COMMUNITY STRUCTURE AND ITS APPLICATIONS IN DYNAMIC COMPLEX NETWORKS

By

NAM P. NGUYEN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2013

ACKNOWLEDGMENTS

I would like express the deepest appreciation to my committee chair, Professor My T. Thai for always being a great advisor of my Ph.D. journey. She continually and convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Her wisdom, support and advices have guided me through all of my difficult moments, not only in doing research but also in my life. Without her guidance and persistent help this dissertation would not have been possible. Also, I am graceful to have excellent labmates who have provided extremely helpful resources to my study.

I would like to thank my committee members, Professor Sanjay Ranka, Professor Panos Pardalos, Professor Tamer Kahveci and Professor Prabhat Mishra who have been very supportive to my dissertation. Their encouragement and advices have helped me a lot not only in my Ph.D. study but also in my future career. Financial support for my Ph.D. program was provided by the University of Florida, NSF CAREER Award Grant number 0953284 and the DTRA Grant number HDTRA1-08-10.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

COMMUNITY STRUCTURE AND ITS APPLICATIONS
IN DYNAMIC COMPLEX NETWORKS

By

Nam P. Nguyen

May 2013

Chair: My T. Thai
Major: Computer Engineering

In this dissertation, we focus on analyzing and understanding the organizational principals, assessing the structural vulnerability as well as exploring practical applications of dynamic complex networks. In particular, we propose two adaptive frameworks for identifying the nonoverlapping and overlapping community structure in dynamic networks. Our approaches have not only the power of quickly and efficiently updating the network communities, but also the ability of tracing the evolution of those communities over time. We also suggest a detection method based on nonnegative matrix factorization which can work on weighted and directed networks. Consequently, we study the discovery of stable communities in the networks, i.e., communities which are tightly connected and remain wealthy even over a long period of time. Furthermore, we investigate on the structural vulnerability of the network community structure via identifying key nodes that play an important role in maintaining the normal function of the whole system. This is a new research direction on the cyber-infrastructure that we have recently introduced. To certify the effectiveness of our suggested frameworks and algorithms, we extensively test them on not only synthesized networks but also on real-world dynamic traces. Finally, we demonstrate the wide applicability of our algorithms via realistic applications, such as the limiting misinformation spread in Online Social Networks as well as the social-based forwarding and routing strategy and worm containment in Mobile networks.

CHAPTER 1
INTRODUCTION

1.1 Community Detection in Dynamic Complex Networks

Many complex systems in reality exhibit the property of containing community structure [1][2], i.e., they naturally divide into groups of vertices with denser connections inside each group and fewer connections crossing groups, where vertices and connections represent network users and their social interactions, respectively. Members in each community of a social network usually share things in common such as interests in photography, movies, music or discussion topics and thus, they tend to interact more frequently with each other than with members outside of their community. Community detection in a network is the gathering of network vertices into groups in such a way that nodes in each group are densely connected inside and sparser outside.

It is noteworthy to differentiate between community detection and graph clustering. These two problems share the same objective of partitioning network nodes into groups; however, the number of clusters is predefined or given as part of the input in graph clustering whereas the number of communities is typically unknown in community detection. Detecting communities in a network provides us meaningful insights to its internal structure as well as its organization principles. Furthermore, knowing the structure of network communities could also provide us more helpful points of view to some uncovered parts of the network, thus helps in preventing potential networking diseases such as virus or worm propagation. Studies on community detection on static networks can be found in an excellent survey [3] as well as in the work of [4][5][6][7] and references therein.

Real-world complex networks, however, are not always static. In fact, most of complex systems in reality (such as Facebook, Bebo and Twitter in OSNs) evolve and witness an expand in size and space as their users increase, thus lend themselves to the field of dynamic networks. A dynamic network is a special type of evolving complex

networks in which changes are frequently introduced over time. In the sense of an online social network, such as Facebook, Twitter or Flickr, changes are usually introduced by users joining in or withdrawing from one or more groups or communities, by friends and friends connecting together, or by new people making friend with each other. Any of these events seems to have a little effect to a local structure of the network on one hand; the dynamics of the network over a long period of time, on the other hand, may lead to a significant transformation of the network community structure, thus raises a natural need of reidentification. However, the rapidly and unpredictably changing topology of a dynamic social network makes it an extremely complicated yet challenging problem.

Although one can possibly run any of the static community detection methods, which are widely available [4][5][6][8], to find the new community structure whenever the network is updated, he may encounter some disadvantages that cannot be neglected: (1) the long running time of a specific static method on large networks (2) the trap of local optima and (3) the almost same reaction to a small change to some local part of the network. A better, much efficient and less time consuming way to accomplish this expensive task is to adaptively update the network communities from the previous known structures, which helps to avoid the hassle of recomputing from scratch. This adaptive approach is the main focus of our study in this paper. In Figure $1-1$, we briefly generalize the idea of dynamic network community structure adaptation. Here, the network evolves from time t to $t+1$ under the change ΔG_t. The adaptive algorithm $\mathcal{A}$ quickly finds the new community structure $\mathcal{C}(G_{t+1})$ based on the previous structure $\mathcal{C}(G_t)$ together with the changes ΔG_t.

1.2 Nonnegative Matrix Factorization for Community Detection

Community identification on complex networks is a well-established field and many efficient graph-based methods have been introduced in the literature (see [9] for an excellent survey). Unfortunately, these methods expose the strong dependence on some local parts of the network topology as well as the implicit meaning and interpretation

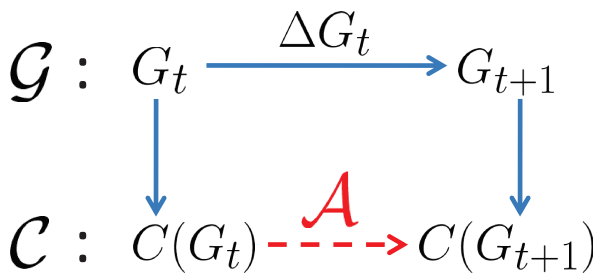


Figure 1-1. The general framework for our adaptive community detection algorithm $\mathcal{A}$.

from the detected overlapping communities. Recently, NMF-based algorithms for detecting network communities have gained great attention due to its meaningful interpretation [10]. In general, an NMF problem asks for, given a nonnegative matrix $X \in \mathbb{R}^{m\times m}$ and a number $k \ll \min\{m, n\}$, nonnegative matrices $W \in \mathbb{R}^{m\times k}$ and $H \in \mathbb{R}^{k\times n}$ such that $||X - WH||$ is minimized, where $||\cdot||$ is a cost function (usually the Frobenius distance or I-divergence). One notable property of NMF is its close relationship to K-mean clustering and graph partitioning [11][12], which also closely relates to community identification.

A few attempts have been suggested on this line of method. Lin et al proposed MetaFac [13], a NMF-based method for extracting community structure through relational hypergraphs. This method, however, is not capable for identifying overlapped structures. In [14], Prorakis et al. recently proposed an approach for finding overlapping communities using a Bayesian NMF based on hyperparameters. This method has the advantages of automatically determining the number of communities and not suffering from the resolution limit. Unfortunately, its built-in estimate of the number of communities could mislead the factorization to return a bad solution. In [15], Wang et al. proposed NMF methods on the Frobenius norm with the capability of extracting overlapped structures. However, we find these approaches do not appear to perform well on weighted directed networks as shown in the experiments.

To overcome the above limitations, we introduce two NMF approaches, namely iSNMF and iANMF, for effectively identifying social network communities with meaningful

interpretations. In particular, we are interested in approximating $X \approx HSH^{\top}$ since this factorization provides us H as the community indicator matrix and S as the community-interaction strength matrix, respectively. This factorization, as a result, nicely reflects the overlap of network communities and promises a meaningful community interpretation that is independent of the network topology.

1.3 Applications of The Network Community Structure

Detecting community structure of a dynamic social network is of considerable uses. To give a sense of it, consider the routing problem in communication network where nodes and links present people and mobile communications, respectively. Due to nodes mobility and unstable links properties of the network, designing an efficient routing scheme is extremely challenging. However, since people have a natural tendency to form groups of communication, there exist groups of nodes which are densely connected inside than outside in the underlying MANET as a reflection, and therefore, forms community structure in that MANET. An effective routing algorithm, as soon as it discovers the network community structure, can directly route or forward messages to nodes in the same (or to the related) community as the destination. By doing this way, we can avoid unnecessary messages forwarding through nodes in different communities, thus can lower down the number of duplicate messages as well as reduce the overhead information, which are essential in MANETs.

Another great example includes the worm containment in cellular networks [16], or in OSNs [17][18]. Nowadays, many social applications such as Facebook, Twitter and FourSquare, are able to run on open-API enabled mobile devices like PDAs and Iphones. However, if such an application is infected with malicious software, such as worms or viruses, this openness will also make it easier for their propagation. A possible solution to prevent worms from spreading out wider is to send patches to critical users and let them redistribute to the others. Intuitively, the smaller the set of important users for sending patches, the better. But how can we effectively choose that set of minimal

size? This is where community structure comes into the picture and helps. In particular, we show that selecting users in the boundaries of the overlapped nodes gives a tighter and more efficient set of influential users, thus significantly lowers the number of sent patches as well as overhead information, which are essential in cellular networks and OSNs.

1.4 The Identification of Stable Communities

OSNs in reality are highly dynamic as social interactions on them tend to come and go quickly. Consequently, their communities are also dynamical and evolve heavily as the networks change over time. However, Palla et al. observe in their seminal work that some communities in social networks are tightly connected and remain wealthy even over a long period of time [19]. The authors also point out that large-size communities with a high internal densities and less external distractions tend to remain stable during the network evolution, which intuitively agree with the findings reported in [20]. These observations reassemble the concept of stable communities in OSNs. For example, stable communities on Facebook can be visualized as groups of users who devoted themselves to one particular interest such as movie, music or photography. Likewise, a stable community in Twitter can be illustrated via a group of users who may follow many but only loyal to a specific celebrity. In a different perspective, stable communities in a citation network may refer to well-established research topics in the field whereas unstable communities may represent topical or recently arising research directions.

The discovery of these stable communities, as a consequence, will provide us valuable insights into the core properties and characteristics of not only each community but also of the network as a whole. This knowledge can further benefit information retrieval in OSNs as searches can be redirected to stable communities sharing the most similar characteristics to the queries for more meaningful answers. For instance, the search for well-established research topics in a citation network can be mined more effectively when one looks at its stable rather than unstable communities, as

discussed above. However, the large-scale and nonreciprocal topologies of OSNs in reality make the detection of stable community structure an extremely challenging yet topical problem.

1.5 The Assessment of Network Community Structure Vulnerability

Complex systems, despite their diversity in physical infrastructures and underlying interactions, expose to be extremely vulnerable under node attacks. In some scenarios, the failures of only a few key nodes are enough to bring the whole network operation down to its knees [21]. More importantly, this vulnerability can further be propagated to a wider population, leading to a much more devastating consequence. In order to develop a comprehensive understanding on this type of attack, it is therefore important to understand not only the impact of nodes' failures on the network components but also the inner and interdependency among those components [22]. Particularly, it is crucial to explore how the failure of a single node, or a set of nodes in general, can significantly change the structure of the network components as well as how these components would affect each other in cases of attacks. However, the large scale and dynamical properties of complex systems in practice make this a complicated problem.

To tackle this problem, we introduce the use of network modules to study both the impact of nodes's failures and the network component interdependency. There are several reasons and benefits behind this approach. First of all, investigating the interdependencies based on the topology of the underlying network structures is a major aspect that must be considered to understand the behavior of structural vulnerability [22]. Secondly, most complex networks commonly exhibit modular property, or in other words, they exhibit to contain community structure in their underlying organizations. That is, the network nodes can be gathered into groups in such a way that each group is densely connected internally and sparsely connected externally [23][24]. Nodes in each community usually share similar functions and characteristics that distinguish themselves from the others. In a broader view, communities displays the whole network

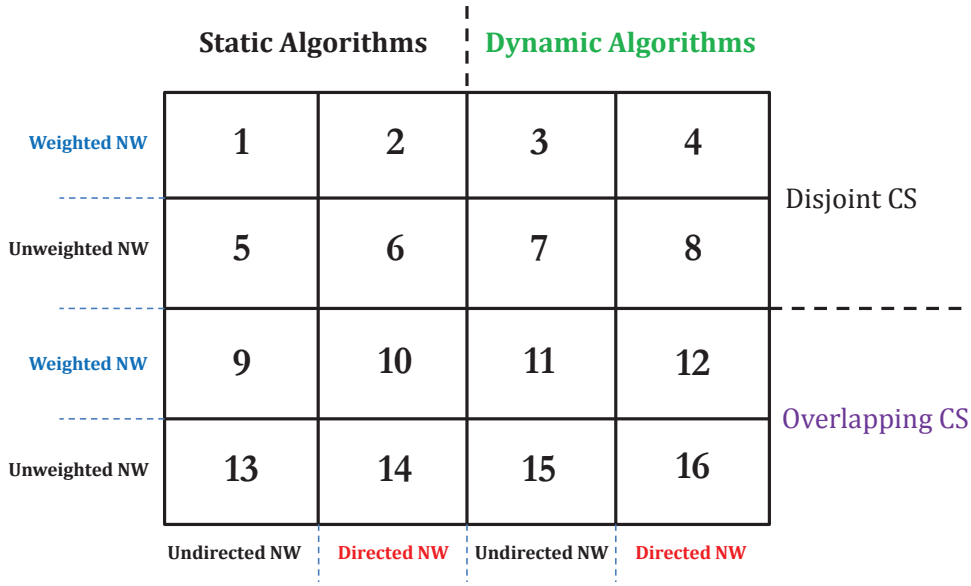


Figure 1-2. The classification of community detection algorithms in complex networks.

structure as a compact and more understandable level where a community may represent an entity or a functional group in the system. At this level, element failures in one community can have a profound impact which can consequently lead to changes of other communities. Therefore, identifying network elements that are essential to its community structure is a fundamental and extremely important issue. To the best of our knowledge, this research direction has not been addressed so far in the literature.

1.6 Literature Review

Community detection in dynamic networks

Community detection in complex networks is a well established field and a tremendous number of identification methods has been proposed in the literature. Some notable approach directions include classical graph clustering algorithms [25][26], dynamic approaches [27], modularity optimization methods [24], statistical inference [28] or random walk for community detection [29] (see [9] and references therein for an excellent survey).

In a general view, community structure detection algorithms for complex network can be classified in different ways: either by nonoverlapping or overlapping detection

algorithms, by static or dynamic algorithms, or by algorithms for directed and undirected networks, etc. Figure 1-2 describes a details classification of 16 different types of identification algorithms.

Community detection on static networks has attracted a lot of attentions and many efficient methods have been proposed for this type of networks. Detecting community structure on dynamic networks, however, has so far been an untrodden area. A recent work of Palla et al. [2] proposed an innovative method for detecting communities on dynamic networks based on the k-clique percolation technique. This approach can detect overlapping communities; however, it is time consuming, especially on large scale networks. Another recent work of Zhang et al. [30] proposed a detection method based on contradicting the network topology and the topology-based propinquity, where propinquity is the probability of a pair of nodes involved in a community. A work in [31] presented a parameter-free methodology for detecting clusters on time-evolving graphs based on mutual information and entropy functions of Information Theory. Hui et al. [32] proposed a distributed method for community detection in which modularity was used as a measure instead of objective function. A part from that, [33] attempted to track the evolving of communities over time, using a few static network snapshots.

In [34], the authors present a framework for identifying dynamic communities with a constant factor approximation. However, this method does not seem to make sense on real-world social networks since it requires some predefined penalty costs which are generally unknown on dynamic networks. A recent work [35], Thang et al. proposed a social-aware routing strategy in MANETs which also makes uses of a modularity-based procedure name MIEN for quickly updating the network structure. In particular, MIEN tries to compose and decompose network modules in order to keep up with the changes and uses fast modularity algorithm [4] to update the network modules. However, this method performs slowly on large scale dynamic networks due to the high complexity of [4].

In [36], Lin et al. proposed FacetNet, a framework for analyzing communities in dynamic networks based on the optimization of snapshot costs. FacetNet is guaranteed to converge to a local optimal solution; however, its convergence speed is slow and its input asks for the number of network communities which are usually unknown in practice. In [37], Duan et al. proposed Stream-Group, an incremental method to solve the community mining and detect the change points in weighted dynamic graphs. This method is modularity-based thus may inherit the resolution limit while discovering network communities. In another attempt, Kim et al. [38] suggested a particle-and-density based clustering method for dynamic networks, based on the extended modularity and the concepts of nano-community and /-quasi-clique-by-clique. Apart from that, the work of Cazabet et al. [39] proposed iLCD method to find the overlapping network communities by adding edges and then merging similar ones. However, this model might not be sufficient in consideration with the dynamic behaviors of the network when new nodes are introduced or removed, or when existing edges are removed from the network. In [40], the author presented OSLOM, a framework for testing the statistical significance of a cluster with respect to a global null model (e.g., a random graph). To expand a community, OSLOM locally computes the value r for each neighbor node and tries to include that node into the current community.

Nonnegative matrix factorization for community detection

Community detection on complex networks is a mature research area and besides NMF-based algorithms, many effective graph-based or topology-based algorithms have been proposed for this purpose. In general, detection methods can be classified into non-overlapping (disjoint) and overlapping algorithms. Traditional non-overlapping algorithms [24][8][41] may return good community identification results, however are not able to reveal the overlapped network structures, particularly on social networks. On the other category, algorithms for graph-based and topological-based detection of overlapping communities have also been proposed in the literature. Most of them

are based on the clique-percolation [42] or clique extension [43] techniques, on the extended modularity [44][45], on a specific fitness function [46], on label propagation [47], or link-based technique [48]. See [9] and references therein for an excellent survey on those detection methods.

Although the success of these aforementioned algorithms have been theoretically and empirically verified, they still expose the following limitations: (1) The strong dependence on some local parts of the network topology, e.g., the clique-percolation method depends on some dense subnetworks in order to percolate, a link-based technique relies on potential links with highest degrees, a modularity-based technique depends on the network hierarchy in order to maximize modularity, etc, and (2) The implicit meaning and interpretation from the detected overlapping communities, e.g., what is the contribution of an overlapped node to these percolated-cliques or why would it even be there? These shortcomings of these methods drive the need for a better approach with a more meaningful interpretation.

Stable community detection

The discovery of stable communities, on the contrary, is still an untrodden area with only a few attempts has been suggested [49][50][51]. This special property of network communities was perhaps first observed by Palla et al. in his seminal work [19], where they point out that tight-knit communities with high internal densities and less external distractions tend to remain strong over time, thereby reassembles the concept of community stability. Delvenne et al. [49] extend this general concept to proposed an measure, called "stability of the clustering $r(t, H)$", to quantify how stable a given cluster (or community structure) H is at a specific time step t based on the Markov Autocovariance model. Under this notation, a cluster H is stable at time t if a high value of $r(t, H)$ is observed. This quantity, instead, is more appropriate for verification rather than identification of stable network communities since it requires the specification of time step t a prior.

In a different approach, Lancichinetti et al. [50] investigate on the consensus of community detection methods. The authors report that, given a particular algorithm A, the consensus on communities found by A after multiple runs dramatically improve the quality of the detection, henceforth suggest that those communities are candidates for stable structures. This is a very interesting approach, however, might encounter some disadvantages of (1) the expensive computational cost and time consuming, and (2) the convergence of the whole iterative process is not guaranteed. In a recent attempt, Yanhua et al. [51] utilize the concept of mutual links and suggest an spectral-clustering-based identification method that tries to maximize the total mutual connections in order to find stable communities. However, there are possibilities that some mutual links are of low magnitudes, and thus, do not significantly contribute to the overall stability at the community level.

Structural vulnerability assessment of community structure

Community structure and complex network vulnerability are the two major and well-developed areas of networking research. Surveys on community structure detection algorithms as well as methods for assessing network vulnerabilities can be found in the work of Fortunatos et. al. [9], and Grubesic et. al. [52], respectively. However, assessing the vulnerability of network community structure has so far been an untrodden area. A large body of work has been devoted to find the node roles within a community by a link-based technique together with a modification of node degree [53], by using the spectrum of the graph [54], by using a within-module degree and their participation coefficient [55], or by the detection of key nodes, overlapping communities and “date” and “party” hubs [56]. However, none of these approaches discuss how the community structure would change in the failure of those important nodes, especially in terms of NMI measure.

The vulnerability of network function and structure has been examined under the node centrality metrics, such as high degree and betweeness centrality, as well as

under the average shortest path which tries to signify the lengths of shortest distances between node pairs [52], under the pairwise connectivity metric whose goal aims to break the network's pairwise connectivity down to a certain level [21], or under the available number of compromised $s - t$ flows [57], etc. However, there is an even more crucial risk that could dramatically affect the normal network functionality that has not been addressed so far: the transformation or restruction of the network community structure. Due to its vital role in the network, any significant restruction or transformation of the community structure, resulted from important node removal, can potentially change the entire network organization and consequently lead to a malfunction or unpredictable corruption of the whole network.

1.7 Dissertation outline

In chapter 2, we propose QCA, a fast and adaptive method for efficiently identifying the nonoverlapping community structure of a dynamic social network. Our approach takes into account the discovered structures and processes on network changes only, thus significantly reduces computational cost and processing time. We study the dynamics of a social network and prove theoretical results regarding its communities' behaviors over time, which are the bases of our method. We extensively evaluate our algorithms on both synthesized and real dynamic social traces. Experimental results show that QCA achieves not only competitive modularity scores but also high quality community structures in a timely manner. We apply QCA method to worm containment problem in OSNs. Simulation results show that QCA outperforms current available methods and confirm its applicability in social network problems.

In chapter 3, we suggest AFOCS, a two-phase adaptive framework for not only detecting and updating the overlapping network communities but also tracing their evolution over time. Theoretical analyses show AFOCS partially achieves more than 74% the internal density of the optimal solution. Second, we evaluate AFOCS on both synthesized and real traces in comparison to both the state-of-the-art and the

most popular static detection methods COPRA and CFinder, as well as to recent adaptive methods FacetNet, iLCD and OSLOM. Empirical results show that AFOCS achieves both competitively results and high quality community structures in a timely manner. Finally, with AFOCS, we suggest a community based forwarding strategy for communication networks that reduces up to 11x overhead information while maintaining competitively delivery time and ratio. We also propose a new social-aware patching scheme for containing worms in OSNs, which helps reducing up to 7x the infection rates on Facebook dataset.

