Local Community Detection in Social Networks

Sahar Bakhtar

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Computer Science) at

Concordia University

Montréal, Québec, Canada

July 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Sahar Bakhtar**

Entitled: **Local Community Detection in Social Networks**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Ahmed Soliman

_____ External Examiner
Dr. Muthucumaru Maheswaran

_____ Examiner
Dr. Lata Narayanan

_____ Examiner
Dr. Tristan Glatard

_____ Examiner
Dr. M. Reza Soleymani

_____ Supervisor
Dr. Hovhannes A. Harutyunyan

Approved by _____
Lata Narayanan, Chair
Department of Computer Science and Software Engineering

September 1, 2022 _____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Local Community Detection in Social Networks

Sahar Bakhtar, Ph.D.

Concordia University, 2022

Recent years have witnessed the rapid growth of social network services and consequently research problems investigated in this area. Community detection is one of the most important problems in social networks. A good community can be defined as a group of vertices that are highly connected and loosely connected to the vertices outside the group. Community detection includes exploring the community partitioning in social networks. Regarding the fact that social networks are huge, having complete information about the whole network is almost impossible. As a result, the problem of local community detection has become more popular in recent years. This problem can be defined as the detection of a community for a given node by using local information. It is noteworthy that the focus of this study is on the problem of local community detection.

One major question to the problem of community detection is how to assess different communities. The most widely used technique to evaluate the quality of communities is to compare them with ground-truth communities. However, for many networks, the ground-truth communities are not known. As a result, it is necessary to have a comprehensive metric to evaluate the quality of communities. In this study, a local quality metric noted as GDM is proposed, several local community detection algorithms are compared by assessing their detected communities. The experimental results, illustrate that the local community detection algorithms are fairly compared using GDM. It is also discussed how GDM covers the drawbacks of other existing local metrics. Moreover, it is shown that the judgment of GDM is almost the same as that of the F-score, i.e. the metric which compares the community with its ground-truth community.

Furthermore, a new metric, called P, and a new local community detection algorithm, Alg_P are proposed. To detect communities locally, researchers mostly utilize an evaluation metric along

with an algorithm to explore communities. The proposed algorithm includes three different steps in which relevant nodes are added in the first step and irrelevant nodes are removed in the second and third steps. It should be mentioned that at each iteration, more than one node is added to the community. Thus, the algorithm is terminated faster than the other algorithms with near-complexity. Regarding the experimental results, it is shown that the proposed algorithm outperforms state-of-the-art local community detection algorithms.

Real-world social networks are dynamic and change over time. In order to model dynamic social networks, network history is partitioned into a series of snapshots, each one of which shows the state of the network at a time. Regarding dynamic networks, the problem of local community detection is not widely investigated. In this concern, a dynamic local community detection algorithm noted as $DevDynaP$, is proposed. The main feature of the proposed algorithm is that it starts from a given node, explores the network incrementally, and detects communities simultaneously at each snapshot. The experimental results show that the community partitioning resulting from the proposed dynamic algorithm outperforms that of the other compared algorithm. Also, the proposed algorithm explores the network faster than the compared algorithm.

Many networks contain both positive and negative relations. A community in signed networks is defined as a group of nodes that are densely connected by positive links within the community and negative links between communities. Considering the problem of local community detection in signed networks, a new algorithm, noted as Alg_{SP}, is developed by extending the metric P for signed networks. Experimental results show that the proposed algorithm can detect the ground-truth communities independently from the starting nodes.

Acknowledgments

First and foremost, I am extremely grateful to my supervisor, Prof. Hovhannes A. Harutyunyan for his invaluable advice, continuous support, and patience during my Ph.D. study. His immense knowledge and experience have encouraged me all the time in my academic research. I would like to thank my committee members, Dr. Lata Narayanan, Dr. Tristan Glatard, and Dr. M. Reza Soleymani for their time, effort, and willingness to serve on my Ph.D. committee throughout my Ph.D. program. I would also like to extend my appreciation to the external examiner Prof. Muthucumaru Maheswaran for his willingness to read through the thesis and serve on my defense committee. Finally, I would like to express my gratitude to my family members. My deepest and endless gratitude goes to my husband for his endless support. Without their tremendous understanding and encouragement over the past few years, it would be impossible for me to complete my study.

Contents

List of Figures ix

List of Tables xi

1 Introduction 1

2 Literature Review and Motivation 6

3 A Local Quality Metric for Communities 21

3.1 Introduction . . . 21

3.2 Related Works . . . 23

3.3 The Proposed Metric . . . 26

3.3.1 Definitions . . . 26

3.3.2 Propositions and proofs . . . 27

3.3.3 The Proposed Metric . . . 31

3.4 The Experiments . . . 32

3.4.1 dataset . . . 33

3.4.2 Small Graphs . . . 35

3.4.3 Two Textbook Experiments . . . 39

3.4.4 Experimental Results . . . 45

3.4.5 Discussion . . . 51

3.5 Conclusion and Future Work . . . 52

4 A New Fast Local Community Detection Algorithm **53**

4.1 Introduction . . . 53

4.2 Related Works . . . 55

4.3 The Proposed Algorithm . . . 57

4.3.1 Complexity . . . 61

4.4 Experimental Results . . . 63

4.5 Conclusion and Future Works . . . 67

5 A Dynamic Local Community Detection Algorithm **69**

5.1 Introduction . . . 69

5.2 Related Works . . . 71

5.3 Dynamic Algorithms . . . 74

5.3.1 A Simple Dynamic Structure . . . 74

5.3.2 Developed Dynamic P (DevDynaP) . . . 74

5.4 Experimental Results . . . 78

5.4.1 Benchmarks . . . 78

5.4.2 Experimental Results on Section 5.3.1 . . . 81

5.4.3 Experimental Results on Section 5.3.2 . . . 83

5.5 Conclusion and Future works . . . 89

6 A Local Community Detection Algorithm in Signed Networks **91**

6.1 Introduction . . . 91

6.2 Related Works . . . 94

6.3 The Proposed Algorithm . . . 96

6.3.1 Definitions . . . 96

6.3.2 Signed Alg_R and Signed Alg_M . . . 97

6.3.3 The Proposed Algorithm . . . 97

6.3.4 Complexity . . . 99

6.4 Experimental Results . . . 102

6.4.1 Dataset . . . 102

6.4.2 Evaluation Metrics 104
6.4.3 Experimental Results 104
6.5 Conclusion and Future Works 106

7 Conclusion **108**
7.1 Conclusion and Future works 108
7.2 Publications 110

Bibliography **111**

List of Figures

Figure 2.1 The taxonomy of proposed methods for community detection 6

Figure 2.2 An example of overlapping communities 7

Figure 2.3 An example of a dendogram tree 8

Figure 2.4 The red link is the edge with the highest betweenness 9

Figure 2.5 The first iteration of propagation of labels in COPRA (Gregory, 2010) . . . 11

Figure 2.6 The rest of iteration of propagation of labels in COPRA with $v = 2$ (Gregory, 2010) 12

Figure 2.7 An example of label propagation algorithm (LPA) (Raghavan, Albert, & Kumara, 2007) 13

Figure 2.8 Local community C, its boundary nodes B, and the neighbors of the community U (Clauset, 2005) 14

Figure 3.1 An arbitrary tree T_d 28

Figure 3.2 An arbitrary tree T_{d+1} 28

Figure 3.3 Zachary Karate Club network with the two ground-truth communities . . . 33

Figure 3.4 First simple example community 34

Figure 3.5 Second simple example community 35

Figure 3.6 Third simple example community 37

Figure 3.7 Fourth simple example community 39

Figure 3.8 The status of node 9 in Karate Club network 41

Figure 3.9 The status of node 3 in Karate Club network 41

Figure 3.10 The status of node 20 in Karate Club network 41

Figure 3.11 An example of wrong assessment of communities by R modularity 42
Figure 3.12 The values of F-score for several communities detected by Alg_R, Alg_M, and Alg_L, regarding a number of random starting nodes on American FC network . . . 48
Figure 3.13 The values of GDM for a number of communities detected by Alg_R, Alg_M, and Alg_L, regarding a number of random starting nodes on American FC network . 48
Figure 3.14 An example of two detected communities C_a and C_b, for a given node v_0 versus its ground-truth community C_g 51
Figure 4.1 The local community detection problem 54
Figure 4.2 The higher illustration of the proposed algorithm 58
Figure 4.3 How to calculate z while node v is added to the community C 62
Figure 4.4 Standard deviation on the values of F-score obtained by the algorithms on the four small real-world networks 65
Figure 5.1 Average GDM on switch, expand/contraction, merge/split, and birth/death networks 84
Figure 5.2 Modularity, Q on switch, expand/contraction, merge/split, and birth/death networks 85
Figure 5.3 The average number of explored nodes on switch, expand/contraction, merge/split, and birth/death networks 86
Figure 6.1 Four possible conditions for a triad in signed networks 93
Figure 6.2 How to calculate z^+ while node v is added to the community C 101
Figure 6.3 Illustrative network 1 (IN1) (J. Chen, Zhang, Liu, & Yan, 2017) 103
Figure 6.4 Illustrative network 2 (IN2) (J. Chen et al., 2017) 104

List of Tables

Table 3.1 Metrics' scores for communities C_1, C_2 and C_3 in Figure 3.4 36

Table 3.2 Metrics' scores for communities C_4 and C_5 in Figure 3.5 36

Table 3.3 Metrics' scores for communities C_6 and C_7 in Figure 3.6 38

Table 3.4 The result of the first textbook experiment on the Karate Club network 40

Table 3.5 The number of wrong evaluations by the four metrics on Dolphins network concerning the first textbook experiment 42

Table 3.6 The number of wrong evaluations by the four metrics on American FC network concerning the first textbook experiment 42

Table 3.7 The result of the second textbook experiment on American FC network . . . 44

Table 3.8 The number of wrong evaluations by the four metrics on American FC network concerning the second textbook experiment 44

Table 3.9 The values of F-score for some communities detected by Alg_R, Alg_M, and Alg_L regarding some random starting nodes on Karate Club network 46

Table 3.10 The values of GDM for some communities detected by Alg_R, Alg_M, and Alg_L regarding some random starting nodes on Karate Club network 46

Table 3.11 The values of F-score for some communities detected by Alg_R, Alg_M, and Alg_L regarding some random starting nodes on Dolphins network 47

Table 3.12 The values of GDM for some communities detected by Alg_R, Alg_M, and Alg_L regarding some random starting nodes on Dolphins network 47

Table 3.13 The results of the comparison of F-score and GDM on Karate club, Dolphins and American FC networks 49

Table 3.14 The results of the comparison of F-score and GDM on LFR synthetic network 49
Table 3.15 The results of the comparison of F-score and GDM on DBLP network 50
Table 3.16 The results of the comparison of F-score and GDM on Amazon network . . . 50
Table 4.1 Average F-score (ratio) on real-world networks 64
Table 4.2 Average F-score (ratio) on real-world networks 65
Table 4.3 Average GDM on real-world networks 66
Table 4.4 Execution time (s) 66
Table 5.1 Average results on LFR switch network 79
Table 5.2 Average results on LFR expand/contraction network 80
Table 5.3 Average results on LFR merge/split network 81
Table 5.4 Average results on LFR birth/death network 82
Table 5.5 Execution time (s) 83
Table 5.6 Average scores of GDM, Q, and the number of explored nodes for $IncL$ and $DevDynaP$ on switch, expand/contraction, merge/split, and birth/death networks (snapshots 1 to 8) 87
Table 5.7 Average scores of GDM, Q, and the number of explored nodes for $IncL$ and $DevDynaP$ on switch, expand/contraction, merge/split, and birth/death networks (snapshots 9 to 15) 88
Table 5.8 Execution time (s) 89
Table 6.1 Average F-score (ratio) on real-world networks 105
Table 6.2 Execution time (s) 106

Chapter 1

Introduction

Real-world networks can be categorized into four main types: social networks, information networks (or knowledge networks), technological networks, and biological networks (Newman, 2003). Social networks can be defined as networks of interactions or relationships, where the nodes play the role of actors or users, and the relationships or interactions are modeled using links or edges. Formally speaking, social networks could be represented as a graph $G = (V, E)$ in which V is the set of entities and E is the relationships among them. Online social networks have revolutionized the way people interact and share information over the Internet; social networking applications such as YouTube, Twitter, Facebook, Snapchat, etc., have millions of active users. Multiple terabytes of information are generated daily as a result of user interactions in such networks. The ability to collect and analyze such data provides unique opportunities to understand the underlying principles of social networks, their formation, evolution, and characteristics.

The underlying structure of social networks is the object of the study of social network analysis. Social network analysis methods and techniques are designed to discover patterns of interaction between entities in social networks. The focus of social network analysis is on the relationships among actors rather in the actors themselves. In other words, the main goal of these techniques is to investigate both the contents and patterns of relationships in social networks to understand the relations among actors and the implications of these relationships (Oliveira & Gama, 2012). There are several critical and challenging problems in social networks that need to be investigated. Some important examples of these problems are the identification of the most influential or central actors,

the identification of hubs and authorities, trust inference, influence maximization, detection of communities, and predicting future links that might be generated regarding the dynamic characteristic of the social networks. These issues are extremely useful in the process of extracting knowledge from social networks. Because of the appealing nature of such issues, social network analysis has become a popular approach in different fields, from biology to business. For instance, some companies use social network analysis to maximize the positive reputation of their products by targeting the customers with higher network influence (Leskovec, Adamic, & Huberman, 2007). Also, mobile telecommunications companies employ social network analysis methods for phone call networks. They try to recognize customers' profiles and then, recommend personalized mobile phone tariffs, according to their profiles (Dasgupta et al., 2008).

Interactions among people through social networks are rather complex because they involve interactions with others who may be strangers (W. Jiang, Wang, Bhuiyan, & Wu, 2016). Thus, it is crucial to study the critical problems in social networks. Addressing these problems helps to construct a safe, efficient, and practical environment through social networks. One problem of great interest in this domain is community detection in which the ultimate objective is to find dense communities within such networks. More precisely, a community is defined as a set of nodes in a graph that have many connections with each other and is loosely connected to other nodes of other communities (Girvan & Newman, 2002). In this regard, the problem of community detection tries to find such structures in networks. Community is an important structure that can be considered a summary of the whole network, thus making the network easy to comprehend (Bedi & Sharma, 2016). Communities include, but are not limited to, three sub-categories (Chakraborty, Dalmia, Mukherjee, & Ganguly, 2017):

- Disjoint (Non-overlapping) communities: a node of the graph could only belong to one community at a time,
- Overlapping communities: some nodes belong to more than one community,
- Hierarchical communities: there is a hierarchical structure in terms of the belonging of nodes to communities.

Being able to explore these sub-structures within social networks has many applications in various fields. It is useful where group decisions are being taken, e.g. multicasting a message of interest to a community instead of each one in the group or recommending a set of products to a community. As an example of community detection applications, citation networks are constructed by citation relationships among papers and researchers. Communities in a citation network indicate either related papers on a single topic or researchers working on the same topic. Identifying these communities in citation networks can provide knowledge about various core topics. Moreover, recommender systems utilize data of similar users or items to create recommendations. This is similar to the detection of groups of similar nodes in a graph. As a result, community detection can improve recommendation algorithms (Cao, Ni, & Zhai, 2015).

Furthermore, detecting communities in social networks lets network providers understand the hidden structures of the user populations, dig into people's views, analyze the information dissemination, and grasp the control of the public sentiment. It allows focusing on regions having some degree of autonomy within the graph. It helps to classify the vertices, based on their roles concerning the communities they belong to. Community detection has major applications in sociology (Sizemore, Phillips-Cremins, Ghrist, & Bassett, 2019), computer science (Kwon et al., 2019), economy (Shao, Yu, & Feng, 2019), and network security (Lee & Huh, 2019), to name a few. As a result, it is essential to detect communities efficiently.

Since the emergence of this problem, several algorithms to detect community partitioning, have been proposed (Blondel, Guillaume, Lambiotte, & Lefebvre, 2008; F. Wang, Zhang, Chai, & Xia, 2018; J. Yang, McAuley, & Leskovec, 2013). However, with the growth of social networks and their corresponding data size, handling the whole structure of the network seems to be impossible. Consequently, the local community detection problem has attracted a great deal of attention from researchers in recent years. This problem includes detecting a community for a given node, v_0, starting from v_0 and only using local information. In this problem, the main goal is to find a high-quality community for a given node v_0. The detection process starts from node v_0 and at each step, one or more nodes from the network (neighboring nodes) are merged into the community. It is noteworthy that the main focus of this study is on the local community detection problem and also, the issue of how to assess communities locally.

The relationships among nodes in real-world social networks are dynamic and change over time. Co-authorship between scholars and email interactions between employees in an organization are two examples. To model dynamic social networks, network history is partitioned into a series of snapshots, each one of which shows the state of the network at a time. In this regard, dynamic networks can be represented in two models: (1) Snapshot network which is a series of snapshots of the network at a time, and (2) Temporal network which is the series of atomic changes of the network. In this way, it is possible to analyze the network's structure over time, explore how the network evolves, and finally anticipate the future topology of the network. A large number of studies have been conducted to address the problem of community detection in dynamic networks (J. He & Chen, 2015; Rossetti, Pappalardo, Pedreschi, & Giannotti, 2017; Zhuang, Chang, & Li, 2019). However, the local community detection problem is not widely investigated in dynamic networks.

Moreover, many networks contain both positive and negative relations. In such networks, a negative link from node A to B indicates A "dislikes" B and a positive relation shows A "likes" B. Negative relations in signed networks cause different structures than in unsigned networks. In this regard, a community in signed networks is defined as a group of nodes with maximum (minimum) positive (negative) interaction within the group and minimum (maximum) positive (negative) relation between groups. The problem of community detection in signed networks tries to find such communities. This problem in signed networks is not widely investigated compared to unsigned networks. Also, the problem of local community detection, i.e. detecting a community for a given node, is not addressed independently.

In this study, the problem of local community detection in unsigned networks, dynamic unsigned networks, and signed networks is addressed. In this concern, the main contributions of this study are itemized as follows:

- Employing geodesic distance, a new local metric to evaluate the quality of detected communities, called GDM, is proposed.
- A new local community detection algorithm, called Alg_P, using the number of common neighbors, is proposed.
- Employing Alg_P, a dynamic local community detection algorithm, called $DevDynaP$, is

presented.

- Finally, Alg_P is extended to detect communities in signed networks. The extended algorithm for signed networks is called Alg_{SP}.

The remainder of this thesis is structured as follows: Chapter 2 includes an overview of presented methods on the problem of community detection. In chapter 3, the proposed metric, GDM, to evaluate communities, is presented and evaluated. In chapter 4, the proposed local community detection algorithm, Alg_P, is introduced, analyzed, and the experimental results are reported. The dynamic local community detection algorithm, $DevDynaP$, is presented and experimented in chapter 5. Also, chapter 6 discusses the problem of local community detection in signed networks and presents the extended version of algorithm Alg_P, noted as Alg_{SP}, for signed networks. Finally, chapter 7 indicates the conclusion and several interesting directions for future works.

Chapter 2

Literature Review and Motivation

Since community structure is an important attribute of social networks, the problem of detecting communities has become one of the most important problems in this area. In this regard, during the last two decades, several methods have been proposed to solve the community detection problem in social networks. C. Wang, Tang, Sun, Fang, and Wang (2015) categorized the methods for the community detection problem into three groups including traditional algorithms, overlapping community detection algorithms, and local community detection algorithms. Figure 2.1 shows the

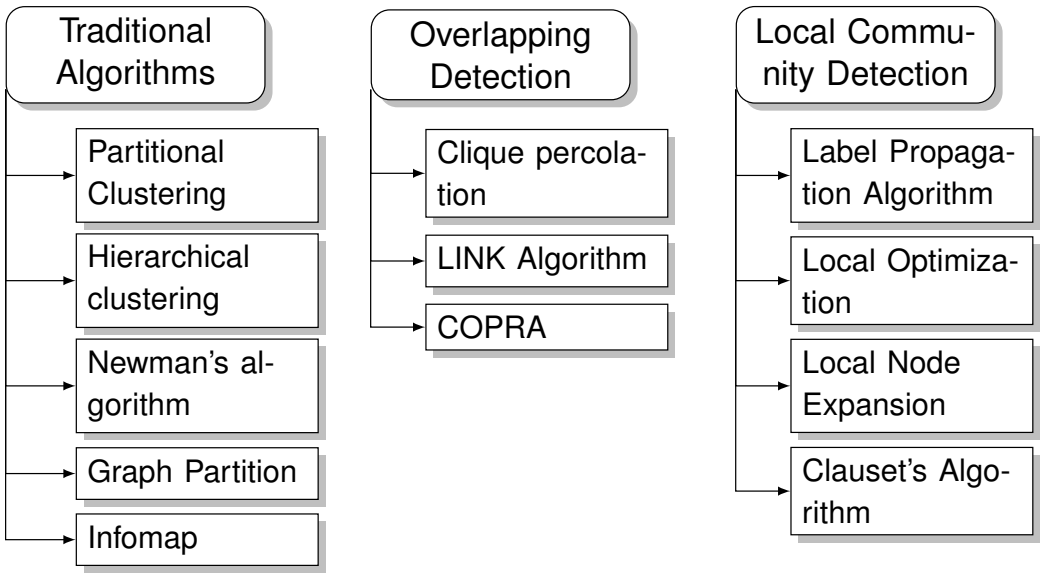

Figure 2.1: The taxonomy of proposed methods for community detection

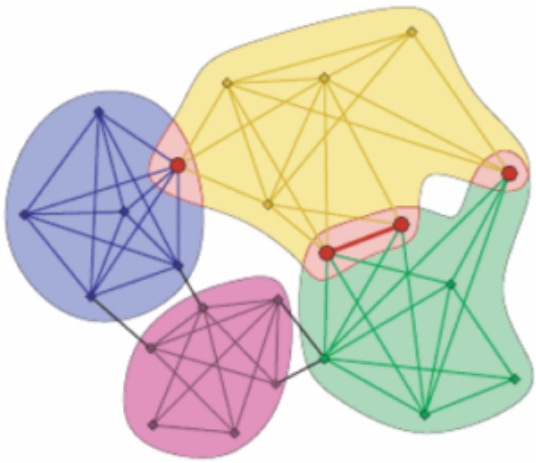

Figure 2.2: An example of overlapping communities

taxonomy of the different proposed methods respecting (C. Wang et al., 2015). Traditional algorithms tend to detect disjoint community partitioning in social networks. It is well understood that people (nodes) in social networks may belong to multiple communities. Thus, many researchers tend to study overlapping community detection. Figure 2.2 demonstrates 4 communities with overlapping nodes. The inference of community structures can generally be reduced to identifying a partitioning of the graph that maximizes some quantitative notions of community structure. However, when global knowledge of the graph's topology is lacking, a measure of community structure must necessarily be independent of those global properties. The measures independent of global properties are called local measures. Detecting communities according to maximizing local measures is local community detection (Clauset, 2005). With the development of social networks, the networks are getting much more complicated and huge. Thus, collecting the whole global information in some networks is almost impossible. As a consequence, some researchers tend to study local perspectives to detect communities. The rest of this section includes brief explanations of some of the most important algorithms belonging to the three mentioned categories in Figure 2.1.

Partitional clustering is a traditional method to detect disjoint community partitioning. This algorithm assumes there are k clusters in the network and the goal is to separate the borders between the k clusters to maximize/minimize a given objective function. Some of the most used functions are minimum k-clustering, k-clustering sum, k-center, and k-median. However, partitional clustering algorithms are easy to implement and have reasonable performance, but the number of communities

needs to be specified in advance (C. Wang et al., 2015). Walktrap algorithm (Pons & Latapy, 2005) uses a random-walk-based similarity between vertices and communities. This algorithm employs a modularity optimization clustering scheme to obtain an optimal clustering structure. Modularity, known as Q, is introduced later in this chapter. In a community detection problem, there is no information about the number of communities. That is why researchers proposed methods to detect communities hierarchically.

The hierarchical clustering methods include agglomerative algorithms and divisive algorithms. The basic idea of agglomerative algorithms is that clusters are iteratively merged bottom-up if their similarities are high enough. Also, divisive algorithms' basic idea is that clusters are iteratively split top-down by removing edges that connect vertices with low similarity. The hierarchical clustering method aims to obtain a dendrogram tree and extract communities by cutting the tree. Figure 2.3 shows an example of a dendrogram tree. In this method, there is no need to know any prior knowledge of communities. However, if the cutting or merging positions are not appropriate, low-quality communities might be extracted. Girvan and Newman (2001) developed a divisive algorithm that made great contributions to the problem of community detection. Girvan and Newman (2001) proposed the concept of edge betweenness which is the number of shortest paths between all vertex pairs that run along the edge. The basic idea of this algorithm is to remove edges with the highest betweenness. According to the definition of edge betweenness, the edge with the highest betweenness is the most between edge among communities in the network. Figure 2.4 shows an edge with

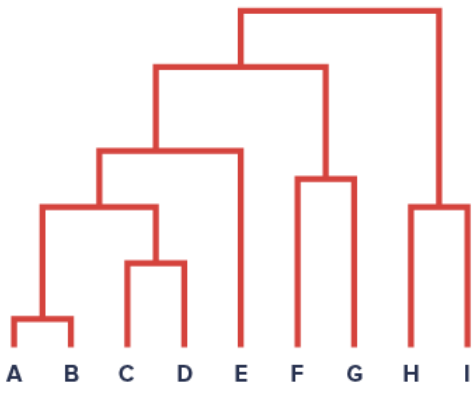

Figure 2.3: An example of a dendogram tree

the highest betweenness which is the best edge to be removed to separate communities. The general structure of Newman's algorithm is presented as follows:

(1) Calculate the betweenness for all edges in the network.

(2) Remove the edge with the highest betweenness.

(3) Recalculate betweennesses for all edges affected by the removal.

(4) Repeat from step 2 until no edges remain.

However, there is no need to know the number of clusters in advance, calculation of edge betweenness is time-consuming. In other words, the principal disadvantage of this algorithm is the high computational demands it makes. The time complexity of this algorithm in the worst-case is $O(m^2n)$ on a network with m edges and n vertices, or $O(n^3)$ on a sparse network. In order to obtain a better complexity, Newman (2004) proposed Newman's fast algorithm.

Newman (2004) proposed the quality function or "modularity", known as Q, to test whether a particular division is meaningful. Q is defined as follows:

$$Q = \frac{1}{2m} \sum_{ij} (a_{ij} - \frac{d_i d_j}{2m}) \sigma(i, j) \tag{1}$$

In (1), m represents the total number of edges in the network. Considering i and j two nodes in the network, d_i and d_j are the degrees of i and j. a_{ij} is 1 if i and j are adjacent and 0 otherwise.

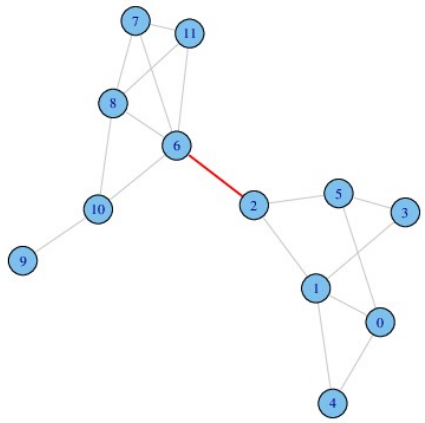

Figure 2.4: The red link is the edge with the highest betweenness

Also, $\sigma(i, j)$ is 1 if i and j belong to the same community and 0 otherwise. The algorithm starts with a state in which each vertex is the only member of a community. Communities repeatedly join together in pairs, choosing at each step the join that results in the greatest increase (or smallest decrease) in Q. The worst-case running time of the algorithm is $O((m + n)n)$ or $O(n^2)$ on a sparse graph. Since the exact modularity optimization is a problem that is computationally hard, approximation algorithms are necessary when dealing with large networks.

Blondel et al. (2008) proposed a very fast heuristic algorithm, called Louvain, to detect communities while optimizing modularity, Q. The first phase of the Louvain algorithm assigns each node a unique label as its community. Then, each node is transferred to its neighbors' community. If this transaction results in an increase of the score of the modularity, Q, the node is removed from its own community and merged into its neighbor's community. This process is repeated until no improvement in the score of modularity is gained. In the second phase of this algorithm, a network is constructed by considering each community as a node. Then, the first phase is reapplied on the newly generated network. The two phases are repeated until no change happens in the structure of the detected communities. This algorithm is capable of identifying communities in a 118 million node network that took only 152 minutes.

Graph partition aims to divide nodes in a graph into a plurality of predetermined size communities which satisfies some objective functions by removing edges. Kernighan-Lin algorithm (Kernighan & Lin, 1970) is a greedy optimization algorithm. The main idea is the maximization of the modularity Q by exchanging nodes in communities. Kernighan-Lin algorithm is a heuristic algorithm. The main drawback of this algorithm is that the size of communities needs to be set in advance. Also, Yuan (2014) proposed a method to try all possible sizes using the Kernighan-Lin algorithm, but its time complexity is quite high, which makes it infeasible for complex networks.

Infomap (J. Chen et al., 2017) was proposed to comprehend the multipartite organization of large-scale biological and social systems. The basic idea of this method is to use the probability flow of random walks on a network as a proxy for information flows in the real system and decompose the network into modules by compressing a description of the probability flow.

The clique percolation method (CPM) (Derényi, Palla, & Vicsek, 2005) is the first proposed method to identify overlapping communities. This algorithm starts by finding all $k-$cliques. Then,

it considers the detected $k-$cliques as nodes of the network. Each two $k-$cliques are adjacent if they share $k-1$ vertices. In the end, the connected parts are the communities.

CPMd (Palla, Derényi, Farkas, & Vicsek, 2005) is an improvement of the CPM algorithm that detects overlapping communities in directed graphs. CPMd presents a new definition of $k-$clique in directed networks as follows: the directed links in a $k-$clique are all from a node with a relative high out-degree to a node with a relative low out-degree; where the directed closed loop in $k-$clique does not exist. A node may exist in multiple $k-$cliques, so CPM and CPMd algorithms could identify overlapping communities.

Moreover, LINK (Ahn, Bagrow, & Lehmann, 2010) is a link partitioning algorithm based on hierarchical clustering that detects overlapping communities. In the LINK algorithm, if two links share the same node belonging to different communities, the vertex must be a node in the overlapping area. It treats each edge as a separate community. Then, it merges the two most similar communities until all the link communities become a single community. Ahn et al. (2010) employ Jaccard similarity to calculate the similarity of pairs of links.

Community Overlap PRopagation Algorithm (COPRA) (Gregory, 2010) is a multi-label propagation algorithm that explores overlapping communities. This algorithm is an improvement of single-label propagation to detect overlapping communities, which is described in the local community detection category. In these methods, each label indicates a community.

