithm basically consists of three iterative steps with the first and third steps derived from the LPAm+ algorithm, as summarized in Algorithm 12 and Algorithm 13. The first step of the proposed algorithm propagates labels of nodes over the network following a greedy search strategy to maximize modularity. The second step of the proposed algorithm propagates node labels following a guided search strategy to avoid local maxima reached by the first step. The guided search

strategy employed by the second step is inspired by the record-to-record travel algorithm [34] for a balance between performance and running time. The third step of our proposed algorithm merges communities resulting from the second step to further improve modularity.

Our proposed algorithm outperforms state-of-the-art algorithms, in term of modularity, on networks with weak community structure, where communities have more inter-community connections than intra-community connections, while preserving a comparable performance on networks with strong community structure, where communities have more intra-community connections than inter-community connections, as shown in Table 4.2 to Table 4.9. The proposed algorithm performs best compared with state-of-the-art algorithms, in term of detection accuracy, on synthetic networks of small size, as shown in Figure 4.1 and Figure 4.2. However, as the network size increases, the proposed algorithm performs worse compared with three other algorithms including LPAm [12], Walktrap [98], and Infomap [108], as shown in Figure 4.3. Therefore, the proposed algorithm serves as an effective complement to existing detection algorithms to identify communities in networks. The implementation of the proposed algorithm can be freely downloaded at https://github.com/badungle/meta-LPAm_plus.

A benchmark for evaluating community detection algorithms with heterogeneous community mixing fractions and outliers

We proposed a new benchmark, called GLFR, that extends the state-of-the-art benchmark LFR [65, 61] by incorporating the heterogeneity of community mixing fractions and the presence of outliers. Our proposed benchmark generates a network with community structure following a sequence of nine construction steps, as summarized in Algorithm 17. The first six steps of the construction algorithm are to determine properties for nodes, outliers, and communities in the network from parameters. Different from the LFR benchmark, which assigns the same fraction of inter-community links, or mixing fraction, to every community in the network and eliminates outliers, our proposed benchmark assigns different mixing fractions to different communities and introduces outliers to the network. The variation in community mixing fractions and the number of outliers are controlled by two additional parameters, to those exist in the LFR benchmark. The validity of the community structure is guaranteed by five additional conditions on the mixing fractions of communities and outliers. The last three steps of the construction algorithm are to generate links among nodes in the same communities, links

among outliers, and links among nodes in different communities and outliers in the network.

We evaluated existing community detection algorithms on our proposed benchmark GLFR. The results show that the variation in community mixing fractions and the presence of outliers changes the performances of the detection algorithms in different ways, for example, decreasing the performances of some algorithms more significantly than other algorithms, as illustrated in Figure 6.1, Figure 6.2, Figure 6.4, and Figure 6.5. This distorts the evaluation results and gives different impressions of the performances of the algorithms on real networks. Therefore, it is important to evaluate the algorithms with heterogeneous community mixing fractions and outliers for a comprehensive understanding of the performances of the algorithms on real networks. Our new benchmark is suitable for this purpose as it provides parameters to control the heterogeneity among community mixing fractions and the number of outliers.

Moreover, we performed a comparative analysis between the evaluation of the detection algorithms with heterogeneous community mixing fractions and outliers and the evaluation of the algorithms with homogeneous community mixing fractions and without outliers, with respect to reflecting the performances of the algorithms on several commonly studied real networks. The comparisons are made based on our proposed metric, denoted as $\tilde{d}_{a,b}$, that measures the difference between the performances of the detection algorithms on a synthetic network and the performances that would be expected on a real network. The results show that the evaluation of the detection algorithms with heterogeneous community mixing fractions better reflects the performances of the algorithms on these real networks than the evaluation of the algorithms with homogeneous community mixing fractions, as indicated in Table 6.5. The results also show that the evaluation of the detection algorithms with outliers more closely resembles the performances of the algorithms on the real networks than the evaluation of the algorithms without outliers, as indicated in Table 6.6. Therefore, evaluating the detection algorithms with heterogeneous community mixing fractions and outliers provides more accurate estimates of the performances of the algorithms on these real networks than evaluating the algorithms with homogeneous community mixing fractions and without outliers. The implementation of the proposed benchmark can be freely downloaded at https://github.com/badungle/GLFR .