We analyze two NMF approaches in chapter 4, namely iSNMF and iANMF, for effectively identifying social network communities with meaningful interpretations. In particular, we are interested in approximating $X \approx HSH^{\top}$ since this factorization provides us H as the foundation feature matrix and S as the feature interaction matrix. Alternatively, H and S can also be thought of as community indicator and inter-community strength matrices whose row elements can further be interpreted as probabilities of nodes belonging to different communities. This factorization, as a result, nicely reflects the overlap of network communities and promises a meaningful community interpretation that is independent of the network topology.

In an application perspective, we illustrate the practical applications of the network community structure via two emerging problems on social and mobile computing, namely the Worm spread containment problem on online social networks (chapter 5) and the forwarding and routing strategy (chapter 6) on mobile networks. We demonstrate that methods and strategies employing QCA and FOCS as community detection cores obtain a significant improvement in term of performance and solution quality. These realistic applications brighten the wide applicability of the network community structure many problems enabled my complex networks.

In chapter 7, we suggest an estimation which provides helpful insights into the stability of links in the input network. Based on that, we propose SCD - a framework

to identify community structure in directional OSNs with the advantage of community stability. We next explore an essential connection between the persistence probability of a community at the stationary distribution and its local topology, which is the fundamental mathematical theory to support the SCD framework. To certify the efficiency of our approach, we extensively test SCD on both synthesized datasets with embedded communities and real-world social traces, including NetHEPT and NetHEPT_WC collaboration networks as well as Facebook social networks, in reference to the consensus of other state-of-the-art detection methods. Highly competitive empirical results confirm the quality and efficiency of SCD on identifying stable communities in OSNs.

In chapter 8, we introduce CSV problem to assess the impact of nodes' failures on the network community structure. To the best of our knowledge, this is the first attempt in this line of research. We analyze possible conditions that can lead to the minimization of NMI on network community structures. We suggest the concept of generating edges of a community and provide an optimal solution for finding a MGES. We propose genEdge, a node selection strategy for CSV based on the MGES solution. We conducted experiments on both synthesized data with known community structures and real world traces. Empirical results reveal that genEdge outperforms other node selection strategies in terms of solution quality as well as in reference to different underlying community detection algorithms. In an application perspective, we demonstrate the critical importance of CSV via the forwarding and routing strategies in delay tolerant networks (DTNs), where the failures of some important devices significantly degrade the entire system's performance.

Finally, we summary our contributions and conclude the dissertation in chapter 9.

CHAPTER 2
NONOVERLAPPING COMMUNITY STRUCTURE DETECTION

In this chapter, we present QCA, our proposed algorithms for detecting nonoverlapping community structure in a dynamic complex network. In the following sections, we first introduce the preliminaries in section 2.1 and then describe our QCA method in detail in section 2.2. Finally, the empirical evaluations of QCA on both synthesized and real datasets are presented in section 2.3.

2.1 Problem Definition

We first present the notations, objective function as well as the dynamic graph model representing a social network that we will use throughout this section.

(Notation) Let $G = (V, E)$ be an undirected unweighted graph with N nodes and M links representing a social network. Let $\mathcal{C} = \{C_1, C_2, .., C_k\}$ denote a collection of disjoint communities, where $C_i \in \mathcal{C}$ is a community of G. For each vertex u, denote by d_u, $C(u)$ and $NC(u)$ its degree, the community containing u and the set of its adjacent communities. Furthermore, for any $S \subseteq V$, let m_S, d_S and e_S^u be the number of links inside S, the total degree of vertices in S and the number of connections from u to S, respectively. The pairs of terms community and module; node and vertex as well as edge and link and are used interchangeably.

(Dynamic social network) Let $G^s = (V^s, E^s)$ be a time dependent network snapshot recorded at time s. Denote by ΔV^s and ΔE^s the sets of vertices and links to be introduced (or removed) at time s and let $\Delta G^s = (\Delta V^s, \Delta E^s)$ denote the change in term of the whole network. The next network snapshot G^{s+1} is the current one together with changes, i.e., $G^{s+1} = G^s \cup \Delta G^s$. A dynamic network $\mathcal{G}$ is a sequence of network snapshots evolving over time: $\mathcal{G} = (G^0, G^1, .., G^s)$.

(Objective function) In order to quantify the goodness of a network community structure, we take into account the most widely accepted measure called modularity Q

[6], which is defined as:

$$Q = \sum_{C \in \mathcal{C}} \left(\frac{m_C}{M} - \frac{d_C^2}{4M^2}\right).$$

Basically, Q is the fraction of all links within communities subtracts the expected value of the same quantity in a graph whose nodes have the same degrees but links are distributed randomly, and the higher modularity Q, the better network community structure is. Therefore, our objective is to find a community assignment for each vertex in the network such that Q is maximized. Modularity, just like other quality measurements for community identifications, has some certain disadvantages such as its non-locality and scaling behavior [7], or resolution limit [58]. However, it is still very well considered due to its robustness and usefulness that closely agree with intuition on a wide range of real world networks.

Problem Definition: Given a dynamic social network $\mathcal{G} = (G^0, G^1, .., G^s)$ where G^0 is the original network and G^1, G^2,.., G^s are the network snapshots obtained through ΔG^1, ΔG^2,.., ΔG^s, we need to devise an adaptive algorithm to efficiently detect and identify the network community structure at any time point utilizing the information from the previous snapshots as well as tracing the evolution of the network community structure.

2.2 Algorithm Description

Let us first discuss how changes to the evolving network topology affect the structure of its communities. We use the term intra-community links to denote edges whose two endpoints belong to the same community, and the term inter-community links to denote those with endpoints connecting different communities. For each community C, the connections linking C with other communities are much fewer than those within C itself, i.e., nodes in C are densely connected inside and less densely connected outside. Intuitively, adding intra-community links inside or removing inter-community links between communities of G will strengthen those communities and make the structure of G more clear. Vice versa, removing intra-community links and inserting inter-community links will loosen the structure of G. The community updating process, as a result, is

challenging since an insignificant change in the network topology can possibly lead to an unexpected transformation of its community structure.

We will discuss in detail possible behaviors of dynamic network communities in Figure 2-1. 2-1A: New edge (u, v): u and v are first checked and memberships are then tested on X and Y. 2-1B: (a) The original community (b) After the dotted edge is removed, two smaller communities arise. 2-1C: (a) The original four communities (b) After the central node is removed, the leftover nodes join in different modules, forming three new communities. 2-1D: (a) The original community (b) When g is removed, a 3-clique is placed at a to discover b, c, d and e. f assigned singleton afterwards.

In order to reflect changes introduced to the social network, its underlying graph is constantly updated by either inserting or removing a node or a set of nodes, or by either introducing or deleting an edge or a set of edges. In fact, the introduction or removal of a set of nodes (or edges) can be decomposed as a sequence of node (or edge) insertions (or removals), in which a single node (or a single edge) is introduced (or removed) at a time. This observation helps us to treat network changes as a collection of simple events where a simple event can be one of newNode, removeNode, newEdge, removeEdge whose details are as follow:

- newNode $(V \cup \{u\})$: A new node u with its associated edges are introduced. u could come with no or more than one new edge(s).
- removeNode $(V \backslash \{u\})$: A node u and its adjacent edges are removed from the network.
- newEdge $(E \cup \{e\})$: A new edge e connecting two existing nodes is introduced.
- removeEdge $(E \backslash \{e\})$: An existing edge e in the network is removed.

Our approach first requires an initial community structure $\mathcal{C}_0$, which we call the basic structure, in order to process further. Since the input model is restricted as an undirected unweighted network, this initial community structure can be obtained by performing any of the available static community detection methods [4][5][8]. To obtain

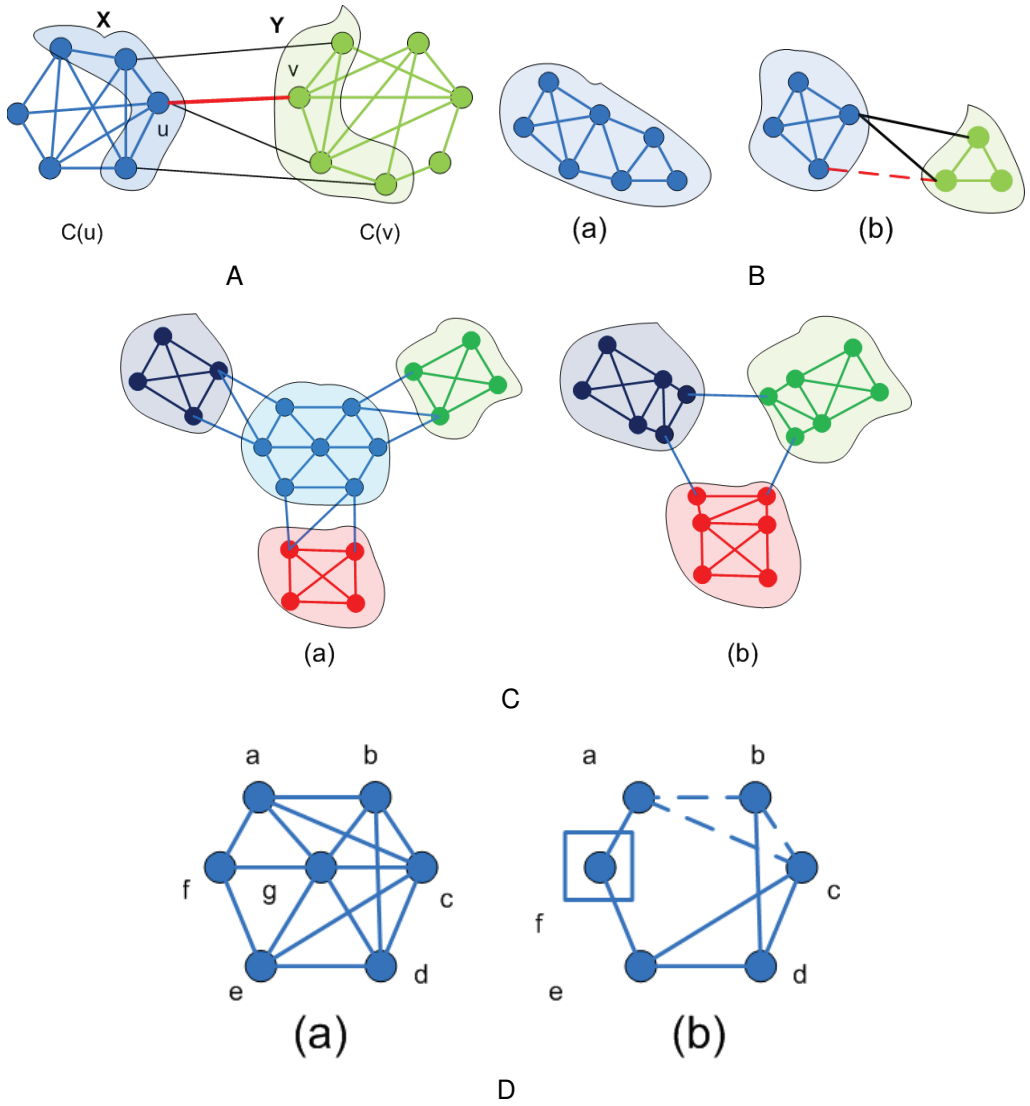


Figure 2-1. Possible behaviors of the network community structure during evolution.

a good basic structure, we choose the method proposed by Blondel et al. in [5] which produces a good network community structure in a timely manner [3].

2.2.1 New Node

Let us consider the first case when a new node u and its associated connections are introduced. Note that u may come with no adjacent edges or with many of them connecting one or more communities. If u has no adjacent edge, we create a new community for it and leave the current structure intact. The interesting case happens, and it usually does, when u comes with edges connecting one or more existing

communities. In this latter situation, we need to determine which community u should join in in order to maximize the gained modularity. There are several local methods introduced for this task, for instance the algorithms of [4][8]. Our method is inspired by a physical approach proposed in [59], in which each node is influenced by two forces: F_{in}^{C} (to keep u stays inside community C) and F_{out}^{C} (the force a community C makes in order to bring u to C) defined as follow:

$$F_{in}^{C}(u) = e_{C}^{u} - \frac{d_u(d_C - d_u)}{2M},$$

and

$$F_{out}^{S}(u) = \max_{S \in NC(u)} \{ e_{S}^{u} - \frac{d_u d_{outS}}{2M} \},$$

where d_{outS} is of opposite meaning of d_S.

Taking into account the above two forces, a node v can actively determines its best community membership by computing those forces and either lets itself join the community S having the highest $F_{out}^{S}(v)$ (if $F_{out}^{S}(v) > F_{in}^{C(v)}(v)$) or stays in the current community $C(v)$ otherwise. By Theorem 2.1, we bridge the connection between those forces and the objective function, i.e., joining the new node in the community with the highest outer force will maximize the local gained modularity. The process is presented in Alg. 1.

Theorem 2.1. *Let C be the community having the maximum $F_{out}^{C}(u)$ when a new node u with degree p is added to G, then joining u in C gives the maximal gained modularity.*

Proof. Let D be a community of G and $D \neq C$, we show that joining u in D contributes less modularity than joining u in C. The overall modularity Q when u joins in C is

$$Q_1 = \frac{m_C + e_C^u}{M + p} - \frac{(d_C + e_C^u + p)^2}{4(M + p)^2} + \frac{m_D}{M + p} - \frac{(d_D + e_D^u)^2}{4(M + p)^2} + A,$$

Algorithm 1 New_Node

Input: New node u with associated links; Current structure $\mathcal{C}_t$.
Output: An updated structure $\mathcal{C}_{t+1}$

Create a new community of only u;
for $v \in N(u)$ **do**
    Let v determine its best community;
end for
for $C \in NC(u)$ **do**
    Find $F_{out}^{C}(u)$;
end for
if $\max_C F_{out}^{C}(u) > F_{in}^{C_u}(u)$ **then**
    Let $C_u \leftarrow \arg\max_C \{F_{out}^{C}(u)\}$;
    Update $\mathcal{C}_{t+1}$: $\mathcal{C}_{t+1} \leftarrow \big(\mathcal{C}_t \backslash C_u\big) \cup \big(C_u \cup u\big)$;
end if

where A is the summation of other modularity contributions. Similarly, joining u to D gives

$$Q_2 = \frac{m_C}{M+p} - \frac{(d_C + e_C^u)^2}{4(M+p)^2} + \frac{m_D + e_D^u}{M+p} - \frac{(d_D + e_D^u + p)^2}{4(M+p)^2} + A,$$

and

$$Q_1 - Q_2 = \frac{1}{M+p}\big(e_C^u - e_D^u + \frac{p(d_D - d_C + e_D^u - e_C^u)}{2(M+p)}\big).$$

Now, since C is the community that gives the maximum $F_{out}^{C}(u)$, we obtain

$$e_C^u - \frac{p(d_C + e_C^u)}{2(M+p)} > e_D^u - \frac{p(d_D + e_D^u)}{2(M+p)},$$

which implies

$$e_C^u - e_D^u + \frac{p(d_D - d_C + e_D^u - e_C^u)}{2(M+p)} > 0.$$

Hence, $Q_1 - Q_2 > 0$ and thus the conclusion follows. □

2.2.2 New Edge

When a new edge $e = (u, v)$ connecting two existing vertices u, v is introduced, we divide it further into two subcases: e is an intra-community link (totally inside a community C) or an inter-community link (connects two communities $C(u)$ and $C(v)$). If e is inside a community C, its presence will strengthen the inner structure of C

according to Lemma 1. Furthermore, by Lemma 2, we know that adding e should not split the current community C into smaller modules. Therefore, we leave the current network structure intact in this case.

The interesting situation occurs when e is a link connecting communities $C(u)$ and $C(v)$ since its presence could possibly make u (or v) leave its current module and join in the new community. Additionally, if u (or v) decides to change its membership, it can advertise its new community to all its neighbors and some of them might eventually want to change their memberships as a consequence. By Lemma 3, we show that should u (or v) ever change its community assignment, $C(v)$ (or $C(u)$) is the best new community for it. But how can we quickly decide whether u (or v) should change its membership in order to form a better community structure with higher modularity? To this end, we provide a criterion to test for membership changing of u and v in Theorem 2.2. Here, if both $\Delta q_{u,C,D}$ and $\Delta q_{v,C,D}$ fail to satisfy the criteria, we can safely preserve the current network community structure (Corollary 1). Otherwise, we move u (or v) to its new community and consequently let its neighbors determine their best modules to join in, using local search and swapping to maximize gained modularity. Figure 2-1A describes the procedure for this latter case. The detailed algorithm is described in Alg. 2.

Lemma 1. *For any $C \in \mathcal{C}$, if $d_C \leq M - 1$ then adding an edge within C will increase its modularity contribution.*

Proof. The portion Q_1 that community C contributes to the overall modularity Q is

$$Q_{1C} = \frac{m_C}{M} - \frac{d_C^2}{4M^2}.$$

When a new edge coming in, the new modularity Q_2 is

$$Q_{2C} = \frac{m_C + 1}{M + 1} - \frac{(d_C + 2)^2}{4(M + 1)^2}.$$

Algorithm 2 New_Edge

Input: Edge $\{u, v\}$ to be added; Current structure $\mathcal{C}_t$.
Output: An updated structure $\mathcal{C}_{t+1}$.

if (u and $v \notin V$) **then**
    $\mathcal{C}_{t+1} \leftarrow \mathcal{C}_t \cup \{u, v\}$;
else if $C(u) \neq C(v)$ **then**
    if $\Delta q_{u,C(u),C(v)} < 0$ and $\Delta q_{v,C(u),C(v)} < 0$ **then**
        return $\mathcal{C}_{t+1} \equiv \mathcal{C}_t$;
    else
        $w = \arg\max\{\Delta q_{u,C(u),C(v)}, \Delta q_{v,C(u),C(v)}\}$;
        Move w to the new community;
        for $t \in N(w)$ **do**
            Let t determine its best community;
        end for
        Update C_{t+1};
    end if
end if

Now, taking the difference between Q_2 and Q_1 gives

$$\begin{aligned}
\Delta Q_C &= Q_{2C} - Q_{1C} \\
&= \frac{4M^3 - 4m_C M^2 - 4d_C M^2 - 4m_C M + 2d_C^2 M + d_C^2}{4(M+1)^2 M^2} \\
&\geq \frac{4M^3 - 6d_C M^2 - 2d_C M + 2d_C^2 M + d_C^2}{4(M+1)^2 M^2} \quad (\text{since } m_C \leq \tfrac{d_C}{2}) \\
&\geq \frac{(2M^2 - 2d_C M - d_C)(2M - d_C)}{4(M+1)^2 M^2} \geq 0
\end{aligned}$$

The last inequality holds since $d_C \leq M - 1$ implies $2M^2 - 2d_C M - d_C \geq 0$. □

Lemma 2. *If C is a community in the current snapshot of G, then adding any intra-community link to C should not split it into smaller modules.*

Proof. Assume the contradiction, i.e, C should be divided into smaller modules when an edge is added into it. Let $X_1, X_2, .., X_k$ be disjoint subsets of C representing these modules. Let d_i and e_{ij} be the total degree of vertices inside X_i and the number of links going from X_i to X_j, respectedly. Assume that, W.L.O.G., when an edge is added inside

C, it is added to X_1. We will show that

$$\frac{\sum_{i\neq j} d_i d_j}{2M} < \sum_{i\neq j} e_{ij} < \frac{\sum_{i\neq j} d_i d_j}{2M} + 1,$$

which can not happen since $\sum_{i\neq j} e_{ij}$ is an natural number. Recall that

$$Q_{1C} = \frac{m_C}{M} - \frac{d_C^2}{4M^2},$$

and

$$Q_{X_i} = \frac{m_i}{M} - \frac{d_i^2}{4M^2},$$

and prior to adding an edge to C, we have

$$Q_{1C} > \sum_{i=1}^{k} Q_{X_i},$$

or equivalently,

$$\frac{m_C}{M} - \frac{d_C^2}{4M^2} > \sum_{i=1}^{k} \left(\frac{m_i}{M} - \frac{d_i^2}{4M^2}\right).$$

Since $X_1, X_2, .., X_k$ are disjoint subsets of C, it follows that

$$d_C = \sum_{i=1}^{k} d_i$$

and

$$m_C = \sum_{i=1}^{k} m_i + \sum_{i<j} e_{ij},$$

(where m_i is the number of links inside X_i). The above inequality equals to

$$\frac{m_C}{M} - \sum_{i=1}^{k} \frac{m_i}{M} > \frac{d_C^2}{4M^2} - \sum_{i=1}^{k} \frac{d_i^2}{4M^2},$$

or

$$\sum_{i<j} e_{ij} > \left\lceil \frac{\sum_{i<j} d_i d_j}{2M} \right\rceil.$$

Now, assume that the new edge is added to X_1 and C is split into $X_1, X_2, .., X_k$ which implies that dividing C into k smaller communities will increase the overall modularity,

i.e.,

$$Q_{2C} < \sum_{i=1}^{k} Q_{2X_i}.$$

Now,

$$\begin{aligned}
&Q_{2C} < \sum_{i=1}^{k} Q_{2X_i} \\
&\Leftrightarrow \frac{\sum_{i=1}^{k} m_i + \sum_{i<j} e_{ij} + 1}{M+1} - \frac{\left(\sum_{i=1}^{k} d_i + 2\right)^2}{4(M+1^2)} < \frac{m_1+1}{M+1} - \frac{(d_1+2)^2}{4(M+1)^2} + \sum_{i=2}^{k} \left(\frac{m_i}{M+1} - \frac{d_i^2}{4(M+1)^2}\right) \\
&\Leftrightarrow \quad \frac{\sum_{i=1}^{k} m_i + \sum_{i<j} e_{ij} + 1}{M+1} - \frac{\left(\sum_{i=1}^{k} d_i + 2\right)^2}{4(M+1^2)} < \frac{\sum_{i=1}^{k} m_i + 1}{M+1} - \frac{(d_1+2)^2}{4(M+1)^2} - \sum_{i=2}^{k} \frac{d_i^2}{4(M+1)^2} \\
&\Leftrightarrow \quad \sum_{i<j} e_{ij} < \frac{\sum_{i=1}^{k} d_i - 2d_1 + \sum_{i<j} d_i d_j}{2(M+1)}
\end{aligned}$$

Moreover, since it is obvious that $\sum_{i=1}^{k} d_i - 2d_1 < 2M$, we have

$$\frac{\sum_{i=1}^{k} d_i - 2d_1 + \sum_{i\neq j} d_i d_j}{2(M+1)} < \left\lceil \frac{\sum_{i<j} d_i d_j}{2M} \right\rceil + 1,$$

and thus the conclusion follows. □

Lemma 3. *When a new edge (u, v) connecting communities $C(u)$ and $C(v)$ is introduced, $C(v)$ (or $C(u)$) is the best candidate for u (or v) if it should ever change its membership.*

Proof. Let $C \equiv C(u)$ and $D \equiv C(v)$. Recall the outer force that a community S applies to vertex u is

$$F_{out}^{S}(u) = e_u^S - \frac{d_u d_{outS}}{2M}.$$

We will show that the presence of edge (u, v) will strengthen $F_{out}^{D}(u)$ while weakening the other outer forces $F_{out}^{S}(u)$, i.e, we show that $F_{out}^{D}(u)$ increases while $F_{out}^{S}(u)$

decreases for all $S \notin \{C, D\}$.

$$\begin{aligned} F^D_{out}(u)_{new} - F^D_{out}(u)_{old} &= \big(e^D_u + 1 - \frac{(d_u+1)(d_{outD}+1)}{2(M+1)}\big) - \big(e^D_u - \frac{d_u d_{outD}}{2M}\big) \\ &= \frac{2M + d_u d_{outD}}{2M} - \frac{d_u d_{outD} + d_{outD} + d_u + 1}{2(M+1)} \\ &\geq \frac{2M + d_u d_{outD}}{2(M+1)} - \frac{d_u d_{outD} + d_{outD} + d_u + 1}{2(M+1)} > 0 \end{aligned}$$

and thus $F^D_{out}(u)$ is strengthened when (u, v) is introduced. Furthermore, for any community $S \in \mathcal{C}$ and $S \notin \{C, D\}$, we have

$$\begin{aligned} F^S_{out}(u)_{new} - F^S_{out}(u)_{old} &= \big(e^S_u - \frac{(d_u+1)d_{outS}}{2(M+1)}\big) - \big(e^S_u - \frac{d_u d_{outS}}{2M}\big) \\ &= d_{outS}\big(\frac{d_u}{2M} - \frac{d_u+1}{2(M+1)}\big) < 0 \end{aligned}$$

which implies $F^S_{out}(u)$ is weakened when (u, v) is connected. Hence, the conclusion follows. □

Theorem 2.2. *Assume that a new edge (u, v) is added to the network. Let $C \equiv C(u)$ and $D \equiv C(v)$. If*

$$\Delta q_{u,C,D} \equiv 4(M+1)(e^u_D + 1 - e^u_C) + e^u_C(2d_D - 2d_u - e^u_C) - 2(d_u+1)(d_u + 1 + d_D - d_C) > 0$$

then joining u to D will increase the overall modularity.