COPRA labels each vertex x with a set of pairs (c, b), where c is a community identifier and b is a belonging coefficient. The belonging coefficient indicates the strength of x's membership to the community c, such that all belonging coefficients for x sum to 1. Each propagation step sets x's

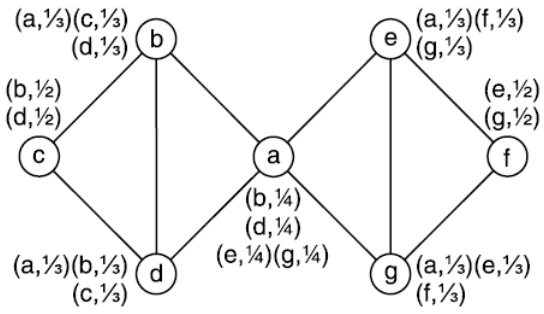

Figure 2.5: The first iteration of propagation of labels in COPRA (Gregory, 2010)

label to the union of its neighbors' labels, sums the belonging coefficients of the communities over all neighbors, and normalizes them. Figure 2.5 shows the result of the first iteration of COPRA. In the first iteration (Figure 2.5), each vertex gets the labels of its neighbors with an equal belonging coefficient for each. For example, node a has 4 neighbors b, d, e and g. Thus, it takes all these labels with a belonging coefficient of $1/4$ for each. During each propagation step, the algorithm first constructs the vertex label as above and then, deletes the pairs whose belonging coefficient is less than some threshold. This threshold is expressed as a reciprocal, $1/v$, where v represents the maximum number of communities to which any vertex can belong. Running COPRA on the network in Figure 2.5, where $v = 2$, gives the results in Figure 2.6. In the first iteration, vertex c is labeled with community identifiers b and d, each with belonging coefficient $1/2$. Since the belonging coefficients are not less than the threshold $(1/2)$, both are retained. Similarly, f is labeled with e and g. All the other five vertices have at least three neighbors, and their belonging coefficients are all below the threshold $(1/3)$. Figure 2.6 shows the rest of iterations. For example, b is labeled at first with $(a, 1/3), (c, 1/3), (d, 1/3)$: c is randomly chosen, a and d are deleted, and the belonging coefficient is re-normalized to $(c, 1)$. The labels for a, d, e and g are similarly randomly chosen. Before the final iteration, a has two neighbors labeled c and two labeled e, and so, it retains both community identifiers: $(c, 1/2), (e, 1/2)$. The final solution, therefore, contains two overlapping communities: a, b, c, d and a, e, f, g.

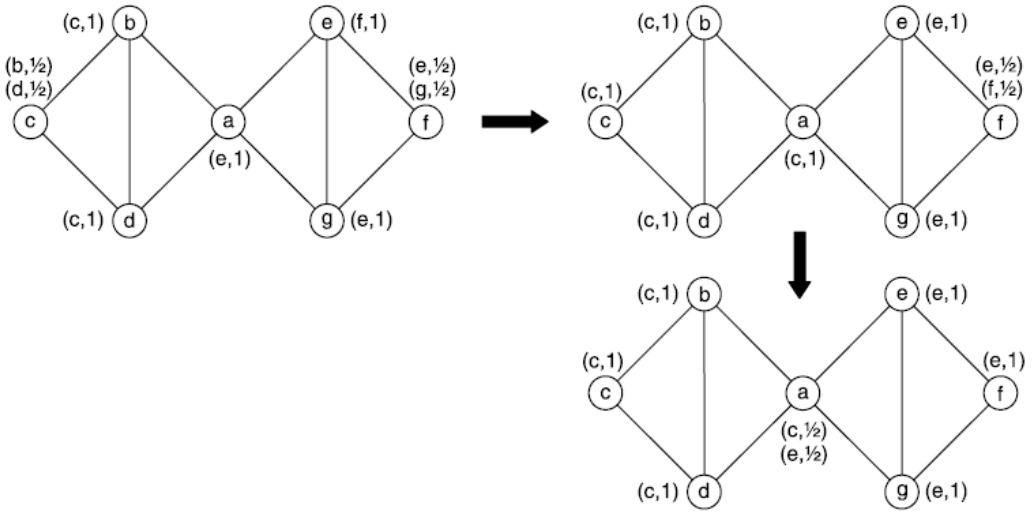

Figure 2.6: The rest of iteration of propagation of labels in COPRA with $v = 2$ (Gregory, 2010)

Local community detection methods are described as follows: Label propagation algorithm (LPA) (Raghavan et al., 2007) is a community detection algorithm that uses local information of nodes to explore communities. LPA works as follows:

(1) To initialize, every vertex is given a unique label.

(2) Then, repeatedly, each vertex x updates its label by replacing it with the label used by the greatest number of its neighbors. If more than one label is used by the same maximum number of neighbors, one of them is chosen randomly. After several iterations, the same label tends to become associated with all members of a community.

(3) All vertices with the same label are added to a separate community.

The algorithm terminates when every vertex has a label that is one of those that is used by a maximum number of neighbors. However, this method has a low time complexity and is easy to operate, the algorithm has great uncertainties. Raghavan et al. (2007) suggested that the number of required iterations is independent of the number of nodes. Also, it is reported that after five iterations, 95% of the nodes are accurately clustered. Also, according to some analysis in (Leung, Hui, Lio, & Crowcroft, 2009), it came to view that this algorithm without any optimization is able to detect communities on a graph with 1×10^9 edges in less than 180 minutes, in a magnitude similar to Louvain algorithm (Blondel et al., 2008). Figure 2.7 depicts an example of the process of community detection using LPA.

Order Statistics Local Optimization Method (OSLOM) (Lancichinetti, Radicchi, Ramasco, & Fortunato, 2011) is a local optimization method that locally optimizes the statistical significance of communities. The statistical significance is defined as the probability of finding the cluster in a random null model, i.e. a class of graphs without community structures. OSLOM consists of three

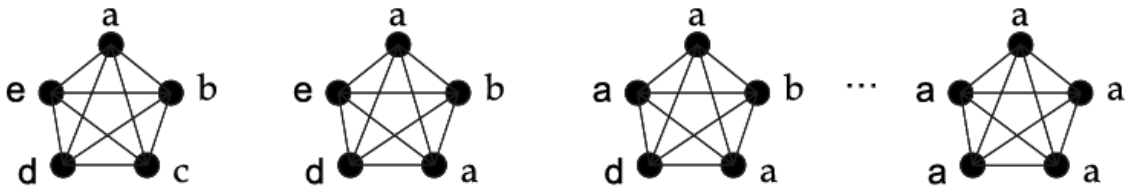

Figure 2.7: An example of label propagation algorithm (LPA) (Raghavan et al., 2007)

phases: first, it looks for significant clusters, until convergence; Second, it analyzes the resulting set of clusters, trying to detect their internal structure or possible unions thereof; Third, it detects the hierarchical structure of the clusters. OSLOM is the first method capable of detecting overlapping communities in directed, weighted, and dynamic networks.

Local node expansion is a local algorithm that starts from a number of nodes. The starting nodes can be selected randomly or by employing some criteria. Then, the selected nodes are expanded to discover a community for each while optimizing a local metric. Seed set expansion (Whang, Gleich, & Dhillon, 2013) is one of the most popular algorithms in the category of local node expansion methods.

In order to employ a local method to detect community partitioning in social networks, it is required to develop a local metric. A local metric evaluates communities independent of the global properties of the network. Clauset (2005) firstly discussed the problem of local community detection i.e. exploring a community for a given node using local information. Clauset (2005) proposed a metric, known as R, by defining 3 types of nodes in a graph (Figure 2.8): C is the set of nodes inside the community, B, boundary (or border) nodes, is the set of nodes in the community that have connections with nodes outside C, and U is the set of unknown nodes with at least one connection inside the community. According to the definition of the local community detection problem, only the introduced nodes are known to be used to evaluate the quality of the detected community. Let us denote the boundary adjacency matrix for B as follows, where $v_i, v_j \in V$:

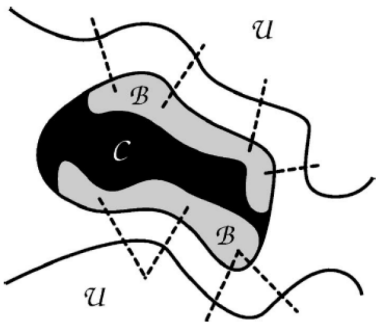

Figure 2.8: Local community C, its boundary nodes B, and the neighbors of the community U (Clauset, 2005)

Algorithm 1 The Clauset's Algorithm (Clauset, 2005)

Input: G, v_0, and n
Output: C: v_0's local community
$C = \{\}, U = \{\}$
$R = 0$
add v_0 to C
add all neighbors of v_0 to U
add v_0 to B
while $|C| < n$ **do**
    for $v \in U$ **do**
        Compute ΔR_v if v is added to C
    end for
    v_{max} = find v such that its ΔR_v is maximum
    add v_{max} to C
    update U, B, and R
end while

$$B_{ij} = \begin{cases} 1 & \text{if vertices } v_i \text{ and } v_i \text{ are connected, and either vertex is in } B; \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

In this regard, the local modularity R is defined as follows:

$$R = \frac{\sum_{ij} B_{ij}\delta(i,j)}{\sum_{ij} B_{ij}} = \frac{I}{T} \tag{3}$$

In (3), $\delta(i,j)$ is 1 when $v_i, v_j \in B$ or $v_i \in B$ and $v_j \in C$, or vice versa, and is 0 otherwise. Here, T is the number of edges with one or more endpoints in B, while I is the number of those edges with neither endpoints in U and at least one endpoint in B. In Clauset's algorithm, initially, the starting node is placed in the community, $C = \{v_0\}$, and also, its neighbors in U. At each step, the algorithm adds to C and to B if necessary, the neighboring vertex that results in the largest increase or smallest decrease in R, breaking ties randomly. For each vertex $v_j \in U$, the ΔR_j that corresponds to the change in local modularity as a result of joining v_j to C is calculated. Clauset's algorithm is shown in Algorithm 1. In this algorithm, n is the size of the final community. Clauset's algorithm is simple and efficient but needs to set the community size in advance. Following this method, several studies have been done and a number of local metrics along with their algorithms

have been developed to detect local communities (J. Chen, Zaïane, & Goebel, 2009; Q. Chen, Wu, & Fang, 2013; F. Luo, Wang, & Promislow, 2006; W. Luo, Zhang, Jiang, Ni, & Hu, 2018). These local community detection algorithms are discussed in Sections 3.2 and 4.2.

As it was mentioned before, real-world networks are dynamic and change over time. In this regard, several methods to detect communities in dynamic networks are proposed. During the past few years, different classifications for dynamic community detection methods are presented (S. Bansal, Bhowmick, & Paymal, 2011; Giatsoglou & Vakali, 2013). The most recent classification introduced by Dakiche, Tayeb, Slimani, and Benatchba (2019) is represented in this chapter. This classification of existing methods for tracking community evolution is based on their functioning principles and algorithmic techniques. Unlike static community detection, in the dynamic detection problem, the evolution of communities during time is very important. That is why, in dynamic community detection methods, the changes in the lifetime of communities are detected and analyzed. There are four main classes of methods as follows:

- **Independent Community Detection and Matching** includes methods that first detect communities at each time step and then, match them across different time steps. The basic idea is to detect communities at each time step (snapshot) independently and then, match those communities with the ones found in the previous step based on their similarity. For example, Y. Sun, Tang, Pan, and Li (2015) applied the Louvain algorithm (Blondel et al., 2008) to find the communities in each snapshot. Then, a correlation matrix is built to describe the relationship between communities in snapshot t and $t + 1$ concerning snapshot t. In this regard, the rules to detect evolution events according to the matrix are defined.

- **Dependent Community Detection** includes methods that detect communities at time t based on the topology of the graph at t and also, the previously found community structures. In other words, the detection of communities at a particular time t depends on the ones detected at time $t - 1$. This avoids the need to match communities, and consequently, provides smoothness in the community identification process. J. He and Chen (2015) improved the Louvain algorithm (Blondel et al., 2008) by including the concept of dynamism when forming communities. In this algorithm, the community structures of the network at time step 1 are initialized by using

the Louvain algorithm. Then, starting time 2 to t_τ and according to the changes during each time interval, a new graph is constructed. In this phase, the Louvain algorithm is applied to detect communities.

- **Simultaneous Community Detection on All Snapshots** includes methods that firstly, construct a single graph by adding changes to the network in different time steps and then, run a classic community detection on this graph. This category includes algorithms that consider all network evolution stages simultaneously. The idea of using snapshots is preserved, but they are all considered at the same time to discover coherent communities. Jdidia, Robardet, and Fleury (2007) built one network from different snapshots by binding the similar nodes appearing in different time steps. Also, they linked nodes connected to at least one common neighbor in two consecutive time-steps and then, employed a community detection algorithm, Walktrap (Pons & Latapy, 2005), to detect communities.

- **Dynamic Community Detection on Temporal Networks**, known as online methods, includes methods that do not detect communities from scratch each time but instead modify the previously found communities according to the network changes. The basic idea is to build and modify communities in an online fashion following the addition and suppression of nodes and edges. Algorithms falling into this category start by finding communities for the initial state of the network and then, update the communities for each incoming change. Rossetti et al. (2017) proposed Tiles, an algorithm that tracks the evolution of communities through time. In this algorithm, every time a new interaction emerges in the network, Tiles employ a label propagation process to propagate the changes to the node's surroundings, adjusting the neighbors' community memberships.

Regarding the local community detection in snapshot and temporal networks and according to the presented classification for dynamic networks, tracking local communities in dynamic networks can briefly be investigated in four different categories as follows:

(1) **Snapshot networks**: There are several changes at each time step.

- **Independent methods**: Any local community detection algorithm can simply be used

at each time step to detect a community independently.

- **Dependent methods**: A method must be able to maintain and update the local community at each snapshot according to the given changes, and the current structure of the community. Takaffoli, Rabbany, and Zaïane (2013) use static Alg_L algorithm (J. Chen et al., 2009) and the structure of the detected communities in the previous snapshots to find local communities in dynamic networks. This algorithm is described and discussed in Section 5.2.

(2) **Temporal networks**: There are streams of atomic changes.

- **Independent methods**: Local methods can be used after each atomic change.
- **Dependent methods**: The goal in this category is to update the structure of a local community after every atomic change without recalculations. Rigi, Moser, Farhangi, and Lui (2019) proposed the Derivative-based Community Detection (DCD) method inspired by geometric active contours (Caselles, Kimmel, & Sapiro, 1997). Geometric Active Contours are used extensively for detecting objects in 2D images in the field of machine vision. This study tried to address some shortcomings of the existing methods, especially efficiency and the ability to trace dynamic communities.

Finally, the problem of community detection in signed networks tries to find groups of users that are densely connected by positive links within the group and negative links between groups. Based on the classification presented by J. Tang, Chang, Aggarwal, and Liu (2016), there are four different methods to solve this problem in signed networks as follows:

- **Clustering-based methods**: In these methods, a positive link or a negative link indicates whether two nodes are similar or not similar. In this regard, community detection in signed networks is simplified to the clustering problem (N. Bansal, Blum, & Chawla, 2004; Sharma, Charls, & Singh, 2009).

- **Modularity-based methods**: These algorithms detect communities by optimizing signed modularity i.e. that is described in Section 6.4, called Q_s, or its variants for signed networks. Anchuri and Magdon-Ismail (2012) presented two new approaches dividing a signed network

into two sub-communities while minimizing frustration and maximizing signed modularity, Q_s. Then, the approaches are extended to detect all communities in signed networks. Frustration is proposed by Doreian and Mrvar (1996) that is described in Section 6.2. Also, Amelio and Pizzuti (2013) tried to find communities in signed networks while minimizing frustration and maximizing modularity, Q_s at the same time.

- **Mixture-model-based methods**: These methods detect the communities based on generative graphical models. This model has two advantages. First, providing soft-partition solutions in signed networks, and second, providing soft memberships that indicate the strength of a node belonging to a community. Algorithms in this category use two different approaches to identify communities. **Stochastic block-based models** create a network from a node perspective in which each node is assigned to a community and links are independently generated for pairs of nodes (J. Q. Jiang, 2015). While **probabilistic mixture-based models** generate a network from the link perspective (Y. Chen, Wang, Yuan, & Tang, 2014).

- **Dynamic model-based methods**: Dynamic-model-based algorithms detect community partitioning in dynamic signed networks. J. Chen, Liu, Hao, and Wang (2020) adopted the differential equations to evaluate the intimacy evolutionary behaviors. During the interactions in dynamic networks, intimacy between two nodes is calculated and updated. Nodes with higher intimacy gather into the same community and nodes with lower intimacy get away. Then, the community structure is formed in dynamic networks.

In this thesis, the problem of local community detection in static networks, unsigned dynamic networks, and signed networks are analyzed and investigated. Moreover, the issue of local evaluation of communities is addressed and a new local metric is presented. Since social networks have become huge in size and complex in structure, the local evaluation and exploration of communities are popular and efficient. In this regard, efficient algorithms employ a local perspective to detect communities while optimizing an objective function (Blondel et al., 2008). It can be concluded that the local community detection problem, i.e. detecting a community for a given node is the core issue of any community detection algorithm that uses local perspectives. Having fast and efficient algorithms to detect a community for a given node only using local information can significantly

enhance the quality of the resulted community partitioning. As a consequence, fast local community detection algorithms that can detect a high-quality community for any randomly given node, regardless of the degree and importance of the starting node in the network, are the main goals of such algorithms. In this thesis, a comprehensive study on the problem of local community detection is conducted. In this concern, a local quality metric to assess communities is proposed and examined. Moreover, three local community detection algorithms are developed to detect a community for a given node in static networks, dynamic networks, and signed networks, respectively.

Chapter 3

A Local Quality Metric for Communities

3.1 Introduction

One major question to the problem of community detection is how to assess different community detection algorithms. Normally, two community detection algorithms are compared via comparing their resulted community partitioning. Since there is not a shared and universally accepted definition of a community, evaluating the results provided by a community detection algorithm is a hard task. In literature, each study provides its own definition of communities, which is often maximizing a specific quality function, e.g. modularity (Newman, 2004), density, conductance (Andersen, Chung, & Lang, 2006). Since the communities, identified by a given algorithm, are based on their quality function, it is not guaranteed that they are able to capture the real sub-topology (community) of the network. For this reason, a common methodology to assess the quality of a community detection algorithm is to compare the resulted community partitioning with the ground-truth communities of the analyzed network. F-score and Normalized Mutual Information (NMI) (Friggeri, Chelius, & Fleury, 2011) are two widely used indexes for measuring the performance of community detection algorithms by comparing their resulted community partitioning with the ground-truth data. However, for a big number of networks, the ground-truth communities are not known. As a result, a comprehensive quality metric to evaluate the quality of the resulted communities and consequently, compare the community detection algorithms is required. Modularity (Newman, 2004), Q, is probably the most widely used quality function for the classic community detection problem. As can be

understood from equation (1), modularity Q needs global information to measure the quality of a community partitioning.

However, several studies have tried to compare different community detection algorithms via comparing their resulted community partitioning (Dao, Bothorel, & Lenca, 2018; Ghasemian, Hosseinmardi, & Clauset, 2019; Jebabli, Cherifi, Cherifi, & Hamouda, 2018), this issue is not widely investigated for local community detection algorithms. In this regard, a new local metric to evaluate the quality of communities using geodesic distance is proposed only by employing local information. To this aim, the shortest distance among all pairs of nodes in a given community is normalized and employed. In summary, the main contributions of this chapter are as follows:

- The minimum and maximum values of geodesic distance for a fixed number of nodes are presented and proved.
- A new metric to evaluate the quality of communities only using local information is proposed.
- Using some small graphs as communities, the deficiencies of other local metrics are discussed,
- Introducing two textbook experiments, the proposed metric is compared with the existing local metrics.
- The proposed metric has been used to compare three local community detection algorithms.
- The performance of the metric is compared with the performance of the F-score, i.e. the metric which compares the community with its ground-truth community.

The remainder of this chapter is structured as follows: Section 3.2 represents the literature review and discusses three famous local community detection algorithms. Section 3.3 introduces the metric, while Section 3.4 presents the experimental results and evaluation of the proposed metric. Finally, this study is concluded in Section 3.5 and also some ideas for possible future works are presented.

3.2 Related Works

During the last few years, many studies have tried to compare different community detection algorithms through evaluation of their detected communities (Dao et al., 2018; Ghasemian et al., 2019; Jebabli et al., 2018). Dao et al. (2018) tried to estimate similarities between two detected community partitionings by using the size distributions of the detected communities. In this regard, two community partitionings detected from two algorithms are compared to calculate how much the partitionings look alike. This approach can be used to calculate how much a community partitioning is similar to the ground-truth communities. Also, Jebabli et al. (2018) proposed a new technique to compare the detected community partitioning with the ground-truth communities. However, the two aforementioned studies (Dao et al., 2018; Jebabli et al., 2018) considered comparing communities with ground-truth data, the ground-truth communities are not available for several real-world networks.

In this regard, modularity Q (Newman, 2006) evaluates community partitionings without using the ground-truth data. As can be seen from (1), modularity Q still needs global information (m) to evaluate communities.

This chapter tries to compare local community detection algorithms by evaluating their detected communities. In this concern, a new metric to evaluate communities without considering the ground-truth data is proposed. Accordingly, only local information about detected communities is available which makes the process of evaluation and comparing more challenging. In the following, several famous local metrics along with their corresponding algorithms are presented. Usually, local community detection algorithms detect local communities, optimizing a local metric. Clauset (2005) firstly proposed the local metric, R, to detectc a community for a given node optimizing R. The metric (refer to (3)) and algorithm (refer to Algorithm 1) are described in Chapter 2.

Following (Clauset, 2005), F. Luo et al. (2006) proposed another metric, namely M, as follows:

$$M = \frac{E_{in}}{E_{ex}} \tag{4}$$

In (4), E_{in} is the number of edges within the community, and E_{ex} is the number of crossing edges, i.e. the edges that only have one endpoint in the community. The proposed algorithm in (F. Luo et

al., 2006) uses two steps to find the local community: addition and deletion steps. In the addition step, the algorithm iteratively adds vertices from U into the community C which results in the greatest increase in the value of M. In the deletion step, the algorithm iteratively removes vertices from C that their removal increases the score of M but does not separate the community. The addition and deletion steps are repeated until no vertex can be added into the community C.

Moreover, J. Chen et al. (2009) proposed another modularity, namely L, to overcome some drawbacks of the previous metrics:

$$L = \frac{L_{in}}{L_{out}} \tag{5}$$

$$L_{in} = \frac{E_{in}}{|C|}, L_{out} = \frac{E_{ex}}{|B|} \tag{6}$$

In (6), $|C|$ is the number of nodes inside community C and $|B|$ is the number of border nodes. The algorithm proposed in (J. Chen et al., 2009), contains two phases called discovery and examination. In the discovery phase, nodes incrementally are added to the community while maximizing L, and in the examination step, some irrelevant nodes are removed from the community. Regarding this algorithm, there are three different cases in which modularity L increases after adding one node into the local community. Assume L'_{in}, L'_{out} and L' are corresponding scores after merging a node v into C. The three cases that result in $L' > L$ are:

(1) $L'_{in} > L_{in}$ and $L'_{out} < L_{out}$,

(2) $L'_{in} < L_{in}$ and $L'_{out} < L_{out}$,

(3) $L'_{in} > L_{in}$ and $L'_{out} > L_{out}$.

In the first case, the node belongs to the community. However, nodes in the second case are outliers and do not belong to the community. Outliers are nodes that do not change the number of crossing edges but weaken the density inside the community. However, the decision regarding adding the third case nodes is made in the modification phase of the algorithm.

Furthermore, Conductance (Andersen et al., 2006) is probably the most widely used metric to evaluate communities locally. Equation (7) defines Conductance as follows:

$$Conductance = \frac{E_{ex}}{2 * E_{in} + E_{ex}} \tag{7}$$

It is noteworthy that some studies have tried to detect local communities while optimizing Conductance (Gao, Zhang, & Zhang, 2019; Van Laarhoven & Marchiori, 2016).

Also, W. Luo, Zhang, Ni, and Lu (2019) tried to define a local modularity, called LQ. LQ is defined as follows:

$$LQ = \frac{e_c}{S} - (\frac{d_c}{2S})^2 \tag{8}$$

In (8), e_c is the number of edges within the detected community, while d_c is the summation of degrees of all nodes belonging to that community. Also, S is the number of edges that have one or two endpoints in the community.

In addition, several local community detection algorithms detect communities, using different strategies (Liu & Xia, 2020; W. Luo et al., 2018; Y. Zhang, Wu, Liu, & Lv, 2019). However, the local metrics employed in these studies, cannot be used as a separate quality metric for communities.

Although the five aforementioned metrics are quite well-known in the literature, they all have their own drawbacks when they are considered as a general quality metric for communities. Regarding the definition of communities in social networks, it is apparent that a community with a denser connection within C and a sparser connection with its neighbors in U is a high-quality community. Modularity R employs the number of edges with at least one endpoint in B and no endpoints in U to control the density of a community (refer to (3)). This information cannot represent the density of the whole community. Because edges within C that do not have any endpoints in B are the valuable information ignored by R. Considering that the ignored information can be helpful to recognize the density of a community, neglecting it reduces the generality of the modularity R.

Also, modularity M uses the number of edges inside the community to represent the density (refer to (4)). However, the number of nodes in the community is neglected. In other words, only employing the number of edges inside the community, E_{in}, without considering the number of community nodes, $|C|$, cannot perfectly demonstrate the density of the community. Assume two communities of size n and n' with the same number of edges, where $n > n'$. It can be concluded that the community with n' nodes is the denser one. As a result, in addition to the number of edges, considering the number of nodes is crucial to represent the density of a community. Moreover, Conductance follows the same pattern as metric M in which the impact of internal density is increased

by adding a coefficient to the number of internal edges (refer to (7)).

Furthermore, metric L represents the density of the community by using the number of internal edges concerning the number of nodes in the community (refer to (5)). In metric L, L_{in} cannot perfectly show the density of the community, because the distribution of edges among nodes is neglected.

Also, it is proved that LQ is equivalent to modularity M by W. Luo et al. (2019). As a result, LQ suffers from the same drawbacks as M. Considering all the limitations of the above-mentioned metrics, it is proposed that the shortest path length between nodes inside the community can perfectly capture the density inside communities.

To overcome the above-mentioned drawbacks, several research papers tried to propose different modularities. However, only a few considered the shortest path length between pairs of nodes (Wu et al., 2013; Zhen-Qing, Ke, Song-Nian, & Jun, 2012) and to the best of our knowledge, all such metrics use global information. In this chapter, using geodesic distance, a new local metric to evaluate communities is proposed. This new metric is compared with the four aforementioned metrics, R, M, L and $Conductance$ and also, used to compare the three algorithms proposed in (Clauset, 2005), (F. Luo et al., 2006) and (J. Chen et al., 2009).

3.3 The Proposed Metric

Considering the limitation of local information, this section proposes a new metric, which uses local information to evaluate the quality of communities. In this section, the definition of geodesic distance, its minimum and maximum scores for fixed numbers of nodes, and the proposed metric are presented.

3.3.1 Definitions

To introduce the proposed metric, some definitions must be presented as follows: in an undirected and unweighted graph $G = (V, E)$, V is the set of nodes and E is the set of edges. A path in graph G is a sequence of edges that connects a sequence of nodes. The length of a path is the number of edges in the path. The shortest path between $v_i \in V$ and $v_j \in V$ is a path connecting

these two nodes, and shorter than any other paths connecting v_i and v_j, in terms of length. By definition, if $v_i, v_j \in V$ are two nodes and $(v_i, v_j) \in E$ is an edge between v_i and v_j, the length of the shortest path between v_i and v_j is 1. Furthermore, if there is no path connecting v_i and v_j, the length of the shortest path is infinite. A connected graph (or connected component) is a (sub-)graph in which the length of the shortest path between every pair of nodes is not infinite.

In addition, geodesic distance between nodes v_i and v_j, $gd(v_i, v_j)$, is the shortest path length between v_i and v_j. Moreover, the sum of the geodesic distances between every pair of nodes in a given graph G is denoted as GD_g and described as follows:

$$GD_g = \sum_{i<j,\ v_i,v_j \in V} gd(v_i, v_j) \tag{9}$$

Also, GD_{min} and GD_{max} show the minimum and maximum possible GD on graph G with n nodes, respectively.

Additionally, a diametral path in graph G is the shortest path between two nodes with the maximum length. The diameter of a graph G is the length of the diametral path. A complete graph K_n (or a clique) is a graph with n nodes and $n(n-1)/2$ edges where every pair of nodes are connected by an edge. In a complete graph, the geodesic distance between every pair of nodes is 1. Furthermore, P_n, is a path with n nodes in which the diameter is $n-1$.

Also, T_d is a graph with n nodes, $n-1$ edges, and the diameter d with no cycles, called a tree. In this regard, GD_{T_d} is the sum of geodesic distances between every pair of nodes in T_d and V_{T_d} is the set of nodes in T_d.

3.3.2 Propositions and proofs

In this section, two propositions corresponding to the main idea of this work are presented and proved. In this regard, two propositions to indicate the minimum and maximum values of GD_g on graph G with n nodes, are presented.

To determine the minimum and maximum scores of GD for a graph with n nodes, it is claimed that GD is minimized in a complete graph, K_n, and maximized in a path P_n.

Proposition 3.3.1. Among all graphs on n nodes, GD_{min} is achieved in a complete graph K_n, and $GD_{min} = \frac{n(n-1)}{2} = \frac{n^2-n}{2}$.

Proof. In any connected graph on n nodes, there are $\binom{n}{2} = \frac{n(n-1)}{2}$ number of paths between every pair of nodes. For every pair of nodes v_i and v_j in any connected graph G, $gd(v_i, v_j) \geq 1$. As a consequence, $GD_g = \sum_{i<j, v_i, v_j \in V} gd(v_i, v_j) \geq \frac{n(n-1)}{2}$. As a result, because $gd(v_i, v_j) = 1$ for every pair of nodes in K_n, $\frac{n(n-1)}{2}$ is achieved in complete graph K_n. □

Proposition 3.3.2. Among all graphs on n nodes, GD_{max} is achieved in a path P_n, and $GD_{max} = \frac{n(n-1)(n+1)}{6} = \frac{n^3-n}{6}$.

Proof. The maximum value of GD is achieved in a connected graph with no cycles (or a tree). Otherwise, removing one edge from an existing cycle increases GD. The following intermediate Lemma is required to complete the proof of Proposition 3.3.2:

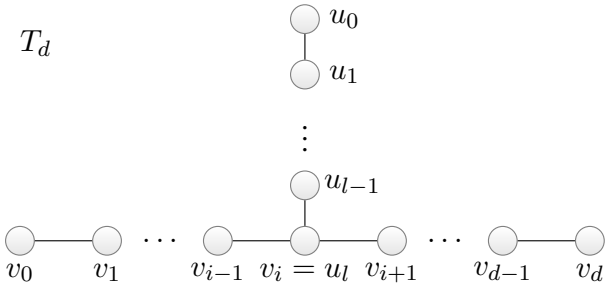

Figure 3.1: An arbitrary tree T_d

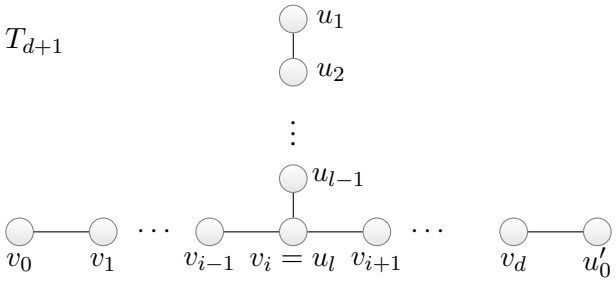

Figure 3.2: An arbitrary tree T_{d+1}

Lemma 3.3.1. For any tree on n nodes and diameter d, T_d, where $2 \leq d \leq n-2$, there exists another tree on n nodes, T_{d+1}, such that $GD_{T_d} < GD_{T_{d+1}}$.

Proof. Figure 3.1 shows tree T_d. Regarding this figure, assume $P = v_0, v_1, \cdots, v_i, \cdots, v_d$ is a diametral path in T_d. Consider a node with degree 1, $u_0 \notin P$, in T_d. Since $d \leq n-2$, there is at least one such node. Suppose the unique path from u_0 to $u_l = v_i$, with length l, intersects path P at node v_i (Figure 3.1). Denote this path as $S = u_0, u_1, \cdots, u_{l-1}, u_l(= v_i)$. Without loss of generality, assume that the length of the path from v_0 to v_i is less than or equal to the length of the path from v_i to v_d (or $gd(v_0, v_i) \leq gd(v_i, v_d)$).