7.2 Future work

Even though this thesis made several significant contributions to research in community detection, some limitations remain that offer directions for future research, as discussed below.

1. **Systematic parameter estimation for meta-LPAm+.** Our proposed algorithm meta-LPAm+ is dependent on the two parameters DEV and max_{no}. The parameter DEV represents the largest allowed decrease in modularity, with respect to the highest modularity value found, and the parameter max_{no} represents the largest allowed number of iterations without improvement in the highest modularity value found. Choosing near-optimal values of DEV and max_{no} for use in meta-LPAm+ is important for this algorithm to achieve the best performance. However, it is not clear how to systematically choose the near-optimal values of DEV and max_{no} for a given network dataset. Until now, they are manually selected based on empirical analysis on artificial networks.

2. **Improving the stability of meta-LPAm+.** One of the important drawbacks of our proposed algorithm meta-LPAm+ is that the modularity values obtained by this algorithm are not stable for different runs since there are many heuristic choices to be made during the iterations of the algorithm. The highest modularity value is therefore not guaranteed to achieve most of the time. Improving the stability of meta-LPAm+ would be an interesting direction for further work. One of the possible approaches in this direction is to search for the consensus of community divisions with high modularity [126].

3. **Resolving the resolution limit of modularity.** Meta-LPAm+ follows the modularity optimization approach that maximizes modularity to identify communities. However, modularity has its own limitations [37, 42, 63, 93] that will also appear in our algorithm. For example, the resolution limit [37] of modularity will prevent the detection of small size-scale communities. This is a long-standing problem of modularity that still remains open for future investigations. Recent studies in this direction include [8, 60, 127, 20].

4. **Extending the GLFR benchmark by incorporating a statistical distribution of community mixing fractions.** Our proposed benchmark GLFR currently assumes that community mixing fractions are uniformly

distributed within a certain range of values. However, a uniform distribution of community mixing fractions seems not always to occur in a wide range of real networks [3]. Future research may explore a statistical distribution of community mixing fractions in real networks and incorporate the statistical distribution of community mixing fractions into the GLFR benchmark. This would help develop a more realistic benchmark for evaluating community detection algorithms with the realistic distribution of community mixing fractions.

5. **Extending the GLFR benchmark by incorporating dynamic communities.** Our proposed benchmark GLFR currently assumes that communities in networks to be static over time. However, real networks change over time through the addition or deletion of nodes and edges [11] and, therefore, communities may grow or shrink [43]. Extending the GLFR benchmark for evaluating community detection algorithms with evolving communities is a promising direction for further studies.

6. **Outliers detection in networks with community structure.** Our proposed benchmark GLFR allows generating networks with community structure and outliers. However, existing community detection algorithms have no mechanism for dealing with outliers along with communities in networks. One possible avenue for future research is to develop community detection algorithms that are able to separate outliers from communities. We believe that this opens new opportunities for the development of more efficient community detection algorithms.

Bibliography

[1] Emile HL Aarts and Jan Karel Lenstra. *Local search in combinatorial optimization*. Princeton University Press, 2003.

[2] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *nature*, 466(7307):761, 2010.

[3] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[4] Daniel Aloise, Sonia Cafieri, Gilles Caporossi, Pierre Hansen, Sylvain Perron, and Leo Liberti. Column generation algorithms for exact modularity maximization in networks. *Physical Review E*, 82(4):046112, 2010.

[5] Arash A Amini, Elizaveta Levina, et al. On semidefinite relaxations for the block model. *The Annals of Statistics*, 46(1):149–179, 2018.

[6] LNF Ana and Anil K Jain. Robust data clustering. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–128. IEEE, 2003.

[7] Alex Arenas, Jordi Duch, Alberto Fernández, and Sergio Gómez. Size reduction of complex networks preserving modularity. *New Journal of Physics*, 9(6):176, 2007.

[8] Alex Arenas, Alberto Fernandez, and Sergio Gomez. Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics*, 10(5):053039, 2008.

[9] James P Bagrow. Evaluating local community methods in networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(05):P05001, 2008.

[10] James P Bagrow and Erik M Bollt. Local method for detecting communities. *Physical Review E*, 72(4):046108, 2005.

[11] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[12] Michael J Barber and John W Clark. Detecting network communities by propagating labels under constraints. *Physical Review E*, 80(2):026129, 2009.