Proof. Node u should leave its current community C and join in D if

$$Q_{D+u} + Q_{C-u} > Q_C + Q_D,$$

or equivalently,

$$\begin{aligned} &\frac{m_D + e_D + 1}{M+1} - \frac{(d_D + d_u + 2)^2}{4(M+1)^2} + \frac{m_C - e_C}{M+1} - \frac{(d_C - d_u - e_C)^2}{4(M+1)^2} \\ &\qquad > \frac{m_D}{M+1} - \frac{(d_D+1)^2}{4(M+1)^2} + \frac{m_C}{M+1} - \frac{(d_C+1)^2}{4(M+1)^2} \end{aligned}$$

The above equation equals to

$$4(M+1)(e_D+1-e_C)+e_C(2d_D-2d_u-e_C)-2(d_u+1)(d_u+1+d_D-d_C)>0,$$

which concludes the Theorem. □

Corollary 1. *If the condition in Theorem 2.2 is not satisfied, then neither u nor its neighbors should be moved to D.*

Proof. The proof follows from Theorem 2.2. □

2.2.3 Node Removal

When an existing node u in a community C is removed, all of its adjacent edges are disregarded as a result. This case is challenging in the sense that the resulting community is very complicated: it can be either unchanged or broken into smaller pieces and could probably be merged with other communities. Let's consider two extreme cases when a single degree node and a node with highest degree in a community is removed. If a single degree node is removed, it leaves the resulted community unchanged (Lemma 5). However, when a highest degree vertex is removed, the current community might be disconnected and broken in to smaller pieces which then are merged to other communities as depicted in Figure 2-1C. Therefore, identifying the leftover structure of C is a crucial part once a vertex in C is removed.

To quickly and efficiently handle this task, we utilize the clique percolation method presented in [2]. In particular, when a vertex u is removed from C, we place a 3-clique to one of its neighbors and let the clique percolate until no vertices in C are discovered (Figure 2-1D). We then let the remaining communities of C choose their best communities to merge in. The detailed algorithm is presented in Alg. 3.

2.2.4 Edge Removal

In the last case when an edge $e = (u, v)$ is removed, we divide further into four subcases (1) e is a single edge connecting only u and v (2) either u or v has

Algorithm 3 Node_Removal

Input: Node $u \in C$ to be removed; Current structure $\mathcal{C}_t$.
Output: An updated structure $\mathcal{C}_{t+1}$.

$i \leftarrow 1$;
while $N(u) \neq \emptyset$ **do**
    $S_i = \{$Nodes found by a 3-clique percolation on $v \in N(u)\}$;
    if $S_i == \emptyset$ **then**
        $S_i \leftarrow \{v\}$;
    end if
    $N(u) \leftarrow N(u) \backslash S_i$;
    $i \leftarrow i + 1$;
end while
Let each singleton in $N(u)$ consider its best communities;
Let each S_i consider its best communities as in [5]
Update $\mathcal{C}_t$;

degree one (3) e is an inter-community link connecting $C(u)$ and $C(v)$ and (4) e is an intra-community link. If e is an single edge, its removal will result in the same community structure plus two singletons of u and v themselves. The same reaction applies to the second subcase when either u or v has single degree due to Lemma 5, thus results in the prior network structure plus u (or v). When e is an inter-community link, the removal of e will strengthen the current network communities (Lemma 4) and hence, we just make no change to the overall network structure.

The last but most complicated case happens when an intra-community link is deleted. As depicted in Figure 2-1B, removing this kind of edge often leaves the community unchanged if the community itself is densely connected; however, the target module will be divided if it contains substructures which are less attractive or loosely connected to each other. Therefore, the problem of identifying the structure of the remaining modules is important. Theorem 2.3 provides us a convenient tool to test for community bi-division when an intra-community link is removed from the host community C. However, it requires an intensive look for all subsets of C, which may be time consuming when C is big. Note that prior to the removal of (u, v), the community C hosting this link should contain dense connections within itself and thus, the removal

of (u, v) should leave some sort of 'quasi-clique' structure [2] inside C. Therefore, we find all maximal quasi-cliques within the current community and have them (as well as leftover singletons) determine their best communities to join in. The detailed procedure is described in Alg. 4.

Algorithm 4 Edge_Removal

Input: Edge (u, v) to be removed; Current structure $\mathcal{C}_t$.
Output: An updated clustering $\mathcal{C}_{t+1}$.

if (u, v) is a single edge **then**
  $\mathcal{C}_{t+1} = (\mathcal{C}_t \backslash \{u, v\}) \cup \{u\} \cup \{v\}$;
else if Either u (or v) is of degree one **then**
  $\mathcal{C}_{t+1} = (\mathcal{C}_t \backslash C(u)) \cup \{u\} \cup \{C(u) \backslash u\}$;
else if $C(u) \neq C(v)$ **then**
  $\mathcal{C}_{t+1} = \mathcal{C}_t$;
else
  % Now (u, v) is inside a community C %
  L = {Maximal quasi-cliques in C};
  Let the singletons in $C \backslash L$ consider their best communities;
end if
Update $\mathcal{C}_{t+1}$;

Lemma 4. *If C and D are two communities of G, then the removal of an inter-community link connecting them will strengthen modularity contributions of both C and D.*

Proof. Let Q_{1C} (resp. Q_{1D}) and Q_{2C} (resp. Q_{2D}) be the modularities of C (resp. D) before and after the removal of that link. We show that $Q_{2C} > Q_{1C}$ (and similarly, $Q_{2D} > Q_{1D}$) and thus, C and D contribute higher modularities to the network.

$$\begin{aligned} Q_{2C} - Q_{1C} &= \left(\frac{m_1}{M-1} - \frac{(d_1-1)^2}{4(M-1)^2}\right) - \left(\frac{m_1}{M} - \frac{d_1^2}{4M^2}\right) \\ &= m_1\left(\frac{1}{M-1} - \frac{1}{M}\right) + \frac{1}{4}\left(\frac{d_1}{M} - \frac{d_1-1}{M-1}\right)\left(\frac{d_1}{M} + \frac{d_1-1}{M-1}\right) \end{aligned}$$

Since all terms are all positive, $Q_{2C} - Q_{1C} > 0$. The same technique applies to show that $Q_{2D} > Q_{1D}$. □

Lemma 5. *The removal of (u, v) inside a community C where only u or v is of degree one will not separate C.*

Proof. Assume the contradiction, i.e., after the removal of (u, v) where $d_u = 1$, C is broken into smaller communities X_1, X_2,..., X_k which contribute higher modularity: $Q_{X_1} + \ldots + Q_{X_k} > Q_C$. W.L.O.G., suppose u was connected to X_1 prior to its removal. It follows that $Q_{X_1+u} > Q_{X_1}$ and thus $Q_{X_1+u} + \ldots + Q_{X_k} > Q_C$, which raises a contradiction since C is originally a community of $\mathcal{C}$. □

Lemma 6. *(Separation of a community) Let $C_1 \subseteq C$ and $C_2 = C \backslash C_1$ be two disjoint subsets of C. $(\mathcal{C} \backslash C) \cup \{C_1, C_2\}$ is a community structure with higher modularity when an edge crossing C_1 and C_2 is removed, i.e., C should be separated into C_1 and C_2, if and only if $e_{12} < \frac{d_1 d_2 - d_C + 1}{2(M-1)} + 1$.*

Proof. Let q_1, q_2 and q_C denote the modularity contribution of C_1, C_2 and C after an edge crossing (C_1, C_2) has been removed. Now,

$$
\begin{aligned}
e_{12} < \frac{d_1 d_2 - d_C + 1}{2(M-1)} + 1 &\Leftrightarrow \frac{2d_1 d_2 - 2d_C + 2}{4(M-1)^2} > \frac{e_{12} - 1}{M-1} \\
&\Leftrightarrow \frac{(d_1 + d_2 - 2)^2}{4(M-1)^2} - \frac{(d_1 - 1)^2}{4(M-1)^2} - \frac{(d_2 - 1)^2}{4(M-1)^2} \\
&\quad > \frac{m_1 + m_2 + e_{12} - 1}{M-1} - \frac{m_1 - 1}{M-1} - \frac{m_2 - 1}{M-1} \\
&\Leftrightarrow \frac{m_1 - 1}{M-1} - \frac{(d_1 - 1)^2}{4(M-1)^2} + \frac{m_2 - 1}{M} - \frac{(d_2 - 1)^2}{4(M-1)^2} \\
&\quad > \frac{m_1 + m_2 + e_{12} - 1}{M-1} - \frac{(d_1 + d_2 - 2)^2}{4(M-1)^2} \\
&\Leftrightarrow q_1 + q_2 > q_C.
\end{aligned}
$$

Thus, the conclusion follows. □

Theorem 2.3. *(Community bi-division) For any community C, let α and β be the lowest and the second highest degree of vertices in C, respectively. Assume that an edge e is removed from C. If there do not exist subsets $C_1 \subseteq C$ and $C_2 \equiv C \backslash C_1$ such that e is crossing C_1 and C_2 and $\frac{\min\{\alpha(d_C - \alpha), \beta(d_C - \beta)\}}{2M} < e_{12} < \frac{(d_C - 2)^2}{8(M-1)} + 1$, then any bi-division of C will not benefit the overall Q.*

Proof. From Lemma 6, it follows that in order to really benefit the overall modularity we must have

$$\frac{d_1 d_2}{2M} < e_{12} < \frac{d_1 d_2 + 1}{2(M-1)} + 1.$$

Now we find an upper bound for the RHS inequality. Since $d_1 + d_2 = d_C$, it follows that

$$\begin{aligned} e_{12} &< \frac{d_1 d_2 - d_C + 1}{2(M-1)} + 1 \leq \frac{\frac{(d_1+d_2)^2}{4} - d_C + 1}{2(M-1)} + 1 \\ &\leq \frac{\frac{d_C^2}{4} - d_C + 1}{2(M-1)} + 1 = \frac{(d_C - 2)^2}{8(M-1)} + 1 \end{aligned}$$

For a lower bound of the LHS inequality, we rewrite $d_1 d_2$ as $d_1 d_2 = d_1(d_C - d_1) = d_1 d_C - d_1^2$ and find the non-zero minimum value on the range $d_1 \in [\alpha, \beta]$. In this interval, $d_1 d_C - d_1^2$ is minimized either at $d_1 = \alpha$ or $d_1 = \beta$. Therefore,

$$\frac{\min\{\alpha(d_C - \alpha), \beta(d_C - \beta)\}}{2M} \leq \frac{d_1 d_2}{2M} < e_{12} \leq \frac{(d_C - 2)^2}{8(M-1)} + 1$$

□

Finally, our QCA method for quickly updating the network community structure is presented in Alg. 5.

Algorithm 5 Quick **C**ommunity **A**daptation (QCA)

Input: $G \equiv G_0 = (V_0, E_0)$, $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, .., \mathcal{E}_s\}$ a collection of simple events
Output: Community structure $\mathcal{C}_t$ of G^t at time t.

```
Use [5] to find an initial community clustering C_0 of G_0;
for (t ← 1 to s) do
    C_t ← C_{t−1};
    if E_t = newNode(u) then
        New_Node(C_t, u);
    else if E_t = newEdge((u, v)) then
        New_Edge(C_t, (u, v));
    else if E_t = removeNode(u) then
        Remove_Node(C_t, u);
    else
        Remove_Edge(C_t, (u, v));
    end if
end for
```

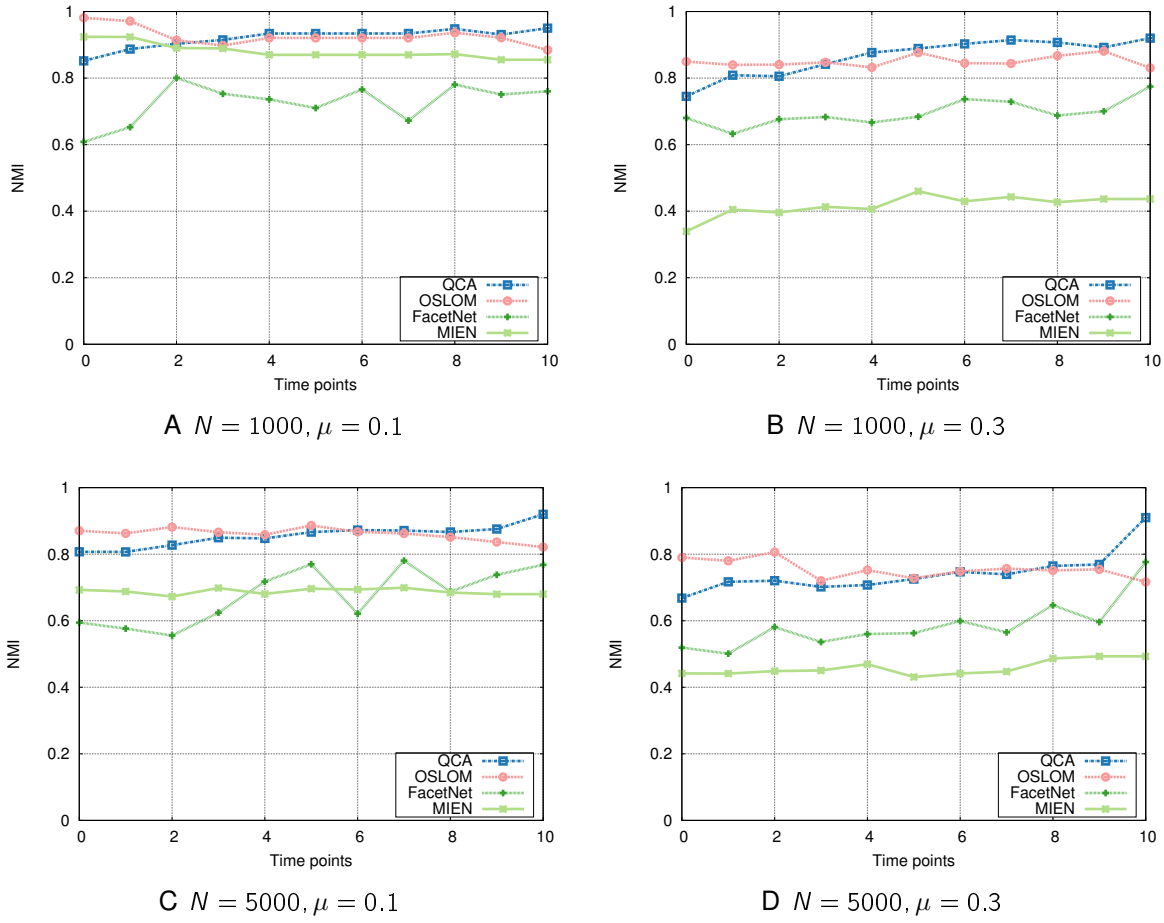
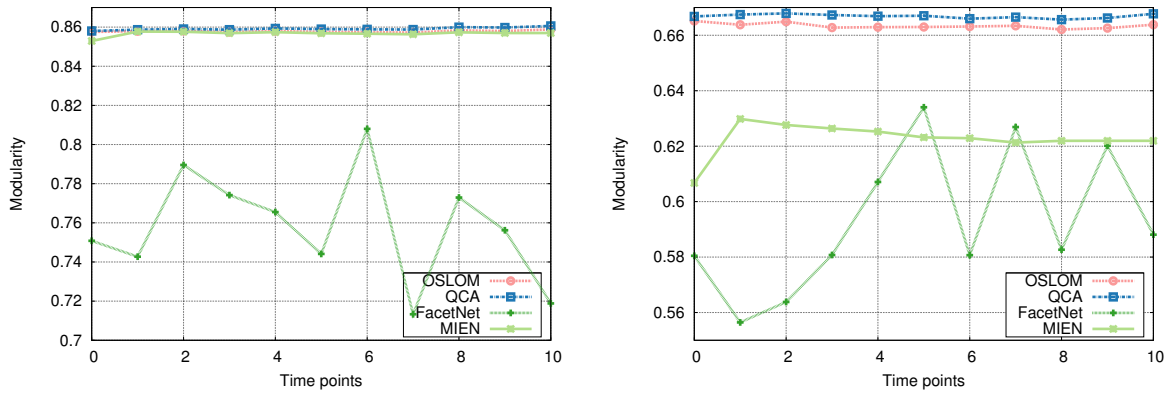


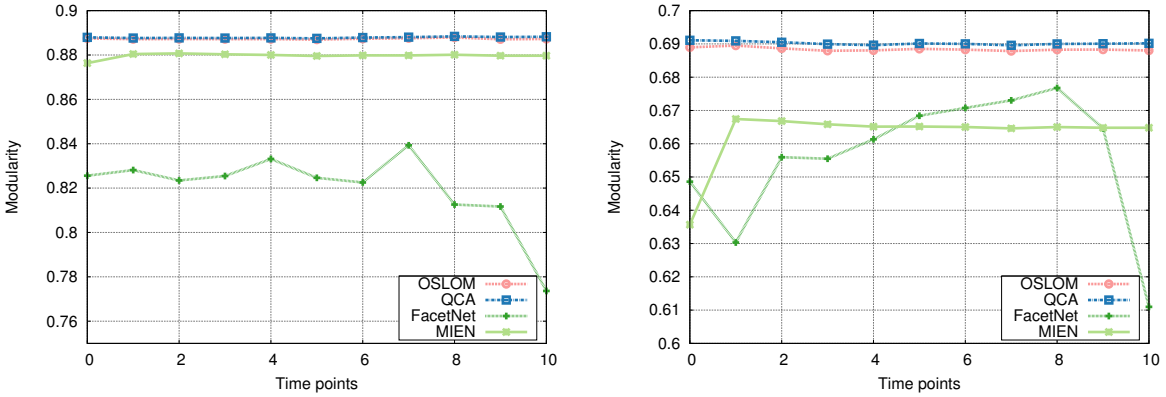
Figure 2-2. NMI scores on synthesized networks with known communities

2.3 Experimental Results

In this section, we first validate our approaches on different synthesized networks with known groundtruths, and then present our findings on real world traces including the Enron email [31], arXiv eprint citation [60], and Facebook social networks [61]. To certify the performance of our algorithms, we compare QCA to other adaptive community detection methods including (1) MIEN algorithm proposed by Thang et al. [35], (2) FacetNet framework proposed by Lin et al. [36], and (3) OSLOM method suggested by Lancichinetti et al. [40].



A $N = 1000, \mu = 0.1$ B $N = 1000, \mu = 0.3$



C $N = 5000, \mu = 0.1$ D $N = 5000, \mu = 0.3$

Figure 2-3. Modularity values on synthesized networks with known communities

2.3.1 Results on Synthesized Networks

Of course, the best way to evaluate our approaches is to validate them on real networks with known community structures. Unfortunately, we often do not know that structures beforehand, or such structures cannot be easily mined from the network topology. Although synthesized networks might not reflect all the statistical properties of real ones, they provide us known ground truths via planted communities, and the ability to vary other parameters such as sizes, densities and overlapping levels, etc. Testing community detection methods on generated data has become an usual practice that is widely accepted in the field [3]. Hence, comparing QCA with other dynamic methods

on synthesized networks not only certifies its performance but also provides us the confidence to its behaviors on real world traces.

Setup. We use the well-known LFR benchmark [3] to generate 40 networks with 10 snapshots. Parameters are: the number of nodes $N = \{1000, 5000\}$, the mixing parameter $\mu = \{0.1, 0.3\}$ controlling the overall sharpness of the community structure. In order to quantify the similarity between the identified communities and the ground truth, we adopt a well known measure in Information Theory called Normalized Mutual Information (NMI). NMI has been proven to be reliable and is currently used in testing community detection algorithms [3]. Basically, $NMI(U, V)$ equals 1 if structures U and V are identical and equals 0 if they are totally separated, and the higher NMI the better.