A new tree T_{d+1} is constructed from T_d by cutting edge (u_0, u_1) and adding the edge (v_d, u'_0) as follows: remove the edge (u_0, u_1) from T_d. Add a new edge from v_d to a new node u'_0. Denote the new tree as T_{d+1}, where $T_{d+1} = T_d \backslash (u_0, u_1) \cup (v_d, u'_0)$. Figures 3.1 and 3.2 illustrate T_d and T_{d+1}, respectively. It must be shown that $GD_{T_d} < GD_{T_{d+1}}$.

As it can be seen from Figures 3.1 and 3.2, the geodesic distance length between many pairs of nodes in two trees T_d and T_{d+1} are the same. As a result, the only unequal distances are the lengths of u_0 to all nodes in T_d versus the lengths of u'_0 to all nodes in T_{d+1}. It should be shown that the latter value is greater than the first one. More formally, It must be proved that:

$$GD_{T_{d+1}} - GD_{T_d} = \sum_{v \in V_{T_{d+1}}} gd(u'_0, v) - \sum_{v \in V_{T_d}} gd(u_0, v) > 0 \tag{10}$$

Note that paths $S_d = u_0, u_1, \cdots, u_{l-1}, v_i, \cdots, v_d$ in T_d and $S_{d+1} = u_1, \cdots, u_{l-1}, v_i, \cdots, v_d, u'_0$ in tree T_{d+1} have the same length, and since both u_0 and u'_0 are leaves, then:

$$\sum_{v \in S_d} gd(u_0, v) = \sum_{v \in S_{d+1}} gd(u'_0, v) \tag{11}$$

It remains to consider the geodesic distances lengths from node u_0 to nodes $v_0, v_1, \cdots, v_{i-1}$ in T_d and also, from node u'_0 to $v_0, v_1, \cdots, v_{i-1}$ in T_{d+1}. Note that in T_d, $gd(u_0, v_k) \leq gd(v_d, v_k)$ for all $k = 0, 1, \cdots, i-1$. If not, then in T_d the path $u_0, u_1, \cdots, u_{l-1}, v_i, v_{i-1}, \cdots, v_k$ is longer than the path $v_d, v_{d-1}, \cdots, v_i, v_{i-1}, \cdots, v_k$. This means that the path $u_0, u_1, \cdots, u_{l-1}, v_i$ is longer than the path $v_d, v_{d-1}, \cdots, v_i$, and as a result, the path $u_0, u_1, \cdots, u_{l-1}, v_i, v_{i-1}, \cdots, v_0$ is longer

than the diametral path $P = v_d, v_{d-1}, \cdots, v_i, v_{i-1}, \cdots, v_0$, which is a contradiction. Therefore, $gd(u_0, v_k) \leq gd(v_d, v_k)$ for all $k = 0, 1, \cdots, i-1$ in T_d. It is clear that in tree T_{d+1}, $gd(u'_0, v_k) = gd(v_d, v_k) + 1$ and so, $gd(u_0, v_k) < gd(u'_0, v_k)$ for all $k = 0, 1, \cdots, i-1$.

Now, combining the latter with Equation 10 and 11, we get:

$$
\begin{aligned}
GD_{T_{d+1}} - GD_{T_d} &= \sum_{v \in V_{T_{d+1}}} gd(u'_0, v) - \sum_{v \in V_{T_d}} gd(u_0, v) \\
&= \sum_{v \in S_{d+1}} gd(u'_0, v) + \sum_{k=0}^{i-1} gd(u'_0, v_k) - \sum_{v \in S_d} gd(u_0, v) - \sum_{k=0}^{i-1} gd(u_0, v_k) \\
&= \underbrace{\left(\sum_{v \in S_{d+1}} gd(u'_0, v) - \sum_{v \in S_d} gd(u_0, v) \right)}_{=0 \text{ (By Equation 11)}} \\
&+ \underbrace{\left(\sum_{k=0}^{i-1} gd(u'_0, v_k) - \sum_{v \in S_d} gd(u_0, v) \right)}_{>0} \\
&> 0
\end{aligned}
\tag{12}
$$

This completes the proof of Lemma 1. □

To continue the proof of Proposition 3.3.2, from Lemma above, the GD is maximized when the diameter of a tree is maximized. Since the diameter in any tree (and any connected graph) on n nodes is upper bounded by $n-1$, and there is a single tree with diameter $n-1$, then the maximum GD is achieved in P_n. To complete the proof, the value of GD_{max} is calculated. In P_n, there are exactly $n-1$ pair of nodes with distance 1 (all pairs of neighboring nodes). Also, there are exactly $n-2$ pairs of nodes with a distance of 2. In general, there are $n-i$ pairs of nodes with distance i for all $1 \leq i \leq n-1$ Thus, it is concluded that:

$$
\begin{aligned}
GD_{max} &= \sum_{i=1}^{n-1} i(n-i) \\
&= n\sum_{i=1}^{n-1} i - \sum_{i=1}^{n-1} i^2 \\
&= n\Big(\frac{n(n-1)}{2}\Big) - \Big(\frac{n(n-1)(2n-1)}{6}\Big) \\
&= \frac{n(n-1)(n+1)}{6} = \frac{n^3-n}{6}
\end{aligned}
\tag{13}
$$

Therefore, the proof of proposition 3.3.2 is completed. □

3.3.3 The Proposed Metric

Regarding the definition of a community, a quality metric must be able to capture the density inside and the sparsity from the outside of the community. In this regard, a new metric for locally evaluating the quality of a community in social networks is proposed. Equation (14) represents the proposed geodesic distance metric GDM_C, by using NGD for the community C as follows:

$$
GDM_C = \frac{X}{NGD_C} \tag{14}
$$

In (14), NGD_C is the Normalized Geodesic Distance of C. Here, the geodesic distance of community C is normalized to be used in GDM_C, as follows:

$$
NGD_C = \frac{GD_C - (GD_{C_{min}} - 1)}{GD_{C_{max}} - (GD_{C_{min}} - 1)} \tag{15}
$$

In (15), GD_C is the sum of the geodesic distances of community C, $GD_{C_{min}}$ and $GD_{C_{max}}$ are the minimum and maximum values for the geodesic distance of community C of size $|C|$. Regarding (15), the value of geodesic distance for any community is normalized between 0 and 1. It is clear that for a complete community, a community with all possible edges among its nodes, the geodesic distance gives its minimum value, $GD_{C_{min}}$. As a result, the normalized geodesic distance for such communities is 0. To avoid having 0 values, instead of $GD_{C_{min}}$, we use $GD_{C_{min}} - 1$. Furthermore,

in (14), X is the parameter that controls the number of crossing edges and is described as follows:

$$X = \frac{E_{in}}{E_{in} + 4 * E_{ex}} \tag{16}$$

Equation (16) represents the parameter X as a function of the number of internal edges E_{in} and the number of crossing edges E_{ex}. In GDM, both E_{in} and NGD are employed to capture the density inside the community. As a result, the impact of the density is more than the number of external edges. To make the number of crossing edges as important as the density inside the community, coefficient 4 is multiplied by E_{ex} in X. In this regard, different coefficients have experimented and 4 seems to make the best balance.

In GDM, both X and NGD range between 0 and 1, which makes them comparable to be used together in a formula. Also, NGD perfectly captures the density inside a community and shows how dense a community is, only using its local information. Besides, the parameter X represents how much a community is separated from its neighbors. Thus, it is expected that the proposed metric performs well while maximizing the density inside the community and minimizing the number of crossing edges. In terms of complexity, the proposed GDM is more complex than the other local metrics. The complexity of the calculation of GDM for a community of size n is $O(n^3)$. Since the proposed GDM is a comprehensive quality metric that is used only once to assess the quality of detected communities, its complexity does not raise any additional concerns.

3.4 The Experiments

In this section, GDM is employed to compare three local community detection algorithms by comparing their detected communities. Also, the performance of GDM is compared with the performance of the F-score, i.e. the metric which compares a given community with its ground-truth community. In addition, using some simple graphs as sample communities and two textbook experiments, it is demonstrated how GDM covers the deficiencies of other local metrics.

3.4.1 dataset

In order to conduct the experiments, several social networks with ground-truth communities are employed which are used by many research studies (Hric, Darst, & Fortunato, 2014; W. Luo et al., 2018). The employed datasets in this chapter are as follows:

- **Zachary Karate Club** (Zachary, 1977) is a 34-node network with 78 edges which represents the club members and the friendship among them. By arising a problem between the masters, the club splits into two smaller communities of sizes 16 and 18. These two groups are considered two ground-truth communities. Figure 3.3 shows the Karate Club network with the two ground-truth communities in two colors.

- **Dolphins** (Lusseau et al., 2003) is a network of frequent association between 62 dolphins. This network contains 62 nodes and 159 edges along with two ground-truth communities of sizes 20 and 42.

- American Football College (**American FC**) (Girvan & Newman, 2002) consists of 115 nodes and 616 edges which represent football teams and regular games among them respectively. Normally, games are more frequent between teams of the same conference than between teams of different conferences. As a result, each conference can be considered as one ground-truth community of the network.

- Lancichinetti-Fortunato-Radicchi (**LFR**) benchmark (Lancichinetti, Fortunato, & Radicchi, 2008) is a synthetic network generator in which the ground-truth communities are known. To

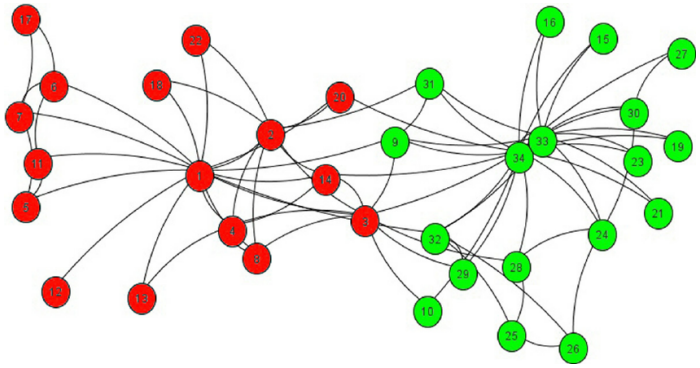

Figure 3.3: Zachary Karate Club network with the two ground-truth communities

generate a network using LFR, some parameters need to be set. The parameters include the number of nodes n, the average degree of nodes d, the maximum degree of nodes $maxd$, the minimum size of communities $minSize$, the maximum size of communities $minSize$ and the mixing parameter $0 < \mu < 1$. The mixing parameter indicates the quality of the ground-truth communities in the network. In this regard, $(1 - \mu)\%$ of a nodes degree belong to the same community as the node, and $\mu\%$ of a nodes neighbors must belong to other communities than the node's community, ($n = 1000, d = 15, maxd = 50, minSize = 20, maxSize = 50$ and $\mu = 0.1, \cdots, 0.8$).

- **DBLP**[1] computer science bibliography provides a comprehensive list of research papers in computer science. In this network, two authors are linked if they publish at least one paper together. Since each publication venue is considered a ground-truth community, authors who published in a certain journal or conference form a community. This network has 317070 nodes and 1049866 edges.

- **Amazon** co-purchasing network[2] was collected from Amazon website[3] and the ground-truth communities are defined by each products category. This is a 334863-node network with

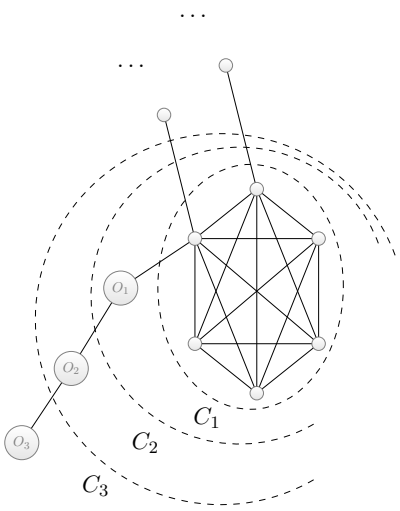

Figure 3.4: First simple example community

[1] http://snap.stanford.edu/
[2] http://snap.stanford.edu/
[3] https://www.amazon.com/

925872 edges.

3.4.2 Small Graphs

This section describes the drawbacks of the four metrics, R (Clauset, 2005), M (F. Luo et al., 2006), L (J. Chen et al., 2009) and Conductance (Andersen et al., 2006), using some small graphs as communities. In this regard, some small communities are created, and the best community among them is selected in advance. Complete graphs are chosen to be considered the best communities. Then, the communities are compared with each other using the metrics and it is expected that the best scores of the metrics go to the selected best community. It is noteworthy that R, M, L, and GDM are maximization metrics and $Conductance$ is a minimization metric. Also, since W. Luo et al. (2019) proved that modularity LQ (refer to (8)) is equivalent to metric M and has the same performance as M, it is removed from this evaluation.

Figure 3.4 demonstrates the first example which is a small sub-graph of a bigger graph. Regarding this figure, it is considered that three communities C_1, C_2, and C_3 are detected by three different algorithms. The purpose of this analysis is to compare three communities C_1, C_2 and C_3, using R (Refer to (3)), M (Refer to (4)), L (Refer to (5)), $Conductance$ (Refer to (7)) and GDM (Refer

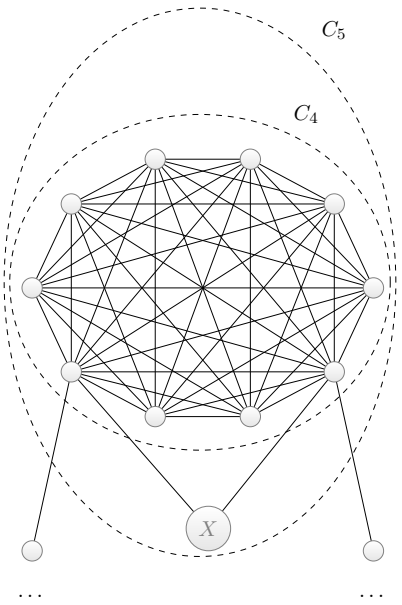

Figure 3.5: Second simple example community

Table 3.1: Metrics' scores for communities C_1, C_2 and C_3 in Figure 3.4

Community	R	M	L	$Conductance$	GDM	NGD	X
C_1	0.75	5	1.6	0.090	**11.46**	0.048	0.55
C_2	0.77	5.3	**2.3**	0.085	3.43	0.166	0.57
C_3	**0.78**	**5.6**	2.1	**0.081**	1.98	0.298	0.59

Table 3.2: Metrics' scores for communities C_4 and C_5 in Figure 3.5

Community	R	M	L	$Conductance$	GDM	NGD	X
C_4	0.8	11.2	2.25	0.04	**90.24**	0.0082	0.74
C_5	**0.9**	**23.5**	**4.45**	**0.02**	15.68	0.0542	0.85

to (14)). Since C_1 is the most compact community, i.e. because it is a complete graph, among the three with the same number of external edges as C_2 and C_3, it is clear that C_1 is by far the best community in comparison with the other two. Thus, it is expected that the highest (best) scores of metrics go to C_1, then C_2, and finally C_3. Table 3.1 summarizes how different metrics, including GDM, assess the three communities. As is clear in Table 3.1, community C_3 is the best community in comparison with the other two communities, according to metrics R, M, and $Conductance$. Also, if we consider L as a general quality metric, community C_2 would be selected as the best among C_1, C_2 and C_3. However, GDM assigns its highest score to C_1 and after merging O_1 to the community C_1, the value for GDM decreases remarkably. Afterward, by joining O_2, GDM drops slightly. It is reasonable to have a huge decrease in the quality of the community after adding O_1 to the community since the density inside the community weakens considerably. As a result, unlike the other metrics, the proposed metric GDM can compare the aforementioned communities very well.

Figure 3.5 shows the second example graph analyzed in this section. In this figure, a complete 10-node graph (or a 10-clique) is considered as a community C_4 that has 4 external edges which makes it a high-quality community. There is a neighboring node x, which is connected to the community by 2 edges. Thus, merging x into the community decreases the number of crossing edges by 2 while weakening the density of the community C_4. In this example, C_4 is considered

a better community than C_5. In this regard, it is expected that the best scores of the metrics are given to C_4. Table 3.2 shows different metrics' results for two communities of Figure 3.5 (C_4 and C_5). However, metrics R, M, L, and $Conductance$ welcome node x into the community C_4, the proposed metric GDM prefers to keep x outside the community. It is due to the fact that merging node x profoundly damages the density of a complete 10-node community while decreasing the number of crossing edges only by 2. Increasing the number of crossing edges by 1 or 2 does not considerably decrease the quality of a complete 10-node community. Thus, GDM works precisely based on the definition of communities in social networks. Since C_4 is a clique, node X needs to be connected to all nodes in the community to be allowed by GDM to be added into C_4. However, usually, communities in the social networks are less dense than C_4 and there is no need for a node like X to be connected to all node to make the community denser. In this regard, a trade off between the density inside and the number of crossing edges lets node like X into the community.

Figure 3.6 indicates the third example. This figure is an extension of Figure 3.5 in which instead of a single node x, there is a complete 6-node community (K_6). In Figure 3.6, the best practice is to consider C_6 as a separate community and keep the K_6 outside of C_6. As a result, it is expected that the best scores of the metrics go to C_6 rather than C_7. Table 3.3 represents the calculation of different metrics regarding communities C_6 and C_7 in Figure 3.6. It can be seen from Table 3.3

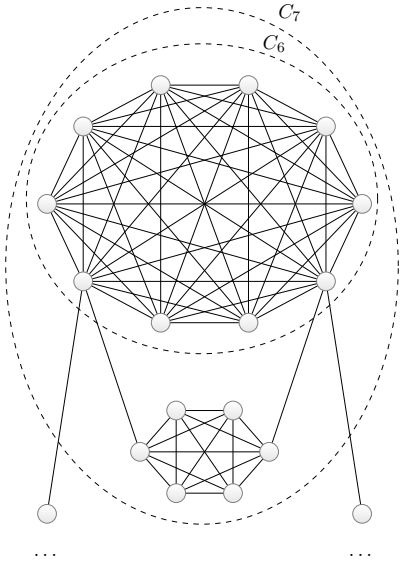

Figure 3.6: Third simple example community

Table 3.3: Metrics' scores for communities C_6 and C_7 in Figure 3.6

Communities	R	M	L	$Conductance$	GDM	NGD	X
C_6	0.8	11.25	2.25	0.04	**90.24**	0.008	0.74
C_7	**0.9**	**31**	**6.872**	**0.01**	6.24	0.141	0.88

that all four metrics R, M, L and $Conductance$ give higher scores to C_7 rather than C_6. However, it is crystal clear that keeping K_6 outside of C_7 and considering it as a separate community result in better community partitioning. As a result, it can be concluded that for metrics R, M, L, and $Conductance$, minimizing the number of crossing edges is much more important than the level of density of the community. This conclusion is more prominent in the next community example when we have a disconnected graph as a community. Furthermore, according to Table 3.3, it can be concluded that GDM to some extent, can solve the problem of resolution limit. In this regard, however, in Figure 3.6, the most obvious sub-graphs (cliques) are chosen as communities and the number of crossing edges is small, metrics R, M, L, and $Conductance$ are unable to correctly compare the communities. As a result, we can claim that GDM, to a very high extent, can solve the resolution limit problem in local community evaluation. In this regard, in the same way as Fortunato and Barthelemy (2007), by considering some restrictions for a group of nodes to be a community, the exact bound for the resolution limit of GDM can be found.

Finally, the example graph in Figure 3.7 shows a disjoint community. For such communities, the proposed metric GDM results in 0, while the other four metrics may give rather high scores. Regarding the community in Figure 3.7, the metric scores for R, M, L and $Conductance$ are 0.7, 2.5, 1.7 and 0.16, respectively. These results indicate that the number of external edges is much more important than the density inside the community for the above-mentioned metrics.

It is noteworthy that the purpose of using complete graphs as best communities is to recognize the best community by observation and without any calculation. In this regard, non-complete graphs with high qualities could be used as well.

3.4.3 Two Textbook Experiments

In this section, two experiments are designed and implemented. The structures of both experiments are according to the fact that the ground-truth communities must be given the highest scores in comparison with any other detected communities for the same given node.

The First Experiment

In the first experiment, assume that there is a ground-truth community RC_n of size n in a given dataset. Regarding ground-truth communities, at each step, one node y from RC_n is chosen to be removed. Here, the removed node is still in the network but not in the community. The resulted community after removing node y is denoted as $RC_n - \{y\}$. Considering $RC_n = \{a_1, a_2, \cdots, a_n\}$, the resulted communities from removing each member a_i, $1 \leq i \leq n$, are $RC_n - \{a_1\}$, $RC_n - \{a_2\}$, $\cdots$, and $RC_n - \{a_n\}$. Also, at each step, one node from all over the network outside of RC_n is chosen to be merged into it. The resulted communities are $RC_n + \{b_1\}$, $RC_n + \{b_2\}$, $\cdots$ and $RC_n + \{b_s\}$, where $S = \{b_1, b_2, \cdots, b_s\}$ is the whole network outside of RC_n. In order to compare the above-mentioned communities with RC_n, the metrics R, M, L, $Conductance$ and GDM are calculated and compared. It is expected that RC_n is given the highest scores from all four mentioned metrics. The purpose of this experiment is to capture the most similar communities to the ground-truth communities and compare them with the ground-truth communities. If a metric can give higher scores to the ground-truth communities in comparison with the values it is given to the most similar communities, the metric is judged as a good metric.

Table 3.4 summarizes the results of the experiment on two ground-truth communities of the

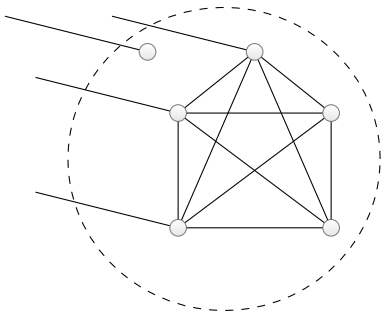

Figure 3.7: Fourth simple example community

Table 3.4: The result of the first textbook experiment on the Karate Club network

RC	Community	R	M	L	Con	GDM
RC_{16}	RC_{16}	0.714	3.3	1.0312	0.1315	2.5877
	$RC_{16} - \{a_3\}$	0.687	2.8	**1.12**	0.1515	2.1582
	$RC_{16} - \{a_5\}$	0.675	2.308	**1.077**	0.1781	1.9398
	$RC_{16} - \{a_6\}$	0.658	2.071	**1.105**	0.1944	1.7679
	$RC_{16} - \{a_7\}$	1.7679	2.071	**1.105**	0.1944	3.495
	$RC_{16} - \{a_{11}\}$	0.675	2.308	**1.077**	0.1781	1.9398
	$RC_{16} - \{a_{13}\}$	0.676	2.583	**1.033**	0.1622	2.1301
	$RC_{16} - \{a_{17}\}$	0.7	2.583	**1.205***	0.1621	2.3858
	$RC_{16} + \{b_9\}$	0.710	**3.4***	**1.2**	0.1358	**2.6699***
	$RC_{16} + \{b_{10}\}$	**0.722***	1.470	0.784	**0.1282***	2.5233
RC_{18}	RC_{18}	0.756*	3.5	1.555*	0.125	2.7628
	$RC_{18} + \{b_3\}$	0.730	**4***	1.053	**0.1111***	**3.1699***
	$RC_{18} + \{b_{20}\}$	0.744	3.273	1.550	0.1325	**2.7802**

Zachary Karate Club network (RC_{16} and RC_{18}). It is noteworthy that $Conductance$ is referred to as Con in the tables. This table includes two categories for two ground-truth communities of this network. The first row of each category represents the metrics' scores for RC_{16} and RC_{18} and the other rows contain the additions or deletions which are given higher scores than RC_{16} or RC_{18} at least by one of the metrics. It is noteworthy that the scores which are higher than that of ground-truth communities are shown in bold and the maximum scores are denoted by an asterisk. According to this table, since modularity R has only one wrong evaluation, i.e., it gave a higher score to $RC_{16} + \{b_{10}\}$ rather than RC_{16}, it has the best performance. Regarding Table 3.4, GDM gives higher score to $RC_{16} + \{b_9\}$ rather than RC_{16}. For more illustration, adding node 9 into RC_{16} increases GDM by 0.082 (from 2.5877 to 2.6699). Fig. 3.8 demonstrates the position of b_9 in the network. In the figures, node a_9 is shown by using the node number. Regarding this figure, node 9 belongs to the community RC_{18} with three connections to nodes 31, 33, and 34 in which node 34 is a core node for RC_{18}. On the other hand, node 9 is connected to RC_{16} with two edges to nodes 1 and 3 (with node 1 being a core node for RC_{16}). By comparing the connections of node 9 to the two ground-truth communities, it will be known that node 9 must belong to RC_{18} since removing 9 from RC_{18} increases the number of crossing edges by 1 and also weaken the density

inside. However, it does not mean that if node 9 is added to RC_{16}, it would negatively affect RC_{16}. Because adding node 9 into RC_{16} increases the number of edges and nodes inside the community by 2 and 1, respectively. The reason is that node 9 is connected to the core node which does not let the density decline. All the above-mentioned reasons imply that the presence of node 9 in both communities can help them to be better communities But removing node 9 from RC_{18} decreases the metric GDM more than 0.202 (It is decreased by 0.451, which is not mentioned in the table.). Thus, one may conclude that node 9 can be considered as an overlapping node, which can belong to both communities. This implies that GDM is able to recognize overlapping communities as well. In the same way, $RC_{18} + \{b_3\}$ and $RC_{18} + \{b_{20}\}$ are given higher scores than RC_{18} by GDM. Fig. 3.9 and 3.10 illustrate the status of nodes 3 and 20 in the network and because of the same reasons as $RC_{16} + \{b_9\}$, they can also be referred to as overlapping nodes.

According to Table 3.4, in 8 cases, metric L results in higher values than that of RC_{16}. So, it

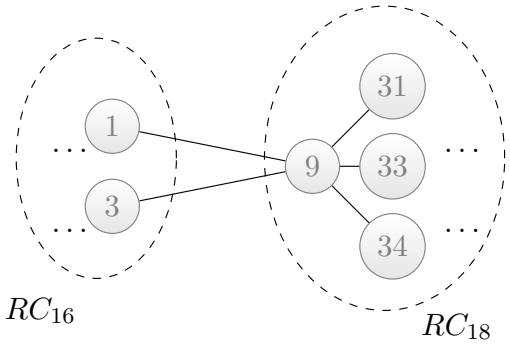

Figure 3.8: The status of node 9 in Karate Club network

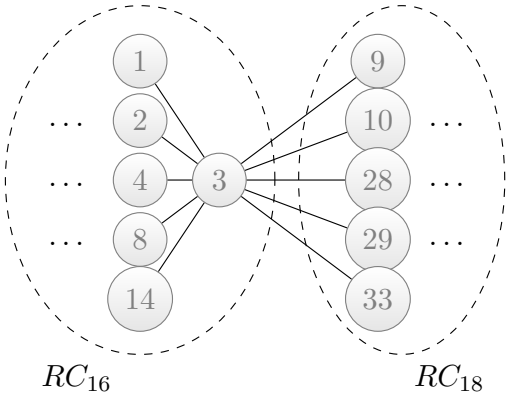

Figure 3.9: The status of node 3 in Karate Club network

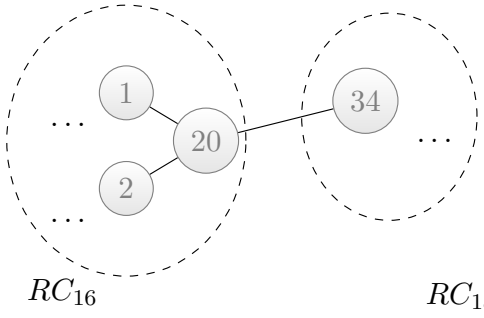

Figure 3.10: The status of node 20 in Karate Club network

Table 3.5: The number of wrong evaluations by the four metrics on Dolphins network concerning the first textbook experiment

RC	Metric	Adding	Removing
RC_{20}	R	0/42	5/20
	M	1/42	0/20
	L	6/42	13/20
	Con	1/42	0/20
	GDM	1/42	1/20
RC_{42}	R	0/20	38/42
	M	0/20	0/42
	L	4/20	26/42
	Con	1/20	0/42
	GDM	1/20	0/42

Table 3.6: The number of wrong evaluations by the four metrics on American FC network concerning the first textbook experiment

RC	Metric	Adding	Removing
RC_{13}	R	2/102	0/13
	M	2/102	0/13
	L	2/102	0/13
	Con	2/102	0/13
	GDM	1/102	0/13

can be concluded that the proposed metric can evaluate the quality of communities at least better than the L metric. Additionally, the same experiment is carried out on Dolphins and American FC networks. Table 3.5 indicates the number of wrong evaluations of the four metrics in the Dolphins network. In this table, the first column shows the name and size of the ground-truth communities and the second and third columns indicate the number of wrong evaluations of adding one node into and removing one node from the ground-truth community. As can be seen from the table, metrics R and L have the worst performance in comparison with the other two metrics. Regarding Table 3.5, deletion is the most challenging part of this experiment for R and L. Fig. 3.11 shows an

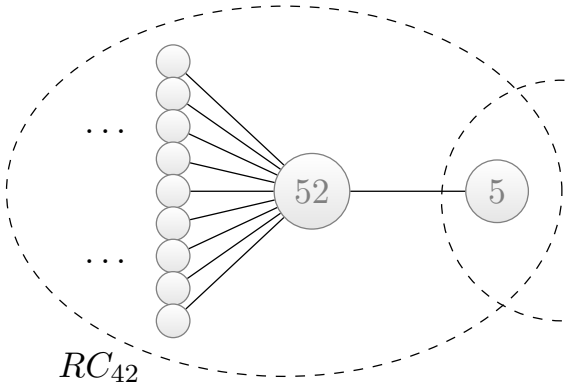

Figure 3.11: An example of wrong assessment of communities by R modularity

example of removing one node from one ground-truth community of Dolphins network in which metric R cannot correctly assess the communities. Concerning this figure, removing node 5 from the community will make node 52 a new border node. Thus, this increases the number of crossing edges by 1 and also increases the number of edges with at least one endpoint in border nodes by 10. As a result, metric R is wrongly increased remarkably. Consequently, we can conclude that metric R is not a quality metric to be used independently to assess communities. However, metrics M, $Conductance$ and GDM perform satisfactorily for this network as well. Moreover, the same experiment is done on the American FC network which does not arise any challenge for any of the metrics. This network has 11 ground-truth communities in which only in one real community of size 13, metrics have wrong evaluations in addition steps, which can be taken into account as overlapping nodes. Table 3.6 summarizes the number of wrong evaluations of metrics for this specific ground-truth community.