[13] Earl R Barnes. An algorithm for partitioning the nodes of a graph. *SIAM Journal on Algebraic Discrete Methods*, 3(4):541–550, 1982.

[14] Asa Ben-Hur, Andre Elisseeff, and Isabelle Guyon. A stability based method for discovering structure in clustered data. In *Pacific symposium on biocomputing*, volume 7, pages 6–17, 2001.

[15] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[16] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[17] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4-5):175–308, 2006.

[18] Marián Boguñá, Romualdo Pastor-Satorras, Albert Díaz-Guilera, and Alex Arenas. Models of social networks based on social distance attachment. *Physical review E*, 70(5):056122, 2004.

[19] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. Maximizing modularity is hard. *arXiv preprint physics/0608255*, 2006.

[20] Mingming Chen, Konstantin Kuzmin, and Boleslaw K Szymanski. Community detection via maximization of modularity and its variants. *IEEE Transactions on Computational Social Systems*, 1(1):46–65, 2014.

[21] Aaron Clauset. Finding local community structure in networks. *Physical review E*, 72(2):026132, 2005.

[22] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[23] Anne Condon and Richard M Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):116–140, 2001.

[24] Michele Coscia, Fosca Giannotti, and Dino Pedreschi. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining*, 4(5):512–546, 2011.

[25] Michele Coscia, Giulio Rossetti, Fosca Giannotti, and Dino Pedreschi. Demon: a local-first discovery method for overlapping communities. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 615–623. ACM, 2012.

[26] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695(5):1–9, 2006.

[27] Leon Danon, Alex Arenas, and Albert Díaz-Guilera. Impact of community structure on information transfer. *Physical Review E*, 77(3):036103, 2008.

[28] Leon Danon, Albert Díaz-Guilera, and Alex Arenas. The effect of size heterogeneity on community identification in complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(11):P11010, 2006.

[29] Leon Danon, Albert Díaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, 2005.

[30] Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique percolation in random networks. *Physical review letters*, 94(16):160202, 2005.

[31] Thang N Dinh, Nam P Nguyen, and My T Thai. An adaptive approximation algorithm for community detection in dynamic scale-free networks. In *INFOCOM, 2013 Proceedings IEEE*, pages 55–59. IEEE, 2013.

[32] Thang N Dinh and My T Thai. Community detection in scale-free networks: Approximation algorithms for maximizing modularity. *Selected Areas in Communications, IEEE Journal on*, 31(6):997–1006, 2013.

[33] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 72(2):027104, 2005.

[34] Gunter Dueck. New optimization heuristics: the great deluge algorithm and the record-to-record travel. *Journal of Computational physics*, 104(1):86–92, 1993.

[35] Gary William Flake, Steve Lawrence, and C Lee Giles. Efficient identification of web communities. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–160. ACM, 2000.

[36] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[37] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[38] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.

[39] Santo Fortunato, Vito Latora, and Massimo Marchiori. Method to find community structures based on information centrality. *Physical review E*, 70(5):056104, 2004.

[40] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[41] Pablo M Gleiser and Leon Danon. Community structure in jazz. *Advances in complex systems*, 6(04):565–573, 2003.

[42] Benjamin H Good, Yves-Alexandre de Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106, 2010.

[43] Clara Granell, Richard K. Darst, Alex Arenas, Santo Fortunato, and Sergio Gómez. Benchmark model to assess community structure in evolving networks. *Phys. Rev. E*, 92:012805, Jul 2015.

[44] Roger Guimera and Luís A Nunes Amaral. Cartography of complex networks: modules and universal roles. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(02):P02001, 2005.

[45] Roger Guimera and Luís A Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, 2005.

[46] Roger Guimera, Leon Danon, Albert Díaz-Guilera, Francesc Giralt, and Alex Arenas. Self-similar community structure in a network of human interactions. *Physical review E*, 68(6):065103, 2003.

[47] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[48] Yanqing Hu, Hongbin Chen, Peng Zhang, Menghui Li, Zengru Di, and Ying Fan. Comparative definition of community and corresponding identifying algorithm. *Physical Review E*, 78(2):026121, 2008.

[49] Jianbin Huang, Heli Sun, Yaguang Liu, Qinbao Song, and Tim Weninger. Towards online multiresolution community detection in large-scale networks. *PloS one*, 6(8):e23829, 2011.