Results. The NMI and Modularity values are reported in Figures 2-2 and 2-3. As depicted in their subfigures, the NMI values and modularities indicated by our QCA method, in general, are very high and competitive with those of OSLOM while are much better than those produced by MIEN and FacetNet methods. On these generated networks, we observe that MIEN and FacetNet perform well when the mixing parameter μ is small, i.e., when the network community structures are clear, however, their performances degrade dramatically when these structures become less clear as μ gets larger. Particularly, MIEN' and FacetNet' NMI scores and modularities in all test cases are fairly low and usually from 10% to 50% and 5% to 15% worst than those produced by QCA. This implies the network communities revealed by these methods are not as high similarity to the ground-truth as QCA algorithm. On the generated networks, OSLOM algorithm performs very well as suggested through its high NMI scores and modularity values. In particular, OSLOM tends to perform better than QCA in the first couple of network snapshots, however, its performance is taken over by QCA when the networks evolve over time, especially at the end of the evolution where OSLM reveals big gaps in similarity to the planted network communities (Note that the higher NMI score at the end of the evolution, the better the final detected community structure). This

concludes that the network communities discovered by QCA are of the best similarity to ones planted in the ground-truth in comparison with other methods.

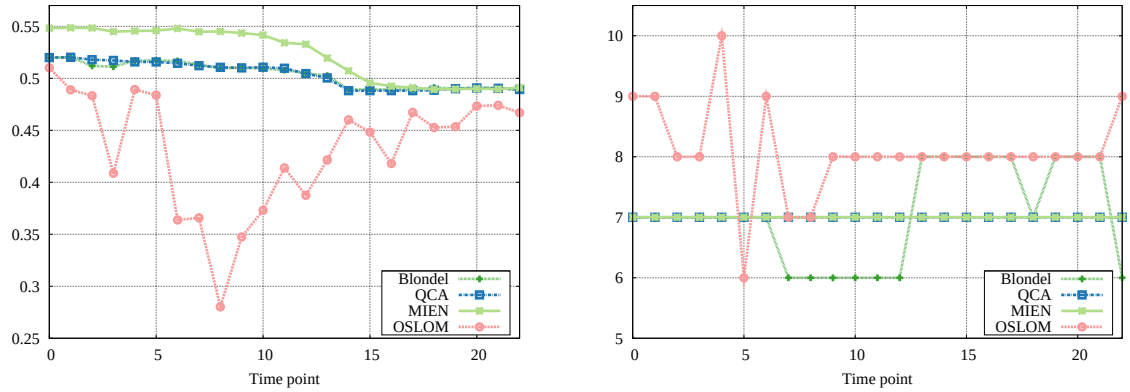
2.3.2 Results on Real World Traces

We next present the results of QCA algorithms on real world dynamic social networks including ENRON email [31], arXiv e-print citation [60], and Facebook networks [61]. Due to the lack of appropriate communities corresponding to these traces, we report the performance of the aforementioned algorithms in reference to the static method proposed by Blondel et al. [5]. In particular, we will show the following quantities (1) modularity values, (2) the quality of the identified network communities through NMI scores, and (3) the processing time of our QCA in comparison with other methods. The above networks possess to contain strong community structures due to their high modularities, which was the main reason for them to be chosen.

For each network, time information is first extracted and a portion of the network data (usually the first snapshot) is then collected to form the basic network community structure. Our QCA method (aslo MIEN and OSLOM) take into account that basic community structure and run on the network changes whereas the static method has to be performed on the whole network snapshot for each time point. In this experiment, FacetNet method does not appear to complete the tasks in a timely manner, and is thus excluded from the plots.

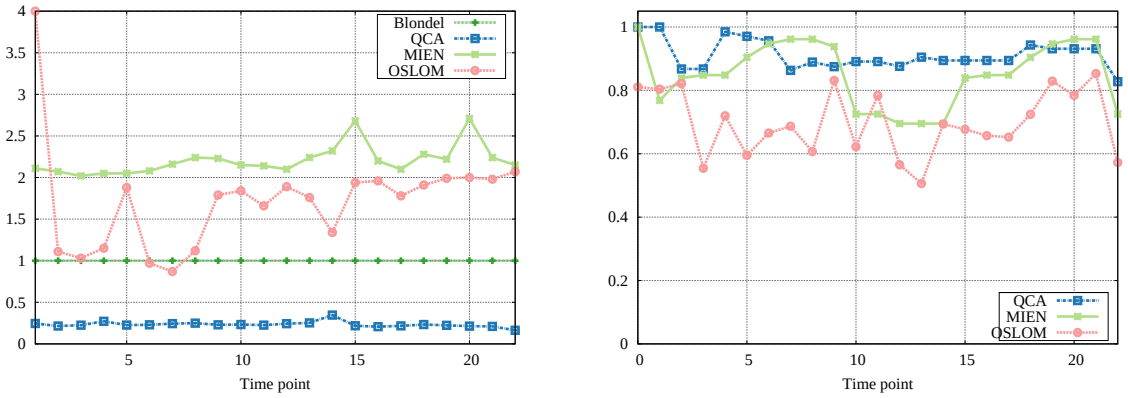
ENRON email network

Data. The Enron email network contains email messages data from about 150 users, mostly senior management of Enron Inc., from January 1999 to July 2002 [31]. Each email address is represented by an unique ID in the dataset and each link corresponds to a message between the sender and the receiver. After a data refinement process, we choose 50% of total links to form a basic community structure of the network with 7 major communities, and simulate the network evolution via a series of 21 growing snapshots.



A Modularity

B Number of Communities



C Running Time(s)

D NMI

Figure 2-4. Simulation results on Enron email network.

Results. We first evaluate the modularity values computed by QCA, MIEN, OSLOM, and Blondel methods. As shown in Figure $2-4A$, our QCA algorithm archives competitively higher modularities than the static method but a little bit less than MIEN, and is far better than those obtained by OSLOM. Moreover, QCA also successes in maintaining the same numbers of communities of the other two methods MIEN and Blondel while OSLOM's are vague (Figure $2-4B$). In particular, the modularity values produced by QCA very well approximate those found by static method with lesser variation. There are reasons for that. Recall that our QCA algorithm takes into account the basic community structures detected by the static method (at the first snapshot) and processes on network changes only. Knowing the basic network community structure

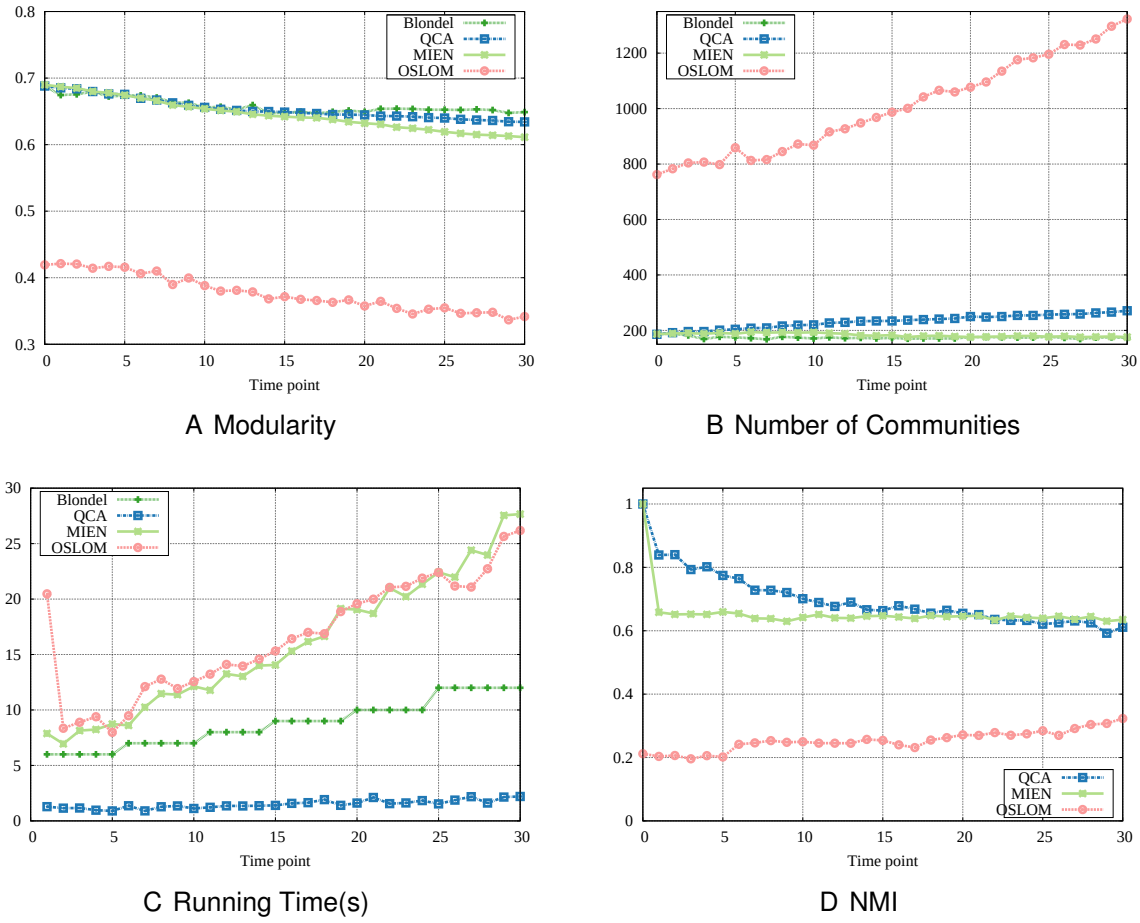


Figure 2-5. Simulation results on arXiv e-print citation network.

is a great advantage of our QCA algorithm: it can avoid the hassle of searching and computing from scratch to update the network with changes. In fact, QCA uses the basic structure for finding and quickly updating the local optimal communities to adapt with changes introduced during the network evolution.

The running time of QCA and the static method in this small network are relatively close: the static method requires one second to complete each of its tasks while our QCA does not even ask for one (Figure 2 – 4*C*). In this dataset, MIEN and OSLOM requires a little more time (1.5 and 2.4 seconds in average for MIEN and OSLOM) to complete their tasks. Time and computational cost are significantly reduced in QCA

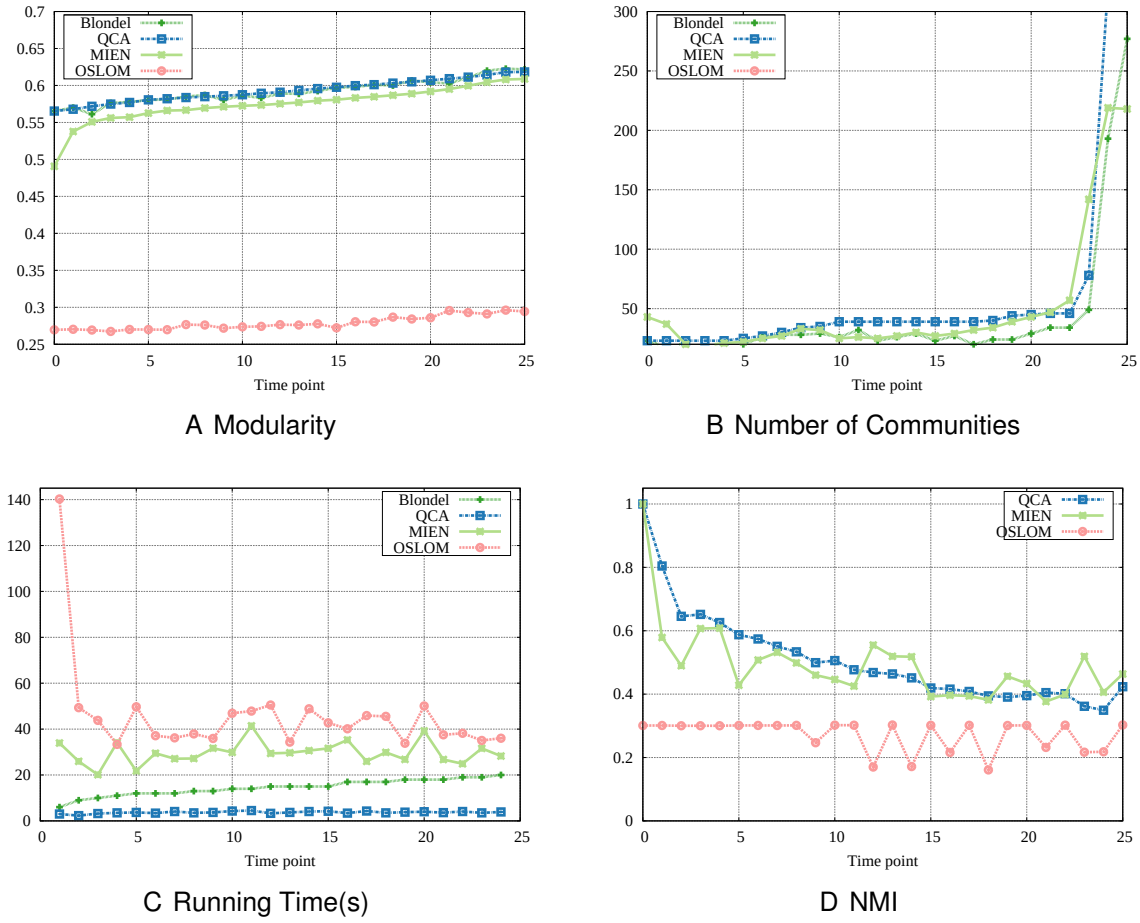


Figure 2-6. Simulation results on Facebook social network.

since our algorithms only take into account the network changes while the static method has to work on the whole network every time.

As reported in Figure $2-4D$, both the NMI scores of ours and MIEN method are very high and relatively close to 1 while those obtained by OSLOM fall short and are far from stable. These results indicate that in this Enron email network, both QCA and MIEN algorithms are able to identify high quality community structure with high modularity and similarity; however, only our method significantly reduces the processing time and computational requirement.

arXiv e-print citation network

Data. The arXiv e-print citation network [60] has become an essential mean of assessing research results in various areas including physics and computer sciences. This network contained more than 225K articles from January 1996 to May 2003. In our experiments, citation links of the first two years 1996 and 1997 were used to form the basic community structure of our QCA method. In order to simulate the network evolution, a total of 30 time dependent snapshots are created on a two-month regular basis from January 1998 to January 2003.

Results. We compare modularity results obtained by QCA algorithm at each network snapshot to Blondel as well as to MIEN and OSLOM methods. It reveals from Figure 2-5A that the modularities returned by QCA are very close to those obtained by the static method with much more stabler and are far higher than those obtained by OSLOM and MIEN. In particular, the modularity values produced by QCA algorithm cover from 94% up to 100% that of Blondel method and from 6% to 10% higher than MIEN and at least 1.5x better than OSLOM. In this citation networks, the numbers of communities detected by OSLOM take off with more than 1200 whereas those found by QCA, MIEN and Blondel methods are relatively small (Figure 2-5B). Our QCA method discovers more communities than both Blondel and MIEN as the network evolves and this can be explained based on the resolution limit of modularity [58]: the static method might disregard some small communities and tend to combine them in order to maximize the overall network modularity.

A second observation on the running time shows that QCA outperforms the static method as well as its competitor MIEN: QCA takes at most 2 seconds to complete updating the network structure while Blondel method requires more than triple that amount of time, MIEN and OSLOM asks for more than 5 times (Figure $2-5C$). In addition, higher NMI scores of QCA than MIEN's and especially OSLOM's scores (Figure $2-5D$) implies network communities identified by our approach are not only

of high similarity to the ground truth but also more precise than that detected by MIEN, while the computational cost and the running time are significantly reduced.

Facebook social network

Data. This dataset contains friendship information among New Orleans regional network on Facebook [61], spanning from September 2006 to January 2009 with more than 60K nodes (users) connected by more than 1.5 million friendship links. In our experiments, nodes and links from September 2006 to December 2006 are used to form the basic community structure of the network, and each network snapshot is recored after every month during January 2007 to January 2009 for a total of 25 network snapshots.

Results. The evaluation depicted in Figure $2-6A$ reveals that QCA algorithm achieves competitive modularities in comparison with the static method, and again far better than those obtained by MIEN and OSLOM method, especially in comparison with OSLOM whose perform was nice on synthesized networks. In the general trend, the line representing QCA results closely approximates that of the static method with much more stability. Moreover, the two final modularity values at the end of the experiment are relatively the same, which means that our adaptive method performs competitively with the static method running on the whole network.

Figure $2-6C$ describes the running time of the three methods on the Facebook data set. As one can see from this figure, QCA takes at least 3 seconds and at most 4.5 seconds to successfully compute and update every network snapshot whereas the static method, again, requires more than triple processing time. MIEN and OSLOM methods really suffer on this large scale network when requiring more than 10x and 11x that amounts of QCA running times. In conclusion, high NMI and modularity scores together with decent executing times on all test cases confirm the effectiveness of our adaptive method, especially when applied to real world social networks where a

centralized algorithm, or other dynamic algorithms, may not be able to detect a good network community structure in a timely manner.

However, there is a limitation of QCA algorithm we observe on this large network and want to point out here: As the the duration of network evolution lasts longer over time (i.e., the number of network snapshots increases), our method tends to divide the network into smaller communities to maximize the local modularity, thus results in an increasing number of communities and a decreasing of NMI scores. Figure $2-6B$ and $2-6D$ describes this observation. For instance, at snapshot 12 (a year after December 2006), the NMI score is approximately $1/2$ and continues decaying after this time point. It implies a refreshment of network community structure is required at this time, after a long enough duration. This is reasonable since activities on an online social network, especially on Facebook social network, tend to come and go rapidly and local adaptive procedures are not enough to reflect the whole network topology over a long period of time.

CHAPTER 3
OVERLAPPING COMMUNITY STRUCTURE DETECTION

In this chapter, we present AFOCS, an adaptive framework to discover and trace the evolution of network communities in dynamic complex systems. In section 3.1, we first state the problem definition including basic notations and the dynamic network model. Next, we present the procedure to detect the basic community structure in section 3.2, and then our AFOCS framework to update and trace the community structure evolution over time in section 3.3. Finally, we demonstrate the empirical results in section 3.4.

3.1 Problem Formulation

3.1.1 Basic Notations

Let $G = (V, E)$ be an undirected unweighted graph representing the network where V is the set of N nodes and E is the set of M connections. Denote by $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ the network community structure, i.e., a collection of subsets of V where each $C_i \in \mathcal{C}$ and its induced subgraph form a community of G. In contrast with the disjoint community structure, we allow $C_i \cap C_j \neq \emptyset$ so that network communities can overlap with each other. For a node $u \in V$, let d_u, $N(u)$ and $Com(u)$ denote its degree, its neighbors and its set of community labels, respectively. For any $C \subseteq V$, let C^{in} and C^{out} denote the set of links having both endpoints in C and the set of links having exactly one endpoint in C, respectively. Finally, the terms node-vertex as well as edge-link-connection are used interchangeably.

3.1.2 Dynamic Network Model

Let $G_0 = (V_0, E_0)$ be the original input network and $G_t = (V_t, E_t)$ be a time dependent network snapshot recorded at time t. Denote by ΔV_t and ΔE_t the sets of nodes and edges to be added to or removed from the network at time t. Furthermore, let $\Delta G_t = (\Delta V_t, \Delta E_t)$ describe this change in terms of the whole network. The network snapshot at next time step $t + 1$ is expressed as a combination of the previous one

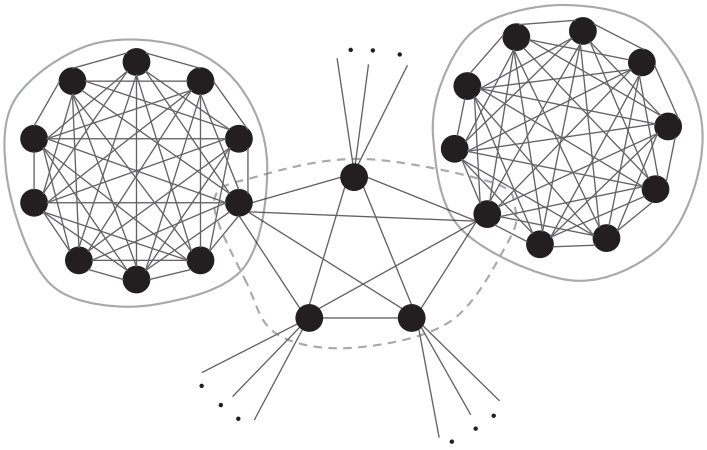


Figure 3-1. Overlapped v.s. non-overlapped community structures.

together with the change, i.e., $G_{t+1} = G_t \cup \Delta G_t$. Finally, a dynamic network $\mathcal{G}$ is defined as a sequence of network snapshots changing over time: $\mathcal{G} = (G_0, G_1, G_2, \ldots)$.

3.1.3 Density Function

In order to quantify the goodness of an identified community, we use the popular density function Ψ [62] defined as: $\Psi(C) = \frac{2|C^{in}|}{|C|(|C|-1)}$ where $C \subseteq V$. Unlike the case of disjoint community structure, in which the number of connections crossing communities should be less than those inside them, our objective does not take into account the number of out-going links from each community. To understand the reason, let us consider a simple example pictured in Figure $3-1$. In the overlapping community structure point of view, it is clear that every clique should form a community on its own, and each community shares with the central clique exactly one node. However, in the disjoint community structure point of view, any vertex at the central clique has n internal and $2n$ external connections, which violates the concept of a community in the strong sense. Furthermore, the internal connectivity of the central clique is also dominated by its external density, which implies the concept of a community in weak sense is also violated. (A community C is in a weak sense if $|C^{in}| > |C^{out}|$, and in a strong sense if any node in C has more links inward than outward C [63]).

In order to set up a threshold on the internal density that suffices for a set of nodes C to be a local community, we propose a function $\tau(C)$ defined as follows:

$$\tau(C) = \frac{\sigma(C)}{\binom{|C|}{2}} \text{ where } \sigma(C) = \binom{|C|}{2}^{1-\frac{1}{\binom{|C|}{2}}}$$

Here $\sigma(C)$ is the threshold on the number of inner connections that suffices for C to be a local community. Particularly, a subgraph induced by C is a local community iff $\Psi(C) \geq \tau(C)$ or equivalently $|C^{in}| \geq \sigma(C)$. Several functions with the same purpose have been introduced in the literature, for instance, in the work of [46][44], and it is worth noting down the main differences between them and ours. First and foremost, our functions $\tau(C)$ and $\sigma(C)$ locally process on the candidate community C only and neither require any predefined thresholds or user-input parameters. Secondly, by Proposition 3.1, $\sigma(C)$ and $\tau(C)$ are increasing functions and closely approach C's full connectivity as well as its maximal density. That makes $\sigma(C)$ and $\tau(C)$ relaxation versions of the traditional density function, yet useful ones as we shall see in the experiments.