The Second Experiment

To show the effectiveness of the proposed metric another experiment is designed and applied to the American FC network. In this experiment, two ground-truth communities are merged, and then the joint community is compared with each of the two ground-truth communities. The purpose of this experiment is to show that the three metrics R, M, L, and $Conductance$ cannot perfectly capture the density inside the community. In this experiment, a wrong evaluation happens when a metric gives a joint community, a score higher than the two real communities which were used to construct the joint community. In this regard, Table 3.7 shows the wrong evaluation of metrics R, M, L, and $Conductance$, only for two ground-truth communities of Football College network. In this table, the first column indicates the community, and the other four columns denoted the metrics scores for the corresponding communities. It should be mentioned that the ground-truth communities of American FC network are numbered from 0 to 10. So, RC_{0_9} is denoted as the ground-truth community of number 0 and size 9. As it can be understood from the table, metrics R, M, L, and $Conductance$ give the combined community higher scores than the ground-truth communities. Moreover, however, the combined community, $RC_{1_8} + RC_{3_{12}}$, results in a disconnected community, it earned higher scores than DC_{1_8} by the four metrics. Furthermore, the total number of wrong

Table 3.7: The result of the second textbook experiment on American FC network

Community	R	M	L	$Conductance$	GDM
RC_{0_9}	0.59	1.44	1.44	0.2577	**22.5**
$RC_{0_9} + RC_{1_8}$	**0.605**	**1.533**	**1.533**	**0.2459**	2.2736
$RC_{0_9} + RC_{3_{12}}$	**0.598**	**1.491**	**1.491**	**0.2511**	1.4998
$RC_{0_9} + RC_{6_{13}}$	**0.6**	**1.5**	**1.5**	**0.25**	1.5172
$RC_{0_9} + RC_{9_{12}}$	**0.65**	**1.89**	**1.89**	**0.2088**	2.5304
$RC_{0_9} + RC_{10_9}$	**0.592**	**1.451**	**1.451**	0.4074	0.7252
RC_{1_8}	0.483	0.933	0.933	0.3488	**10.7837**
$RC_{1_8} + RC_{3_{12}}$	**0.543**	**1.187**	**1.187**	**0.2963**	0.0
$RC_{1_8} + RC_{4_9}$	**0.542**	**1.185**	**1.185**	**0.3367**	0.0
$RC_{1_8} + RC_{6_{13}}$	**0.625**	**1.666**	**1.666**	**0.2307**	2.4314
$RC_{1_8} + RC_{7_8}$	**0.487**	**0.95**	**0.95**	**0.3448**	0.9528
$RC_{1_8} + RC_{8_{10}}$	**0.543**	**1.19**	**1.19**	**0.2959**	1.2005
$RC_{1_8} + RC_{9_{12}}$	**0.551**	**1.226**	**1.226**	**0.2897**	0.0
$RC_{1_8} + RC_{10_9}$	**0.516**	**1.066**	**1.066**	0.4834	0.3466

Table 3.8: The number of wrong evaluations by the four metrics on American FC network concerning the second textbook experiment

Dataset	R	M	L	Conductance	GDM
American FC	77/110	78/110	78/110	68/110	**0/110**

evaluations of the combined community experiment is indicated in Table 3.8 for the five metrics. Respecting this table, among 110 possible comparisons, GDM has no wrong evaluation. However, the other four metrics R, M, L, and $Conductance$ have 77, 78, 78, and 68 wrong assessments, respectively. This experiment shows that the three above-mentioned metrics are designed to compare the same communities with only one node difference. However, the proposed metric in this paper is capable of comparing every community for a given node. This metric is aimed to assess the quality of detected communities instead of only comparing them with ground-truth communities. Since ground-truth communities of some real-world networks are unknown, there is a need for a comprehensive local quality metric to evaluate the quality of detected communities.

3.4.4 Experimental Results

In this experiment, three local community detection algorithms, Alg_R (Clauset, 2005), Alg_M (F. Luo et al., 2006) and Alg_L (J. Chen et al., 2009), are compared with each other via comparing their detected communities. After randomly choosing one node as the starting node for the community detection algorithms, each algorithm gives one community corresponding to the starting node. Among the three communities detected by the three algorithms, the one with the maximum F-score is the best community compared to the ground-truth one. F-score is a measure of a model's accuracy on a dataset which is defined as follows:

$$F1score = 2 * \frac{precision_s * recall_s}{precision_s + recall_s} \tag{17}$$

$$recall_s = \frac{|C_{found} \cap C_{true}|}{|C_{true}|} \tag{18}$$

$$precision_s = \frac{|C_{found} \cap C_{true}|}{|C_{found}|} \tag{19}$$

Equations (18) and (19) show the formula of recall and precision for sets, respectively. In these equations, C_{found} is the set of nodes of the community found by an algorithm, and C_{true} is the set of nodes that belong to the ground-truth community. In (18), $C_{found} \cap C_{true}$ is the intersection of sets C_{found} and C_{true} which shows the set of correctly detected nodes (true positive) by the algorithm. As a result, $recall_s$ indicates the fraction of the number of true positive nodes over the number of nodes that must be detected as positive. Also, equation (19) shows the fraction of the number of true positive nodes over the whole number of detected nodes. In other words, $precision_s$ deals with the nodes which are correctly detected as negative (true negative). A perfect model (or community) has an F-score of 1 which means the detected community is identical to its ground-truth community. In this experiment, the final judgment of the F-score is compared with the final judgment of GDM and it is shown how much the results of GDM are similar to that of the F-score. It is expected that GDM compares the communities, in the same way as F-score. In other words, both metrics choose the same community as the best in the same comparison.

Table 3.9: The values of F-score for some communities detected by Alg_R, Alg_M, and Alg_L regarding some random starting nodes on Karate Club network

Starting node	Alg_R	Alg_M	Alg_L	Starting node	Alg_R	Alg_M	Alg_L
2	**0.90**	0.73	0.58	11	**0.96**	0.47	0.22
3	**0.84**	0.73	0	28	0.971	**0.973**	0.36
4	**0.90**	0.73	0.54	26	**0.97**	0.36	0.36

Table 3.10: The values of GDM for some communities detected by Alg_R, Alg_M, and Alg_L regarding some random starting nodes on Karate Club network

Starting node	Alg_R	Alg_M	Alg_L	Starting node	Alg_R	Alg_M	Alg_L
2	**2.40**	1.61	0.39	11	**2.52**	0.60	0.059
3	**1.88**	1.61	0	28	2.60	**3.16**	0.20
4	**2.40**	1.61	0.47	26	**2.61**	0.20	0.20

Table 3.9 shows the values of the F-score for several communities detected by the three algorithms Alg_R, Alg_M, and Alg_L on the Karate Club network. In this table, the first column shows the name of the randomly selected starting node and the next three columns illustrate the values of the F-score for the detected local communities by three algorithms Alg_R, Alg_M, and Alg_L for the corresponding starting node. As can be seen from the table, the community with the maximum F-score is indicated in bold. Also, Table 3.10 shows the values of the GDM metric for the same communities as in Table 3.9. As in Table 3.9, in Table 3.10, the best community among the three is bold. It can easily be seen from the two tables that the best communities according to F-score are identical to the best communities from the perspective of GDM.

The same experiment has been done on the Dolphins network and the values of F-score and GDM are shown in Tables 3.11 and 3.12. In these tables, 10 nodes are chosen randomly as starting nodes and in all cases, the judgments of GDM are the same as F-score. Moreover, Figures 3.12 and 3.13 are the bar charts that show the values of F-score and GDM regarding the same experiment on the American FC network. According to these figures, in 6 out of 7 cases, the judgments of GDM are the same as F-score. It can also be seen from the figures that in some cases, e.g. starting node = 35, Alg_L does not provide any community. As it was mentioned before, algorithm Alg_L

Table 3.11: The values of F-score for some communities detected by Alg_R, Alg_M, and Alg_L regarding some random starting nodes on Dolphins network

Starting node	Alg_R	Alg_M	Alg_L	Starting node	Alg_R	Alg_M	Alg_L
27	**0.69**	0.26	0.26	26	**0.68**	0.26	0.26
3	**0.44**	0.35	0.17	21	0.44	**0.98**	0.47
6	0.69	**0.80**	0	38	0.44	**1.0**	0.50
28	**0.69**	0.53	0.26	47	0.44	**0.46**	0.09
19	0.44	**0.47**	0.28	16	0.44	**0.47**	0.41

Table 3.12: The values of GDM for some communities detected by Alg_R, Alg_M, and Alg_L regarding some random starting nodes on Dolphins network

Starting node	Alg_R	Alg_M	Alg_L	Starting node	Alg_R	Alg_M	Alg_L
27	**0.77**	0.26	0.26	26	**0.77**	0.26	0.26
3	**0.428**	0.425	0.09	21	0.96	**7.10**	1.37
6	1.41	**1.91**	0	38	0.80	**7.20**	1.27
28	**0.77**	0.70	0.26	47	0.43	**1.64**	0.11
19	1.20	**1.26**	1.16	16	1.12	**1.26**	0.97

includes two phases: discovery and examination. In the examination phase, some irrelevant nodes are removed from the community. In cases, where Alg_L has no detected communities, the starting node is mistakenly removed from the community in this phase.

Table 3.13 shows the overall information regarding the three networks Karate Club, Dolphins, and American FC. In this table, the first column indicates the name of the networks, the second column, noted as N, shows the number of randomly selected starting nodes, and the third column, denoted by S, illustrates the success percentage. Success percentage is the percentage of cases in which GDM and F-score have chosen the same community as the best among the three communities. The more the success percentage is, the better the judgment of GDM is. Moreover, columns 4, 5, and 6 show the percentage of cases in which the best community (among the three detected communities) according to the F-score, is created by Alg_R, Alg_M, or Alg_L, respectively. Similarly, columns 7, 8, and 9 show the percentage of cases in which the best community according to GDM

is created by Alg_R, Alg_M, and Alg_L, respectively. Regarding Table 3.13, in some cases, e.g. American FC, the sum of three percentages is greater than 100, e.g. 32 + 70 + 40 = 142. The reason is that different algorithms with the same starting node might create the same communities. As a result, in some cases, two communities among the three are the same and also the best. Thus, the percentages for both algorithms grow. It can be seen from Table 3.13 that both metrics F-score and GDM choose Alg_M as the best in comparison with the other two algorithms. Also, it is understandable that by 100%, 96.66%, and 70%, the judgments of GDM are the same as that of the F-score for three networks Karate Club, Dolphins, and American FC, respectively.

Table 3.14 shows the same values as Table 3.13 for LFR synthetic network. In LFR, the best

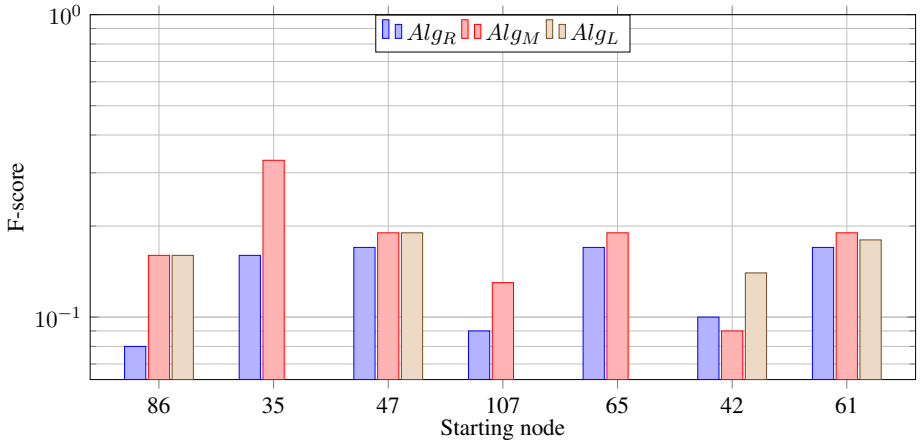

Figure 3.12: The values of F-score for several communities detected by Alg_R, Alg_M, and Alg_L, regarding a number of random starting nodes on American FC network

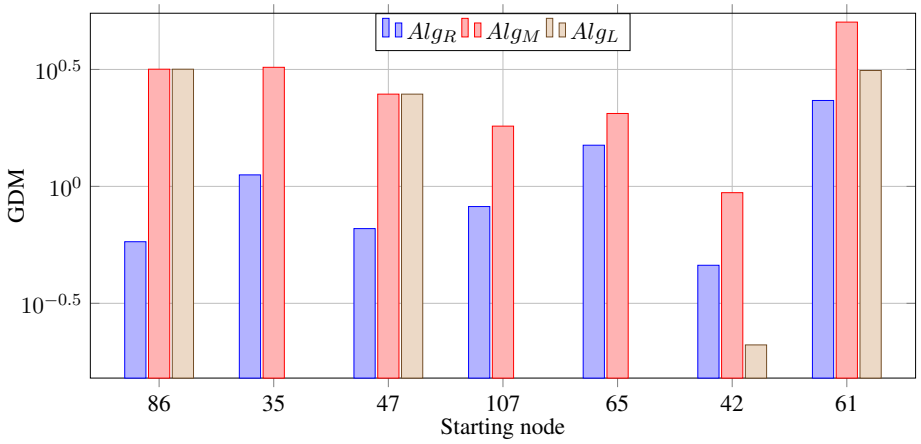

Figure 3.13: The values of GDM for a number of communities detected by Alg_R, Alg_M, and Alg_L, regarding a number of random starting nodes on American FC network

Table 3.13: The results of the comparison of F-score and GDM on Karate club, Dolphins and American FC networks

Dataset	N	S (%)	F-score (%)			GDM (%)		
			Alg_R	Alg_M	Alg_L	Alg_R	Alg_M	Alg_L
Karate club	20	100	35	**65**	0	35	**65**	0
Dolphins	30	96.66	16.66	**83.33**	0	13.33	**86.66**	0
American FC	50	70	32	**70**	40	16	**88**	42

Table 3.14: The results of the comparison of F-score and GDM on LFR synthetic network

Dataset	N	S (%)	F-score (%)			GDM (%)		
			Alg_R	Alg_M	Alg_L	Alg_R	Alg_M	Alg_L
LFR($\mu = 0.1$)	100	100	83	**100**	47	83	**100**	47
LFR($\mu = 0.2$)	100	100	69	**100**	51	69	**100**	51
LFR($\mu = 0.3$)	100	100	63	**100**	44	63	**100**	44
LFR($\mu = 0.4$)	100	98	48	**97**	35	46	**95**	37
LFR($\mu = 0.5$)	100	94	26	**92**	26	23	**95**	24
LFR($\mu = 0.6$)	100	87	31	**77**	11	26	**87**	5
LFR($\mu = 0.7$)	100	70	28	**61**	13	20	**75**	7
LFR($\mu = 0.8$)	100	41	34	**40**	28	**50**	49	1

community partitioning is obtained when $\mu = 0.1$. As the score of μ increases, the quality of ground-truth communities decreases. According to Table 3.14, when $\mu = 0.1, \cdots, 0.7$, the success percentage is over 70% and the best and the second-best algorithms are the same based on F-score and GDM. However, when $\mu = 0.8$, we can barely call the partitioning communities. As a result, the success percentage decreases to 40%.

Furthermore, Tables 3.15 and 3.16 show the same values for DBLP and Amazon networks, respectively. In these tables, the same experiment with different numbers of random starting nodes is conducted and the results are reported. It is shown in Table 3.15 that by over 70%, the GDM has the same judgment as F-score in the DBLP network. Also, according to the values of F-score and GDM, the best and the second-best community detection algorithms are the same. In Table 3.16,

Table 3.15: The results of the comparison of F-score and GDM on DBLP network

Dataset	N	S (%)	F-score (%)			GDM (%)		
			Alg_R	Alg_M	Alg_L	Alg_R	Alg_M	Alg_L
DBLP	200	74.5	28.5	**70**	7.5	13.5	**85.5**	6.5
DBLP	500	74.5	29	**69.6**	6.66	10.59	**89.21**	4.71
DBLP	800	73.5	29.9	**68.8**	6.4	11.7	**88.3**	6.5
DBLP	1000	72.5	31.7	**68**	6.4	12.3	**88.5**	6.2
DBLP	1500	71.63	32.70	**65.67**	6.60	12.60	**87.61**	5.68

Table 3.16: The results of the comparison of F-score and GDM on Amazon network

Dataset	N	S (%)	F-score (%)			GDM (%)		
			Alg_R	Alg_M	Alg_L	Alg_R	Alg_M	Alg_L
Amazon	200	86.5	10.5	**87**	8	9.5	**87**	8
Amazon	500	85.9	11.6	**88.8**	6.6	10	**87.2**	5.6
Amazon	800	84.2	11.1	**87.3**	6.7	10.2	**87.2**	4.8
Amazon	1000	83.1	11.6	**85.6**	7.7	10.3	**86.0**	6.1
Amazon	1500	83.0	11.0	**85.9**	7.6	10.8	**86.4**	5.9

the success percentage is over 83% and the best and the second-best algorithms are chosen the same for the Amazon network as well.

It needs to be mentioned that the three algorithms have their drawbacks. In this regard, the most important deficiency of Alg_R is that this algorithm does not have a termination condition and it must be determined before running. In this experiment, the termination condition of Alg_R for small networks e.g. Karate Club, Dolphins, and American FC, is considered the maximum degree of the network, for LFR and DBLP, is equal to the size of the corresponding ground-truth community and for Amazon network is the average ground-truth communities sizes. As it can be understood from the results, in all cases, Alg_R performs better than Alg_L while worse than Alg_M. Also, the employed information for the three termination conditions is not local. In addition, the most remarkable drawback of Alg_M in comparison with the other two is that this algorithm is more time-consuming. Moreover, regarding Alg_L, as was mentioned before, this algorithm cannot detect any

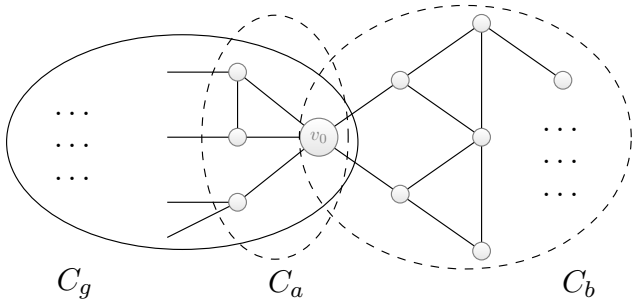

Figure 3.14: An example of two detected communities C_a and C_b, for a given node v_0 versus its ground-truth community C_g

community for some starting nodes. This problem is even more prominent in bigger networks such as DBLP, where it cannot find any communities in more than 50% of cases. As a result, according to the experimental results and the drawbacks of the algorithms, we can conclude that Alg_M can be considered the best among the three algorithms.

3.4.5 Discussion

As it is obvious from the experimental results, in some samples, the community which is chosen as the best by GDM is different from that of the F-score. Figure 3.14 shows a simple example of such cases. In this figure, consider two different algorithms, Alg_a and Alg_b, detected two communities, C_a and C_b, for the same starting node, v_0. The ground-truth community of v_0 is also shown in the figure as C_g. If $Fscore_a$ is the value of F-score for C_a and GDM_a is the score of GDM for C_a, comparing two communities, C_a and C_b, it will be known that $F1score_a > Fscore_b$ while $GDM_a < GDM_b$. The reason is that F-score evaluates the detected communities, C_a and C_b, based on C_g. Since C_b includes a greater number of irrelevant nodes compared to C_a, it is given less F-score value than C_a. However, GDM or any other local quality metrics evaluate the quality of the detected community without considering the starting node. Consequently, since C_b contains more edges and also has better density rather than C_a, it is given greater GDM.

3.5 Conclusion and Future Work

This chapter presents a local evaluation metric to evaluate the quality of communities using the geodesic distance. In this chapter, proposing a new evaluation metric, several local community detection algorithms are compared via evaluation of their detected communities. Also, the GDM is compared with several local metrics and the drawbacks of the existing metrics are discussed. Therefore, based on the results, analyses, and discussions, the contributions of this work could be summarized as follows:

- Considering geodesic distance as an auxiliary element to show the density inside a community, the proposed metric, GDM, comprehensively evaluates the quality of a given community.
- GDM can recognize overlapping nodes of communities.
- GDM is capable of measuring the quality of communities without using any global information.
- GDM is able to handle some important deficiencies of previous well-known metrics including R, M, L, and $Conductance$.
- To evaluate the quality of detected communities, GDM can be used as an alternative instead of comparing them with ground-truth communities.
- GDM can be used to compare different local community detection algorithms.

Considering the advantages of GDM compared to the other metrics and also the experimental results, the following directions are interesting for future works:

- Working on the resolution limit problem using GDM,
- Expanding GDM for directed, weighted, and/or signed networks.

Chapter 4

A New Fast Local Community Detection Algorithm

4.1 Introduction

Social networks are rapidly expanding in the virtual world. The advantages of the virtual world have allowed people to easily communicate with their friends and relatives regardless of geographical distances and temporal constraints. This area, which has attracted the interest of many researchers over the last few years, provides an appropriate description of how individuals communicate with one another. One of the fields that can improve these descriptions is the formation and detection of communities in social networks (Tabarzad & Hamzeh, 2017).

Since the emergence of the problem of community detection, a large number of algorithms have been proposed to detect community partitioning (Blondel et al., 2008; F. Wang et al., 2018; J. Yang et al., 2013). Most studies on community structures in social networks have focused on the classic detection problem which is not local. In the problem of global community detection, the information of the whole network is available in advance. Detection methods based on global perspectives try to detect communities of nodes by exploring the structure information of the entire network. Thus, their running time is positively correlated with the scale of the input complex network. For large-scale complex networks, global community partitioning demands higher performance on computer hardware systems and the efficiency of the algorithm.

However, with the growth of social networks and their corresponding data size, handling the whole structure of the network seems to be impossible. Consequently, local community detection problem has attracted a great deal of attention from researchers in recent years. Several studies (Blondel et al., 2008; Bouyer & Roghani, 2020; Ding, Zhang, & Yang, 2020; Li, Tang, Tang, Zhao, & Huang, 2018; Z. Tang et al., 2021) showed that community detection methods based on local perspectives are easier to detect high-quality communities in complex networks. The local community detection methods can easily solve the problems which block global clustering methods. This problem includes exploring a community for a given node or a group of nodes only using local information. This approach does not need the information of the entire network or any prior knowledge of the network. Therefore, in large-scale complex networks, community detection methods based on local optimization have certain advantages in speed and scalability. In other words, the formation of a community is only based on the local information of the network instead of the entire network.

In this thesis, the main goal of a local community detection problem is to find a high-quality community for a given node v_0. The detection process starts from node v_0 and at each step, one or more nodes from the network (neighboring nodes) are merged into the community. Figure 4.1 shows a sample community with 4 nodes detected for the given node v_0. In the next step, the community can be extended only by using the 6 neighboring nodes.

In this chapter, a new fast local community detection algorithm is presented. This algorithm tries to detect communities by optimizing a new metric, called P. Also, the proposed metric, P, evaluates communities employing the number of common neighbors. The proposed algorithm contains

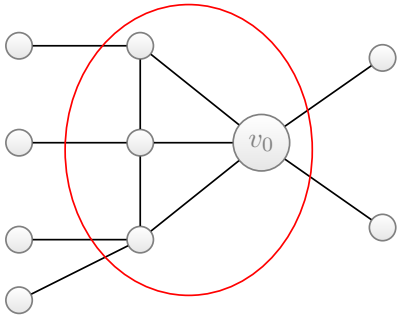

Figure 4.1: The local community detection problem

three steps to add relevant nodes into and also, remove irrelevant nodes from the detected community. The main goal of the proposed algorithm is to find high-quality communities regardless of the importance of the starting nodes in the network, as fast as possible. Experimental results show that the proposed algorithm outperforms state-of-the-art local community detection algorithms. Furthermore, the proposed algorithm is considerably faster than other compared algorithms.

The remainder of this chapter is structured as follows: Section 4.2 represents the literature review and some local community detection algorithms. Section 4.3 proposes the algorithm, while Section 4.4 discusses the experimental results. Finally, this study is concluded in Section 4.5 and also, some ideas for possible future works are presented.

4.2 Related Works

During the last few years, several algorithms have been proposed to locally detect communities in social networks (J. Chen et al., 2009; Q. Chen et al., 2013; F. Luo et al., 2006; W. Luo et al., 2018; T. Zhang & Wu, 2012). Generally, local community detection algorithms detect local communities by optimizing a local metric. Some algorithms, including (J. Chen et al., 2009; Clauset, 2005; F. Luo et al., 2006), cannot capture the density inside the community good enough. These algorithms try to evaluate the quality of communities by only using the number of internal and external edges. As it is concluded in Chapter 2, using the number of internal and external edges alone, cannot give fair scores for the density inside the community. Furthermore, some algorithms are dependent on the starting point of the algorithm (Q. Chen et al., 2013; Su, Wang, & Zhang, 2017; Whang et al., 2013; T. Zhang & Wu, 2012). In this regard, firstly, they search for nodes with specific features, i.e. nodes with high degrees, and then apply the local community detection algorithm to such nodes. It can be concluded that the quality of the detected communities is significantly dependent on the importance of the starting nodes. Also, some algorithms which use more information to capture the density inside the community can still be improved (Q. Chen et al., 2013; W. Luo et al., 2018; T. Zhang & Wu, 2012).

In this section, several local community detection algorithms are presented. Concerning the local community detection problem, the three important local community detection algorithms Alg_R

(Clauset, 2005), Alg_M (F. Luo et al., 2006), and Alg_L (J. Chen et al., 2009) are discussed in Chapter 2 and Section 3.2.

T. Zhang and Wu (2012) proposed a new method by finding core nodes of the community and then expanding the core node's clique to detect the local community for a given node. In this algorithm, the core node is detected based on the nodes' degrees and the number of common neighbors. Also, Q. Chen et al. (2013) propose a new algorithm, called LMD_R, using the same perspective as (T. Zhang & Wu, 2012). In this algorithm, at first, two local degree central nodes for a given node are discovered. Then, starting from those central nodes and using the R metric, the local community is detected. The main limitation of these algorithms is that in some cases, the algorithm needs to walk through all nodes in the network to find the core nodes. For more illustration, the complexity to find the local degree central nodes in (Q. Chen et al., 2013) is $O(nd)$, where n is the number of nodes and d is the average degree of nodes in the community. In the worst case, n is the number of nodes and d is the average degree of nodes in the network.

Furthermore, W. Luo et al. (2018) proposed two algorithms, DMF_M and DMF_R, which have three different stages. In the initial stage of DMF_M, a number of nodes are added to the community while maximizing metric M (refer to (4)) and a new metric, called μ_{M1_tot}. $\mu_{M1\_tot}(v_i)$ for node v_i is defined as follows:

$$\mu_{M1}(v_i) = \begin{cases} \max_{v_j \in NC} \frac{|N(v_i) \cap N(v_j)|+1}{|N(v_j)|} & \Delta M \geq 0 \\ 0 & \Delta M < 0 \end{cases}$$
$$\mu_{M1\_sl}(v_i) = \begin{cases} \max_{v_j \in NC} \frac{|N\_sl(v_i) \cap N\_sl(v_j)|+1}{|N\_sl(v_j)|} & \Delta M \geq 0 \\ 0 & \Delta M < 0 \end{cases} \tag{20}$$

In (20), $N(v_i)$ is the set of neighbor nodes of v_i, and NC is the set of neighbors of v_i that are inside community C. Also, $N\_sl(v_i)$ is the set of second-layer neighbors of v_i. ΔM can be calculated by (4). In the middle stage, more nodes are added to the community while maximizing M. Furthermore, in the closing stage, optimizing a new metric, called μ_{M3}, final nodes are inserted into the

community. $\mu_{M3}(v_i)$ for node v_i is defined as follows:

$$\begin{aligned}\mu_{M3'}(v_i) &= \max_{v_j \in NC} \frac{|N(v_i) \cap N(v_j)| + 1}{|N(v_i)|} \\ \mu_{M3''}(v_i) &= \frac{|\{n | n \in N(v_i), n \in C\}|}{|N(v_i)|} \\ \mu_{M3}(v_i) &= \max\{\mu_{M3'}(v_i), \mu_{M3''}(v_i)\}\end{aligned} \tag{21}$$

DMF_R follows the same pattern as DMF_M, only using metric R (refer to (3)) instead of M. In these algorithms, the number of common neighbors between nodes is the key parameter to evaluate communities. As a result, DMF_M and DMF_R try to evaluate and incrementally explore the communities using the number of common neighbors and metrics M and R. It is also reported that the complexities of these algorithms are $O(n^2 d^2 \log d)$, where n and d are the number of nodes and the average degree of nodes in the community, respectively. It is noteworthy that the major limitation of these algorithms is that they are quite time-consuming.

In this chapter, a fast local community detection algorithm using the number of common neighbors is presented. The algorithms Alg_R (Clauset, 2005), Alg_M (F. Luo et al., 2006), LMD_R (Q. Chen et al., 2013), DMF_R, and DMF_M (W. Luo et al., 2018) are implemented and compared with the proposed algorithm.

4.3 The Proposed Algorithm

In this section, a local community detection algorithm is presented. This algorithm detects a community for a given node using local information while optimizing a local metric. In this regard, a new local quality function is presented to evaluate communities. This local metric tries to capture the quality of communities using the number of common neighbors between every pair of nodes in the community. Following GDM (refer to (14)), it is shown that the geodesic distance can perfectly capture and evaluate the density inside the communities locally. Due to the high complexity of the calculation of the shortest-path lengths, it is not efficient to use it in a metric that is optimized in an algorithm. In this regard, the number of common neighbors is used to examine the density of communities. Common neighbors are the key factors for a group of nodes (graph or community) to

1: **while** more nodes can be added to the community **do**
2: **step 1**: from neighboring nodes, add nodes to the community that increase P.
3: **step 2**: from nodes that are added in Step 1, remove those whose removal increase P.
4: **end while**
5: **while** more nodes can be removed **do**
6: **step 3**: from nodes in the community (i.e. except for the starting node), remove those that their removal increase metric M.
7: **end while**

Figure 4.2: The higher illustration of the proposed algorithm

have lower scores of geodesic distance. Using the number of common neighbors, it is expected that the proposed metric captures the density inside the community in the same pattern as the geodesic distance while demanding less calculation complexity.

The proposed metric is defined as follows:

$$P = \frac{NCN_C + E_{in}}{NCN_C + E_{in} + E_{ex}} \tag{22}$$

In (22), NCN_C is the sum of the number of common neighbors that are inside the community C, between every pair of nodes. It is noteworthy that only the common neighbors that are inside the community C are taken into consideration. Also, E_{in} is the number of edges in the community, and E_{ex} is the number of crossing edges. It is concluded in Chapter 3 that only using the number of internal and crossing edges is not enough to evaluate communities. As a result, the number of common neighbors (NCN) is used to evaluate the density inside the community. As can be seen from (22), E_{in} is used along with the number of common neighbors in this metric. E_{in} is employed to cover the limitation of the number of common neighbors in small communities. For more details, in a community with two nodes and one edge, $NCN = 0$. As e result, to avoid having $P = 0$, E_{in} is added to the metric. Furthermore, E_{ex} is employed to evaluate how separate the community is from the rest of the network.

The remaining of this section presents the proposed local community detection algorithm. The proposed local community detection algorithm discovers a community for a given node while increasing P and using only local information. Figure 4.2 is a high-level illustration of the proposed

algorithm. As it can be seen from Figure 4.2, the proposed algorithm has three different steps. In the first step, it selects nodes whose addition into the community increases the metric P. In the second phase, the algorithm checks all the nodes that were added in phase 1 and removes the nodes that increase P by their removal. The two above-mentioned steps are repeated until no more nodes can be added to the community. In the first two steps, a significant number of correct nodes are added to the community. Metric P is a looser metric in comparison with other existing local metrics, e.g. R, M, L, $Conductance$, LQ. In this concern, metric P allows more nodes to be added to the community. If the addition of such nodes results in the exploration of more nodes that can help to detect a better community, it is worth adding them. However, some irrelevant nodes along with the correct nodes may be inserted into the community as well. As a result, a tighter metric is needed to remove the extra nodes. Thus, there is a third step in which nodes are removed, if their removal increases metric M (refer to (4)).