[50] Hawoong Jeong, BÃ¡lint Tombor, RÃ©ka Albert, Zoltan N Oltvai, and A-L BarabÃ¡si. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, 2000.

[51] Lucas GS Jeub, Prakash Balachandran, Mason A Porter, Peter J Mucha, and Michael W Mahoney. Think locally, act locally: Detection of small, medium-sized, and large communities in large networks. *Physical Review E*, 91(1):012821, 2015.

[52] Emily M Jin, Michelle Girvan, and Mark EJ Newman. Structure of growing social networks. *Physical review E*, 64(4):046132, 2001.

[53] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1):016107, 2011.

[54] Tatsuro Kawamoto and Martin Rosvall. The map equation and the resolution limit in community detection. *arXiv preprint arXiv:1402.4385*, 2014.

[55] Tatsuro Kawamoto and Martin Rosvall. Estimating the resolution limit of the map equation in community detection. *Physical Review E*, 91(1):012809, 2015.

[56] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. Optimization by simmulated annealing. *science*, 220(4598):671–680, 1983.

[57] Stephan Kitchovitch and Pietro Liò. Community structure in social networks: Applications for epidemiological modelling. *PLOS ONE*, 6(7):1–17, 07 2011.

[58] V. Krebs. A network of co-purchased books about us politics sold by the online bookseller amazon.com, 2008.

[59] Balachander Krishnamurthy and Jia Wang. On network-aware clustering of web clients. *ACM SIGCOMM Computer Communication Review*, 30(4):97–110, 2000.

[60] Darong Lai, Hongtao Lu, and Christine Nardini. Enhanced modularity-based community detection by random walk network preprocessing. *Physical Review E*, 81(6):066118, 2010.

[61] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):016118, 2009.

[62] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80(5):056117, 2009.

[63] Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Physical Review E*, 84(6):066122, 2011.

[64] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.

[65] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110, 2008.

[66] Andrea Lancichinetti, Filippo Radicchi, José J Ramasco, and Santo Fortunato. Finding statistically significant communities in networks. *PloS one*, 6(4):e18961, 2011.

[67] Steve Lawrence and C Lee Giles. Accessibility of information on the web. *intelligence*, 11(1):32–39, 2000.

[68] Ba-Dung Le, Hung Nguyen, and Hong Shen. Community detection in networks with less significant community structure. In *International Conference on Advanced Data Mining and Applications*, pages 65–80. Springer, 2016.

[69] Ba-Dung Le, Hung X Nguyen, Hong Shen, and Nickolas Falkner. Glfr: A generalized lfr benchmark for testing community detection algorithms. In *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*, pages 1–9. IEEE, 2017.

[70] Juyong Lee, Steven P Gross, and Jooyoung Lee. Modularity optimization by conformational space annealing. *Physical Review E*, 85(5):056702, 2012.

[71] Sune Lehmann and Lars Kai Hansen. Deterministic modularity optimization. *The European Physical Journal B*, 60(1):83–88, 2007.

[72] Elizabeth A Leicht and Mark EJ Newman. Community structure in directed networks. *Physical review letters*, 100(11):118703, 2008.

[73] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th international conference on World Wide Web*, pages 695–704. ACM, 2008.

[74] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[75] Xin Liu and Tsuyoshi Murata. Advanced modularity-specialized label propagation algorithm for detecting communities in networks. *Physica A: Statistical Mechanics and its Applications*, 389(7):1493–1500, 2010.

[76] Fabrizio Luccio and Mariagiovanna Sami. On the decomposition of networks in minimally interconnected subnetworks. *IEEE Transactions on Circuit Theory*, 16(2):184–188, 1969.

[77] R Duncan Luce. Connectivity and generalized cliques in sociometric group structure. *Psychometrika*, 15(2):169–190, 1950.

[78] R Duncan Luce and Albert D Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949.

[79] David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[80] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[81] Fragkiskos D Malliaros and Michalis Vazirgiannis. Clustering and community detection in directed networks: A survey. *Physics Reports*, 533(4):95–142, 2013.

[82] Marina Meilă. Comparing clusterings by the variation of information. In *Learning theory and kernel machines*, pages 173–187. Springer, 2003.

[83] Marina Meilă. Comparing clusterings—an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895, 2007.