Proposition 3.1. *The function $f(n) = n^{1-\frac{1}{n}}$ is strictly increasing for $n \geq 4$ and $\lim_{n\to\infty} f(n) = n$.*

3.1.4 Objective Function

Our objective is to find a community assignment for the set of nodes V which maximizes the overall internal density function $\Psi(\mathcal{C}) = \sum_{C\in\mathcal{C}} \Psi(C)$ since the higher the internal density of a community is, the clearer its structure would be. Although our objective puts more focus on the internal edges and less focus on the external edges, these external edges are not completely ignored but are considered in the following senses: they will be tested later for the formation of another community if the number of edges suffices. Only when these external edges are really sparse, they will not be considered.

3.1.5 Problem Definition

Given a dynamic network $\mathcal{G} = (G_0, G_1, G_2, \ldots)$ where G_0 is the input network and $G_1, G_2, \ldots$ are network snapshots obtained through a collection of network topology changes $\Delta G_1, \Delta G_2, \ldots$ over time. The problem asks for an adaptive framework to efficiently detect and update the network overlapping community structure $\mathcal{C}_t$ at any time point t by only utilizing the information from the previous snapshot $\mathcal{C}_{t-1}$, as well as tracing the evolution of the network communities.

In the next section, we present our main contribution: an adaptive framework for (1) identifying basic overlapped community structure in a network snapshot and (2) updating as well as tracing the evolution of the network communities in a dynamic network model. First, we describe FOCS, a procedure to identify the basic communities in a static network, and then discuss in great detail how AFOCS adaptively updates these basic communities to cater with the evolution of the dynamic network.

3.2 Basic Community Structure Detection

We describe FOCS, the first phase of our framework that quickly discovers the basic overlapping network community structure. In general, FOCS works toward the classification of network nodes into different groups by first locating all possible densely connected parts of the network (3.2.1), and then combining those who highly overlap with each other, i.e., those share a significant substructure (3.2.2). Finally, a final refinement to group unassigned nodes into different communities is conducted in (3.2.3).

In FOCS, β (the input overlapping threshold) defines how much substructure two communities can share. Note that FOCS fundamentally differs from [48] in the way it allows $|C_i \cap C_j| \geq 2$ for any subsets C_i, C_j of V, and consequently allows network communities to overlap not only at a single vertex but also at a part of the whole community.

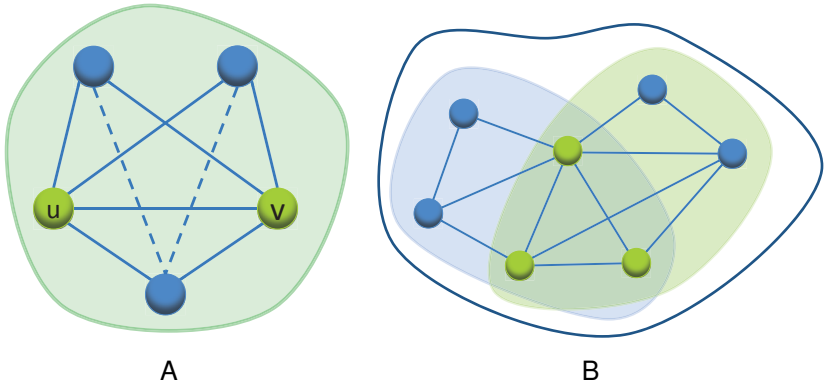


Figure 3-2. Locating and merging local communities.

3.2.1 Locating Local Communities

Local communities are connected parts of the network whose internal densities are greater than a certain level. In FOCS, this level is automatically determined based on the function $\tau()$ and the size of each corresponding part. Particularly, a local community is defined based on a connection (u, v) when the number of internal connections in the subgraph induced by $C \equiv \{u, v\} \cup (N(u) \cap N(v))$ exceeds $\sigma(C)$, or in other words, when $\Psi(C) \geq \tau(C)$ as illustrated in Figure 3-2A. Here, (a) A local community C defined by a link (u, v). Here $\Psi(C) = 0.9 > \tau(C) = 0.794$ (b) Merging two local communities sharing a significant substructure (OS score $= 1.027 > \beta = 0.8$).

However, there is a problem that might eventually arise: the containment of sub communities in an actual bigger one. Intuitively, one would like to detect a bigger community unified by smaller ones if the bigger community is itself densely connected. In order to filter this undesired case, we impose $\Psi\left(\bigcup_{i=1}^{s} C_i\right) < \tau\left(\bigcup_{i=1}^{s} C_i\right) \quad \forall s = 1 \ldots |\mathcal{C}|$ (note that some of these unifications do not contain all the nodes). In addition, we allow this locating procedure to skip over tiny communities of size less than 4. This condition is carried out from Proposition 3.1. This makes sense in terms of mobile or social networks where a group of mobile devices or a social community usually has size larger than 3, and intuitively agrees with the finding of [64][65]. Thus, the condition $|C| \geq 4$ is

imposed for any community C we discuss hereafter. The tiny communities will then be identified later. Alg. 6 describes this procedure.

Algorithm 6 Locating local communities

Input: $G = (V, E)$
Output: A collection of raw communities $\mathcal{C}_r$.

```
1: C_r ← ∅;
2: for ((u, v) ∈ E) do
3:     if (Com(u) ∩ Com(v) = ∅) then
4:         C ← {u, v} ∪ N(u) ∩ N(v);
5:         if (|C^in| ≥ σ(C) and |C| ≥ 4) then
6:             Check C's connectivity if |C| = 5;
7:             Define C a local community;
8:             /*Include C into the raw community structure*/
9:             C_r ← C_r ∪ {C};
10:        end if
11:    end if
12: end for
```

Lemma 7. *All local communities C's detected by Alg. 6 satisfy $\Psi(C) \geq \tau(4) \approx 0.74$. Furthermore, other communities satisfying these conditions will also be detected by Alg. 6.*

Proof. Alg. 6 will examine every edge $(u, v) \in E$ (except those whose endpoints are already in the same community), and by this greedy nature, any local community it detects has $|C| > 4$ and $\Psi(C) \geq \tau(C) \geq \tau(4) \approx 0.74$.

We now show that any community C statisfying $|C| \geq 4$ and $\Psi(C) \geq \tau(C) \geq \tau(4)$ will also be detected by Alg. 6. Suppose otherwise, that is there exists a community C satisfying these two conditions and is not detected by Alg. 6. To prove that this is not the case, we do the following: (1) Construct a community D which is not detected by Alg. 6 with $|D| = n \equiv |C|$ and $\Psi(D)$ is maximized, and (2) show that $\Psi(D) < \tau(D)$.

Because $|D| = |C|$, it implies $\tau(D) = \tau(C)$. However, since $\Psi(D)$ is maximized, $\Psi(D) \geq \Psi(C)$ which in turn implies $\Psi(C) \leq \Psi(D) < \tau(D) = \tau(C)$. This raises a contradiction to our original assumption, and thus concludes the proof.

To construct D, we do as follow (i) make D a clique of size n, and (ii) remove edges from D one by one until D cannot be detected by Alg. 6. By doing in this way, $\Psi(D)$ is maximized iff the number of removed edges is minimized.

It is easy to find the least number of edges we have to remove from D is $n/2$ if n is even and $n/2 - 1$ if n is odd. Therefore, $m_D = n(n-1)/2 - n/2$ if n is even, and $m_D = n(n-1)/2 - (n-1)/2$ if n is odd. Now, $\Psi(D) < \tau(D)$ iff $m_D < \left(\frac{n(n-1)}{2}\right)^{1-\frac{2}{n(n-1)}}$. Let $f(n)$ be the difference between the left and the right hand sides, we show that $f(n) < 0$ as n increases. Taking the derivative of $f(n)$ gives $\delta f(4) < 0$ and $f(n) < f(4) < 0$ for all even $n > 4$, and $\delta f(7) < 0$ and $f(n) < f(7) < 0$ for all odd $n > 7$. When $n = 5$, $f(5) > 0$ but this is the only exception and thus, can be handled easily in line 6 of Alg. 6. Therefore, we have $\Psi(D) < \tau(D)$, and hence, the conclusion follows. □

Theorem 3.1. *The local community structure $\mathcal{C}_r$ detected by Alg. 6 satisfies $\Psi(\mathcal{C}_r) \geq \tau(4) \times \Psi(OPT)$ where OPT is the optimal dense community assignment satisfying $\Psi(S) \geq \tau(4)$ for any $S \in OPT$.*

Proof. Let $\mathcal{C}_r$ be the local community structure returned by Alg. 6, and OPT be the optimal solution of the dense community assignment satisfying $\Psi(S) \geq \tau(4)$ for any $S \in OPT$. Let $k = |OPT|$. Clearly $\Psi(OPT) \leq k$. By Lemma 7, we know that Alg. 6 can detect as many communities as OPT but probably with less internal density. Moreover, since Alg. 6 only skips over edges in a community, it ensures that no real community is a substructure of a bigger one. Hence, we have $\Psi(\mathcal{C}_r) \geq \tau(4) \times k \approx 0.74 \times \Psi(OPT)$. This also implies that Alg. 6 is an 0.74-approximation algorithm for finding local densely connected communities. □

Lemma 8. *The time complexity of Alg. 6 is $O(dM)$ where $d = \max_{v \in V} d_v$.*

Proof. Time to examine an edge (u, v) is $|N(u)| + |N(v)| = d_u + d_v$. However, when u and v are in the same community, (u, v) will be skipped. Therefore, the total time complexity is upper bounded by $d \sum_{u \in V} d_u = O(dM)$. □

3.2.2 Combining Overlapping Communities

After Alg. 6 finishes, the raw network community structure is pictured as a collection of (possibly overlapped) dense parts of the network together with outliers. As some of those dense parts can possibly share significant substructures, we need to merge them if they are highly overlapped. To this end, we introduce the overlapping score of two communities defined as follow

$$OS(C_i, C_j) = \frac{|I_{ij}|}{\min\{|C_i|, |C_j|\}} + \frac{|I_{ij}^{in}|}{\min\{|C_i^{in}|, |C_j^{in}|\}}$$

where $I_{ij} = C_i \cap C_j$. Basically, $OS(C_i, C_j)$ values how important the common nodes and links shared between C_i and C_j mean to the smaller community. In comparison with the distance metric suggested in [43], our overlapping score not only takes into account the fraction of common nodes but also values the fraction of common connections, which is crucial in order to combine network communities. Furthermore, $OS(\cdot, \cdot)$ is symmetric and scales well with the size of any community, and the higher the overlapping score, the more those communities in consideration should be merged. In this merging process, we combine communities C_i and C_j if $OS(C_i, C_j) \geq \beta$ (Figure 3-2B).

Algorithm 7 Combining local communities

Input: Raw community structure $\mathcal{C}_r$
Output: A refined community structure $\mathcal{D}$.

```
1: D ← C_r;
2: Done ← false;
3: while (!Done) do
4:     Done ← true;
5:     Order (C_i, C_j)'s by their OS(C_i, C_j) scores;
6:     for (C_i, C_j ∈ C_r) do
7:         if (OS(C_i, C_j) > β and ) then
8:             C ← Combine C_i and C_j;
9:             /*Update the current structure*/
10:            D ← (C_f \ {C_i ∪ C_j}) ∪ C;
11:            Done ← False;
12:        end if
13:    end for
14: end while
```

The time complexity of Alg. 7 is $O(N_0^2)$ where N_0 is the number of local communities. Clearly, $N_0 \leq M$ and thus, it can be $O(M^2)$. However, when the intersection of two communities is upper bounded, by Lemma 9 we know that the number of local communities is also upper bounded by $O(N)$, and thus, the time complexity of Alg. 7 is $O(N^2)$. In our experiments, we observe that the running time of this procedure is, indeed, much less than $O(N^2)$.

Lemma 9. *The number of raw communities detected in Alg. 6 is $O(N)$ when the number of nodes in the intersection of any two communities is upper bounded by a constant α.*

Proof. For each $C_i \in \mathcal{C}$, decompose it into overlapped and non-overlapped parts, denoted by C_i^{ov} and C_i^{nov}. We have $C_i = C_i^{ov} \cup C_i^{nov}$ and $C_i^{ov} \cap C_i^{nov} = \emptyset$. Therefore, $|C_i| = |C_i^{ov}| + |C_i^{nov}|$.

Now,

$$\sum_{C_i \in \mathcal{C}} |C_i| = \sum_{C_i \in \mathcal{C}} (|C_i^{ov}| + |C_i^{nov}|) \leq N + \sum_{i<j} |C_i^{ov} \cap C_j^{nov}|,$$

where $N = \sum_{C_i \in \mathcal{C}} |C_i^{nov}| + \left| \bigcup_{C_i \in \mathcal{C}} |C_i^{ov}| \right|$. For an upper bound of the second term, rewrite

$$\sum_{i<j} |C_i^{ov} \cap C_j^{nov}| \leq N + \sum_{|C_i \cap C_j| \geq 2} |C_i \cap C_j| \leq N(1+\alpha),$$

where $\alpha = \max\{|C_i \cap C_j| : |C_i \cap C_j| \geq 2\}$

Hence, $\sum_{C_i \in \mathcal{C}} |C_i| \leq N(2+\alpha)$. Let N_0 be the number of raw communities, it follows that $N_0 \min\{|C_i|\} \leq \sum_{C_i \in \mathcal{C}} |C_i| \leq (2+\alpha)N$. Since $\min\{|C_i|\} \geq 4$, we have $N_0 \leq \frac{(2+\alpha)}{4} N = O(N)$. □

Remark

After the above community merging process, detected communities can possibly be of very large sizes. The explanation is as follow: small quasi-cliques are discovered in the first phase (Alg. 6) as densely connected parts of the network, and are regarded as candidate elements for bigger communities in the merging process. If these small cliques are loosely connected to the rest of the network, they will retain as local

communities afterwards. Otherwise, they can be merged to other dense parts to become new bigger communities. As a result, if the communities are highly overlapped, some of them can potentially grow to very large sizes at the end of the merging process, beside the small cliques detected at the first place. Larger dense quasi-cliques, though rare in many networks, will surely be detected by FOCS as we observed in Theorem 3.1.

3.2.3 Revisiting Unassigned Nodes

Even when the above two procedures are executed, there would still exist leftover nodes or edges due to their less attraction to the rest of the network. Because of its size constraint, the first procedure skips over tiny communities of sizes less than four and thus, may leave out some nodes unlabeled. These nodes will not be touched in the second phase since they do not belong to any local communities, and consequently, will remain unassigned afterwards. Moreover, they are mostly nodes with less connection to the rest of the network, and thus, are very likely supplement nodes possibly to their adjacent communities. Therefore, we need to revisit those nodes to either group them into appropriate communities or classify them as outliers based on their connectivity structures.

Algorithm 8 Revisit Unassigned Nodes

Input: The refined community structure $\mathcal{D} = \{D_1, D_2, \ldots, D_t\}$
Output: The basic community structure $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$

$\mathcal{C} \leftarrow \mathcal{D}$;
for $(u \in V$ and $Com(u) == \emptyset)$ **do**
    $NC(u) \leftarrow \{C_j \in \mathcal{C} | u$ is adjacent to $C_j\}$;
    for $(C_j \in NC(u))$ **do**
        if $(F_{C_j \cup \{u\}} \geq F_{C_j})$ **then**
            $C_j \leftarrow C_j \cup \{u\}$;
            $Com(u) \leftarrow Com(u) \cup \{j\}$;
        end if
    end for
    if $(Com(u) == \emptyset)$ **then**
        Classify u as an outlier;
    end if
end for

Alternatively, this process can be thought of as a community trying to hire adjacent unassigned nodes which are similar to the host community. However, the internal density function might be too strict for them to be included in any community (which was also the reason why they are left unassigned). To this end, we need a community fitness function in order to quantify the similarity between a node u and a neighbor community C. We find the fitness function $F_S = \frac{|S^{in}|}{2|S^{in}|+|S^{out}|}$ (where $S \subseteq V$) commonly used in [46][66][43] performs competitively in both synthesized and real-world datasets. Taking into account this fitness function, a community C will keep hiring any unassigned adjacent vertex of maximum similarity in a greedy manner, provided the newly joined vertex does not shrink down the community's current fitness value. If there is no such node, C is defined as a final network community. Nodes remained unlabeled through this last procedure are identified as outliers. This algorithm is presented in Alg. 8.

3.3 Detecting Evolving Network Communities

We describe AFOCS, the second phase and also the main focus of our detection framework. In particular, we use AFOCS to adaptively update and trace the network communities, which were previously initialized by FOCS, as the dynamic network evolves over time. Note that FOCS is executed only once on G_0, after that AFOCS will take over and handle all changes introduced to the network.

Let us first discuss the various behaviors of the community structure when the network topology evolves over time. Suppose $G = (V, E)$ and $\mathcal{C} = \{C_1, C_2, .., C_n\}$ is the current network and its corresponding overlapping community structure, respectively. We use the term intra links to denote edges whose two endpoints belong to the same community, inter links to denote those with endpoints connecting different disjoint communities and the term hybrid links to stand for the others. For each community C of G, the number of connections joining C with the others are lesser than the number of connections within C itself by definition

Intuitively, the addition of intra links or removal of inter links between communities of G will strengthen them and consequently, will make the structure of G more clear. Similarly, removing intra links from or introducing inter links to a community of G will decrease its internal density and as a result, loosen its internal structure. However, when two communities have less distraction to each other, adding or removing links makes them more attractive to each other and therefore, leaves a possibility that they can overlap with each other or can be combined to form a new community. The updating process, as a result, is very complicated and challenging since any insignificant change in the network topology could possibly lead to an unpredictable transformation of the network community structure.

In order to reflect these changes to a complex network, its underlying graph model is frequently updated by either inserting or removing a node or a set of nodes, or an edge or a set of edges. A scrutiny look into these events reveals that the introduction or removal of a set of nodes (or edges) can furthermore be decomposed as a collection of node (or edge) insertions (or removals), in which only a node (or only an edge) is inserted (or removed) at a time. Therefore, changes to the network at each time step can be viewed as a collection of simpler events whose details are as follow:

- newNode $(V + u)$: A new node u and its adjacent edge(s) are introduced
- removeNode $(V - u)$: A node u and its adjacent edge(s) are removed from the network.
- newEdge $(E + e)$: A new edge e connecting two existing nodes is introduced.
- removeEdge $(E - e)$: An edge e in the network is removed.

As we mentioned earlier, our adaptive framework initially requires a basic community structure $\mathcal{C}_0$. To obtain this basic structure, we apply FOCS algorithm at the first network snapshot, i.e., we execute FOCS on the network G_0 and then let AFOCS adaptively handle this structure as the network evolves.

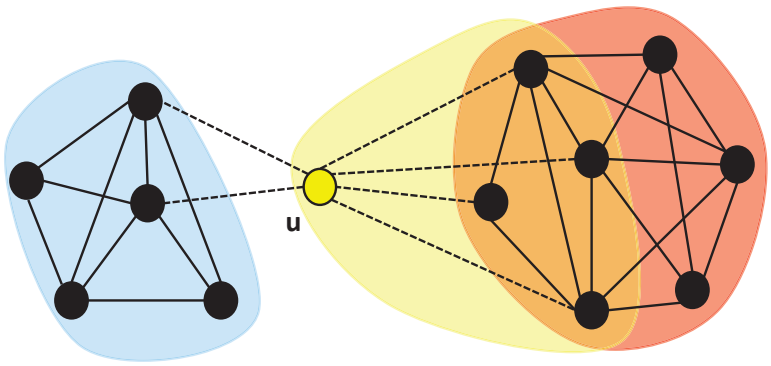


Figure 3-3. A possible scenario when a new node is introduced.

3.3.1 Handling a New Node

Let us discuss the first case when a new node u and its associated links are introduced to the network. Possibilities are (1) u may come with no adjacent edge or (2) with many of them connecting one or more possibly overlapped communities. If u has no adjacent edge, we simply join u in the set of outliers and preserve the current community structure.

The interesting case happens, and it usually does, when u comes with multiple links connecting one ore more existing communities. Since network communities can overlap each other, we need to determine which ones u should join in in order to maximize the gained internal density. But how can we quickly and effectively do so? By Lemma 10, we give a necessary condition for a new node in order to join in an existing community, i.e., our algorithm will join node u in C if the number of connections u has to C suffices: $d_{ui} > \max\{\frac{2|C_i^{in}|}{|C_i|-1}, f(|C_i|+1) - |C_i^{in}|\}$. However, failing to satisfy this condition does not necessarily imply that u should not belong to C, since it can potentially gather some substructure of C to form a new community (Figure 3-3). Thus, we also need to handle this possibility. Alg. 9 presents the algorithm.

Lemma 10. *Suppose u is a newly introduced node with d_{ui} connections to each adjacent community C_i. u will join in C_i if $d_{ui} > \max\{\frac{2|C_i^{in}|}{|C_i|-1}, f(|C_i|+1) - |C_i^{in}|\}$.*

Algorithm 9 Handling a new node u

Input: The current community structure $\mathcal{C}_{t-1}$
Output: An updated structure $\mathcal{C}_t$.

$C_1, C_2, \ldots, C_k \leftarrow$ Adjacent communities of u;
for i = 1 **do** to k
  if $(d_{ui} > \max\{\frac{2|C_i^{in}|}{|C_i|-1}, f(|C_i|+1) - |C_i^{in}|\})$ **then**
    $C_i \leftarrow C_i \cup \{u\}$;
  else
    $C \leftarrow N(u) \cap C_i$;
    if $(\Psi(C) \geq \tau(C)$ and $|C| \geq 4)$ **then**
      $C_i \leftarrow C_i \cup \{u\}$;
    end if
  end if
end for
/*Checking new communities formed from outliers*/
for $(v \in N(u)$ and $Com(v) \cap Com(u) = \emptyset)$ **do**
  $C \equiv N(u) \cap N(v)$;
  if $(\Psi(C) \geq \tau(C)$ and $|C| \geq 4)$ **then**
    Define C a new community;
  end if
end for
Merging overlapping communities on $C_1, C_2, \ldots, C_k$;
Update $\mathcal{C}_t$;

Proof. Prior to u joining to C_i, the internal density is $\Psi(C_i) = \frac{2|C_i^{in}|}{|C_i|(|C_i|-1)}$. Similarly, after u joining in C_i, the density function is $\Psi(C_i \cup \{u\}) = \frac{2|C_i^{in}|+2d_{ui}}{|C_i|(|C_i|+1)}$. Taking the difference between these two quantities gives $\Psi(C_i \cup \{u\}) > \Psi(C_i) \iff d_{ui} > \frac{2|C_i^{in}|}{|C_i|-1}$. Moreover, u should also satisfy $\Psi(C_i \cup \{u\}) \geq \tau(C_i \cup \{u\})$, which in turn implies $d_{u,i} \geq f(|C_i|+1) - |C_i^{in}|$. Therefore, $d_{ui} > \max\{\frac{2|C_i^{in}|}{|C_i|-1}, f(|C_i|+1) - |C_i^{in}|\}$. □

The analysis of Alg. 9 is shown by Lemma 11. In particular, we show that this procedure achieves at least 74% the internal density of the optimal assignment for u, given the prior community structure.