The detailed illustration of the proposed algorithm (Alg_P) is presented in Algorithm 2. The input of the algorithm is the network G, and the starting node v_0. Also, C is the discovered local community for the starting node v_0, and N is the set of neighboring nodes of C. According to Algorithm 2, in the initialization phase, the given node (v_0) is added to the community C. Consequently, all its neighbors are inserted into the neighboring set, N. In the first step, all nodes in the neighboring set, $v \in N$, are examined and metric P is calculated if v is added to C. If this movement increases the score of P, node v is added to C and $Q1$, and removed from N. $Q1$ keeps the nodes that are inserted into the community in the current iteration of the algorithm. Then, in the second step, each node v in $Q1$ is checked and metric P is calculated if v is removed from C. If this transition increases the score of P, v is removed from C and $Q1$. Then, the neighboring set N is updated regarding the added nodes in the community and their neighbors that are not in the community. The first two steps are repeated until no more nodes can be inserted into the community. In this regard, the size of the community of the last iteration is kept in C_l and is compared with the current community size. Thus, under two conditions the algorithm terminates. The first condition is that no node is added to the community in the first step and $Q1$ is empty. The second condition is that the same nodes that are added in the step 1, are removed in the step 2. As a result, the algorithm terminates at some point. In the final step, metric M decides to keep or remove nodes.

Algorithm 2 The proposed algorithm Alg_P

Input: G and v_0
Output: C: v_0's local community
$C = \{\}$, $N = \{\}$
$P = 0$
add v_0 into C
add all neighbors of v_0 into N
do
    $C_l = |C|$, $Q1 = \{\}$
    step 1:
    for v in N **do**
        calculate P_v if v is added to C
        if $P_v > P$ **then**
            $P = P_v$
            add v to C, add v to $Q1$, remove v from N
        end if
    end for
    step 2:
    for v in $Q1$ **do**
        calculate P_v if v is removed from C
        if $P_v > P$ **then**
            $P = P_v$
            remove v from C, remove v from $Q1$
        end if
    end for
    update N
while $|C| > C_l$
do
    $Q2 = \{\}$
    step 3:
    for v in C **do**
        calculate ΔM if v is removed from C
        if $\Delta M > 0$ and v is not v_0 **then**
            remove v from C, Add v into $Q2$
        end if
    end for
while Q2 is not empty

In this regard, each node v in C is examined and the value of M is computed if v is removed from the community. If this transition increases the score of M and $v \neq v_0$, then v is removed from C and added to $Q2$. $Q2$ keeps nodes that are removed in the current iteration of the last step of the algorithm. The third step is repeated until no node can be removed from the community. Once the algorithm stops, the community C is explored for the given node v_0.

The proposed algorithm, Alg_P, tries to find high-quality communities for the given nodes without considering the importance of the node in the network. Normally, having nodes with higher degrees as starting nodes, results in the detection of high-quality communities. However, walking through the network to find such nodes for the algorithms is time-consuming and in some cases costs $O(n)$ where n is the number of nodes in the network. As a result, having a fast local community detection algorithm that is capable of exploring communities regardless of the starting point of the algorithm is required and also efficient. Moreover, it might be assumed that the calculation of the number of common neighbors at each iteration of the algorithm in comparison with other local metrics including R, M, and L, is time-consuming and complex. In this regard, the complexity of the algorithm, Alg_P, and the metric, P, are analyzed and described in the following section.

4.3.1 Complexity

In this section, the complexity analysis of the proposed algorithm Alg_P and the metric P are analyzed and presented. One of the limitations of using the number of common neighbors NCN to evaluate and explore communities is that the computation of NCN is time-consuming and complex. To explore communities, the proposed algorithm Alg_P calculates NCN, at each iteration. To overcome this limitation, the proposed algorithm Alg_P reduces the number of iterations by adding more than one node to the community, at each iteration. Usually, the local community detection algorithms () add one node at each time to the community to explore the community. This method is required for some metrics to have a better judgment of the selection of nodes to be merged into the community. In this regard, Alg_P reduces the number of iterations of the detection process and consequently, the algorithm terminates faster than other near-complexity algorithms.

To analyze the complexity of the proposed algorithm it is assumed that at each iteration, only one node is added to the community. Assuming n as the number of nodes in the local community,

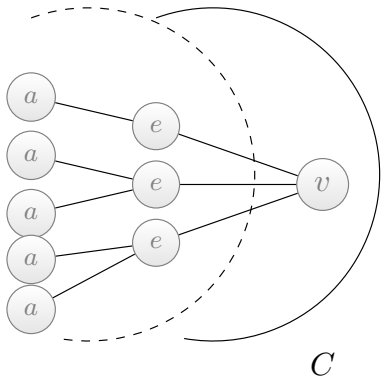

Figure 4.3: How to calculate z while node v is added to the community C

and d as the average degree of nodes inside the community, in the first iteration, one node (the starting node) is in the community. Thus, d neighboring nodes must be traversed. Similarly, in the last iteration, n nodes are added and nd neighboring nodes must be traversed. Consequently, it is required to traverse $d + 2d + \cdots + nd$ nodes. As a result, considering p as the required time complexity to calculate metric P for each node, we have $\sum_{i=1}^{n}(i)dp = dp[n(n+1)/2]$.

Regarding the calculation of P, consider node v is added into the community C. The updated value of P, P', after addition of node v, is calculated using formula 23 as follows:

$$P' = \frac{(NCN_C + z) + (E_{in} + x)}{(NCN_C + z) + (E_{in} + x) + (E_{ex} + y - x)} \tag{23}$$

On the other hand, assuming node v is removed from the community C, the updated score of P, P', is calculated using formula 24 as follows:

$$P' = \frac{(NCN_C - z) + (E_{in} - x)}{(NCN_C - z) + (E_{in} - x) + (E_{ex} - y + x)} \tag{24}$$

In (23) and (24), x is the number of neighbors of node v that are inside the community, and y is the number of v's neighbors that are not inside the community. Moreover, z is the number of common neighbors that node v adds into/subtracts from the community by its addition/removal. According to these equations, for each node at each iteration, it is required to calculate x, y, and z. Since $x + y = d$, the complexity of the calculation of x and y is $O(d)$. Moreover, Algorithm 3 shows how to calculate z, when a new node v is added into the community C, in graph G. In this regard, two

Algorithm 3 count-added-common-neighbors(G, C, v)

1: **Input**: C and v
2: **Output**: z: the number of added common neighbors
3: $N_v =$ Neighbors of v inside C
4: $z = 0$
5: **for** i in N_v **do**
6: $\quad N_i =$ All neighbors of i inside C except v
7: $\quad z = z + |N_i|$
8: **end for**
9: $z = z + |N_v| * (|N_v| - 1)/2$

groups of nodes are counted to compute z. Figure 4.3 shows the two groups of nodes while adding node v to the community C. The first group of nodes includes those that are the neighbors of the neighbors of node v. These nodes are shown with a tags. Regarding Algorithm 3, the first group of nodes (a nodes in Figure 4.3) is calculated in lines 5-7. Moreover, node v is a common neighbor to its instant neighbors inside the community. According Algorithm 3, the number of times that v is a common neighbors itself is computed via $|N_v| * (|N_v| - 1)/2$, where $|N_v|$ is the number of instant neighbors of v that are inside the community (e nodes in Figure 4.3).

Considering Algorithm 3, the complexity of the calculation of z is $O(d^2)$, where d is the average degree of nodes inside the community C. As a result, considering $p \approx O(d^2)$ and $\sum_{i=1}^{n}(i)dp = dp[n(n+1)/2]$, the complexity of the proposed algorithm Alg_P is $O(n^2d^3)$. It should be mentioned that in social networks, the average degree of nodes is not big and does not raise any concern in the complexity of the algorithms. The number of nodes in the community n is the main parameter for the complexity of such algorithms. Traversing the neighboring node is the step that takes $O(n^2)$ and all such algorithms include this step. To the best of our knowledge all local community detection algorithms such as Alg_R, Alg_M, LMD_R, DMF_R, and DMF_M, have the complexity of at least n^2. Although algorithms using the same pattern as LMD_R can have ever bigger complexity.

4.4 Experimental Results

In this section, a number of state-of-the-art local community detection algorithms including Alg_M (F. Luo et al., 2006), Alg_R (Clauset, 2005), LMD_R (Q. Chen et al., 2013), DMF_M, and

Table 4.1: Average F-score (ratio) on real-world networks

Dataset	Evaluation	Alg_M	Alg_R	LMD_R	DMF_M	DMF_R	Alg_P
Karate Club	*Recall*	0.7353	0.5527	0.6556	0.9226	0.9226	**0.9781**
	Precision	0.8784	0.9088	**0.9228**	0.6980	0.7035	0.8172
	F-score	0.7570	0.6474	0.7369	0.7638	0.7669	**0.8757**
Dolphins	*Recall*	0.4362	0.2955	0.4677	0.6515	0.6386	**0.8120**
	Precision	0.9180	0.9528	0.9797	0.9892	**0.9898**	0.9422
	F-score	0.5301	0.4204	0.6162	0.7428	0.7324	**0.8326**
Polbook	*Recall*	0.5741	0.4357	0.7065	0.7319	0.7249	**0.7583**
	Precision	0.7454	0.7785	0.7787	0.7821	**0.7876**	0.7777
	F-score	0.5837	0.4971	0.7184	0.7291	0.7290	**0.7350**
American FC	*Recall*	0.9066	0.7343	0.8468	0.8954	0.8954	**0.9045**
	Precision	0.8428	0.6655	0.7745	**0.8895**	**0.8895**	0.7405
	F-score	0.8604	0.6907	0.8045	**0.8893**	**0.8893**	0.7864
Average	*Recall*	0.6630	0.5045	0.6691	0.8003	0.7954	**0.8632**
	Precision	0.8461	0.8264	**0.8639**	0.8397	0.8426	0.8194
	F-score	0.6828	0.5639	0.719	0.7812	0.7794	**0.8074**

DMF_R (W. Luo et al., 2018) are implemented and compared with the proposed algorithm Alg_P. In order to conduct the experiment, Karate Club (Zachary, 1977), Dolphins (Lusseau et al., 2003), US Political Book (Polbook) (Krebs, 2004), American FC (Girvan & Newman, 2002), Amazon, and DBLP are employed. All the above-mentioned datasets are defined in Section 3.4.4 except for Polbook. Polbook is described as follows:

- **US political book (Polbook)** (Krebs, 2004) is a co-purchasing network. Each node in the dataset represents a book about US politics. An edge between two books indicates that they are often purchased together by customers. This network has 105 nodes, 441 edges, and three ground-truth communities.

In this experiment, every single node is chosen as a starting node, and the results are reported in average scores. The detected communities are evaluated using $Recall$ (refer to 18), $Precision$ (refer to 19), and F-score (refer to Equations (18)(19)(17)) and GDM (refer to Equation (14)).

Table 4.1 shows the resulted average $Recall$, $Precision$, and F-score for all algorithms on the four small datasets including Karate Club, Dolphins, Polbooks, and American FC. It should be

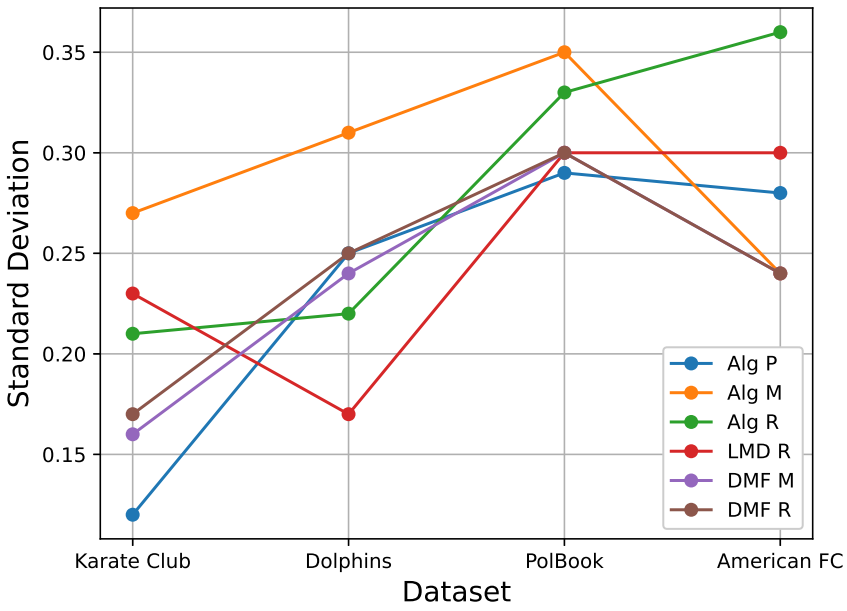

Figure 4.4: Standard deviation on the values of F-score obtained by the algorithms on the four small real-world networks

Table 4.2: Average F-score (ratio) on real-world networks

Dataset	N	Evaluation	Alg_M	Alg_R	LMD_R	DMF_M	DMF_R	Alg_P
Amazon	500	*Recall*	0.8989	0.8276	0.8721	0.9250	0.9242	**0.9275**
		Precision	0.9289	**0.9335**	0.8589	0.9283	0.9331	0.9212
		F-score	0.8906	0.8497	0.8204	0.9092	0.9091	**0.9095**
DBLP	500	*Recall*	0.7209	0.5896	—	—	0.6519	**0.7516**
		Precision	0.6025	**0.6232**	—	—	0.4383	0.5992
		F-score	0.5998	0.5610	—	—	0.4988	**0.6083**

mentioned that in all tables, the bold numbers indicate the best scores in comparison to the other scores in the same row. According to Table 4.1, the proposed algorithm, Alg_P, has the highest F-score on Karate Club, Dolphins, and Polbook networks. Also, the highest average $Recall$ goes to the proposed algorithm for all four datasets. However, the best average $Precision$ is distributed among LMD_R, DMF_M, and DMF_R algorithms. Moreover, according to the results on the American FC network, it turns out that the best score for F-score goes to the two algorithms DMF_M and DMF_R. Although Alg_P does not provide the best scores of $Precision$, values of $Precision$ provided by the

Table 4.3: Average GDM on real-world networks

Dataset	Alg_M	Alg_R	LMD_R	DMF_M	DMF_R	Alg_P
Karate Club	2.0942	1.2617	2.0762	1.3148	1.3171	**3.4062**
Dolphins	1.9068	0.8330	1.4685	2.3116	2.2632	**4.7298**
Polbook	5.8061	3.7481	7.8627	7.9170	7.8380	**8.1934**
American FC	6.2799	3.7796	1.1399	**6.9127**	**6.9127**	6.0355
Amazon	4.522	4.0353	4.2205	4.7485	4.7818	**4.7875**
DBLP	4.0930	4.0295	—	—	3.9324	**4.3322**

Table 4.4: Execution time (s)

Dataset	Alg_M	Alg_R	LMD_R	DMF_M	DMF_R	Alg_P
Karate Club	0.53	0.36	1.25	1.95	3.28	**0.50**
Dolphins	2.24	1.08	5.02	3.75	6.97	**2.11**
Polbook	15.22	13.21	53.71	53.05	84.24	**10.21**
American FC	3.23	4.99	8.27	5.60	9.28	**3.71**
Amazon	2712.87	892.42	153047.05	8241.88	5934.81	**1096.07**
DBLP	10113.68	3780.23	—	—	123853.73	**24366.76**

proposed algorithm are big enough to result in the best values of the F-score. It should be mentioned that a community detection algorithm must be able to improve F-score for all networks as much as possible at the same time. As a result, the last three rows are added to Table 4.1 which indicate the average of the obtained $Recall$, $Precision$, and F-score on the four networks. According to the last three rows of Table 4.1, Alg_P has the best average $Recall$ and F-score. Figure 4.4 shows the standard deviation of the values of F-score obtained by the algorithms on the four small networks.

Also, Table 4.2 shows the same results on the Amazon and DBLP networks with 500 starting nodes that are selected randomly. In this table, N is the number of starting nodes. It should be noted that the 500 starting nodes are chosen randomly from 500 distinct communities. Regarding this table, the proposed algorithm has the best $Recall$ and F-score on both networks, Amazon and DBLP. Also, the best scores of Precision go to Alg_R for Amazon and DBLP. As it can be seen from Table 4.2, LMD_M and DMF_M do not give any result for DBLP in a reasonable amount of time.

Moreover, Table 4.3 illustrates the average GDM for the resulted communities from the six algorithms on the same datasets. According to this table, the proposed algorithm has the highest scores on Karate Club, Dolphins, and Polbook networks. Also, the judgment of GDM is the same as that of the F-score, and DMF_M and DMF_R are given the highest scores on the American FC network. Moreover, the proposed algorithm, Alg_P, is given the best average GDM on the Amazon and DBLP networks.

Furthermore, Table 4.4 shows the execution time of the algorithms for the same experiment. All experiments in this thesis is done using python programming language and on a computer with Intel(R) Core i7 processor and 16GB RAM. According to this table, comparing the execution time of LMD_R, DMF_R, DMF_M, and Alg_P, which obtained comparable results, the execution time of the proposed algorithm is faster on all datasets. It can be seen that however, the complexity of the Alg_P is more than the reported complexities for DMF_M and DMF_R, the execution time of the proposed algorithm is faster. It is noteworthy that in the Amazon network, Alg_P is more than twice faster than Alg_M, which is known to be a low-complexity algorithm. Since at each iteration of the proposed algorithm more than one node is added into the community, the total number of iterations decreases, and consequently, the algorithm is terminated faster.

4.5 Conclusion and Future Works

In this chapter, a new fast local algorithm to detect communities is proposed using the number of common neighbors. In order to develop the proposed algorithm, a new metric, P, is proposed. The proposed algorithm Alg_P locally detects communities while optimizing the proposed metric P. Moreover, several state-of-the-art algorithms are employed to be compared with the proposed one. According to the experimental results, it can be concluded that Alg_P outperforms the state-of-the-art algorithms. For more illustration, the detected communities by Alg_P have the best average scores of $Recall$ on all datasets and also, the best average scores of F-score on all datasets, except for the American FC network. Furthermore, since at each iteration more than one node may be added to the community, the execution time of the proposed algorithm is much faster than the other algorithms with comparable results. In this regard, a fast and efficient community detection

algorithm is presented to detect a community for a given node without considering the importance of the starting nodes in the network.

Considering the performance of the proposed algorithm, the following directions can be considered for future works:

- Employing networks without ground-truth data to evaluate the proposed algorithm using GDM.
- Using Alg_P in a community detection algorithm to detect the whole community partitioning of networks.
- Employing the proposed algorithm in an evolutionary algorithm to detect local communities in dynamic networks.

Chapter 5

A Dynamic Local Community Detection Algorithm

5.1 Introduction

In terms of temporal changes, social networks can be divided into two different categories: static and dynamic. The structure of static networks, unlike the structure of dynamic ones, does not change over time. However, dynamic networks allow changes in the relations among nodes at different time steps. Real-world social networks are dynamic and change over time. Co-authorship between scholars and email interactions between employees in an organization are two examples. In dynamic networks, time plays a crucial role in shaping network topologies. As a consequence, the task of describing these time-evolving networks is extremely important. Traditionally, a network is represented as a graph with nodes and their in-between links. But this definition of network needs some modification to incorporate the other important dimension, time. The temporal dimension facilitates improvised understanding of the network by embedding valuable information to it (Mishra, Singh, Mishra, & Biswas, 2021). To model dynamic social networks, two different approaches are developed as follows:

- **Snapshots network**: In this model, network history is partitioned into a series of snapshots, each one of them showing the state of the network at a time.

Definition 5.1.1. (Snapshot Network) A snapshot graph G_τ is defined by an ordered set $\langle G_1, G_2 ... G_t \rangle$ where at each snapshot $1 < i < t$, $G_i = (V_i, E_i)$ is the status of the network which is denoted by the sets of nodes V_i and edges E_i (Rossetti & Cazabet, 2018).

- **Temporal network**: A temporal network models a dynamic structure in which both nodes and edges may appear and disappear as time goes by. More formally, temporal networks can be defined as follows:

  Definition 5.1.2. (Temporal Network) A temporal network is a graph $G = (V, E, T)$, where V is a set of triplets (v, ts, te), with v is a vertex of the graph and $t_s, t_e \in T$ respectively being the birth and death time of the corresponding vertex. Also, E is a set of quadruplets (u, v, t_s, t_e), where $u, v \in V$ being vertices of the graph, and $t_s, t_e \in T$ respectively being the birth and death timestamps of the corresponding edge (Rossetti & Cazabet, 2018).

By modeling dynamic social networks, it is possible to analyze the network's structure over time, explore how the network evolves, and finally anticipate the future topology of the network. Dynamic community detection includes finding a series of similar communities in different snapshots. A dynamic community is represented by its constituent communities ordered by time snapshots. Given a dynamic network G_d, a dynamic community C_d is represented by a series of community partitioning denoted by $C_d = \{C_{t_0}, C_{t_1}, \cdots, C_{t_\tau}\}$, where $t_0 < t_1 < \cdots < t_\tau$ and C_{t_i} represents the corresponding community partitioning at time t_i. If k communities are detected at time t_i, then $C_{t_i} = \{C^1_{t_i}, C^2_{t_i}, \cdots, C^k_{t_i}\}$, where $C^j_{t_i}$ is the j-th community detected at time t_i (Dakiche et al., 2019).

Dynamic community detection is a complicated problem because of rapid and unpredictable changes in social networks (Z. Wang et al., 2018). A large number of studies have been conducted to address the problem of community detection in dynamic networks (J. He & Chen, 2015; Rossetti et al., 2017; Zhuang et al., 2019). However, the local community detection problem is not widely investigated in dynamic networks. In this chapter, the problem of local community detection in snapshot dynamic networks is addressed. First, a simple dynamic structure is introduced which employs any local community detection algorithm. The goal of this dynamic structure is to analyze

different local community detection algorithms and evaluate their strengths in detecting communities in dynamic networks. To the best of our knowledge, no such analysis has been conducted before. The reported results help to analyze the weaknesses and strengths of the existing algorithms and consequently develop an efficient algorithm.

Secondly, a dynamic local community detection algorithm, called $DevDynaP$ is proposed to overcome the existing drawbacks. The main feature of the proposed algorithm is that it starts from a given node, explores the network, and detects communities simultaneously at each snapshot. The main goal of the proposed algorithm is to explore the network as fast as possible and detect communities at the same time to identify the whole community partitioning of the network in the upcoming snapshots.

The remainder of this chapter is structured as follows: Section 5.2 presents the literature review. Section 5.3 presents the dynamic structures, while Section 5.4 discusses the experimental results. Finally, this chapter and the proposed dynamic algorithm are concluded in Section 5.5.

5.2 Related Works

During the last few years, a large number of studies have been conducted to address the problem of community detection in dynamic networks (J. He & Chen, 2015; Rossetti et al., 2017; Zhuang et al., 2019). In this section, some dynamic community detection algorithms are presented.

The linear time complexity of label propagation algorithms (LPA) (Garza & Schaeffer, 2019; Sattari & Zamanifar, 2018; X.-K. Zhang, Ren, Song, Jia, & Zhang, 2017) proves its efficiency in large and dynamic networks. Xie, Chen, and Szymanski (2013) proposed a deterministic variation of the label propagation algorithm, called LabelRankT. Also, Boudebza, Cazabet, Azouaou, and Nouali (2018) proposed an online clique percolation method (OLCPM) which amalgamates the label propagation technique with the clique percolation algorithm.

Also, J. He and Chen (2015) proposed a simple dynamic community detection algorithm using Louvain algorithm (Blondel et al., 2008). In their method, the Louvain algorithm is employed to detect the community partitioning of the whole network in the first snapshot. Then, starting the second

snapshot, a network is generated using the information of the network and the community partitioning of the previous snapshot. Next, the Louvain algorithm is executed to detect the community partitioning of the current snapshot.

Moreover, Rossetti et al. (2017) proposed Tiles algorithm to detect overlapping communities and track their evolution in time using an online iterative structure. In this algorithm, the memberships of nodes to communities are recalculated whenever a change occurs in the network. In other words, if one single change happens in the network, Tiles runs a label propagation structure to diffuse the changes to recalculate the neighbors' community memberships.

Furthermore, Zhuang et al. (2019) proposed an incremental algorithm to maximize the modularity scores while updating the community structure of dynamic networks. In this regard, six different types of change in the network are introduced, and a series of actions are taken for each. Also, the Louvain algorithm (Blondel et al., 2008) is the employed community detection algorithm. Also, Mishra et al. (2021) presented a tree-based community detection algorithm (TCD2) in dynamic social networks which exploits two important properties of social networks, connectedness, and influence, for finding communities in the network. TCD2 uses a tree structure to maintain the information on dynamically changing community structures of the network.

It is noteworthy that mostly the presented algorithms need the whole community partitioning of the network at the first snapshot. In other words, however, the presented algorithms use a local perspective to fasten the execution time, they are not local algorithms. Because they need global information of the network, at some point. Also, proposed algorithms by (Rossetti et al., 2017), (Zhuang et al., 2019), and (Mishra et al., 2021) are efficient in temporal networks. These algorithms are triggered by every single change in the network.

Takaffoli et al. (2013) proposed a local dynamic community detection algorithm called Incremental L ($IncL$) which only uses local information. This algorithm detects communities by capturing connected components according to the explored communities at the previous snapshot, and the structure of the network at the current snapshot. Then, new communities are detected at the current snapshot by applying algorithm Alg_L (J. Chen et al., 2009) on each captured connected component. This algorithm explores as many communities at each snapshot based on the detected communities at previous snapshots. Algorithm 4 shows the $IncL$ algorithm. In this algorithm, G_i is the network

Algorithm 4 $IncL$ Algorithm

Input: $G_d = \{G_0, G_2, \cdots, C_n\}$ and v_0
Output: $C = \{C_0, C_1, \cdots, C_n\}$
C_0 = $Alg_L(G_0, v_0)$
$C = C \cup C_0$
for each snapshot $i = 1, 2, ..., n$ **do**
    $CP_i = C_{i-1}$ at snapshot i
    for each connected component C_c in CP_i **do**
        $C_i \cup Alg_L(G_i, C_c)$
    end for
    $C = C \cup C_i$
end for
Return C

and C_i is the detected community partitioning at snapshot i. To the best of our knowledge, $IncL$ is the only local dynamic algorithm that starts from a single node and detects the community partitioning at the same time as exploring the network incrementally. One of the main drawbacks of $IncL$ is that the speed to explore the network is low. Another drawback of this algorithm is that it keeps detecting repetitive communities.

In this chapter, algorithms Alg_R (Clauset, 2005), Alg_M (F. Luo et al., 2006), LMD_R (W. Luo et al., 2018), LMD_M (W. Luo et al., 2018) and Alg_P (refer to Algorithm 2) are implemented in a simple local dynamic structure to detect communities in dynamic networks. The results are compared to evaluate the flexibility of different local algorithms being used in a dynamic structure. Moreover, a dynamic local community detection algorithm, called $DevDynaP$, is developed and proposed, regarding the drawbacks of the existing algorithms. The main focus of the proposed algorithm ($DevDynaP$) is to explore the nodes of the network as fast as possible and detect the community partitioning at the same time to identify the whole community partitioning of the network in the upcoming snapshots. It is noteworthy that $DevDynaP$ employs Alg_P (refer to Algorithm 2) to explore communities. In this regard, the results of the proposed algorithm are compared with that of $IncL$ (Takaffoli et al., 2013).

5.3 Dynamic Algorithms

This section presents a simple dynamic structure to detect a community for a given node at each snapshot in dynamic networks. In order to explore a community in the current snapshot, this structure improves the detected community in the previous snapshot. This dynamic structure can employ any local community detection algorithm. In this regard, the flexibility of different local community detection algorithms of being used in dynamic networks can be examined. Also, a local community detection algorithm, called $DevDynaP$ is presented. $DevDynaP$ starts from a given node and detects a community in the first snapshot, and by using the detected nodes and communities, tries to capture community partitioning in the next snapshots.

5.3.1 A Simple Dynamic Structure

In this section, a simple dynamic structure is presented. In this dynamic structure, any local community detection algorithm can be used to detect a community for a given node in a dynamic network. It is noteworthy that the main focus of this dynamic structure is to detect a community for a given node at each snapshot according to the current network structure and the detected community at the previous snapshot. Algorithm 5 shows the dynamic structure. In this algorithm, G_i is the network at i-th snapshot and C_i is the detected community for the given node v_0 at snapshot i. Employing any local community detection algorithm, this dynamic structure explores a community at the first snapshot. Starting the second snapshot, the detected community in the previous snapshot is analyzed in the current network structure, C_c. If C_c is connected, the local community detection is applied to it to explore the community in the current snapshot. If C_c is not connected, the connected part of C_c that contains the given starting node v_0, C_v is exploit. Then, the local community detection algorithm is applied on C_v to detect the community for the current snapshot. In this regard, one community for each snapshot for the given node v_0 is detected, $C = \{C_0, C_1, \cdots, C_n\}$.

5.3.2 Developed Dynamic P (DevDynaP)

In this section, the proposed local dynamic algorithm ($DevDynaP$) is presented. The main goal of $DevDynaP$ is to start from a given node at the first snapshot and explore the network as

Algorithm 5 The dynamic structure

Input: $G_d = \{G_0, G_1, \cdots, C_n\}$ and v_0
Output: $C = \{C_0, C_1, \cdots, C_n\}$
$C_0 =$ Local-Detection-Algorithm(G_0, v_0)
$C = C \cup C_0$
for each snapshot $i = 1, 2, ..., n$ **do**
    $C_c = C_{i-1}$ at snapshot i
    if C_c is still connected **then**
        $C_i =$ Local-Detection-Algorithm(G_i, C_c)
    else
        $C_v =$ the connected part of C_c at snapshot i that includes v_0
        $C_i =$ Local-Detection-Algorithm(G_i, C_v)
    end if
    $C = C \cup C_i$
end for
Return C

fast as possible and detect community partitioning at the same time in the next snapshots.

As it was mentioned before, the other local community detection algorithm ($IncL$) (Takaffoli et al., 2013) suffers from some drawbacks in dynamic networks. One of the drawbacks is the low speed of networks' exploration. In this regard, $IncL$ is unable to explore the whole network even after several snapshots. The other drawback is that repetitive communities or hierarchical communities are detected. Detection of repetitive communities results in spending more time detecting a community that has been already detected. $DevDynaP$ covers the drawbacks of the other algorithm in which it explores the network faster and also, avoids detecting repetitive communities. The experiments in Chapter 4 show that Alg_P (refer to Algorithm 4.2) is a fair algorithm to detect communities locally versus other compared algorithms. Therefore, Alg_P is employed in $DevDynaP$ to explore communities locally in the network.

Algorithm 6 shows the proposed local dynamic algorithm ($DevDynaP$). In this algorithm different spanshots of the network $G_d = \{G_0, G_2, \cdots, C_n\}$, and also, a starting node, v_0, are given as inputs of the algorithm. The output of $DevDynaP$ includes a community partitioning for each snapshot $C = \{C_0, C_1, \cdots, C_n\}$. In this regard, G_i is the network at snapshot i, and C_i is the community partitioning detected by $DevDynaP$ at snapshot i, which contains several communities.