[84] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[85] Michael Molloy and Bruce Reed. A critical point for random graphs with a given degree sequence. *Random Structures & Algorithms*, 6(2-3):161–180, 1995.

[86] Mark EJ Newman. The structure of scientific collaboration networks. *Proceedings of the national academy of sciences*, 98(2):404–409, 2001.

[87] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[88] Mark EJ Newman. Analysis of weighted networks. *Physical review E*, 70(5):056131, 2004.

[89] Mark EJ Newman. Coauthorship networks and patterns of scientific collaboration. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5200–5205, 2004.

[90] Mark EJ Newman. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):321–330, 2004.

[91] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.

[92] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[93] Mark EJ Newman. Equivalence between modularity optimization and maximum likelihood methods for community detection. *Physical Review E*, 94(5):052315, 2016.

[94] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[95] Günce Keziban Orman and Vincent Labatut. The effect of network realism on community detection algorithms. In *Advances in Social Networks*

Analysis and Mining (ASONAM), 2010 International Conference on, pages 301–305. IEEE, 2010.

[96] Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[97] Leto Peel, Daniel B Larremore, and Aaron Clauset. The ground truth about metadata and community detection in networks. *Science advances*, 3(5):e1602548, 2017.

[98] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005*, pages 284–293. Springer, 2005.

[99] Mason A Porter, Jukka-Pekka Onnela, and Peter J Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097, 2009.

[100] Filippo Radicchi. Decoding communities in networks. *Physical Review E*, 97(2):022316, 2018.

[101] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663, 2004.

[102] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007.

[103] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[104] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67(2):026112, 2003.

[105] Sidney Redner. How popular is your paper? an empirical study of the citation distribution. *The European Physical Journal B-Condensed Matter and Complex Systems*, 4(2):131–134, 1998.

[106] Colin R Reeves. *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, Inc., 1993.

[107] Martin Rosvall, Daniel Axelsson, and Carl T Bergstrom. The map equation. *The European Physical Journal-Special Topics*, 178(1):13–23, 2009.

[108] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[109] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River, 2003.

[110] Michael T Schaub, Jean-Charles Delvenne, Martin Rosvall, and Renaud Lambiotte. The many facets of community detection in complex networks. *Applied Network Science*, 2(1):4, 2017.

[111] Philipp Schuetz and Amedeo Caflisch. Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Physical Review E*, 77(4):046112, 2008.

[112] Philipp Schuetz and Amedeo Caflisch. Multistep greedy algorithm identifies community structure in real-world and computer-generated networks. *Physical Review E*, 78(2):026112, 2008.

[113] Marco Scibetta, Fulvio Boano, Roberto Revelli, and Luca Ridolfi. Community detection as a tool for complex pipe network clustering. *EPL (Europhysics Letters)*, 103(4):48001, 2013.

[114] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[115] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, 2003.

[116] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

[117] Mursel Tasgin, Amac Herdagdelen, and Haluk Bingol. Community detection in complex networks using genetic algorithms. *arXiv preprint arXiv:0711.0491*, 2007.

[118] Amanda L Traud, Peter J Mucha, and Mason A Porter. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, 2012.

[119] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 1275–1276. ACM, 2007.

[120] Stanley Wasserman, K Faust, et al. Social network analysis. 1994. *Cambridge University, Cambridge*, 1994.

[121] Duncan J Watts. *Small worlds: the dynamics of networks between order and randomness*. Princeton university press, 1999.

[122] Warren Weaver. The mathematics of communication. *Scientific American*, 181(1):11–15, 1949.

[123] Lilian Weng, Filippo Menczer, and Yong-Yeol Ahn. Virality prediction and community structure in social networks. *Scientific reports*, 3, 2013.

[124] Jierui Xie, Stephen Kelley, and Boleslaw K Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Comput. Surv.*, 45(4):1–35, 2013.

[125] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pages 452–473, 1977.

[126] Pan Zhang and Cristopher Moore. Scalable detection of statistically significant communities and hierarchies, using message passing for modularity. *Proceedings of the National Academy of Sciences*, 111(51):18144–18149, 2014.

[127] Zhong-Yuan Zhang, Kai-Di Sun, and Si-Qi Wang. Enhanced community structure detection in complex networks with partial background information. *Scientific reports*, 3:3241, 2013.