Lemma 11. *Alg. 9 produces a community assignment that, prior to the community combination process, achieves* $\Psi(\mathcal{C}_t) \geq \tau(4) \times \Psi(OPT(u)_t)$ *where* $OPT(u)_t$ *is the optimal community assignment for* u *at time* t*, given the prior community structure* $\mathcal{C}_{t-1}$.

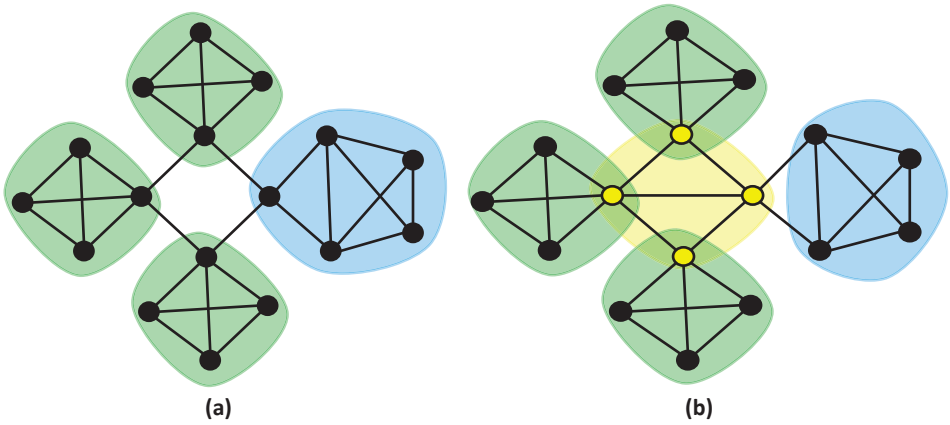


Figure 3-4. Possible scenarios when a new edge is introduced.

Proof. Let $C_1, C_2, \ldots, C_k$ be the communities (including the newly formed ones) in $\mathcal{C}_t$ that Alg. 9 assigns the new node u to. Note that in the optimal solution $OPT(u)_t$, the number of communities u belongs to should not exceed k since each C_i is also a candidate for $OPT(u)_t$ (of course, $OPT(u)_t$ could possibly rearrange nodes differently). Therefore, the optimal internal density gained is upper bounded by k. On the other hand, Alg. 9 makes sure that each community C_i that u joins in should have $\Psi(C_i) \geq \tau(C_i) \geq \tau(4)$ since $|C_i| \geq 4$. Thus, Alg. 9 will achieve at least $\tau(4) \times k \approx 0.74 \times \Psi(OPT(u)_t)$. □

3.3.2 Handling a New Edge

In case where a new edge $e = (u, v)$ connecting two existing vertices u and v is introduced, we divide it further into two four smaller cases: (1) e is solely inside a single community C (2) e is within the intersection of two (or more) communities (3) e is joining two separated communities and (4) e is crossing overlapped communities. If e is totally inside a community C, its presence will strengthen C's internal density and by Lemma 12, we know that adding e should not split the current community C into smaller substructures.

In the second subcase, the introduction of the new edge might increase the density of some part of C and it is reasonable to think of that part (say D) as a new separated community. However, since D originally shared a significant substructure with C, the

merging process will then combine C and D (if they were separated) to be a bigger community, thus raising the same community as if C was kept intact. Therefore, the same reaction applies in the second subcase when e is within the intersection of two communities since their inner densities are both increased. Thus, in these first two cases, we leave the current network structure intact.

Algorithm 10 Handling a new edge (u, v)

Input: The current community structure $\mathcal{C}_{t-1}$.
Output: An updated community structure $\mathcal{C}_t$.

if $((u, v) \in$ a single community OR $(u, v) \in C_u \cap C_v)$ **then**
    $\mathcal{C}_t \leftarrow \mathcal{C}_{t-1}$;
else if $(Com(u) \cap Com(v) == \emptyset)$ **then**
    $C \leftarrow N(u) \cap N(v)$;
    if $(\Psi(C) \geq \tau(C))$ **then**
        Define C a new community;
        Check for combining on $Com(u)$, $Com(v)$ and C;
    else
        for $(D \in Com(u)$ (or $D* \in Com(v)))$ **do**
            if $(\Psi(D \cup \{v\}) \geq \tau(D \cup \{v\}))$ (or $\Psi(D* \cup \{u\}) \geq \tau(D* \cup \{v\}))$ **then**
                $D \leftarrow D \cup \{v\}$ (or $D* \leftarrow D* \cup \{u\}$)
            end if
        end for
        Merging overlapping communities for D's (or $D*$);
    end if
    Update $\mathcal{C}_t$;
end if

Handling the last two subcases is complicated since any of them can either have no effect on the current network structure or unpredictably form a new network community, and furthermore can overlap or merge with the others (Figure 3-4). However, there is still a possibility that the introduction of this new link, together with some substructure of C_u or C_v, suffices to form a new community that can overlap with not only C_u and C_v but also with some of the others. The other subcases can be handled similarly. Alg. 10 describe this procedure.

Lemma 12. *If an new edge (u, v) is introduced solely inside a community C, it should not split C into smaller substructures.*

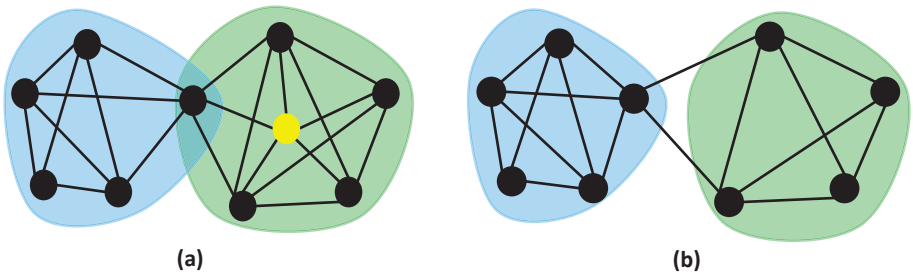


Figure 3-5. Possible scenarios when an existing node is removed.

Proof. Suppose otherwise, that is C is divided into smaller parts C_1 and C_2. Prior to the introduction of (u, v), we have $\Psi(C) = \Psi(C_1 \cup C_2) \geq \tau(C) = \tau(C_1 \cup C_2)$. Now, when C_1 and C_2 are formed, they imply that $\Psi(C_1 \cup C_2 + (u, v)) < \tau(C_1 \cup C_2 + (u, v))$. Putting all together, we have $\tau(C_1 \cup C_2 + (u, v)) = \tau(C_1 \cup C_2) > \Psi(C_1 \cup C_2 + (u, v)) > \Psi(C) > \tau(C_1 \cup C_2)$, which raises a contradiction. Thus, the conclusion follows. □

3.3.3 Removing an Existing Node

When an existing node u is about to be removed from the network, all of its adjacent edges will also be removed as a consequence. If u is an outlier, we can simply exclude u and its corresponding links from the current structure and safely keep the network communities unchanged.

In unfortunate situations where u is not an outlier, the problem becomes very challenging in the sense that the resulting community is complicated: it can either be unchanged, or broken into smaller communities, or could probably be further merged with the other communities. To give a sense of this effect, let's consider two examples illustrated in Figure 3-5. In the first example, when C is almost a full clique, the removal of any node will not break it apart. However, if we a remove node that tends to connect the others within a community, the leftover module is broken into a smaller one together with a node that will later be merged to one of its nearby communities. Therefore, identifying the leftover structure of C is a crucial task once a vertex u in C is removed.

To quickly handle this task, we first examine the internal density of C excluding the removed node u. If the number of internal connections still suffices, e.g., $\Psi(C\backslash\{u\}) \geq \tau(C\backslash\{u\})$, we can safely keep the current network community structure intact because C is still tightly connected itself with a sufficient internal density. Otherwise, this community is of a weak strength and shall be broken into smaller ones. These substructures might further be merged with other communities if C origianlly overlaps with them. To efficiently detect these new substructures, we apply Alg. 6 on the subgraph induced by $C\backslash\{u\}$ to quickly identify the leftover modules in C, and then let these modules hire a set of unassigned nodes $\Psi(C)$ that help them increasing their inner densities. Finally, we locally check for community combination, if any, by using an algorithm similar to Alg. 7. Alg. 11 presents the procedure.

Algorithm 11 Removing a node u

Input: The current community structure $\mathcal{C}_{t-1}$.
Output: An updated structure $\mathcal{C}_t$.

```
for (C ∈ Com(u) and Ψ(C\{u}) < τ(C\{u})) do
    LC ← Local communities by Alg 6 on C\{u};
    for (C_i ∈ LC and |C_i| ≥ 4) do
        S_i ← Nodes such that Ψ(C_i ∪ S_i) ≥ τ(C_i ∪ S_i);
        C_i ← C_i ∪ S_i;
    end for
    Merging overlapping communities on LC;
end for
Update C_t;
```

3.3.4 Removing an Existing Edge

In the last situation when an edge $e = (u, v)$ is about to be removed, we divide it further into four subcases similar to those of a new edge (1) e is between two disjoint communities (2) e is inside a sole community (3) e is within the intersection of two (or more) communities and finally (4) e is crossing overlapping communities.

In the first subcase, when e is crossing two disjoint communities, its removal will make the network structure more clear (since we now have less connections between groups), and thus, the current communities should be keep unchanged. When e is

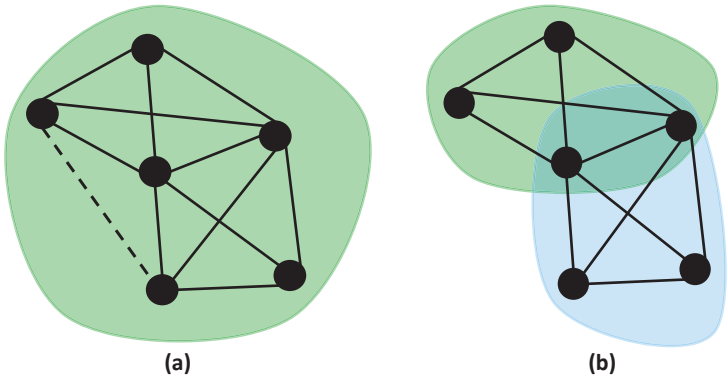


Figure 3-6. Possible scenarios when an existing edge is removed.

totally within a sole community C, handling its removal is complicated since this can lead to an unpredictable transformation of the host module: C could either be unchanged or broken into smaller modules if it contains substructures which are less attractive to each other, as depicted in Figure 3-6. Therefore, the problem of identify the structure of the remaining module becomes the central part for not only this case but also for the others.

Algorithm 12 Removing an edge (u, v)

Input: The current structure $\mathcal{C}_{t-1}$.
Output: An updated community structure $\mathcal{C}_t$.

if $((u, v)$ is an isolated edge) **then**
    $\mathcal{C}_t = (\mathcal{C}_{t-1} \backslash \{u, v\}) \cup \{u\} \cup \{v\}$;
else if $(d_u = 1$ (or $d_v = 1))$ **then**
    $\mathcal{C}_t = (\mathcal{C}_{t-1} \backslash C(u)) \cup \{u\} \cup C(v)$;
else if $(C \equiv C(u) \cap C(v) = \emptyset)$ **then**
    $\mathcal{C}_t = \mathcal{C}_{t-1}$;
else if $(\Psi(C \backslash (u, v)) < \tau(C \backslash (u, v)))$ **then** /*Here $C \neq \emptyset$*/
    $LC \leftarrow$ Local communities by Alg 6 on $C \backslash (u, v)$;
    Define each $L \in LC$ a local community of $\mathcal{C}_{t-1}$;
    Merging overlapping community on L's;
end if
Update $\mathcal{C}_t$;

To quickly handle these tasks, we first verify the inner density of the remaining module and, again utilize the local community location method (Alg. 6) to locally identify the leftover substructures. Next, we check for community combination since

these structures can possibly overlap with existing network communities. The detailed procedure is described in Alg. 12.

3.3.5 Remarks

Note that the ultimate goal of our framework is to adaptively detect and update the community structure as the network evolves, i.e., to mainly deal with the dynamics of a mobile network. As a result, we mainly put our focus on AFOCS. Although FOCS, the first detection phase, appears to be a centralized algorithm, it is executed only once at the very first network snapshot whereas AFOCS stays up and locally handles all changes as the network evolves over time. That said, we do not execute FOCS again. Furthermore, AFOCS can be run independently with FOCS, i.e., one can use any localized detection algorithm to identify a basic community structure at the first phase. Thus, AFOCS can be easily apply to solve mobile network problems.

3.3.6 Complexity

Our main algorithm consists of two parts: (1) finding the basic community structure and (2) updating the network community structure through changes introduced at every time step. The complexity of quickly unfolding the basic network community structures has been claimed to be linear in terms of number of nodes and links $O(M + N)$ [3]. To handle the case of a new node of degree p coming in, our algorithm computes p forces this new node applies to its neighbors, which results in linear time complexity $O(p)$. When a new edge connecting nodes u and v is introduced to the network, our algorithm just simply computes the forces applied to communities adjacency nodes, which takes $O(|C(u)| + |C(v)|)$ in the best case and $O(k \times \max\{|C(u)|, |C(v)|\})$ in the worst case when some nodes in a module are pulled out to form new communities (where k is the number of communities in G). The time taken to handle the last two cases is essentially the time complexity of the clique percolation, which is roughly $O(|C(u)|^3)$ in the worst case. Although the time complexity is in the third order of number of nodes, the total nodes inside a single community is relatively small in comparison with the total number

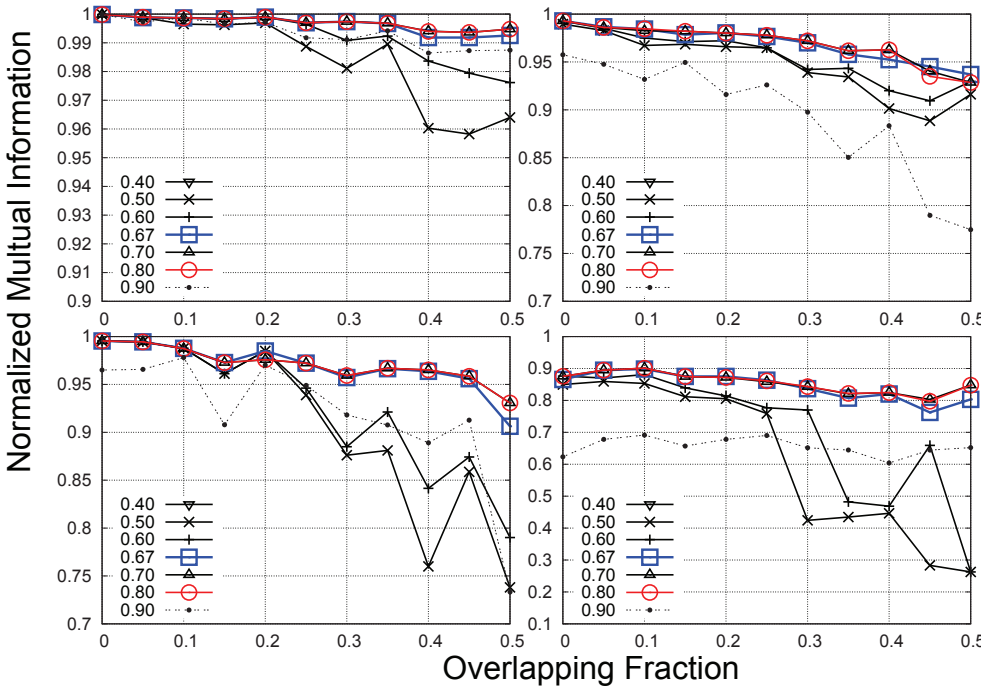


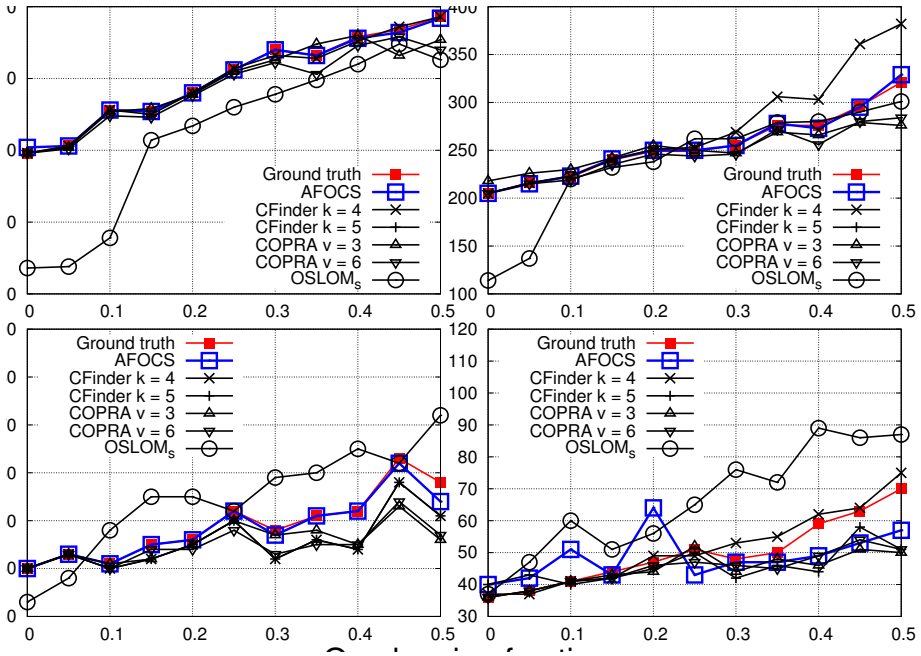
Figure 3-7. NMI scores for different values of β. $N = 5000$ (top), $N = 1000$ (bottom), $\mu = 0.1$ (left), $\mu = 0.3$ (right).

of vertices N, and thus, does not affect the actual running time. Experimental results in Section 4 show that our algorithm performs quickly and smoothly in large social online networks.

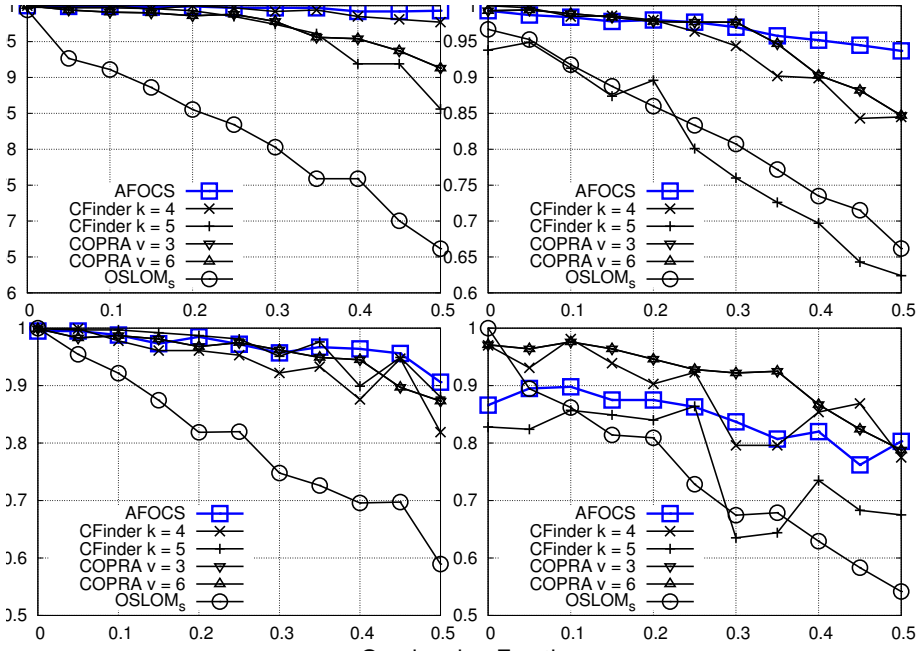
3.4 Experimental Results

In this section, we first present the empirical results of AFOCS in comaprison with two static detection methods: CFinder - the most popular method [42], and COPRA - the most effective method [47]. We next compare the performance of AFOCS with other dynamic methods including OSLOM [40], FacetNet [67] and iLCD [39].

Data Sets: We use networks generated by the well-known LFR overlapping benchmark [3], the 'de facto' standard for evaluating overlapping community detection algorithms. Generated networks follow power-law degree distributions and contain embedded overlapping communities (the ground truth) of varying sizes that capture the internal characteristics of real-world networks.



A Number of communities



B NMI scores

Figure 3-8. Comparison among AFOCS, COPRA and CFinder methods. $N = 5000$ (top), $N = 1000$ (bottom), $\mu = 0.1$ (left), $\mu = 0.3$ (right).

Set up: To fairly compare with COPRA and to avoid being biased, we keep the parameters close to [47]: the minimum and maximum community sizes are $c_{min} = 10$ and $c_{max} = 50$, each vertex belongs to at most two communities, $o_m = 2$. $N = 1000$ and $N = 5000$. The mixing rate $\mu = 0.1$ and $\mu = 0.3$. The overlapping fraction γ, which determines the fraction of overlapped nodes, is from 0 to 0.5. Since COPRA is nondeterministic, we run it 10 times on each instance and select the best result.

Metrics: We evaluate following metrics.

(1) The generalized Normalized Mutual Information (NMI) [46] specially built for overlapping communities. NMI scores the similarity between the detected network communities and the ground truth. This is an standardized measure since NMI(U,V)=1 if structures U and V are identical and 0 if they are totally separated.

(2) The number of communities, ignoring singleton communities and unassigned nodes. A good community detection method should produce roughly the same number of communities with the known ground truth.

3.4.1 Choosing the Overlapping Threshold β

The overlapping threshold β is the only input parameter required by our framework, and thus, determining its appropriate value plays an important role in assessing AFOCS's performance. To best determine this threshold, we run AFOCS on generated networks with different values of β, and record the similarities between the detected communities and the ground-truth via NMI scores (Figure 3-7). Of course, the higher NMI scores imply the better β values.

As a threshold parameter, β controls how much substructure communities can have in common. The smaller values of β imply the more we allow network communities to overlap with each other, and vice versa. Similarly, β can be thought of as the zooming scale of the network structure where lower β's reveal the coarser and higher β's reveal the finer structure. As depicted in Figure 3-7, the best values for β are ranging from 0.67 to 0.80, among which $\beta = 0.70$ yields the best community similarity (NMI scores are

ranging from 0.8 to 1) in all of the generated networks. Therefore, we fix the overlapping threshold in AFOCS to be 0.70 hereafter.