Algorithm 6 The proposed dynamic algorithm ($DevDynaP$)

Input: $G_d = \{G_0, G_1, \cdots, C_n\}$ and v_0
Output: $C = \{C_0, C_1, \cdots, C_n\}$
At snapshot 0:
$C_0 = Alg_P\ (G_0, v_0)$
CRN = nodes which are removed from the third step of Alg_P
$C = C \cup C_0$
for each snapshot $i = 1, 2, ..., n$ **do**
    Step 1:
    $RN = CRN$
    $CRN = []$
    $CP = C_{i-1}$ at snapshot i
    for each connected component C_c in CP **do**
        if C_c is not a sub graph of any communities in C_i **then**
            $C_i = C_i \cup Alg_P(G_i, C_c)$
            $CRN = CRN \cup$ nodes which are removed from the third step of $Alg_P(G_i, C_c)$
        end if
    end for
    Step 2:
    for each node v in RN **do**
        for each community com in C_i **do**
            if v is a neighbor of com & $validate(v, com)$ **then**
                $com = com \cup v$
                $RN = RN \backslash v$
            end if
        end for
    end for
    Step 3:
    if RN is not empty **then**
        for each node v in RN **do**
            if v does not exist in any communities of C_i **then**
                $C_i = C_i \cup Alg_P(G_i, v)$
                $CRN = CRN \cup$ nodes which are removed from the third step of $Alg_P(G_i, v)$
            end if
        end for
    end if
    $C = C \cup C_i$
end for
Return C

Algorithm 7 The validation algorithm ($validate(v, com)$)

1: **Input**: community com and v
2: $C_v = com \cup v$
3: **if** $M_{C_v} >= M_{com}$ & $P_{C_v} >= P_{com}$ **then**
4: **return** True
5: **else**
6: **return** False
7: **end if**

As can be seen from Algorithm 6, $DevDynaP$ consists of three different steps. Before the steps start, one community, C_0, for the starting node is detected at snapshot 0 (first snapshot). Then, the removed nodes from the third step of Alg_P are added to the current removed nodes list, CRN. These removed nodes are employed to explore the network as fast as possible. In the first step, and starting snapshot 1, the nodes in CRN, are moved to the removed nodes list, RN. Next, the community partitioning detected at the previous snapshot is analyzed at the current snapshot, CP, and the connected components are extracted. Then, each one of the extracted connected components, C_c, is checked if it is a sub-graph of any detected community at the current snapshot. If not, Alg_P is executed on C_c, $Alg_P(G_i, C_c)$, to explore a community for the current snapshot. This condition is inserted to avoid the detection of repetitive communities. As a result, the number of execution of Alg_P is decreased, which results in reducing the execution time of the whole algorithm. Moreover, the removed nodes in the third step of Alg_P are stored in CRN.

In the second step, some nodes in RN that meet some criteria are joined to their corresponding communities and removed from RN. The criteria for a node, v, is to be a neighbor of the community com and validate by $validate(v, com)$. The purpose of this step is to avoid running Alg_P for nodes that can belong to an existing community. $validate(v, com)$ is presented in Algorithm 7. In this algorithm, M_C and P_C show the scores of metrics M and P for community C.

In the final step, the remaining nodes in RN are considered new starting nodes for the community detection algorithm Alg_P. In this regard, if $v \in RN$ in the third step, Alg_P is executed on v, and a new community is added to C_i. Also, the removed nodes from the third step of Alg_P are added to CRN. This step may increase the execution time of the algorithm due to some extra execution of Alg_P. This is justified by the reduction of the execution time of the algorithm at the first step by

taking out several executions of Alg_P. In other words, some execution of Alg_P which results in the detection of repetitive communities are removed, and instead, some runs of Alg_P which cause fast exploration of the network are inserted. The three steps are repeated for all snapshots.

In this regard, $DevDynaP$ explores the network and detects the whole community partitioning of the network as fast as possible in the upcoming snapshots. To analyze the complexity of the algorithm, the number of times that Alg_P is executed is the key parameter. The number of times that $DevDynaP$ executes algorithm Alg_P is significantly based on the structure of the network and the evolution of the network through snapshots. Considering s as the number of snapshots, n_p as the number of times that $DevDynaP$ executes Alg_P, and α as the complexity of Alg_P, the complexity of $DevDynaP$ is approximately $O(sn_p\alpha\beta)$. β indicated the complexity of line 13 of Algorithm 6 for each connected component C_c.

5.4 Experimental Results

In order to evaluate the strengths of local community detection algorithms in dynamic networks, some local detection algorithms are implemented and used in the dynamic structure (refer to Algorithm 5). The experimental results are presented in Section 5.3.1. Also, $DevDynaP$ is compared with $IncL$ (Takaffoli et al., 2013), and the results are reported in Section 5.3.2. To conduct the experiments, dynamic Lancichinetti-Fortunato-Radicchi (LFR) benchmarks (Lancichinetti et al., 2008) are employed.

5.4.1 Benchmarks

Lancichinetti-Fortunato-Radicchi (LFR) benchmark (Lancichinetti et al., 2008) is a synthetic network generator in which the ground-truth communities are known. To generate a network using LFR, some parameters need to be set. The parameters include the number of nodes n, the average degree of nodes d, the maximum degree of nodes $maxd$, and the mixing parameter $0 < \mu < 1$. Also, Greene, Doyle, and Cunningham (2010) designed the dynamic version of the LFR network generator in which different types of networks are created according to events in dynamic networks. A short description of the most common events includes (Dakiche et al., 2019): (1) Birth: a new

Table 5.1: Average results on LFR switch network

Snapshots	Metrics	Alg_M	Alg_R	DMF_M	DMF_R	Alg_P
1	$Recall$	0.956	0.7801	**1.0**	**1.0**	**1.0**
	$Precision$	0.9301	0.7816	**1.0**	**1.0**	0.9918
	F-score	0.9400	0.7791	**1.0**	**1.0**	0.9950
	GDM	3.5110	3.0237	**3.8250**	**3.8250**	3.7985
2	$Recall$	0.9682	0.8958	**1.0**	**1.0**	0.9861
	$Precision$	0.9541	0.8905	0.9165	0.9165	**0.9825**
	F-score	0.9586	0.8910	0.9470	0.9470	**0.9838**
	GDM	3.8405	3.2381	3.4134	3.4134	**4.0125**
3	$Recall$	0.984	0.9335	**0.9962**	**0.9962**	0.9853
	$Precision$	0.9721	0.9259	0.8961	0.8961	**0.9853**
	F-score	0.9763	0.9292	0.9314	0.9314	**0.9852**
	GDM	3.5523	3.3200	3.2941	3.2941	**3.6381**
4	$Recall$	0.972	0.9263	**0.9963**	**0.9963**	0.9892
	$Precision$	0.9593	0.9155	0.9041	0.9041	**0.9875**
	F-score	0.9638	0.9202	0.9328	0.9328	**0.9881**
	GDM	3.4411	3.2391	3.3035	3.3035	**3.5860**
5	$Recall$	0.98	0.9368	**0.9964**	**0.9964**	0.9891
	$Precision$	0.9698	0.9277	0.9113	0.9113	**0.9878**
	F-score	0.9737	0.9315	0.9410	0.9410	**0.9883**
	GDM	3.3532	3.1140	3.1432	3.1432	**3.4352**

community emerges at a snapshot. (2) Death: a community disappears. All nodes belonging to this community lose their membership. (3) Expand: a community gets some new members. (4) Contraction: a community loses some of its members. (5) Merging: several communities merge to create a new community. (6) Splitting: a community is divided into several communities.

In this regard, four different types of LFR dynamic networks are created in different numbers of snapshots including:

- **Switch**: in this network, the nodes flip memberships between communities at each snapshot ($n = 250$, $d = 10$, $maxd = 20$, $\mu = 0.2$ and $pr = 0.1$). pr is the probability of a node switching community membership between snapshots.

- **Expand/Contraction**: in this network, communities are expanded/contracted at each snapshot ($n = 250$, $d = 10$, $maxd = 20$, $\mu = 0.2$, $expand = 5$, $contract = 5$ and $r = 0.1$).

Table 5.2: Average results on LFR expand/contraction network

Snapshots	Metrics	Alg_M	Alg_R	DMF_M	DMF_R	Alg_P
1	$Recall$	0.956	0.7801	**1.0**	**1.0**	**1.0**
	$Precision$	0.9301	0.7816	**1.0**	**1.0**	0.9918
	F-score	0.9400	0.7791	**1.0**	**1.0**	0.9950
	GDM	3.5110	3.0237	**3.8250**	**3.8250**	3.7985
2	$Recall$	0.944	0.8719	**0.98**	**0.98**	0.9587
	$Precision$	0.9092	0.8606	0.8603	0.8603	**0.9543**
	F-score	0.9218	0.8653	0.9014	0.9014	**0.9559**
	GDM	3.7692	3.4860	3.6124	3.6124	**4.0016**
3	$Recall$	0.952	0.8832	**0.98**	**0.98**	0.9727
	$Precision$	0.9435	0.8812	0.8983	0.8983	**0.9724**
	F-score	0.9468	0.8815	0.9281	0.9281	**0.9725**
	GDM	3.2390	3.0574	3.07915	3.0791	**3.3876**
4	$Recall$	0.964	0.9277	**0.98**	**0.98**	0.9725
	$Precision$	0.9556	0.9237	0.9167	0.9167	**0.9668**
	F-score	0.9587	0.9252	0.9352	0.9352	**0.9687**
	GDM	3.8528	3.7151	3.8165	3.8165	**3.9337**
5	$Recall$	0.96	0.9057	**0.972**	**0.972**	0.9646
	$Precision$	0.9491	0.9041	0.8958	0.8958	**0.9536**
	F-score	0.9534	0.9042	0.9126	0.9126	**0.9569**
	GDM	3.4229	3.2545	3.6608	3.6608	**3.4812**

$expand$ is the number of expansion events per snapshot, $contract$ is the number of contraction events per snapshot and r is rate of $expand/contract$.

- **Merge/Split**: in this network, communities are merged/split at each snapshot ($n = 250$, $d = 10$, $maxd = 20$, $\mu = 0.2$, $merge = 5$ and $split = 5$). $merge$ is the number of merge events per snapshot and $split$ is the number of split events per snapshot.

- **Birth/Death**: in this network, communities are permanently added/removed at each snapshot ($n = 250$, $d = 10$, $maxd = 20$, $\mu = 0.2$, $birth = 3$ and $death = 3$). $birth$ is the number of community birth events per snapshot and $death$ is the number of community death events per snapshot.

Table 5.3: Average results on LFR merge/split network

Snapshots	Metrics	Alg_M	Alg_R	DMF_M	DMF_R	Alg_P
1	*Recall*	0.956	0.7801	**1.0**	**1.0**	**1.0**
	Precision	0.9301	0.7816	**1.0**	**1.0**	0.9918
	F-score	0.9400	0.7791	**1.0**	**1.0**	0.9950
	GDM	3.5110	3.0237	**3.8250**	**3.8250**	3.7985
2	*Recall*	0.952	0.8619	**0.9964**	**0.9964**	0.9928
	Precision	0.9073	0.8553	0.9392	0.9392	**0.9487**
	F-score	0.9225	0.8515	0.9580	0.9580	**0.9638**
	GDM	3.5058	3.3129	3.5409	3.5409	**3.6938**
3	*Recall*	0.964	0.8649	**0.9925**	**0.9925**	0.9851
	Precision	0.9462	0.8691	0.9279	0.9280	**0.9648**
	F-score	0.9533	0.8614	0.9509	0.9510	**0.9720**
	GDM	3.5602	3.2124	3.4373	3.4373	**3.7082**
4	*Recall*	0.9648	0.8643	**0.9928**	**0.9928**	0.9927
	Precision	0.9026	0.8432	0.8955	0.8955	**0.9184**
	F-score	0.9247	0.8376	0.9335	0.9335	**0.9453**
	GDM	3.7741	3.4127	3.6636	3.6637	**3.9750**
5	*Recall*	0.9547	0.8405	**0.9912**	**0.9912**	0.9873
	Precision	0.9237	0.8438	0.8694	0.8694	**0.9424**
	F-score	0.9353	0.8270	0.9169	0.9169	**0.9566**
	GDM	3.6322	2.9666	3.8480	3.8480	**3.9062**

5.4.2 Experimental Results on Section 5.3.1

In this experiment, algorithms Alg_R (Clauset, 2005), Alg_M (F. Luo et al., 2006), LMD_R (W. Luo et al., 2018), LMD_M (W. Luo et al., 2018) and Alg_P (Algorithm 2) are implemented and used in the dynamic structure (refer to Algorithm 5). The dynamic structure detects a community for a given node at each snapshot. In this regard, four different datasets including switch, expand/contraction, merge/split, and birth/death in five snapshots are generated. Every single node of the first snapshot is selected as a starting node for the algorithms, and the results are reported as average values per snapshot. Then, the detected communities at each snapshot are evaluated using F-score (refer to (17)) and GDM (refer to (14)).

Table 5.1 shows the average results of running the dynamic structure in Algorithm 5 by employing the five algorithms Alg_R (Clauset, 2005), Alg_M (F. Luo et al., 2006), LMD_R (W. Luo et

Table 5.4: Average results on LFR birth/death network

Snapshots	Metrics	Alg_M	Alg_R	DMF_M	DMF_R	Alg_P
1	$Recall$	0.956	0.7801	**1.0**	**1.0**	**1.0**
	$Precision$	0.9301	0.7816	**1.0**	**1.0**	0.9918
	F-score	0.9400	0.7791	**1.0**	**1.0**	0.9950
	GDM	3.5110	3.0237	**3.8250**	**3.8250**	3.7985
2	$Recall$	0.86	0.7859	**0.9045**	**0.9045**	0.8800
	$Precision$	0.8230	0.7689	0.7954	0.7954	**0.8622**
	F-score	0.8352	0.7748	0.8328	0.8328	**0.8681**
	GDM	3.2337	2.9136	3.0601	3.0601	**3.4720**
3	$Recall$	0.852	0.7789	**0.88**	**0.88**	0.8692
	$Precision$	0.8289	0.7757	0.7785	0.7785	**0.8577**
	F-score	0.8368	0.7757	0.8120	0.8120	**0.8616**
	GDM	3.0383	2.7556	2.7693	2.7693	**3.1904**
4	$Recall$	0.788	0.7660	**0.8124**	**0.8124**	0.7984
	$Precision$	0.7734	0.7344	0.6614	0.6616	**0.7829**
	F-score	0.7792	0.7457	0.7059	0.7062	**0.7891**
	GDM	3.0555	2.8360	2.8240	2.8228	**3.1385**
5	$Recall$	0.72	0.6726	**0.74**	**0.74**	0.7274
	$Precision$	0.7024	0.6331	0.6358	0.6359	**0.7079**
	F-score	0.7084	0.6470	0.6637	0.6639	**0.7141**
	GDM	2.5455	2.2561	2.4250	2.4180	**2.6110**

al., 2018), LMD_M (W. Luo et al., 2018) and Alg_P (refer to Algorithm 2) on the dynamic LFR switch network. It should be mentioned that in all tables, the bold numbers indicate the best scores in comparison to the other scores in the same row. According to Table 5.1, algorithms DMF_R and DMF_M have the best average results in the first snapshot. Also, the scores of Alg_P are very close to scores of DMF_R and DMF_M in snapshot 1. In the next four snapshots, Alg_P results in the best average scores of $Precision$, F-score, and GDM. Comparing DMF_R and DMF_M with Alg_P, it turns out that DMF_R and DMF_M result in the best $Recall$ in all cases. However, algorithm Alg_P has the best scores of $Precision$ in all cases which results in the best scores of F-score in all snapshots. Also, the best GDM scores go to algorithm Alg_P in snapshots 2, 3, 4, and 5. As it can be seen from Table 5.1, almost in all cases, Alg_M has better average results in comparison with Alg_R. Furthermore, in snapshots 2, 3, and 4, Alg_M results in higher averages of $Precision$,

Table 5.5: Execution time (s)

Dataset	Alg_M	Alg_R	DMF_M	DMF_R	Alg_P
LFR Switch	28.8986	32.74205	54.4883	95.1228	**19.24631**
LFR Expand/Contraction	31.1020	36.384	75.0148	133.3144	**20.1922**
LFR Merge/Split	102.9329	150.6100	272.0781	496.4822	**64.5965**
LFR Birth/Death	23.4686	28.5201	62.6446	109.4156	**15.7374**

F-score, and GDM in comparison with DMF_R and DMF_M.

Also, tables 5.2, 5.3 and 5.4 illustrate the average results of the same experiment on the same algorithms for LFR expand/contraction, merge/split, and birth/death networks, respectively. As it can be seen, the results of these tables follow the same pattern as Table 5.1 and Alg_P results in the best average scores of $Precision$, F-score, and GDM in snapshots 2, 3, 4, and 5. As it can be understood from tables 5.1, 5.2, 5.3 and 5.4, Alg_P is the best local community detection algorithm to be used in a dynamic structure in comparison with the other four algorithms.

Since real-world networks are huge in size and change over time, the execution time of the algorithms is important. In this regard, Table 5.5 shows the execution time of the implemented algorithms in the first experiment on the four aforementioned networks. Regarding this table, Alg_P is the fastest algorithm in the proposed dynamic structure in comparison with the other four algorithms. Moreover, algorithms Alg_M and Alg_R are the next fastest algorithms, respectively. As can be seen, DMF_R is by far the slowest algorithm in this experiment.

5.4.3 Experimental Results on Section 5.3.2

In order to conduct the experiment, the proposed algorithm $DevDynaP$ and $IncL$ (Takaffoli et al., 2013) are executed on the four generated datasets including switch, expand/contraction, merge/split, and birth/death in fifteen snapshots, and the results are reported.

In this respect, three metrics are employed to evaluate the resulted communities in dynamic networks. The first metric is GDM (refer to (14)). It is noteworthy that the score of GDM for a

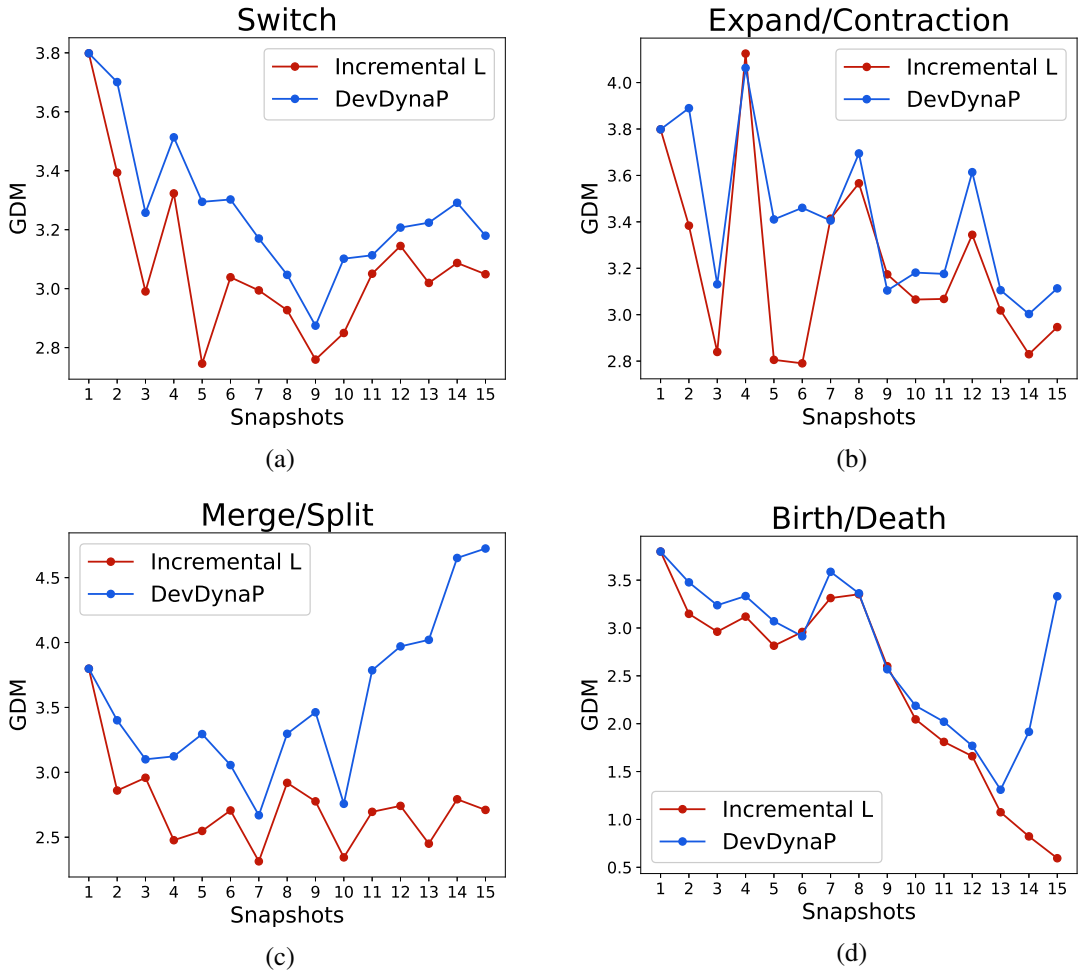

Figure 5.1: Average GDM on switch, expand/contraction, merge/split, and birth/death networks

community partitioning is calculated as follows:

$$GDM = \frac{1}{s}\sum_{i=1}^{s} GDM_i \tag{25}$$

In (25), s is the number of detected communities and GDM_i is the score of GDM for i-th community. The second metric is modularity Q (Newman, 2006) (refer to (1)). Furthermore, the average number of explored nodes in each snapshot is considered the third evaluation metric. Since the purpose of the proposed algorithm is to explore the network as fast as possible, the average number of detected nodes at each snapshot is calculated and reported.

In this experiment, each node of the network is considered a starting node for the algorithms,

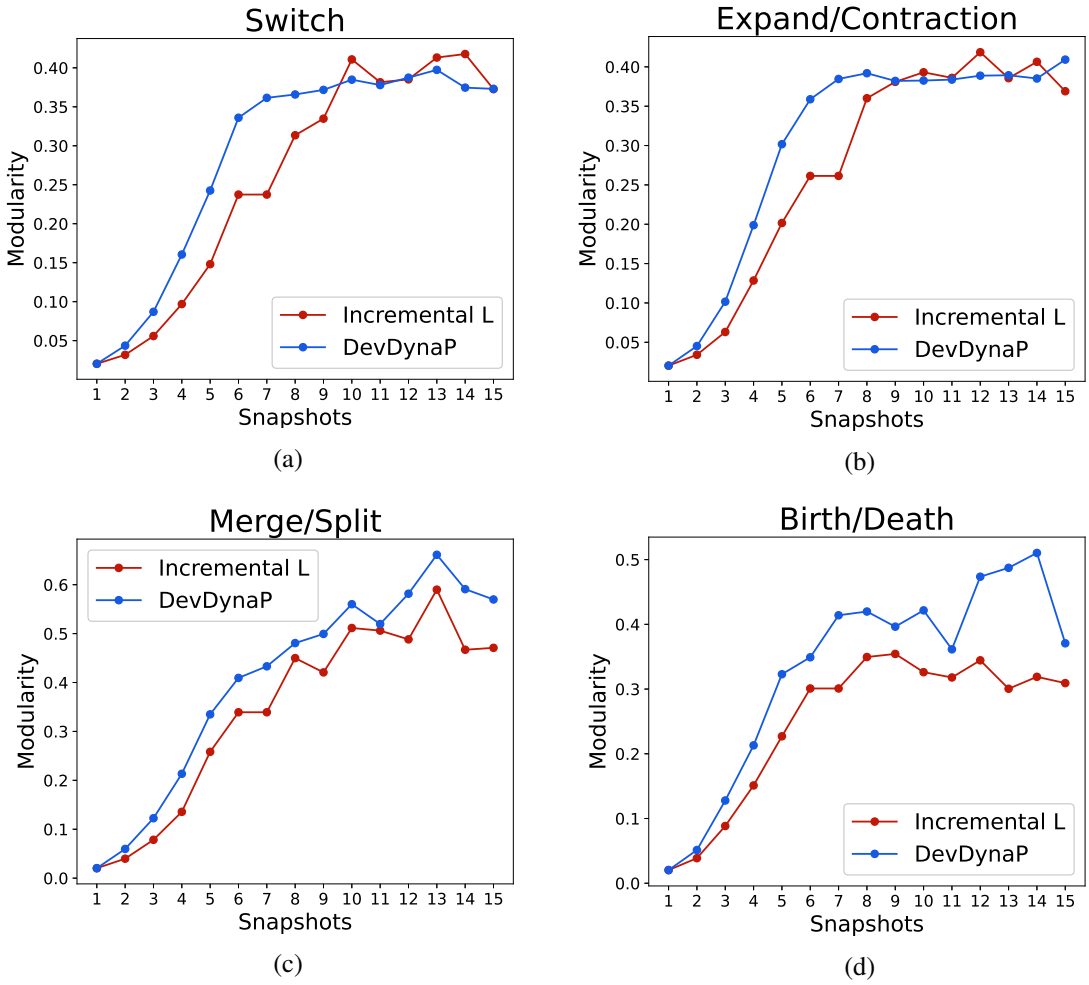

Figure 5.2: Modularity, Q on switch, expand/contraction, merge/split, and birth/death networks

and consequently, the results are calculated as average scores for each snapshot. It is noteworthy that in this experiment, Alg_P is employed to detect the community in the starting snapshot. Then, algorithms $IncL$ and $DevDynaP$ are executed, starting the second snapshot from the same communities. In this regard, the results of the algorithms are compared in equal circumstances. As it was mentioned before, $IncL$ explores repetitive and hierarchical communities which makes it unfair to compare with other community partitionings. That is why the resulted communities from $IncL$ are trimmed. In this regard, the repeated and hierarchical communities are removed. However, communities that are almost the same might still exist.

Figure 5.1, 5.2, and 5.3 demonstrates the average results of the execution of $IncL$ (Takaffoli et al., 2013) and $DevDynaP$ on four generated datasets Switch, Expand/Contraction, Merge/Split,

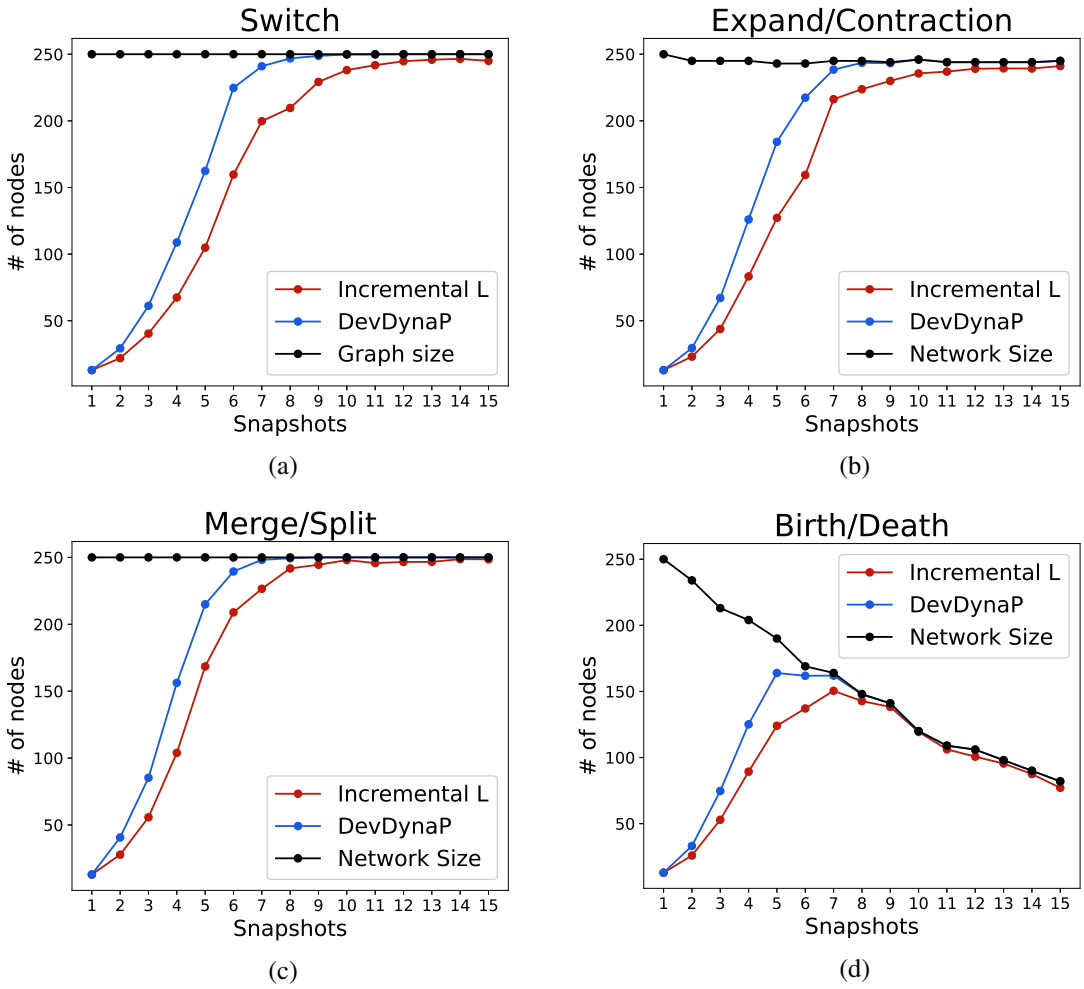

Figure 5.3: The average number of explored nodes on switch, expand/contraction, merge/split, and birth/death networks

and Birth/Death, respectively. As can be seen from Figures 5.1a, 5.1b, 5.1c and 5.1d, the average scores of GDM for $DevDynaP$ outperform that of $IncL$, almost in all snapshots. More specifically, in later snapshots of Merge/Split and Birth/Death, the differences in the results are more significant.

Moreover, considering Figures 5.2a, 5.2b, 5.2c and 5.2d, $DevDynaP$ results in higher values of modularity in all four datasets and almost in all snapshots. In datasets Switch and Expand/Contraction, there are a couple of snapshots in which the scores of modularity for $IncL$ are higher than that of $DevDynaP$. Since the modularity results are the summation of the scores of modularity for each community, and concerning the fact that the average values of GDM are higher

Table 5.6: Average scores of GDM, Q, and the number of explored nodes for $IncL$ and $DevDynaP$ on switch, expand/contraction, merge/split, and birth/death networks (snapshots 1 to 8)

Snapshots	Metrics	$IncL$				$DevDynaP$			
		Switch	Expand	Merge	Birth	Switch	Expand	Merge	Birth
1	GDM	**3.7985**	**3.7985**	**3.7985**	**3.7985**	**3.7985**	**3.7985**	**3.7985**	**3.7985**
	Q	**0.0203**	**0.0203**	**0.0203**	**0.0203**	**0.0203**	**0.0203**	**0.0203**	**0.0203**
	# of nodes	**12.94**	**12.94**	**12.94**	**12.94**	**12.94**	**12.94**	**12.94**	**12.94**
2	GDM	3.3936	3.3836	2.8598	3.1478	**3.7008**	**3.8897**	**3.4009**	**3.4766**
	Q	0.0318	0.0341	0.0398	0.0387	**0.0436**	**0.0452**	**0.0597**	**0.0513**
	# of nodes	21.92	23.06	27.792	25.884	**29.244**	**29.488**	**40.616**	**33.14**
3	GDM	2.9905	2.8394	2.9580	2.9611	**3.2575**	**3.1309**	**3.1001**	**3.2378**
	Q	0.0559	0.0631	0.0784	0.0884	**0.0870**	**0.1016**	**0.1225**	**0.1278**
	# of nodes	40.364	43.812	55.736	52.94	**61.204**	**67.084**	**85.292**	**74.74**
4	GDM	3.3231	**4.1249**	2.4769	3.1186	**3.5134**	4.0630	**3.1230**	**3.3338**
	Q	0.0969	0.1285	0.1356	0.1511	**0.1605**	**0.1988**	**0.2134**	**0.2130**
	# of nodes	67.42	83.268	103.896	89.328	**108.768**	**126.012**	**156.212**	**125.132**
5	GDM	2.7326	3.3848	3.4046	3.1617	**3.2945**	**3.4104**	**3.2946**	**3.0701**
	Q	0.1481	0.2016	0.2583	0.2270	**0.2426**	**0.3017**	**0.3349**	**0.3229**
	# of nodes	104.84	127.244	168.472	124.08	**162.4**	**184.244**	**214.9**	**163.952**
6	GDM	3.0386	2.7904	2.7062	2.9585	**3.3025**	**3.4604**	**3.0558**	**2.9133**
	Q	0.2374	0.2614	0.3392	0.3009	**0.3360**	**0.3588**	**0.4094**	**0.3491**
	# of nodes	159.692	159.376	208.888	137.108	**224.744**	**217.392**	**239.476**	**161.852**
7	GDM	2.9940	**3.4137**	2.3140	3.3124	**3.1701**	3.4057	**2.6690**	**3.5868**
	Q	0.3076	0.3383	0.3822	0.3555	**0.3615**	**0.3846**	**0.4331**	**0.4140**
	# of nodes	199.752	216.228	226.54	150.48	**240.964**	**238.452**	**248.176**	**161.968**
8	GDM	2.9270	3.5660	2.9188	3.3522	**3.0468**	**3.6946**	**3.2968**	**3.3622**
	Q	0.3135	0.3601	0.4499	0.3495	**0.3658**	**0.3919**	**0.4806**	**0.4198**
	# of nodes	209.664	223.732	241.76	142.68	**246.868**	**243.544**	**249.34**	**147.736**

for $DevDynaP$ in the same snapshots, it seems that there are still communities in the resulted community partitioning of $IncL$ which are almost the same. These communities are different only in a small number of nodes, which still is not good to have almost the same communities in a community partitioning.