3.4.2 Reference to Static Methods

We show our results in groups of four. For each case we vary the overlapping fraction γ from 0 to 0.5 and analyze the results found by AFOCS, CFinder, COPRA and (static) OSLOM methods (OSLOM$_s$). We only present results when corresponding parameters give top performance for CFinder (clique size $k = 4, 5$) and COPRA (max. communities per vertex $v = 3, 6$).

Figure 3-8A shows the number of communities found by AFOCS, COPRA and CFinder, OSLOM$_s$ and the ground truth. It reveals from this figure that the numbers of communities found by AFOCS, marked with squares, are the closest and almost identical to the ground truth as the overlapping fraction gets higher. There is an exception when $N = 1000$ and $\mu = 0.3$ which we will discuss later. In terms of NMI scores, as one can infer from Figure 3-8B, AFOCS achieves the highest performance among all methods with much more stable. A common trend in this test is the performances of all methods degrade (1) when the mixing rate μ increases, i.e., when the community structure becomes more ambiguous or (2) when the size of network decreases while the mixing rate μ stays the same. Even though AFOCS is not very competitive only when both negative factors happen in the bottom-right char as $N = 1000$ and $\mu = 0.3$, it is in general the best performer. OSLOM$_s$, the static version of OSLOM method, does not appear to perform well on these synthesized data as its NMI scores are low and degrade quickly when the network communities become more stochastic. The NMI scores of AFOCS, on the other hand, remain high and stable even when the network community structure becomes unclear when the overlapping fraction increases.

The significant gap is observed when the mixing rate gets higher ($\mu = 0.3$) and the network size gets smaller ($N = 1000$). AFOCS provides less numbers of communities

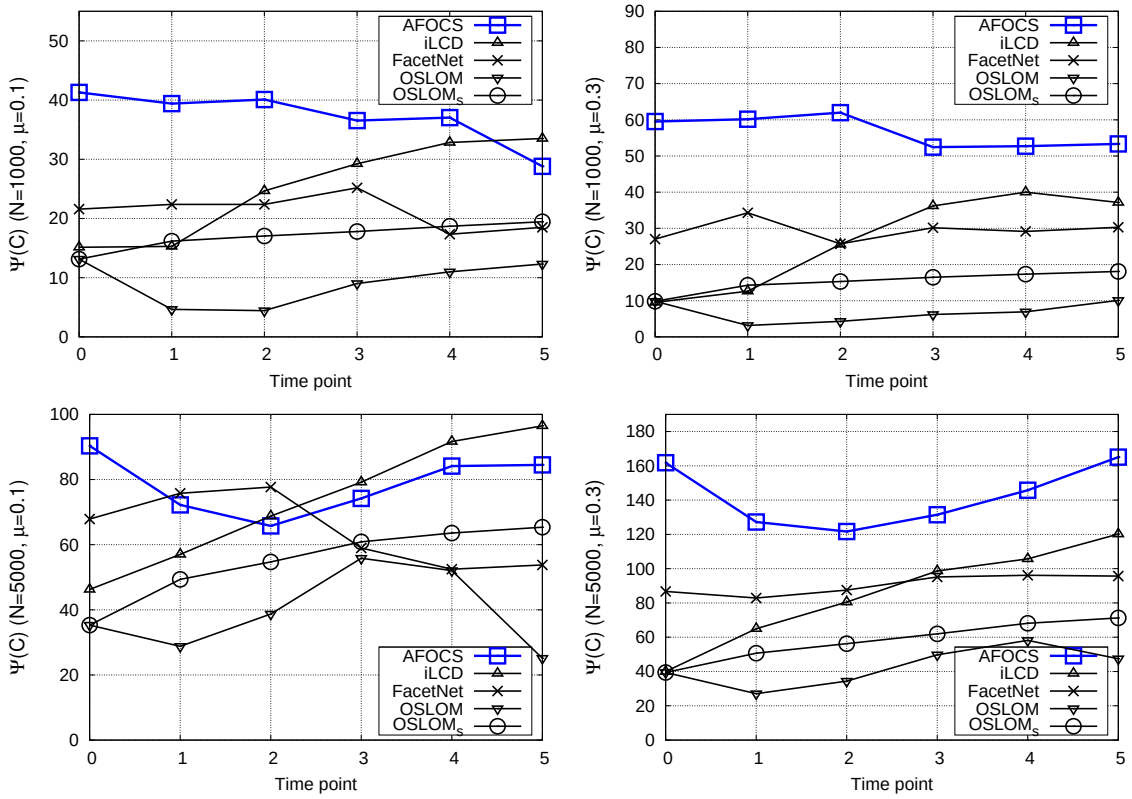
than those of the ground truth but with much higher overlapping rates. The reason is with a larger mixing rate μ, a node will have more edges connecting vertices in other communities, thus increases the chance that AFOCS will merge highly overlapped communities. Hence, AFOCS creates less but with larger size communities. We note that this 'weakness' of AFOCS is controversial as when the mixing rate increases, the ground truth does not necessarily coincide with the structure implied by the network's topology. Extensive experiments show the ability of AFOCS in identifying high quality overlapping communities. In addition, we found AFOCS runs substantially faster than the other competitors: on the Facebook regional network [61] containing 63K nodes, AFOCS is 150x faster than COPRA while CFinder is unable to finish its tasks.

3.4.3 Reference to Other Dynamic Methods

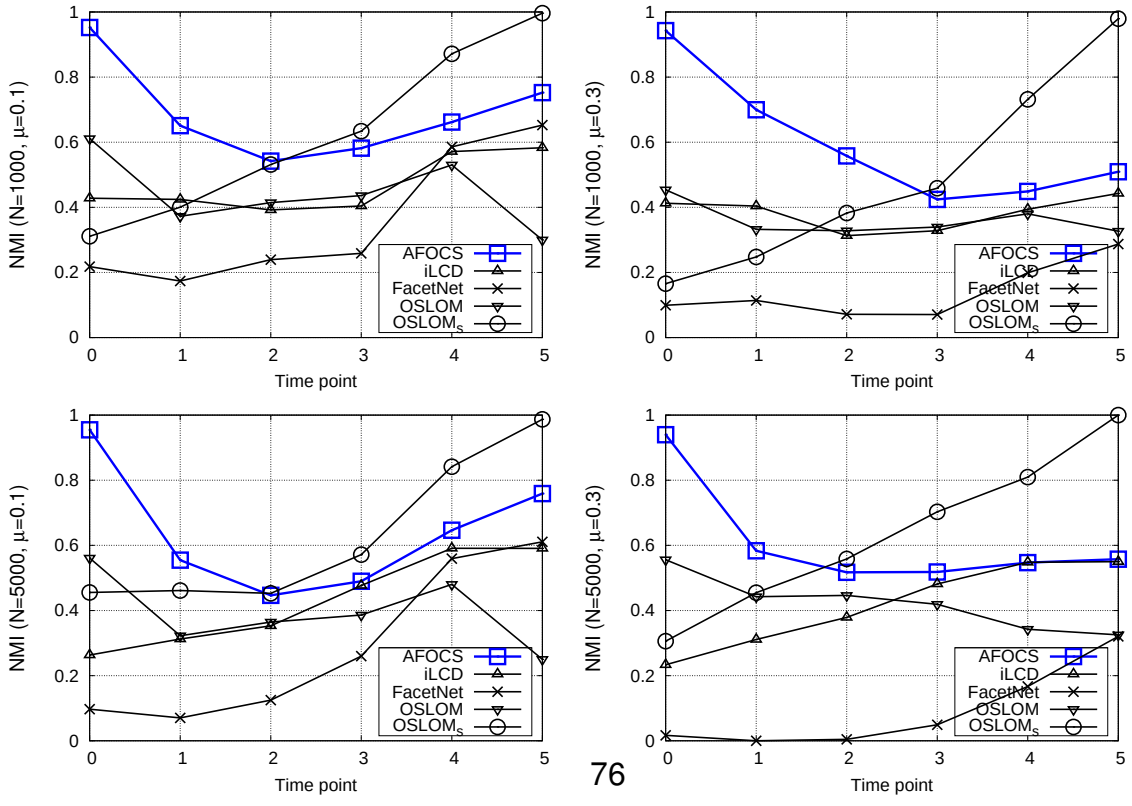
We next observe the performance of AFOCS in reference to two dynamic methods FacetNet, iLCD and OSLOM. Since the ground-truth communities are known on synthesized datasets, fair comparisons among three methods can be obtained via their NMI scores and running times. Of course, the higher its NMI scores with less time consuming, the better the method seems to be.

Each synthesized dynamic network is simulated via 5 snapshots, in which the basic communities are formed by using 50% of the network data with approximately 10% of the network evolution (node/edge additions and removals) added to each growing snapshot at a time. Since FacetNet requires the number of communities a priori, we input this method the actual number as mined form the ground-truth. For iLCD and OSLOM methods, we keep the default setting as provided in their deliverable.

We first evaluate the objective function, i.e., the total internal density obtained by all methods in Figure 3-9A. Although internal density is not necessarily the objective of other methods, this metric can provide us the concept of how strong the community structure detected by each approach is. As revealed Figure 3-9A, AFOCS obtained the highest internal density in all tests and is only lagged behind iLCD approach.



A Objective values



B NMI scores

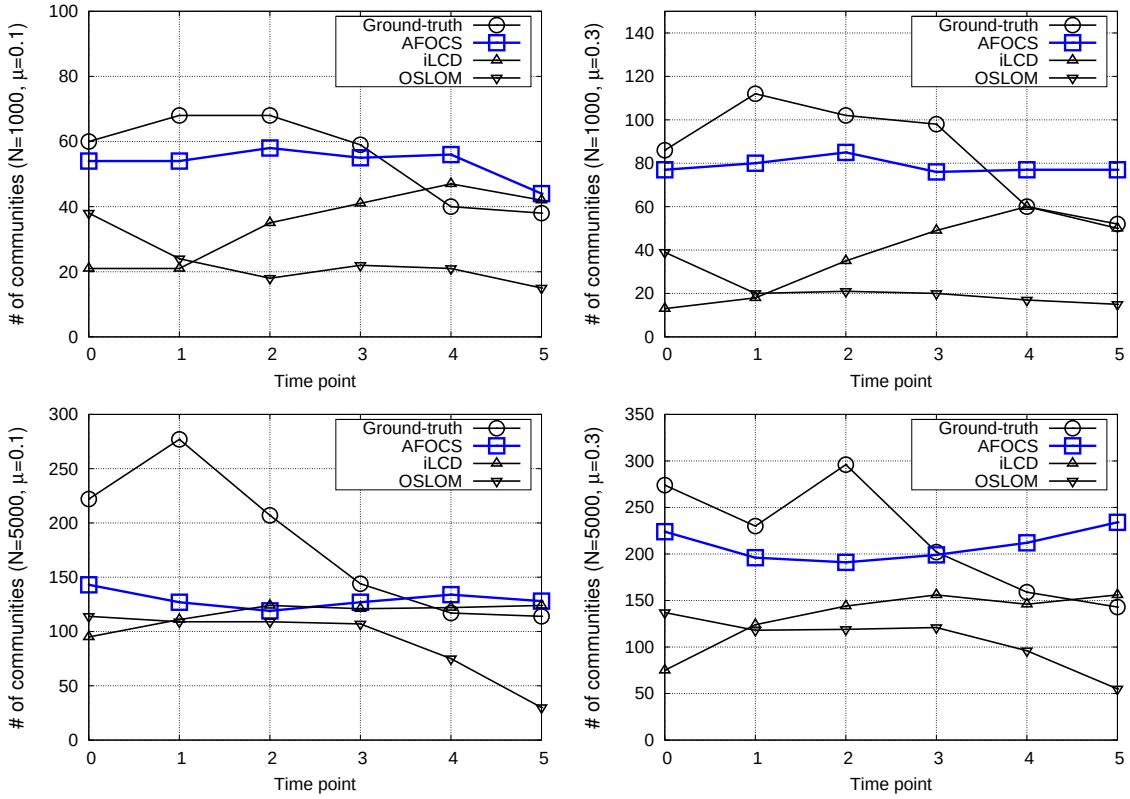


Figure 3-10. The number of communities obtained by AFOCS, iLCD, FacetNet and OSLOM and OSLOM$_s$ methods.

The NMI scores of four methods are presented in Figure 3-9B and 3-10. It reveals from these subfigures that the NMI scores of AFOCS are higher than those of FacetNet, iLCD and OSLOM. In particular, the NMI scores of AFOCS are about just 5-7% lag behind that of OSLOM and iLCD in the first 2 network snapshots, while are much better than the others at the end of the evolution. OSLOM's NMI values are very high at the very beginning, however, they tend to decrease quickly as more connections and nodes are introduced. The NMI scores of iLCD and FacetNet tend to fluctuate and also decrease significantly at the last snapshot. AFOCS, in the other trend, keeps its NMI scores high and wealthy, especially at the end of the network evolution. This implies communities discovered by AFOCS are of higher similarity to the ground-truth than

the other dynamic methods, especially in the long run. The number of communities found by all methods are reported in Figure 3-10. Of course, the closer these detected numbers of communities to the ground-truth, the better the method are believed to be. As revealed in the subfigures of Figure 3-10, these quantities discovered by AFOCS tend to closely approach the actually numbers, even when the mixing rates are high (right figures). The highest similarity between these numbers of communities is possibly the best explanation for the high NMI scores of AFOCS over the other competitors.

We next take a look at the running time of all methods in these synthesized networks. AFOCS requires at most 5 seconds to finish updating each network snapshot whereas FacetNet asks for more than 25 seconds (5x more time consuming) in the networks with just 5000 nodes. iLCD and OSLOM also perform fast in these generated datasets; however, the similarity of the detected communities and the ground-truth is surprisingly poor, as revealed from the results. Therefore, in terms of dynamic approaches, we strongly believe that AFOCS achieves competitive community detection results in a timely manner. These results also provide us the confidence when applying AFOCS to analyze real-world networks.

CHAPTER 4
COMMUNITY STRUCTURE DETECTION USING NONNEGATIVE MATRIX FACTORIZATION

In this chapter, analyze two approaches, namely iSNMF and iANMF, for effectively identifying social network communities using Nonnegative Matrix Factorization (NMF) with I-divergence (Kullback-Leibler divergence) as the cost function. Our approaches work by iteratively factorizing a nonnegative input matrix through derived multiplicative update rules and the Quasi-Newton method. By doing so, we can not only extract meaningful overlapping communities via soft community assignments produced by NMF but also nicely handle both directed and undirected networks with or without weights. We give the complete multiplicative update rules for factorizing $X \approx HH^{\top}$ (iSNMF problem) and $X \approx HSH^{\top}$ (iANMF problem) to effectively identify overlapping communities on social networks. These approaches are topology-independent and their solutions can be easily interpreted. We provide in detail the foundation properties as well as the proofs of correctness and convergence of both iSNMF and iANMF problems. We also propose the Quasi-Newton method to speed up the performance of iSNMF update rule. Furthermore, we validate the performance of our approaches through extensive experiments on not only synthesized datasets but also real-world networks. Empirical results show that iSNMF is among the best efficient detection methods on undirected networks while iANMF outperforms current available methods in directed networks, especially in terms of detection quality.

4.1 Problem Definition and Properties

4.1.1 Motivation for NMF in Community Detection

Let us first get some insight about how NMF can be helpful in detecting network communities, especially overlapping ones. Consider the toy network $G = (V, E)$ pictured in Figure 4-1. This network contains clear communities C_1 and C_2 having node 4 in common. The adjacency matrix X of this ideal network can be represented as

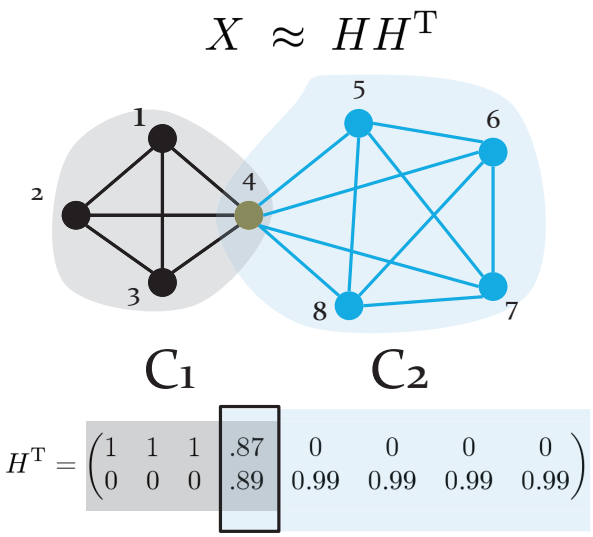


Figure 4-1. An illustrative example motivating NMF in community detection

$X = \begin{pmatrix} S_1 & 0 \\ 0 & S_2 \end{pmatrix}$, where S_1 and S_2 are 4×4 and 5×5 square matrices corresponding to C_1 and C_2, respectively. This adjacent matrix X summarizes all the network information and is the only thing we have. So, how can we derive back the appropriate communities (or the community indicators) only from this matrix? This is where NMF comes into the picture and helps. In particular, the special NMF factorization $X \approx HSH^{\mathrm{T}}$ gives us H and S as the community indicator and the community internal-strength indicator matrices, respectively. In this example, $X \approx HSH^{\mathrm{T}}$ factorization realizes $S = I_2$ and

$$H^{\mathrm{T}} = \begin{pmatrix} 1 & 1 & 1 & .87 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .89 & 0.99 & 0.99 & 0.99 & 0.99 \end{pmatrix}$$

Matrix H clearly indicates that nodes 1-3 should be in a community and nodes 5-8 should belong to another one. H also advises that node 4 should be an overlapping node due to its significant contribution to both communities. These assignments indeed reflect the true nodes' labels. In addition, matrix S indicates that each detected community attains its perfect internal strength, which intuitively agrees with the original clique structures. This illustrative example, though simple, motivates the application

of the NMF factorization $X \approx HSH^\top$ in community detection. Note that when X is symmetric (i.e., the network is undirected), S is also symmetric and thus, can be further absorbed into H by the assignment $H \leftarrow HS^{1/2}$. Hence, the problem is reduced to $X \approx HH^\top$ only when X is symmetric.

4.1.2 Problem Definitions

In order to quantify the goodness of the approximation, we use the I-divergence (Kullback-Leibler (KL) divergence) between two nonnegative matrices A and B suggested by [68] as

$$D(A||B) = \sum_{ij} \left(A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij} \right)$$

Due to the inequality $x \log x \geq x - 1 \; \forall x > 0$, it is easy to see $D(A||B)$ is lower bounded by zero and vanishes if and only if $A = B$. However, unlike the Euclidean distance, this function is not symmetric in A and B, so we refer to it as the "divergence" from A to B. The smaller the divergence between A and B, the more similar they are. Therefore, our objectives seek for the factorizations $X \approx HH^\top$ and $X \approx HS^\top H$ such that $D(X||HH^\top)$ and $D(X||HSH^\top)$ are minimized. Formally, the problems we are interested in can be stated as follows (here the little "i" comes from the I-divergence)

Problem 1 (iSNMF) Given a nonnegative **symmetric** matrix X, find a matrix $H \geq 0$ that minimizes $D_X(HH^\top) \equiv D(X||HH^\top)$

Problem 2 (iANMF) Given a nonnegative **asymmetric** matrix X, find matrices $H, S \geq 0$ that minimize $D_X(HSH^\top) \equiv D(X||HSH^\top)$

4.1.3 Properties of iSNMF and iANMF factorizations

By Lemma 13, we give important properties of iSNMF and iANMF: the divergences $D_X(HH^\top)$ and $D_X(HSH^\top)$ are convex in S only or H only; however, they are not convex in both variables together. Although the same observations have been proposed for the general NMF problem on both Frobenius and I-divergence cost functions [68], no

claim has been made particularly for the iSNMF and iANMF problems, especially on the I-divergence function.

Lemma 13. *The divergences $D_X(HH^\mathsf{T})$ and $D_X(HSH^\mathsf{T})$ in iSNMF and ANMF are convex in H or S only but not in both S and H together.*

Proof. (Convexity in S) Suppose H is a fixed matrix. For any number $\alpha, \beta \in [0, 1]$ and $\alpha + \beta = 1$, we have

$$D_X(H(\alpha S_1 + \beta S_2)H^\mathsf{T}) \leq \alpha D_X(HS_1H^\mathsf{T}) + \beta D_X(HS_2H^\mathsf{T}),$$

if and only if

$$-\sum_{ij} \log\left(\alpha[HS_1H^\mathsf{T}]_{ij} + \beta[HS_2H^\mathsf{T}]_{ij}\right) \leq -\alpha \sum_{ij} \log\left[HS_1H^\mathsf{T}\right]_{ij} - \beta \sum_{ij} \log\left[HS_2H^\mathsf{T}\right]_{ij}$$

for any matrices $S_1, S_2 \geq 0$. The later inequality holds true due to the convexity of $-\log()$ function and Jensen's inequality. Thus, $D_X(HSH^\mathsf{T})$ is convex in S when H is fixed.

(Convexity in H) Assume S is a fixed matrix. Rewrite

$$D_X(HSH^\mathsf{T}) = \sum_{ij} X_{ij}(\log X_{ij} - 1) - \sum_{ij} X_{ij} \log\left[HSH^\mathsf{T}\right]_{ij} + \sum_{ij}[HSH^\mathsf{T}]_{ij}$$

. Since the first term is a constant and $-\log()$ is a convex function, we need to show that the last term is also convex in H. Let $f(H) = \sum_{ij}[HSH^\mathsf{T}]_{ij}$. Now,

$$\begin{aligned}\alpha f(H_1) + \beta f(H_2) - f(\alpha H_1 + \beta H_2) &= \alpha\beta \sum_{ij} \left([H_1SH_1^\mathsf{T}]_{ij} - [H_2SH_1^\mathsf{T}]_{ij} - [H_1SH_2^\mathsf{T}]_{ij} + [H_2SH_2^\mathsf{T}]_{ij}\right) \\ &= \alpha\beta \sum_{ij} [(H_1 - H_2)S(H_1 - H_2)^\mathsf{T}]_{ij} \geq 0\end{aligned}$$

(since $S \geq 0$ and $\sum_{ij}[AA^\mathsf{T}]_{ij} \geq 0$ for any matrix A). This implies the convexity in H of $D_X(HSH^\mathsf{T})$.