Also, Figures 5.3a, 5.3b, 5.3c and 5.3d illustrate the average number of explored nodes at each snapshot, in four datasets. In these figures, the black line shows the actual number of nodes in the corresponding snapshot for each dataset. As can be observed from the figures, in all four datasets, $DevDynaP$ reaches the actual number of nodes of the network. Moreover, the average explored

Table 5.7: Average scores of GDM, Q, and the number of explored nodes for $IncL$ and $DevDynaP$ on switch, expand/contraction, merge/split, and birth/death networks (snapshots 9 to 15)

Snapshots	Metrics	$IncL$				$DevDynaP$			
		Switch	Expand	Merge	Birth	Switch	Expand	Merge	Birth
9	GDM	2.7593	**3.1738**	2.7762	2.6004	**2.8744**	3.1043	**3.4626**	**2.5686**
	Q	0.3348	0.3809	0.4207	0.3543	**0.3717**	**0.3821**	**0.4993**	**0.3964**
	# of nodes	229.232	229.98	244.48	138.292	**248.736**	**243.344**	**250.0t**	**141.0**
10	GDM	2.8495	3.0653	2.3445	2.0451	**3.1016**	**3.1813**	**2.7574**	**2.1871**
	Q	**0.4108**	**0.3931**	0.5115	0.3261	0.3848	0.3826	**0.5601**	**0.4216**
	# of nodes	238.076	235.604	247.996	119.372	**249.776**	**246.0**	**250.0**	**120.0**
11	GDM	3.0506	3.0679	2.6955	1.8100	**3.1132**	**3.1758**	**3.7860**	**2.0205**
	Q	**0.3815**	**0.3860**	0.5061	0.3179	0.3779	0.3838	**0.5196**	**0.3614**
	# of nodes	241.788	236.956	245.76	106.204	**249.808**	**244.0**	**250.0**	**109.0**
12	GDM	3.1448	3.3442	2.7415	1.6607	**3.2072**	**3.6140**	**3.9702**	**1.7691**
	Q	0.3852	**0.4186**	0.4882	0.3444	**0.3874**	0.3888	**0.5816**	**0.4736**
	# of nodes	244.736	239.088	246.572	100.68	**250.0**	**244.0**	**250.0**	**106.0**
13	GDM	3.0194	3.0183	2.4501	1.0749	**3.2238**	**3.1055**	**4.0207**	**1.3103**
	Q	**0.4132**	0.3856	0.5899	0.3005	0.3974	**0.3893**	**0.6612**	**0.4873**
	# of nodes	245.868	239.36	246.66	95.448	**250.0**	**244.0**	**250.0**	**98.0**
14	GDM	3.0871	2.8296	2.7923	0.8228	**3.2913**	**3.0029**	**4.6516**	**1.9152**
	Q	**0.4177**	**0.4064**	0.4670	0.3189	0.3746	0.3852	**0.5909**	**0.5103**
	# of nodes	246.5	239.252	248.672	87.52	**250.0**	**244.0**	**250.0**	**90.0**
15	GDM	3.0489	2.9465	2.7103	0.5946	**3.1796**	**3.1133**	**4.7244**	**3.3310**
	Q	0.3727	0.3690	0.4709	0.3092	**0.3730**	**0.4092**	**0.5698**	**0.3707**
	# of nodes	245.016	241.092	248.588	77.092	**250.0**	**245.0**	**250.0**	**82.0**

nodes by $DevDynaP$ is more than that of $IncL$ at all snapshots in the four datasets. In this regard, from the moment the network is fully explored by $DevDynaP$, there is no limitation of lack of information on the network, and any dynamic algorithm which needs the whole information of the network can be employed to detect communities.

Furthermore, Tables 5.6 and 5.7 show the same results more specifically. The name of the datasets are shortened in the table and referred to as Switch, Expand, Merge, and Birth, respectively. Table 5.6 shows the results on snapshots 1 to 8 and Table 5.6 reports the results on snapshots 9 to 15. Regarding these tables, $DevDynaP$ explores the whole network at snapshots 12, 10, 9, and 9 for Switch, Expand/Contraction, Merge/Split, and Birth/Death networks, respectively. However, algorithm $IncL$ cannot explore the whole network even in fifteen snapshots for all four networks.

Table 5.8: Execution time (s)

Algorithms	LFR Switch	LFR Expand	LFR Merge/Split	LFR Birth/Death
$IncL$	5108.59	5141.62	36747.18	1137.47
$DevDynaP$	**947.36**	**1139.26**	**6073.24**	**684.52**

Also, Table 5.8 reports the execution time of $IncL$ and $DevDynaP$. Regarding this table, $DevDynaP$ is much faster than $IncL$ in all four LFR networks. $IncL$ detects repetitive communities which increases the execution time of the algorithm while no more parts of the network are being discovered.

5.5 Conclusion and Future works

In this chapter, the problem of dynamic local community detection is addressed. In this regard, several existing local community detection algorithms are implemented and employed in a dynamic structure to detect communities locally in dynamic networks. Experimental results show that Alg_P outperforms other compared algorithms. Furthermore, the execution time of Alg_P is less than other algorithms in all cases. The importance of the results is to show the strengths and weaknesses of the existing algorithms when applied to dynamic networks. Also, by analyzing the reported results, a more efficient algorithm can be improved to detect local communities in dynamic networks.

As a result, a local dynamic community detection algorithm, $DevDynaP$ is proposed. The main goal of $DevDynaP$ is to explore the network as fast as possible and detect communities at the same time. Starting from a single node, and employing a local community detection algorithm (Alg_P), $DevDynaP$ incrementally detects communities. Moreover, using a simple structure, it explores the network faster than the compared algorithm. Also, the experimental results show that the community partitioning resulting from the proposed algorithm outperforms that of the other compared algorithm.

In order to further investigate the problem of dynamic local community detection, the following directions can be considered for future works:

- Employing big real-world networks to evaluate the proposed algorithm in dynamic networks.

- Analyzing and employing AI/ML techniques to tackle this problem. Since AI/ML techniques are not widely investigated in the dynamic local community detection problem, it is an interesting direction for future work.
- Analyzing the local community detection problem is temporal networks.

Chapter 6

A Local Community Detection Algorithm in Signed Networks

6.1 Introduction

Many networks contain both positive and negative relations. Positive relations in signed networks denote positive links, such as ”friend”, “trust”, “like”, “support” and “cooperative” relationships. On the contrary, negative links denote negative relationships, such as “enemy”, “distrust”, “dislike”, “oppose”, and “hostile” relationships. Signed networks are ubiquitous in the real world, such as social networks containing trust and distrust relationships, protein interaction networks containing activation and inhibition relationships, and international relationship networks containing cooperation and hostility relationships. In addition to these naturally formed signed networks, we can also artificially construct signed networks from interactions among data objects by using specific algorithms. For example, Hassan, Abu-Jbara, and Radev (2012) applied linguistic analysis techniques to identify attitudes (support or oppose) from online discussion texts and then built a debater-signed network. Also, Maniu, Cautis, and Abdessalem (2011) inferred a trust-signed network by aggregating various user interactions on Wikipedia content. Hoang and Lim (2017) computed the cosine similarities of all pairs of documents and then constructed a document-signed network by treating the similarities as the weights of the corresponding links.

Signed networks are represented as a graph $G_s = (V, E^+, E^-)$ in which V is the set of nodes,

E^+ and E^- show the positive and negative relations, respectively. In such networks, a negative link from node A to B indicates A "dislikes" B and a positive relation shows A "likes" B. To some certain theoretical approaches in network analysis, negative relations are fundamental. In the past few years, some important negative interactions such as bullying and social exclusion have been the subject of extensive research (DeWall, 2013).

Negative relations in signed networks cause different structures than in regular networks. For instance, however, high levels of transitivity in positive tie networks are expected (e.g., the friends of friends are often friends), but levels of transitivity in signed networks are very low (e.g., enemies of enemies do not tend to be enemies). As a result, the definition of communities is different in signed networks. The challenge of community detection in networks with positive and negative links was firstly addressed by social balance theory (Heider, 1946). This theory is based on the notion that if two people are positively related, their attitudes toward a third person should match. For example, if Bob and Mary are positively related as friends, and both of them are related to John, they should both be related to him either positively or negatively. In either case, their triad is said to be socially balanced.

Definition 6.1.1. A complete signed graph G is structurally balanced whenever all triads are balanced.

Theorem 6.1.1. *(Structure theorem) If the graph G is structurally balanced, then it can be partitioned into two clusters such that there are only positive links within each cluster and negative links between them (Harary et al., 1953).*

Figure 6.1 shows two possible conditions for a balanced triad in signed networks. There are four possible configurations for having positive or negative links between a triad. Here, a solid line represents a positive link and a dashed line shows a negative link. The upper two triads are structurally balanced because the product of their signs is positive. Similarly, the lower two triads are not structurally balanced because the product of their signs is negative. If all triads (in a complete network) are structurally balanced, the network can be partitioned into two factions such that they are internally positively linked, with negative links between the two factions, as illustrated on the right. The definition of balanced triads can be extended for cycles. A cycle is balanced if its sign is

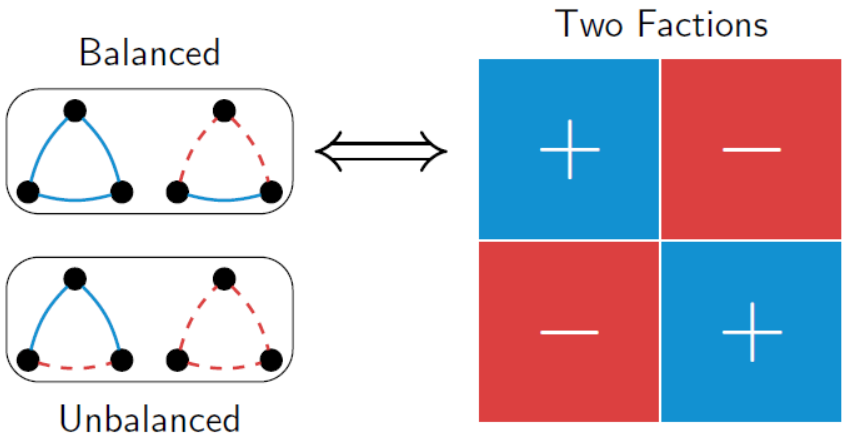

Figure 6.1: Four possible conditions for a triad in signed networks

positive. If a cycle contains m negative edges, then $sgn(C) = (-1)^m$. In other words, a cycle is balanced if it contains an even number of negative links.

Definition 6.1.2. A signed graph G is called balanced if all its cycles C are balanced.

However, it was later proved that it is not necessary to determine the structural balance of all cycles, and the balance of chord-less cycles would make the network balanced as well, the conditions to make a network structurally balanced are still strict and almost unlikely to happen. As a result, the definitions of weakly balanced cycles and graphs are defined as follows:

Definition 6.1.3. A cycle C is termed weakly balanced if it does not contain exactly a single negative link.

Theorem 6.1.2. *Let G be a signed network. Then G is weakly structurally balanced if and only if all chordless cycles are weakly balanced.*

Similar to structural balance, a weakly structurally balanced graph can be partitioned, but now in possibly more than two clusters. This is called the second structure theorem by Doreian and Mrvar (1996). Therefore, according to the aforementioned definitions, a community in signed networks can be defined as groups of nodes with positive links with each other and negative links between groups. In this regard, the community detection problem in signed networks tries to find groups of users that are densely connected by positive links within the group and negative links

between groups. Anchuri and Magdon-Ismail (2012) represents a comprehensive analysis of signed networks.

In this chapter, by extending algorithm Alg_P for signed networks, a new fast local community detection algorithm for signed networks is developed. The main contribution of this chapter is summarized as follows:

- A new signed local community detection algorithm is proposed.
- Algorithms Alg_R (Clauset, 2005) and Alg_M (F. Luo et al., 2006) are extended for signed networks to be compared with the proposed algorithm.
- Several local community detection algorithms for signed networks are implemented, compared, and the results are reported.

The remainder of this chapter is structured as follows: Section 6.2 presents the literature review. Section 6.3 represents the signed local community detection algorithm, while Section 6.4 discusses the experimental results. Finally, this chapter is concluded in Section 6.5.

6.2 Related Works

However, compared to unsigned networks, the problem of community detection in signed networks is not widely investigated, several algorithms have been proposed to detect community partitioning in signed networks (Che, Yang, & Wang, 2020; J. Chen, Wang, Wang, & Liu, 2016; R. Sun, Chen, Wang, Zhang, & Wang, 2020). Mostly, algorithms and their objective functions employ information that is dependent on the global properties of the network to detect communities.

C. He et al. (2021) proposed a similarity preserving overlapping community detection (SPOCD) method. Firstly, SPOCD computes node similarity information and geometric structure information from the link topology. Then, it uses a graph regularized binary semi-nonnegative matrix factorization (GRBSNMF) model to utilize these two sources of information to detect communities.

Also, Xia, Luo, Wang, and Li (2021) proposed a signed modularity-based community detection algorithm to explore community partitioning. This algorithm includes two phases. In the first phase, each node is considered a community. Then, the communities are iterated so that the community

is merged into its neighboring community to obtain the larger value of modularity, which will be increasingly improved. This iteration process will be ended until no improvement can be made to the value of modularity. In the second phase, all nodes within the same community given by the first phase are folded as a new node, and then all new nodes are used to reconstruct the network. The two steps are repeated until no changes happen. This algorithm uses the same pattern as the Louvain algorithm (Blondel et al., 2008).

Moreover, some studies have been conducted to detect polarized communities (Bonchi, Galimberti, Gionis, Ordozgoiti, & Ruffo, 2019; Tzeng, Ordozgoiti, & Gionis, 2020; Xiao, Ordozgoiti, & Gionis, 2020). Polarized communities are two communities that are opposed to each other. In this regard, the metrics used in these algorithms are based on two communities.

To the best of our knowledge, the problem of local community detection is not independently investigated in signed networks. However, there are some studies in which employing a local perspective and using a local metric, tried to address the community detection problem (Anchuri & Magdon-Ismail, 2012; Doreian & Mrvar, 1996; Su, Wang, Cheng, et al., 2017).

Doreian and Mrvar (1996) proposed an objective function for communities in signed networks, called Frustration. Also, they present a community detection algorithm in signed networks optimizing Frustration. Frustration, $Frus$, is defined as follows:

$$Frus = E_{in}^{-} + E_{ex}^{+} \tag{26}$$

In (26), E_{in}^{-} is the number of negative edges inside the community and E_{ex}^{+} is the number of positive bordering edges. In this regard, the network is partitioned into k random communities. Then, by examining the neighbors of each community, if merging a node into the community decreases $Frus$, the node is added to the community. The algorithm continues until no change happens.

Also, Anchuri and Magdon-Ismail (2012) proposed a community detection algorithm in signed networks optimizing a metric based on the definition of Frustration as follows:

$$Cri = E_{in}^{+} + E_{ex}^{-} - E_{in}^{-} - E_{ex}^{+} \tag{27}$$

In (27), E_{in}^{+} and E_{in}^{-} are the number of positive and negative edges inside the community. Moreover,

E^+_{ex} and E^-_{ex} are the number of positive and negative bordering edges. The algorithm proposed in (Anchuri & Magdon-Ismail, 2012), employs a local community detection algorithm optimizing Cri, to improve its community partitioning.

Furthermore, Su, Wang, Cheng, et al. (2017) used negative and positive probability functions based on positive and negative similarities to improve communities locally. The positive similarity is defined as follows:

$$Similarity(u,v)^+ = \frac{|N^+_u \cap N^+_v|}{|N^+_u| \cup |N^+_v|} \tag{28}$$

In (28), N^+_u is the number of positive neighbors of u, where $u \in V$. Also, the negative similarity can be defined using the same pattern as the positive similarity.

In this chapter, three local community detection algorithms from (Doreian & Mrvar, 1996), (Anchuri & Magdon-Ismail, 2012), and (Su, Wang, Cheng, et al., 2017), are implemented and their results are compared with the proposed algorithm. The implemented algorithms are named FrusOpt (Doreian & Mrvar, 1996), CriOpt (Anchuri & Magdon-Ismail, 2012), and SimOpt (Su, Wang, Cheng, et al., 2017).

6.3 The Proposed Algorithm

In this section, the proposed local community detection algorithm is presented. Also, the extended version of algorithms M and R are presented. Moreover, some definitions regarding signed networks are introduced.

6.3.1 Definitions

In order to extend metric P for signed networks, it is required to introduce the definition of positive common neighbors. Positive common neighbors in signed networks are defined as follows.

Definition 6.3.1. In an undirected and signed graph $G_s = (V, E^+, E^-)$, V is the set of nodes, E^+ is the set of positive edges, and E^- is the set of negative edges. $u, v \in V$ are two nodes. If $(u,v) \in E$, where $E = E^+ \cup E^-$, is an edge between u and v, then S_{uv} is the sign of link (u,v) which can be positive or negative. By definition, if $(u,x),(x,v) \in E^+$ or $(u,x),(x,v) \in E^-$ where $x \in V$, x

is a positive common neighbor of u and v. In other words, (u, x) and (x, v) should have the same signs.

6.3.2 Signed Alg_R and Signed Alg_M

The extended version of metric R for signed networks, named SR, is described as follows:

$$SR = \frac{B_{in}^{+} + B_{ex}^{-}}{B_{in}^{+} + B_{ex}^{-} + B_{ex}^{+} + B_{in}^{-}} \tag{29}$$

In (29), B_{in}^{+} and B_{in}^{-} are the number of positive and negative links respectively from bordering nodes to the nodes which are inside the community and also, other bordering nodes. Moreover, B_{ex}^{+} and B_{ex}^{-} are the number of positive and negative links from bordering nodes to the nodes that are outside of the community, respectively. This metric is employed in the same algorithm as Alg_R (Clauset, 2005) and the new algorithm for signed networks is named Alg_{SR}. Alg_{SR} is implemented and the results are compared with the proposed algorithm.

Also, the extended version of metric M for signed networks, name SM, is described as follows:

$$SM = \frac{E_{in}^{+} + E_{ex}^{-}}{E_{in}^{-} + E_{ex}^{+}} \tag{30}$$

In (30), E_{in}^{+} and E_{in}^{-} are the number of positive and negative edges inside the community. Furthermore, E_{ex}^{+} and E_{ex}^{-} are the numbers of positive and negative edges from the community to the outside. SM is used in the same algorithm as Alg_M (F. Luo et al., 2006), called Alg_{SM}, and the results are reported.

6.3.3 The Proposed Algorithm

In this section, a local community detection algorithm for signed networks, denoted as Alg_{SP}, is presented. Alg_{SP} is an extension of Alg_S for signed networks. In this regard, the extension of metric P, called SP, is presented and used in Alg_{SP} to explore a community for a given node in signed networks.

The extension of metric P, named SP, is defined as follows:

$$SP = \frac{NCN_C^+ + E_{in}^+ + E_{ex}^-}{NCN_C^+ + E_{in}^+ + E_{in}^- + E_{ex}^+ + E_{ex}^-} \tag{31}$$

In equation (31), E_{in}^+ and E_{in}^- are the number of positive and negative links inside the community. Also, E_{ex}^+ and E_{ex}^- represent the number of positive and negative bordering links. Moreover, NCN_C^+ is the number of positive common neighbors between every pair of nodes inside the community.

Also, another metric, named A_s, is proposed as follows:

$$A_s = \frac{E_{in}^+ + E_{ex}^-}{E_{in}^+ + E_{in}^- + E_{ex}^+ + E_{ex}^-} \tag{32}$$

Algorithm 8 presents the proposed local community detection algorithm (Alg_{SP}) in signed networks. This algorithm follows almost the same pattern as Alg_P, only using metrics SP and A instead of P and M respectively. As it can be seen from Algorithm 8, the proposed algorithm has three different steps. In the first step, it selects nodes whose addition into the community increases metric SP. It is noteworthy that the addition process starts from positive neighbors. Then, neighbors with both positive and negative links to the community are evaluated and added. Finally, neighbors with only negative links to the community are considered. In this regard, at each iteration, the neighboring nodes are traversed three times. In the second step, the algorithm checks all the nodes that were added in phase 1 and removes the nodes that increase SP. The above-mentioned two steps are repeated until no more nodes can be added to the community. Lastly, in the third step, nodes are removed, if their removal increases metric A_s (refer to (32)). In comparison with the other local signed metrics, metric SP is a loose metric. That is why it allows more nodes to be added to the community. However, this increases the opportunity to capture all possible nodes for the community detection process, some irrelevant nodes may be merged into the community. As a result, a tighter metric A_s is used to remove irrelevant nodes added in step 1. It should be mentioned that the tightness of metrics SM and A_s are the same. To avoid having zero values in the denominator of SM, metric A_s is improved and employed.

Algorithm 8 The extension of Alg_P (Algorithm 2) for signed networks, Alg_{SP}

Input: G_s and v_0
Output: C: v_0's local community
$C = \{\}$, $N = \{\}$
$SP = 0$
add v_0 to C
add all neighbors of v_0 to N
do
    $Q1 = \{\}$
    step 1:
    for v in N (starting from $+$, then $+$ and $-$, and finally $-$ neighbors) **do**
        calculate SP_v if v is added to C
        if $SP_v > SP$ **then**
            $SP = SP_v$
            add v to C, add v to $Q1$, remove v from N
        end if
    end for
    step 2:
    for v in $Q1$ **do**
        calculate SP_v if v is removed from C
        if $SP_v > SP$ **then**
            $SP = SP_v$
            remove v from C, remove v from $Q1$
        end if
    end for
    update N
while Q1 is not empty
do
    $Q2 = \{\}$
    step 3:
    for v in C **do**
        calculate ΔA_s if v is removed from C
        if $\Delta A_s > 0$ and v is not v_0 **then**
            remove v from C, Add v into $Q2$
        end if
    end for
while Q2 is not empty

6.3.4 Complexity

In this section, the complexity analysis of the proposed algorithm Alg_{SP} is presented. To explore communities, the proposed algorithm calculates NCN^+ at each iteration which is time-consuming. To overcome this limitation, the proposed algorithm reduces the number of iterations

Algorithm 9 count-added-positive-common-neighbors(G_s, C, v)

Input: C and v
Output: z^+: the number of added positive common neighbors
$N_v =$ neighbors of v inside C
$z^+ = 0$
for i in N_v **do**
    if S_{vi} is positive **then**
        $N_i^+ =$ positive neighbors of i inside C
        $z^+ = z^+ + |N_i^+|$
    else
        $N_i^- =$ negative neighbors of i inside C
        $z^+ = z^+ + |N_i^-|$
    end if
end for
$z^+ = z^+ + |N_v^+| * (|N_v^+| - 1)/2$
$z^+ = z^+ + |N_v^-| * (|N_v^-| - 1)/2$

by adding more than one node to the community, at each iteration.

To analyze the complexity of the proposed algorithm it is assumed that at each iteration, only one node is added to the community. Assuming n as the number of nodes in the local community, and d as the average degree of nodes inside the community, in the first iteration, one node (the starting node) is in the community. Thus, d neighboring nodes must be traversed. It should be mentioned that d neighboring nodes are traversed 3 times to first capture positive neighbors, then neighbors with both positive and negative links to the community, and finally negative neighbors. As a result, $3d$ nodes are traversed at the first iteration. Similarly, in the last iteration, n nodes are added, and $3nd$ neighboring nodes must be traversed. Consequently, it is required to traverse $3(d + 2d + \cdots + nd)$ nodes. As a result, considering p_s as the required time complexity to calculate metric SP for each node, we have $\sum_{i=1}^{n} 3(i)dp_s = 3dp_s[n(n+1)/2]$.

To calculate SP at each iteration and using the same perspective as metric P, consider node v is added into the community C. The updated value of SP, SP', after addition of node v, is calculated using formula (33) as follows:

$$SP' = \frac{(NCN_C^+ + z^+) + (E_{in}^+ + x^+) + (E_{ex}^- + y^- - x^-)}{(NCN_C^+ + z^+) + (E_{in}^+ + x^+) + (E_{in}^- + x^-) + (E_{ex}^+ + y^+ - x^+) + (E_{ex}^- + y^- - x^-)} \tag{33}$$

On the other hand, assuming node v is removed from the community C, the updated score of SP,

SP' is calculated using formula (34) as follows:

$$SP' = \frac{(NCN_C^+ - z^+) + (E_{in}^+ - x^+) + (E_{ex}^- - y^- + x^-)}{(NCN_C^+ - z^+) + (E_{in}^+ - x^+) + (E_{in}^- - x^-) + (E_{ex}^+ - y^+ + x^+) + (E_{ex}^- - y^- + x^-)} \tag{34}$$

In (33) and (34), x^+ (x^-) is the number of positive (negative) neighbors of node v that are inside the community, and y^+ (y^-) is the number of v's positive (negative) neighbors that are not inside the community. Moreover, z^+ is the number of positive common neighbors that node v adds into (subtracts from) the community by its addition (removal). According to these equations, for each node at each iteration, it is required to calculate x^+, x^-, y^+, y^-, and z^+. Since $x^+ + x^- + y^+ + y^- = d$, the complexity of the calculation of x^+, x^-, y^+, and y^- is $O(d)$. Moreover, Algorithm 9 shows how to calculate z^+, when a new node v is added into the community C, in graph G_s. In this algorithm, S_{vi} is the sign of the link (v, i) and $|N_v^+|$ is the number of positive neighbors of node v that are inside C. Two groups of nodes are counted to compute z^+. Figure 6.2 shows the two groups of nodes while adding node v to the community C. In this figure, dashed links indicate negative, and solid links illustrate positive relations. The first group of nodes includes those that are the second layer neighbors of node v that have the same signs as its first layer neighbors. These nodes are shown with the a letter in them. Regarding Algorithm 9, the first group of nodes (a nodes in Figure 6.2) is calculated in lines 5-12. Moreover, node v is a positive common neighbor to its same sign instant neighbors inside the community. According Algorithm 9, the number of times that

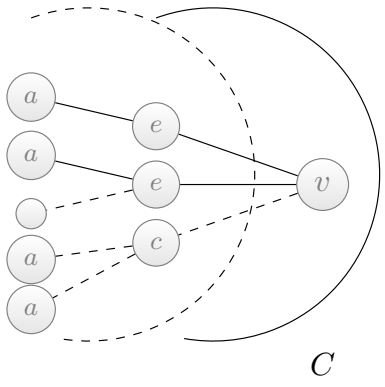

Figure 6.2: How to calculate z^+ while node v is added to the community C

v is a positive common neighbors to its same sign neighbors itself is computed via $|N_v^+| * (|N_v^+| - 1)/2 + |N_v^-| * (|N_v^-| - 1)/2$, where $|N_v^+|$ is the number of instant positive neighbors of v (e nodes in Figure 6.2) and $|N_v^-|$ is the number of instant negative neighbors of v (c nodes in Figure 6.2) that are inside the community (lines 14-15). Regarding Algorithm 9, the complexity of the calculation of z^+ is $O(d^2)$. As a result, the complexity of the proposed algorithm is $O(n^2d^3)$.

6.4 Experimental Results

In this section, the proposed community detection algorithm (Alg_{SP}) is compared with five other algorithms and the results are reported. In this regard, the community development algorithms, FrusOpt (Doreian & Mrvar, 1996), CriOpt (Anchuri & Magdon-Ismail, 2012), and SimOpt (Su, Wang, Cheng, et al., 2017) are extracted from their community partitioning detection algorithms, implemented, and compared with Alg_{SP}. Also, two more algorithms are developed to be compared with the proposed algorithm. In this concern, algorithms Alg_R (Clauset, 2005) and Alg_M (F. Luo et al., 2006) are extended for signed networks.

6.4.1 Dataset

To conduct the experiment, two illustrative signed networks and four real-world networks are employed as follows:

- **Illustrative Network 1 (IN1)** (B. Yang, Cheung, & Liu, 2007): A synthetic network with 28 nodes that are partitioned into three communities. This network is partitionable and balanced. Figure 6.3 shows this illustrative network with three ground-truth communities. In this figure, dashed links indicate negative relations and solid links indicate positive relations.

- **Illustrative Network 2 (IN2)** (B. Yang et al., 2007): A synthetic network with 28 nodes that are partitioned into three communities. This network is partitionable and not balanced. IN2 is modified from IN1 by adding 7 negative links to unbalance the network. Figure 6.4 shows this illustrative network with three ground-truth communities. In this figure, dashed links indicate negative relations and solid links indicate positive relations.

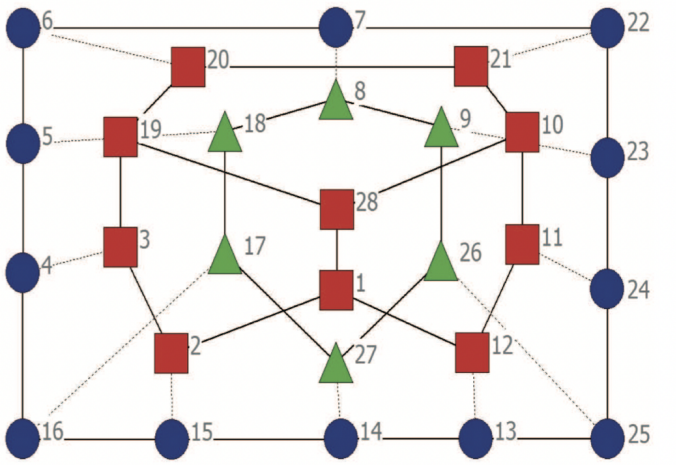

Figure 6.3: Illustrative network 1 (IN1) (J. Chen et al., 2017)

- **Slovene Parliamentary Party (SPP)** (Ferligoj & Kramberger, 1996): The network contains 10 Slovene Parliamentary parties which shows the relationships among the 10 political parties in 1994. The positive and negative links separately represent the similar and dissimilar relationships between the parties. This network is partitioned into three communities.

- **U.S. Supreme Court (USC)** (Doreian & Mrvar, 2009): This network describes the voting behavior of nine justices in the supreme court of the United States between 2006 and 2007. The positive and negative links mean whether one justice supports the other one or not. The U.S. supreme court justices' network is divided into two communities.

- **Gahuku-Gama Subtribes (GGS)** (Read, 1954): This network demonstrates the political alliance and enmities among the 16 Gahuku-Gama subtribes, which were distributed in a particular area in 1954. The positive and negative links of the network represent the positive and negative political arrangements, respectively. GGS is partitioned into three communities.

- **Congress** (Thomas, Pang, & Lee, 2006): In this network, nodes are politicians who speak in the United States Congress. An edge denotes that a speaker mentions another speaker. The sign of an edge (positive or negative) denotes whether the mention is in support of or opposition to the mentioned politician. Congress contains 219 members and 521 positive and negative links. There are no ground-truth communities for this network.

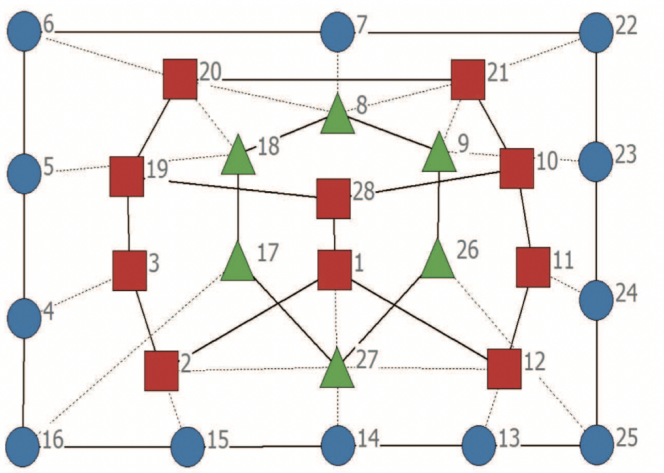

Figure 6.4: Illustrative network 2 (IN2) (J. Chen et al., 2017)

6.4.2 Evaluation Metrics

In this experiment, each node of the network is selected as a starting node for the algorithms and the results are reported in average. The detected communities are evaluated using F-score (refer to (17)) and Signed Q (Q_s) (Gómez, Jensen, & Arenas, 2009).

Gómez et al. (2009) generalized the modularity Q for networks with positive and negative links. The modularity is generalized for signed networks, Q_s as follows:

$$Q_s = \frac{1}{2m^+ + 2m^-} \sum_{ij} [w_{ij} - (\frac{w_i^+ w_j^+}{2m^+} - \frac{w_i^- w_j^-}{2m^-})]\sigma(i,j) \tag{35}$$

In equation (35), m^+ and m^- are the numbers of positive and negative links in the network. Considering w as the adjacency matrix of the signed network, $w = w^+ - w^-$. In this regard, $w_i^+ = \sum_j w_{ij}^+$ and $w_i^- = \sum_j w_{ij}^-$. Also, $\sigma(i,j)$ is 1 if i and j belong to the same community and 0 otherwise. The main property of equation (35) is that the standard modularity Q is recovered when there are no negative weights.

6.4.3 Experimental Results

Table 6.1 shows the average results of $Recall$, $Precision$, F-score, and Q_s for the algorithms FrusOpt (Doreian & Mrvar, 1996), CriOpt (Anchuri & Magdon-Ismail, 2012), SimOpt (Su, Wang,

Table 6.1: Average F-score (ratio) on real-world networks

Dataset	Evaluation	FrusOpt	CriOpt	SimOpt	Alg_{SR}	Alg_{SM}	Alg_{SP}
IN1	*Recall*	0.2928	0.7428	0.2357	**1.0**	0.8178	**1.0**
	Precision	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	F-score	0.4416	0.7922	0.3516	**1.0**	0.8690	**1.0**
	Q_s	0.0195	0.0740	0.0104	**0.1025**	0.0817	**0.1025**
IN2	*Recall*	0.2928	0.8535	0.1821	**1.0**	0.7857	**1.0**
	Precision	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	F-score	0.4416	0.8928	0.2916	**1.0**	0.8333	**1.0**
	Q_s	0.0172	0.0736	0.0057	**0.0904**	0.0667	**0.0904**
SPP	*Recall*	0.3600	**1.0**	0.2600	**1.0**	0.9	**1.0**
	Precision	**1.0**	**1.0**	**1.0**	**1.0**	0.9	**1.0**
	F-score	0.4666	**1.0**	0.4047	**1.0**	0.9	**1.0**
	Q_s	0.0209	**0.1088**	-0.0013	**0.1088**	0.0983	**0.1088**
USC	*Recall*	0.2222	0.9111	0.3111	**1.0**	0.8889	**1.0**
	Precision	**1.0**	0.9111	0.3444	0.5062	0.8889	**1.0**
	F-score	0.3629	0.9111	0.3132	0.6703	0.8889	**1.0**
	Q_s	0.0	0.1016	0.0273	0.0029	0.0912	**0.1020**
GGS	*Recall*	0.2892	**1.0**	0.4196	**1.0**	**1.0**	**1.0**
	Precision	**1.0**	0.9453	0.9375	0.6354	**1.0**	**1.0**
	F-score	0.4232	0.9708	0.5466	0.7562	**1.0**	**1.0**
	Q_s	0.0065	0.0751	0.0153	**0.0791**	0.0730	0.0730
Congress	*Recall*	-	-	-	-	-	-
	Precision	-	-	-	-	-	-
	F-score	-	-	-	-	-	-
	Q_s	0.0008	0.0034	0.0040	0.0087	0.0137	**0.0318**

Cheng, et al., 2017), Alg_{SR}, Alg_{SM}, and Alg_{SP} on the six introduced datasets. As can be seen from Table 6.1, Alg_{SP} has the best possible results on the five datasets with ground-truth data. In other words, Alg_{SP} obtains an average score of 1 for metrics $Recall$, $Precision$, and F-score. In this regard, Alg_{SP} explored the ground-truth communities for all five datasets including IN1, IN2, SPP, USC, and GGS. It is noteworthy that Alg_{SP} obtains the ground-truth data, independent of the starting nodes. This is a valuable feature for the proposed algorithm. Because this feature makes it possible for Alg_{SP} to be used in any global community detection algorithm to detect and extend the communities without considering the starting points of the detection.

Table 6.2: Execution time (s)

Dataset	FrusOpt	CriOpt	SimOpt	Alg_{SR}	Alg_{SM}	Alg_{SP}
IN1	0.003	0.11	**0.04**	0.24	0.08	0.09
IN2	0.005	0.14	**0.03**	0.28	0.12	0.12
SPP	0.008	0.03	**0.02**	0.06	**0.02**	**0.02**
USC	0.003	**0.02**	0.07	0.06	0.03	**0.02**
GGS	0.007	0.10	0.09	0.09	0.07	**0.05**
Congress	0.13	**5.65**	105.94	543.47	15.20	29.91

Considering the average scores of Q_s, the best scores go to Alg_{SP} for all networks except for GGS. However, Alg_{SP} reaches the ground-truth communities on GGS, Alg_{SR} has the best score of Q_s. Since the value of $Precision$ for Alg_{SR} on GGS is low, a relatively big number of wrong nodes are inside the resulted community. This increases the score of Q_s.

Comparing other algorithms in Table 6.1, it can be observed that algorithms Alg_{SR} and Alg_{SP} have better results than the other three algorithms. Alg_{SR} obtains the best possible scores for $Recall$, $Precision$, and F-score on IN1, IN2, and SPP. Also, Alg_{SM} has F-score scores of greater than 0.8 for all networks. After Alg_{SR} and Alg_{SM}, it can be concluded that CriOpt has higher scores than FrusOpt and SimOpt.

Moreover, Table 6.2 represents the execution time of the algorithms for this experiment on the six networks. Regarding this table, FrusOpt has the fastest algorithm among all algorithms. Since the results of FrusOpt are not comparable with other algorithms, it is not participating in the execution time comparison. As can be seen from Table 6.2, algorithms CriOpt, Alg_{SM}, and Alg_{SP} are among the fastest algorithms. In contrast, regarding the time execution on the Congress network, SimOpt and Alg_{SR} are among the slowest algorithms.

6.5 Conclusion and Future Works

In this chapter, the extended version of the algorithm Alg_P and metric P, called Alg_{SP} and SP for signed networks are presented. In this regard, the definition of positive common neighbors in signed networks is introduced. Also, the extended version of algorithms Alg_R and Alg_M for signed

networks are discussed and introduced. The experimental results show that algorithm Alg_{SP} obtains the ground-truth communities of the employed real-world networks. Also, it shows that Alg_{SP} detects the ground-truth communities of the employed dataset, independent from the starting nodes. Moreover, the execution time of the algorithm represents that Alg_{SP} is among the fast algorithms in community detection.

Regarding the reported experimental results and the efficiency of Alg_{SP}, there are some interesting directions for future works as follows:

- Employing big real-world signed networks to evaluate the proposed algorithm,
- Extending the proposed metric SP and Alg_{SP} for weighted and also, directed signed networks and using directed weighted networks to analyze the efficiency of the proposed algorithm.
- Using Alg_{SP} in a community detection algorithm to explore the whole community partitioning of signed social networks.

Chapter 7

Conclusion

7.1 Conclusion and Future works

Community is an important structure of social networks that is defined as a group of well-connected nodes. During the last two decades, several works and studies have been done to detect different types of communities in social networks. Today, with increasing the size and complexity of real-world social networks, working with the whole information of the networks seems almost impossible. As a result, the problem of local community detection has become more popular. This problem includes detecting a community for a given node only using local information. In this regard, the best community partitioning algorithms employ local perspectives to explore the community partitioning of complex social networks. Accordingly, these algorithms need an efficient algorithm to detect communities or improve already detected communities locally. It can be concluded that the local community detection problem, i.e. detecting a community for a given node is the core issue of exploring community partitioning in big real-world networks. That is why, a comprehensive study of community detection considering the local point of view, is conducted, in this thesis. In this regard, having local community detection algorithms that can detect a high-quality community for any randomly given node, regardless of the degree and importance of the node in the network, is the main goal of this study.

The main focus of this thesis is on the problem of local community detection and how to locally evaluate communities. In this concern, the problem of local community detection in static networks,

dynamic networks, and signed networks is analyzed and investigated.

In order to evaluate the quality of communities, a new metric, GDM, is proposed by employing geodesic distance. Summing up the length of the shortest path between every pair of vertices, the geodesic distance can accurately represent the density of a given community. According to the experimental results, GDM fairly compares the communities resulting from different algorithms only using local information.

Moreover, employing the number of common neighbors and proposing a new metric, a new local community detection algorithm, Alg_P, is presented. Also, several local community detection algorithms are implemented to be compared with the proposed one. The experimental results show that Alg_P outperforms the above-mentioned algorithms. Furthermore, it has been shown that the execution time of the proposed algorithm is faster than the other near-complexity algorithms.

As it was mentioned before, the problem of local community detection is addressed in dynamic social networks. In this regard, a dynamic local community detection algorithm, $DevDynaP$, is proposed, employing algorithm Alg_P. This algorithm explores the network and detects communities at the same time. The main goal of $DevDynaP$ is to explore the network as fast as possible and detect high-quality communities simultaneously. Experimental results show that $DevDynaP$ explores the network faster than the compared algorithm. Also, the community partitioning resulting from $DevDynaP$ outperforms that of the other compared algorithm.

Furthermore, the proposed local community detection algorithm, Alg_P, is extended to explore communities in signed networks. The extended algorithm for signed networks is called Alg_{SP}. Alg_{SP} uses the same perspective as algorithm Alg_P, only optimizing metric SP, instead of P. Also, two classic local community detection algorithms Alg_R and Alg_M, are extended to explore local communities in signed networks and be compared with Alg_{SP}. The experimental results indicate that Alg_{SP} obtains the ground-truth communities of the employed real-world networks. It can be concluded that algorithm Alg_{SP} detects communities independent from the starting nodes. In this regard, the goal of the thesis is fulfilled. Moreover, algorithm SP is among the fast algorithms in community detection.

Finally, regarding all the reported results in this thesis, the most interesting directions for future work are listed as follows:

- Extending metric GDM for directed, weighted, and signed networks.
- Employing algorithm Alg_P in a community detection algorithm to detect the whole community partitioning of the real-word networks.
- Employing big real-world networks to evaluate $DevDynaP$.
- Employing algorithm Alg_{SP} in a community detection algorithm to detect the whole community partitioning of the real-word signed networks.

7.2 Publications

This PhD degree resulted in the following scientific publications:

- A conference paper (Bakhtar, Gholami, & Harutyunyan, 2020) and a journal paper (Bakhtar & Harutyunyan, 2021b) are published considering the proposed metric GDM in Chapter 3.
- Moreover, another conference paper (Bakhtar & Harutyunyan, 2021a) is published regarding algorithm Alg_P and Chapter 4.
- Furthermore, a workshop paper (Bakhtar & Harutyunyan, 2022b) and a journal paper (Bakhtar & Harutyunyan, 2022a) are published and and submitted, respectively regarding Chapter 5.
- Finally, a conference paper will be written and submitted considering Chapter 6 and algorithm Alg_{SP}.

References

Ahn, Y.-Y., Bagrow, J. P., & Lehmann, S. (2010). Link communities reveal multiscale complexity in networks. *nature*, *466*(7307), 761.

Amelio, A., & Pizzuti, C. (2013). Community mining in signed networks: a multiobjective approach. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (pp. 95–99).

Anchuri, P., & Magdon-Ismail, M. (2012). Communities and balance in signed networks: A spectral approach. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (pp. 235–242).

Andersen, R., Chung, F., & Lang, K. (2006). Local graph partitioning using pagerank vectors. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)* (pp. 475–486).

Bakhtar, S., Gholami, M. S., & Harutyunyan, H. A. (2020). A New Metric to Evaluate Communities in Social Networks Using Geodesic Distance. In *9th International Conference on Computational Data and Social Networks (CSoNet 2020)* (pp. 202–216).

Bakhtar, S., & Harutyunyan, H. A. (2021a). A New Fast Local Community Detection Algorithm Using the Number of Common Neighbours. In *8th International Conference on Social Networks Analysis, Management and Security (SNAMS 2021)* (pp. 1–8).

Bakhtar, S., & Harutyunyan, H. A. (2021b). A new metric to compare local community detection algorithms in social networks using geodesic distance. *Journal of Combinatorial Optimization*, 1–23.

Bakhtar, S., & Harutyunyan, H. A. (2022a). DevDynaP: A Dynamic Local Community Detection Algorithm. *Submitted to the Journal of Computational Social Networks*.

Bakhtar, S., & Harutyunyan, H. A. (2022b). Dynamic Local Community Detection Algorithms. In *IEEE/IFIP Network Operations and Management Symposium (NOMS 2022)* (pp. 1–6).

Bansal, N., Blum, A., & Chawla, S. (2004). Correlation clustering. *Machine learning*, *56*(1-3), 89–113.

Bansal, S., Bhowmick, S., & Paymal, P. (2011). Fast community detection for dynamic complex networks. In *Complex Networks* (pp. 196–207). Springer.

Bedi, P., & Sharma, C. (2016). Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *6*(3), 115–135.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, *2008*(10), P10008.

Bonchi, F., Galimberti, E., Gionis, A., Ordozgoiti, B., & Ruffo, G. (2019). Discovering polarized communities in signed networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (pp. 961–970).

Boudebza, S., Cazabet, R., Azouaou, F., & Nouali, O. (2018). Olcpm: An online framework for detecting overlapping communities in dynamic social networks. *Computer Communications*, *123*, 36–51.

Bouyer, A., & Roghani, H. (2020). Lsmd: a fast and robust local community detection starting from low degree nodes in social networks. *Future Generation Computer Systems*, *113*, 41–57.

Cao, C., Ni, Q., & Zhai, Y. (2015). An improved collaborative filtering recommendation algorithm based on community detection in social networks. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (pp. 1–8).

Caselles, V., Kimmel, R., & Sapiro, G. (1997). Geodesic active contours. *International journal of computer vision*, *22*(1), 61–79.

Chakraborty, T., Dalmia, A., Mukherjee, A., & Ganguly, N. (2017). Metrics for community analysis: A survey. *ACM Computing Surveys (CSUR)*, *50*(4), 1–37.

Che, S., Yang, W., & Wang, W. (2020). A memetic algorithm for community detection in signed networks. *IEEE Access*, *8*, 123585–123602.

Chen, J., Liu, D., Hao, F., & Wang, H. (2020). Community detection in dynamic signed network: an intimacy evolutionary clustering algorithm. *Journal of Ambient Intelligence and Humanized Computing*, *11*(2), 891–900.

Chen, J., Wang, H., Wang, L., & Liu, W. (2016). A dynamic evolutionary clustering perspective: Community detection in signed networks by reconstructing neighbor sets. *Physica A: Statistical Mechanics and its Applications*, *447*, 482–492.

Chen, J., Zaïane, O., & Goebel, R. (2009). Local community identification in social networks. In *International Conference on Advances in Social Network Analysis and Mining (ASONAM 2009)* (pp. 237–242).

Chen, J., Zhang, L., Liu, W., & Yan, Z. (2017). Community detection in signed networks based on discrete-time model. *Chinese Physics B*, *26*(1), 018901.

Chen, Q., Wu, T.-T., & Fang, M. (2013). Detecting local community structures in complex networks based on local degree central nodes. *Physica A: Statistical Mechanics and its Applications*, *392*(3), 529–537.

Chen, Y., Wang, X., Yuan, B., & Tang, B. (2014). Overlapping community detection in networks with positive and negative links. *Journal of Statistical Mechanics: Theory and Experiment*, *2014*(3), P03021.

Clauset, A. (2005). Finding local community structure in networks. *Physical review E*, *72*(2), 026132.

Dakiche, N., Tayeb, F. B.-S., Slimani, Y., & Benatchba, K. (2019). Tracking community evolution in social networks: A survey. *Information Processing & Management*, *56*(3), 1084–1102.

Dao, V.-L., Bothorel, C., & Lenca, P. (2018). Estimating the similarity of community detection methods based on cluster size distribution. In *International Conference on Complex Networks and their Applications* (pp. 183–194).

Dasgupta, K., Singh, R., Viswanathan, B., Chakraborty, D., Mukherjea, S., Nanavati, A. A., & Joshi, A. (2008). Social ties and their relevance to churn in mobile telecom networks. In *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology* (pp. 668–677).

Derényi, I., Palla, G., & Vicsek, T. (2005). Clique percolation in random networks. *Physical review letters*, *94*(16), 160202.

DeWall, C. N. (2013). *The oxford handbook of social exclusion*. Oxford University Press.

Ding, X., Zhang, J., & Yang, J. (2020). Node-community membership diversifies community structures: An overlapping community detection algorithm based on local expansion and boundary re-checking. *Knowledge-Based Systems*, *198*, 105935.

Doreian, P., & Mrvar, A. (1996). A partitioning approach to structural balance. *Social networks*, *18*(2), 149–168.

Doreian, P., & Mrvar, A. (2009). Partitioning signed social networks. *Social Networks*, *31*(1), 1–11.

Ferligoj, A., & Kramberger, A. (1996). An analysis of the slovene parliamentary parties network. *Developments in statistics and methodology*, *12*, 209–216.

Fortunato, S., & Barthelemy, M. (2007). Resolution limit in community detection. *Proceedings of the national academy of sciences*, *104*(1), 36–41.

Friggeri, A., Chelius, G., & Fleury, E. (2011). Egomunities, exploring socially cohesive person-based communities. *arXiv preprint arXiv:1102.2623*.

Gao, Y., Zhang, H., & Zhang, Y. (2019). Overlapping community detection based on conductance optimization in large-scale networks. *Physica A: Statistical Mechanics and its Applications*, *522*, 69–79.

Garza, S. E., & Schaeffer, S. E. (2019). Community detection with the label propagation algorithm: a survey. *Physica A: Statistical Mechanics and its Applications*, *534*, 122058.

Ghasemian, A., Hosseinmardi, H., & Clauset, A. (2019). Evaluating overfit and underfit in models of network community structure. *IEEE Transactions on Knowledge and Data Engineering*.

Giatsoglou, M., & Vakali, A. (2013). Capturing social data evolution using graph clustering. *IEEE Internet Computing*, *17*(1), 74–79.

Girvan, M., & Newman, M. E. (2001). Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, *99*(cond-mat/0112110), 8271–8276.

Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the national academy of sciences*, *99*(12), 7821–7826.

Gómez, S., Jensen, P., & Arenas, A. (2009). Analysis of community structure in networks of correlated data. *Physical Review E*, *80*(1), 016114.

Greene, D., Doyle, D., & Cunningham, P. (2010). Tracking the evolution of communities in dynamic social networks. In *International Conference on Advances in Social Networks Analysis and Mining* (pp. 176–183).

Gregory, S. (2010). Finding overlapping communities in networks by label propagation. *New Journal of Physics*, *12*(10), 103018.

Harary, F., et al. (1953). On the notion of balance of a signed graph. *The Michigan Mathematical Journal*, *2*(2), 143–146.

Hassan, A., Abu-Jbara, A., & Radev, D. (2012). Extracting signed social networks from text. In *Workshop proceedings of textgraphs-7: Graph-based methods for natural language processing* (pp. 6–14).

He, C., Liu, H., Tang, Y., Liu, S., Fei, X., Cheng, Q., & Li, H. (2021). Similarity preserving overlapping community detection in signed networks. *Future Generation Computer Systems*, *116*, 275–290.

He, J., & Chen, D. (2015). A fast algorithm for community detection in temporal network. *Physica A: Statistical Mechanics and its Applications*, *429*, 87–94.

Heider, F. (1946). Attitudes and cognitive organization. *The Journal of psychology*, *21*(1), 107–112.

Hoang, T.-A., & Lim, E.-P. (2017). Highly efficient mining of overlapping clusters in signed weighted networks. In *Proceedings of the 2017 acm on conference on information and knowledge management* (pp. 869–878).

Hric, D., Darst, R. K., & Fortunato, S. (2014). Community detection in networks: Structural communities versus ground truth. *Physical Review E*, *90*(6), 062805.

Jdidia, M. B., Robardet, C., & Fleury, E. (2007). Communities detection and analysis of their dynamics in collaborative networks. In *2nd International Conference on Digital Information Management* (Vol. 2, pp. 744–749).

Jebabli, M., Cherifi, H., Cherifi, C., & Hamouda, A. (2018). Community detection algorithm evaluation with ground-truth data. *Physica A: Statistical Mechanics and its Applications*, *492*, 651–706.

Jiang, J. Q. (2015). Stochastic block model and exploratory analysis in signed networks. *Physical Review E*, *91*(6), 062805.

Jiang, W., Wang, G., Bhuiyan, M. Z. A., & Wu, J. (2016). Understanding graph-based trust evaluation in online social networks: Methodologies and challenges. *ACM Computing Surveys (CSUR)*, *49*(1), 10.

Kernighan, B. W., & Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, *49*(2), 291–307.

Krebs, V. (2004). Political books network. *Unpublished, retrieved from Mark Newman's website: www-personal. umich. edu/˜ mejn/netdata*.

Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., & Kim, K. J. (2019). A survey of deep learning-based network anomaly detection. *Cluster Computing*, 1–13.

Lancichinetti, A., Fortunato, S., & Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Physical review E*, *78*(4), 046110.

Lancichinetti, A., Radicchi, F., Ramasco, J. J., & Fortunato, S. (2011). Finding statistically significant communities in networks. *PloS one*, *6*(4), e18961.

Lee, S., & Huh, J.-H. (2019). An effective security measures for nuclear power plant using big data analysis approach. *The Journal of Supercomputing*, *75*(8), 4267–4294.

Leskovec, J., Adamic, L. A., & Huberman, B. A. (2007). The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, *1*(1), 5–es.

Leung, I. X., Hui, P., Lio, P., & Crowcroft, J. (2009). Towards real-time community detection in large networks. *Physical Review E*, *79*(6), 066107.

Li, C., Tang, Z., Tang, Y., Zhao, J., & Huang, Y. (2018). Community detection algorithm with local-first approach in social networks. *J. Front. Comput. Sci. Technol.*, *12*(8), 1263–1277.

Liu, S., & Xia, Z. (2020). A two-stage bfs local community detection algorithm based on node transfer similarity and local clustering coefficient. *Physica A: Statistical Mechanics and its Applications*, *537*, 122717.

Luo, F., Wang, J. Z., & Promislow, E. (2006). Exploring local community structures in large networks. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI Main Conference Proceedings)(WI'06)* (pp. 233–239).

Luo, W., Zhang, D., Jiang, H., Ni, L., & Hu, Y. (2018). Local community detection with the dynamic membership function. *IEEE Transactions on Fuzzy Systems*, *26*(5), 3136–3150.

Luo, W., Zhang, D., Ni, L., & Lu, N. (2019). Multiscale local community detection in social networks. *IEEE Transactions on Knowledge and Data Engineering*.

Lusseau, D., Schneider, K., Boisseau, O. J., Haase, P., Slooten, E., & Dawson, S. M. (2003). The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, *54*(4), 396–405.

Maniu, S., Cautis, B., & Abdessalem, T. (2011). Building a signed network from interactions in wikipedia. In *Databases and social networks* (pp. 19–24).

Mishra, S., Singh, S. S., Mishra, S., & Biswas, B. (2021). Tcd2: Tree-based community detection in dynamic social networks. *Expert Systems with Applications*, *169*, 114493.

Newman, M. E. (2003). The structure and function of complex networks. *SIAM review*, *45*(2), 167–256.

Newman, M. E. (2004). Fast algorithm for detecting community structure in networks. *Physical review E*, *69*(6), 066133.

Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the national academy of sciences*, *103*(23), 8577–8582.

Oliveira, M., & Gama, J. (2012). An overview of social network analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(2), 99–115.

Palla, G., Derényi, I., Farkas, I., & Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, *435*(7043), 814.

Pons, P., & Latapy, M. (2005). Computing communities in large networks using random walks. In *International symposium on computer and information sciences* (pp. 284–293).

Raghavan, U. N., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, *76*(3), 036106.

Read, K. E. (1954). Cultures of the central highlands, new guinea. *Southwestern Journal of Anthropology*, *10*(1), 1–43.

Rigi, M. A., Moser, I., Farhangi, M. M., & Lui, C. (2019). Finding and tracking local communities by approximating derivatives in networks. *World Wide Web*, 1–33.

Rossetti, G., & Cazabet, R. (2018). Community discovery in dynamic networks: a survey. *ACM Computing Surveys (CSUR)*, *51*(2), 1–37.

Rossetti, G., Pappalardo, L., Pedreschi, D., & Giannotti, F. (2017). Tiles: an online algorithm for community discovery in dynamic social networks. *Machine Learning*, *106*(8), 1213–1241.

Sattari, M., & Zamanifar, K. (2018). A spreading activation-based label propagation algorithm for overlapping community detection in dynamic social networks. *Data & Knowledge Engineering*, *113*, 155–170.

Shao, L., Yu, X., & Feng, C. (2019). Evaluating the eco-efficiency of China's industrial sectors: A two-stage network data envelopment analysis. *Journal of environmental management*, *247*, 551–560.

Sharma, T., Charls, A., & Singh, P. (2009). Community mining in signed social networks-an automated approach. *ICCEA09, Manila*, *9*, 163–168.

Sizemore, A. E., Phillips-Cremins, J. E., Ghrist, R., & Bassett, D. S. (2019). The importance of the whole: topological data analysis for the network neuroscientist. *Network Neuroscience*, *3*(3), 656–673.

Su, Y., Wang, B., Cheng, F., Zhang, L., Zhang, X., & Pan, L. (2017). An algorithm based on positive and negative links for community detection in signed networks. *Scientific reports*, *7*(1), 1–12.

Su, Y., Wang, B., & Zhang, X. (2017). A seed-expanding method based on random walks for community detection in networks with ambiguous community structures. *Scientific reports*, *7*(1), 1–10.

Sun, R., Chen, C., Wang, X., Zhang, Y., & Wang, X. (2020). Stable community detection in signed social networks. *IEEE Transactions on Knowledge and Data Engineering*.

Sun, Y., Tang, J., Pan, L., & Li, J. (2015). Matrix based community evolution events detection in online social networks. In *IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity 2015)* (pp. 465–470).

Tabarzad, M. A., & Hamzeh, A. (2017). A heuristic local community detection method (hlcd). *Applied Intelligence*, *46*(1), 62–78.

Takaffoli, M., Rabbany, R., & Zaïane, O. R. (2013). Incremental local community identification in dynamic social networks. In *IEEE/ACM International Conference on Advances in Social*

Networks Analysis and Mining (ASONAM 2013) (pp. 90–94).

Tang, J., Chang, Y., Aggarwal, C., & Liu, H. (2016). A survey of signed network mining in social media. *ACM Computing Surveys (CSUR)*, *49*(3), 1–37.

Tang, Z., Tang, Y., Li, C., Cao, J., Chen, G., & Lin, R. (2021). A fast local community detection algorithm in complex networks. *World Wide Web*, *24*(6), 1929–1955.

Thomas, M., Pang, B., & Lee, L. (2006). Get the Out Vote: Determining Support or Opposition from Congressional Floor-Debate Transcripts. In *Proc. Conf. on Empir. Methods in Nat. Lang. Process.* (pp. 327–335).

Tzeng, R.-C., Ordozgoiti, B., & Gionis, A. (2020). Discovering conflicting groups in signed networks. *Advances in Neural Information Processing Systems*, *33*, 10974–10985.

Van Laarhoven, T., & Marchiori, E. (2016). Local network community detection with continuous optimization of conductance and weighted kernel k-means. *The Journal of Machine Learning Research*, *17*(1), 5148–5175.

Wang, C., Tang, W., Sun, B., Fang, J., & Wang, Y. (2015). Review on community detection algorithms in social networks. In *IEEE International Conference on Progress in Informatics and Computing (PIC 2015)* (pp. 551–555).

Wang, F., Zhang, B., Chai, S., & Xia, Y. (2018). An extreme learning machine-based community detection algorithm in complex networks. *Complexity*.

Wang, Z., Li, Z., Yuan, G., Sun, Y., Rui, X., & Xiang, X. (2018). Tracking the evolution of overlapping communities in dynamic social networks. *Knowledge-Based Systems*, *157*, 81–97.

Whang, J. J., Gleich, D. F., & Dhillon, I. S. (2013). Overlapping community detection using seed set expansion. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management* (pp. 2099–2108).

Wu, L., Bai, T., Wang, Z., Wang, L., Hu, Y., & Ji, J. (2013). A new community detection algorithm based on distance centrality. In *The 10th International Conference on Fuzzy Systems and*

Knowledge Discovery (FSKD 2013) (pp. 898–902).

Xia, C., Luo, Y., Wang, L., & Li, H.-J. (2021). A fast community detection algorithm based on reconstructing signed networks. *IEEE Systems Journal*, *16*(1), 614–625.

Xiao, H., Ordozgoiti, B., & Gionis, A. (2020). Searching for polarization in signed graphs: a local spectral approach. In *Proceedings of The Web Conference* (pp. 362–372).

Xie, J., Chen, M., & Szymanski, B. K. (2013). Labelrankt: Incremental community detection in dynamic networks via label propagation. In *Proceedings of the workshop on dynamic networks management and mining* (pp. 25–32).

Yang, B., Cheung, W., & Liu, J. (2007). Community mining from signed social networks. *IEEE transactions on knowledge and data engineering*, *19*(10), 1333–1348.

Yang, J., McAuley, J., & Leskovec, J. (2013). Community detection in networks with node attributes. In *IEEE 13th International Conference on Data Mining* (pp. 1151–1156).

Yuan, L. (2014). Research on community detection and graph partitioning.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of anthropological research*, *33*(4), 452–473.

Zhang, T., & Wu, B. (2012). A method for local community detection by finding core nodes. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (pp. 1171–1176).

Zhang, X.-K., Ren, J., Song, C., Jia, J., & Zhang, Q. (2017). Label propagation algorithm for community detection based on node importance and label influence. *Physics Letters A*, *381*(33), 2691–2698.

Zhang, Y., Wu, B., Liu, Y., & Lv, J. (2019). Local community detection based on network motifs. *Tsinghua Science and Technology*, *24*(6), 716–727.

Zhen-Qing, Y., Ke, Z., Song-Nian, H., & Jun, Y. (2012). A new definition of modularity for community detection in complex networks. *Chinese Physics Letters*, *29*(9), 098901.

Zhuang, D., Chang, J. M., & Li, M. (2019). Dynamo: Dynamic community detection by incrementally maximizing modularity. *IEEE Transactions on Knowledge and Data Engineering*, *33*(5), 1934–1945.