The convexity of H in iSNMF is derived similarly as above when S is similar to I, the identity matrix. The nonconvexity in both S and H follows from the general NMF case [68]. □

The above properties are nontrivial since they tell us it is unrealistic to solve either iSNMF or iANMF problem for global minima, and consequently give us the hope to use other techniques such as Project Gradient [69], Quasi-Newton [70] or particularly the Alternating Lease Square (ALS) [71] methods to quickly find a local minima. However, by Lemma 14, we show that for iSNMF and iANMF problems, employing the traditional ALS does not provide any speed up since we can neither independently update the columns of S nor H at the same time, thus prevent the employment of this technique to our problems.

Lemma 14. *Employing ALS method does not provide any speed up to either iSNMF or iANMF.*

Proof. Let us first review the ALS method's working mechanism on the general NMF problem $X \approx WH$. Given $X \geq 0$, the ALS method does the following steps [72]

1. Randomly initialize $W^1_{ia} \geq 0, H^1_{bj} \geq 0, \quad \forall i, a, b, j$
2. For $k = 1, 2, \ldots$ alternatively update W^{k+1} and H^{k+1} by $W^{k+1} = \arg\min_{W \geq 0} D_X(WH^k)$, and $H^{k+1} = \arg\min_{H \geq 0} D_X(W^{k+1}H)$;

The main idea of the ALS method is to solve each minimization problem as the collection of several non-negative independent least square problems, due to the uncorrelated relationship between W and H. For instance, one can write $H^{k+1} = \arg\min_{H \geq 0} D(X||W^{k+1}H)$ as H^{k+1}'s jth column $= \min_{\mathbf{h} \geq 0} D(\mathbf{x}||W^{k+1}\mathbf{h})$, where $\mathbf{x}$ is the jth column of X and $\mathbf{h}$ is a column vector of appropriate size. Therefore, each sub-minimization problem requires only the values of a specific column and consequently can be done in a parallel manner. Since H and $H^\top$ are strongly related, it is inappropriate to apply ALS method to iSNMF problem. For $ANMF$ problem, we first note that $[HSH^\top]_{ij} = \sum_{tk} H_{ik}H_{jt}S_{kt}$, which implies an entry in $HSH^\top$ already requires all values of S even when H is fixed. Therefore, should one try to update a single column of S independently as suggested in S-phase of the ALS method, he has to repeatedly

solve for all elements S_{kt}'s, which may incur even more computational requirements. Thus, the conclusion follows. □

4.2 The Update Rule for iSNMF

4.2.1 Multiplicative Update Rule

We present our solution for iSNMF when the input matrix X is symmetric. Formally, given a nonnegative symmetric matrix X of size $n \times n$ and an integer number $K \ll n$, we need to find a nonnegative matrix H of size $n \times K$ such that $D_X(HH^\top) \equiv D(X||HH^\top)$ is minimized.

We solve this problem using the Karush-Kuhn-Tucker (KKT)[73] conditions. In particular, we introduce the Lagrange multipliers α_{ij} for the constraints $H_{ij} \geq 0$ and consider the objective function $J = D(X||HH^\top) - \sum_{ij} \alpha_{ij} H_{ij}$, or,

$$J = \sum_{ij} \left(X_{ij} \log \frac{X_{ij}}{[HH^\top]_{ij}} - X_{ij} + [HH^\top]_{ij} \right) - \sum_{ij} \alpha_{ij} H_{ij}$$

The KKT conditions require

$$\frac{\partial J}{\partial H_{ab}} = 0 \text{ (or } \frac{\partial D_X(]HH^\top)}{\partial H_{ab}} = \alpha_{ab})$$

as the optimality condition and

$$\alpha_{ab} H_{ab} = 0$$

as a complementary slackness condition for any H_{ab}.

For the ease of computation, we construct the derivative matrix $HH^\top$ with respect to H_{ab} in Figure 4-2. For each position (a, b), this derivative matrix is zero elsewhere except for the a^{th} column and a^{th} row whose elements are from the b^{th} column of H. Using this matrix, we obtain

$$\frac{\partial D_X(HH^\top)}{\partial H_{ab}} = 2 \left(\sum_k H_{kb} - \sum_k H_{kb} \frac{X_{ak}}{[HH^\top]_{ak}} \right). \tag{4–1}$$

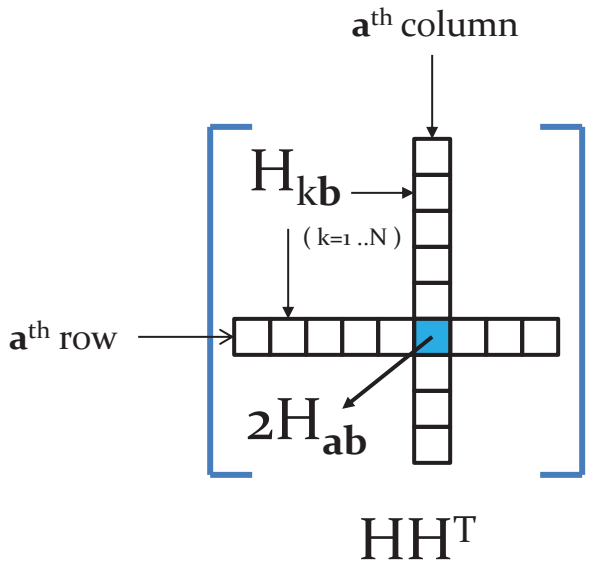


Figure 4-2. The partial derivative matrix of $HH^{\top}$ with respect to H_{ab}.

Hence, the optimality condition implies

$$\alpha_{ab} = 2\Big(\sum_k H_{kb} - \sum_k H_{kb}\frac{X_{ak}}{[HH^{\top}]_{ak}}\Big),$$

and thus, the complementary slackness condition requires

$$2\Big(\sum_k H_{kb} - \sum_k H_{kb}\frac{X_{ak}}{[HH^{\top}]_{ak}}\Big)H_{ab} = 0,$$

which suggests the following update rule

$$H_{ab} \leftarrow H_{ab}\frac{\sum_k H_{kb}X_{ak}/[HH^{\top}]_{ak}}{\sum_t H_{tb}}. \tag{4–2}$$

In terms of projected gradient method, the rule above can be obtained by using the update rule

$$H_{ab} \leftarrow H_{ab} - \nu_{ab}\frac{\partial D_X(HH^{\top})}{\partial H_{ab}},$$

with the magnitude ν_{ab} set to some appropriate small positive number. Here, setting

$$\nu_{ab} = \frac{H_{ab}}{2\sum_t H_{tb}}$$

leads to the same update rule as (4–2).

The iSNMF community detection algorithm is described in Alg. 13. Here, n_0 is the maximum number of iterations, ϵ is the allowed threshold for the quality of iSNMF approximation and α is a given scale to determine community memberships. We assume that K, the number of communities, is predetermined or given as part of the input. Also, the choice of α will be described later.

Algorithm 13 SNMF for community detection

Input: Undirected, unweighted (weighted) adjacent matrix X, K, n_0, ϵ, α;
Output: Community indicator matrix H;

Initialize H to be a random nongnegative matrix;
$iter \leftarrow 0$;
while $(iter \leq n_0)$ and $(D_X(HH^{\mathsf{T}}) > \gamma)$ **do**
    Update $H_{ab} \leftarrow H_{ab} \frac{\sum_k H_{kb} X_{ak} / [HH^{\mathsf{T}}]_{ak}}{\sum_t H_{tb}}$;
    $iter \leftarrow iter + 1$;
end while
% Inferring community labels from H%
$C_b \leftarrow \emptyset \; \forall b = 1 \ldots K$;
$P \leftarrow normalized(H)$;
for $b \leftarrow 1 \ldots p$ **do**
    if $P(a, b) \geq \alpha * max(P(a, :))$ **then**
        $C_b \leftarrow C_b \cup \{a\}$;
    end if
end for

Remark

In contrast to those update rules found in [68], we have shown an important fact: These rules can be derived similarly for this special case. However, our multiplicative update rule (4–2) is not trivial in the sense that we can obtain the convergence proof for our proposed rule whereas one may find it inappropriate to adapt the proof of [68][74] which assumed absolutely no correlation between W and H.

Analysis

We provide the convergence analysis for our proposed update rule (4–2) using an auxiliary function defined as follow:

(Auxiliary function) $G(h, \widetilde{h})$ is the auxiliary function for $F(h)$ if the conditions $G(h, \widetilde{h}) \geq F(h)$ and $G(h, h) = F(h)$ are satisfied

Lemma 15. *[68] If G is an auxiliary function, then F is nonincreasing under the update $h^{t+1} = \arg\min_h G(h, \widetilde{h})$.*

To prove the convergence of the proposed multiplicative update rule, we construct an auxiliary function $G(H, \widetilde{H})$ of $F(H) \equiv D_X(HH^\top)$ as follow

$$G(H, \widetilde{H}) = \sum_{ij} \left(X_{ij} \log X_{ij} - X_{ij} + [HH^\top]_{ij}\right) - \sum_{ijk} X_{ij} \frac{H_{ik}\widetilde{H}_{jk}}{\sum_t H_{it}\widetilde{H}_{jt}} \left(\log H_{ik}H_{jk} - \log \frac{H_{ik}\widetilde{H}_{jk}}{\sum_t H_{it}\widetilde{H}_{jt}}\right)$$

Theorem 4.1. *The divergence $D_X(HH^\top)$ is nonincreasing under the update rule (4–2) and is invariant when H is at its stationary point of the divergence.*

Proof. When $\widetilde{H} = H$, it is easy to verify that $G(H, H) = F(H)$, thus we only need to check $G(H, \widetilde{H}) \geq F(H)$. Now, $G(H, \widetilde{H}) \geq F(H)$ iff

$$\begin{aligned}
&- \sum_{ijk} X_{ij} \frac{H_{ik}\widetilde{H}_{jk}}{\sum_t H_{it}\widetilde{H}_{jt}} \left(\log H_{ik}H_{jk} - \log \frac{H_{ik}\widetilde{H}_{jk}}{\sum_t H_{it}\widetilde{H}_{jt}}\right) \\
&\geq - \sum_{ij} X_{ij} \log [HH^\top]_{ij} = - \sum_{ij} X_{ij} \log \left(\sum_k H_{ik}H_{jk}\right) \\
\iff &- \sum_{ijk} X_{ij} \frac{H_{ik}\widetilde{H}_{jk}}{\sum_t H_{it}\widetilde{H}_{jt}} \left(\log \frac{H_{ik}H_{jk} \times \sum_t H_{it}\widetilde{H}_{jt}}{H_{ik}\widetilde{H}_{jk}}\right) \\
&\geq - \sum_{ij} X_{ij} \log \left(\sum_k H_{ik}H_{jk}\right).
\end{aligned}$$

To prove the above inequality, we apply Jensen's inequality to the convex function $-\log\left(\sum_k H_{ik}H_{jk}\right)$, yielding

$$-\log \sum_k \alpha_k \frac{H_{ik}H_{jk}}{\alpha_k} \leq - \sum_k \alpha_k \log \frac{H_{ik}H_{jk}}{\alpha_k},$$

where $\alpha_k \equiv \alpha_{ijk} = \frac{H_{ik}\widetilde{H}_{jk}}{\sum_t H_{it}\widetilde{H}_{jt}}$. It is obvious that α_k's are nonnegative and sum up to unity. Thus, we have $G(H, \widetilde{H}) \geq F(H)$. Taking the derivative of $G(H, \widetilde{H})$ with respect to H also gives the same update rule (4–2). □

4.2.2 Quasi-Newton Method for iSNMF

One of the problems with the multiplicative update rule is its slow convergence, i.e., it does converge to (possible) stationary point but may be slow, taking more iterations and time, as well as easily getting into local minima trap [72]. One way to speed up the convergence is to adjust the learning rate in a sequential manner, using the second-order estimate of the objective function, e.g. the Quasi-Newton method. In [70], the authors already addressed this method for the general NMF $X \approx WH$ but with the uncorrelated relationship assumption between W and H. Obviously, that assumption does not hold when $X \approx HH^\top$ and hence, it is not trivial to derive proper Quasi-Newton formulation for iSNMF problem. In fact, we show that the second-order, or Hessian, matrix $\mathcal{H}_{D_X}^{(H)}$ of iSNMF is much different from that of the general NMF.

The general Quasi-Newton method, when applied to iSNMF problem, takes the form

$$H \leftarrow \max\left\{H - [\mathcal{H}_{D_X}^{(H)}]^{-1}\frac{\partial D_X}{\partial H}, \epsilon\right\}, \tag{4–3}$$

where D_X is short for $D_X(HH^\top)$, $\frac{\partial D_X}{\partial H}$ is the $n \times K$ first-order matrix of $D_X(HH^\top)$ w.r.t H, $\mathcal{H}_{D_X}^{(H)}$ is the $nK \times nK$ second-order derivative (or Hessian) matrix of D_X w.r.t to H and ϵ is a small nonnegative number to enforce the nonnegativity of H. Thanks to equation (4–1), the first-order derivative matrix $\frac{\partial D_X}{\partial H}$ can be found as

$$\frac{\partial D_X}{\partial H} = 2\left(\mathbf{1} - X./HH^\top\right)H,$$

where $\mathbf{1}$ is a $N \times N$ matrix of all 1's and $./$ is the Hadamard (element-wise) division. For any pair (i, j), the Hessian matrix $\mathcal{H}_{D_X}^{(H)}$ can be found by: $[\mathcal{H}_{D_X}^{(H)}]_{ij} = \frac{\partial D_X}{\partial H_{ij} H_{ab}} =$

$$\begin{cases} 2\left(1 - \frac{X_{ii}([HH^\top]_{ii} - 2H_{ij}^2)}{[HH^\top]_{ii}^2} + \sum_{k\neq i}\frac{H_{ik}^2 X_{ik}}{[HH^\top]_{ik}^2}\right) & a = i, b = j \\ 2\left(\frac{2H_{ib}H_{ij}X_{ii}}{[HH^\top]_{ii}^2} + \sum_{k\neq i}\frac{H_{kb}H_{kj}X_{ik}}{[HH^\top]_{ik}^2}\right) & a = i, b \neq j \\ 2\left(1 - \frac{X_{ak}([HH^\top]_{ai} - H_{ij}H_{aj})}{[HH^\top]_{ai}^2}\right) & a \neq i, b = j \\ 2\left(\frac{H_{ib}H_{aj}X_{ai}}{[HH^\top]_{ai}^2}\right) & a \neq i, b \neq j \end{cases} \tag{4–4}$$

There are two important differences between the Hessian $\mathcal{H}_{NMF}$ for the general case [70] and $\mathcal{H}_{D_X}^{(H)}$. Firstly, $\mathcal{H}_{NMF}$'s elements are zeros everywhere except when $a = i, b = j$ whereas $\mathcal{H}_{D_X}^{(H)}$ obtains values for all combinations of a, b, i and j. Secondly, due to its sparseness, H_{NMF} can be written under matrix block form while $\mathcal{H}_{D_X}^{(H)}$ might not be, particularly when it is a full matrix. Therefore, updating H in iSNMF problem is much more complicated than usual since finding $[\mathcal{H}_{D_X}^{(H)}]^{-1}$ in (4–3) shall require more numerical computations.

The authors in [70] also proposed a numerical technique to overcome the ill-conditioned Hessian matrix and to speed up the computing process, which we find it useful when applied to our problems. Here, we briefly state their technique so that the paper is self-contained (note that (4–5) and (4–6) are not our equations): To reduce the computational cost, the inversion of the Hessian is replaced with the Q-less QR factorization computed by LAPACK. The final form of the Quasi-Newton method is

$$H \leftarrow \max\left\{H - \gamma R_H | W_H, \epsilon\right\} \tag{4–5}$$

$$W_H = Q_H^{\top} \frac{\partial D_X}{\partial H}, \quad Q_H R_H = \mathcal{H}_{D_X}^{(H)} + \lambda I_H \tag{4–6}$$

where I_H is the $nK \times nK$ identical matrix, $\gamma = 10^{-12}$ and $\lambda = 0.9$ are the small fixed regularization and the relax parameters, respectively. The $|$ operator in (4–5) means the Gaussian elimination.

4.3 Update Rules for iANMF

4.3.1 Multiplicative Update Rules

In this section, we present our solution for the iANMF problem when X is not symmetric. Formally, given a nonegative asymmetric matrix X of size $n \times N$, we find nonnegative matrices H and S of size $n \times K$ and $K \times K$, respectively, such that $D_X(HSH^{\top}) \equiv D(X||HSH^{\top})$ is minimized. We again solve this problem using the KKT conditions by introducing the Lagrange multipliers α_{ij} and β_{ij} for the constraints

$H_{ij} \geq 0$ and $S_{ij} \geq 0$, respectively, and then consider the objective function $J = D(X||HSH^\top) - \sum_{ij} \alpha_{ij} H_{ij} - \sum_{ij} \beta_{ij} S_{ij}$. Equivalently, J can be written as

$$J = \sum_{ij} \left(X_{ij} \log \frac{X_{ij}}{[HSH^\top]_{ij}} - X_{ij} + [HSH^\top]_{ij} \right) - \sum_{ij} \alpha_{ij} H_{ij} - \sum_{ij} \beta_{ij} S_{ij}.$$

The KKT conditions require

$$\frac{\partial J}{\partial H_{ab}} = 0 \text{ and } \frac{\partial J}{\partial S_{ab}} = 0,$$

or equivalently,

$$\frac{\partial D_X(HSH^\top)}{\partial H_{ab}} = \alpha_{ab} \text{ and } \frac{\partial D_X(HSH^\top)}{\partial S_{ab}} = \beta_{ab}$$

as the optimality conditions, as well as

$$\alpha_{ab} H_{ab} = 0 \text{ and } \beta_{ab} S_{ab}$$

as a complementary slackness condition for any H_{ab} and S_{ab}. For the ease of computation, we construct the matrix for finding the derivative of an entry $[HSH^\top]_{ij}$ with respect to any H_{ab} in Figure 4-3. Here $r(\mathbf{A}, i)$ and $c(\mathbf{B}, j)$ mean the i^{th} row of A and j^{th} column of B, respectively. Elements outside of the plotted column and row are zeros. The elements of this matrix are zeros elsewhere except for the a^{th} column and a^{th} row. Using this conventional partial derivative matrix, we obtain

$$\frac{\partial \sum_{ij} X_{ij} \log [HSH^\top]_{ij}}{\partial H_{ab}} = \sum_k X_{ka} \frac{[HS]_{kb}}{[HSH^\top]_{ka}} + \sum_k X_{ak} \frac{[SH^\top]_{bk}}{[HSH^\top]_{ak}}$$

$$\text{and} \qquad \frac{\partial \sum_{ij} [HSH^\top]_{ij}}{\partial H_{ab}} = \sum_k ([HS]_{kb} + [SH^\top]_{bk})$$

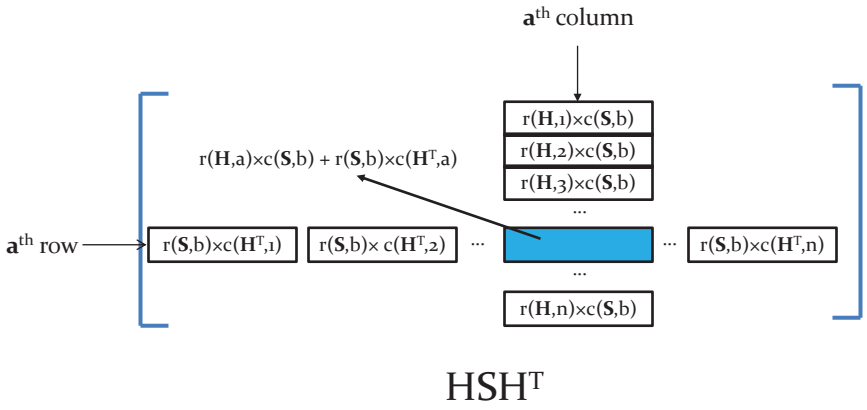


Figure 4-3. The partial derivative matrix of $HSH^\top$ with respect to H_{ab}.

Therefore,

$$\begin{aligned}\frac{\partial D_X(HSH^\top)}{\partial H_{ab}} &= \frac{-\partial \sum X_{ij} \log [HSH^\top]_{ij} + \sum [HSH^\top]_{ij}}{\partial H_{ab}} \\ &= -\sum_k X_{ka} \frac{[HS]_{kb}}{[HSH^\top]_{ka}} - \sum_k X_{ak} \frac{[SH^\top]_{bk}}{[HSH^\top]_{ak}} \\ &\quad + \sum_k ([HS]_{kb} + [SH^\top]_{bk}). \end{aligned} \tag{4–7}$$

The optimality condition $\frac{\partial D_X(HSH^\top)}{\partial H_{ab}} = \alpha_{ab}$ and the complementary slackness condition $\alpha_{ab} H_{ab} = 0$ together give the following update rule for H_{ab}

$$H_{ab} \leftarrow H_{ab} \left(\frac{\sum_k X_{ka} [HS]_{kb} / [HSH^\top]_{ka}}{\sum_t [HS]_{tb} + [SH^\top]_{bt}} + \frac{\sum_k X_{ak} [SH^\top]_{bk} / [HSH^\top]_{ak}}{\sum_t [HS]_{tb} + [SH^\top]_{bt}} \right) \tag{4–8}$$

Alternatively, this update rule can also be achieved by using projected gradient method, in particular by updating

$$H_{ab} \leftarrow H_{ab} - \nu_{ab} \frac{\partial D_X(HSH^\top)}{\partial H_{ab}}$$

with the magnitude

$$\nu_{ab} = \frac{H_{ab}}{\sum_t ([HS]_{tb} + [SH^\top]_{bt})}.$$

Now we give the multiplicative update rule for any S_{ab}. The partial derivative of $D_X(HSH^\top)$ w.r.t S_{ab} is derived as

$$\frac{\partial D_X(HSH^\top)}{\partial S_{ab}} = \sum_{st} H_{sa}H_{tb} - \sum_{st} X_{st}\frac{H_{sa}H_{tb}}{[HSH^\top]_{st}}$$

The KKT conditions $\frac{\partial D_X(HSH^\top)}{\partial S_{ab}} = \beta_{ab}$ and $\beta_{ab}S_{ab} = 0$ together imply the following update rule for S_{ab}

$$S_{ab} \leftarrow S_{ab}\frac{\sum_{st} H_{sa}H_{tb}(X_{st}/[HSH^\top]_{st})}{\sum_{st} H_{sa}H_{tb}} \tag{4–9}$$

Alternatively, this rule can be derived by the projected gradient method

$$S_{ab} \leftarrow S_{ab} - \nu_{ab}\frac{\partial D_X(HSH^\top)}{\partial S_{ab}}$$

with magnitude

$$\gamma_{ab} = \frac{S_{ab}}{\sum_{st} H_{sa}H_{tb}}.$$

The iANMF community detection is presented in Alg. 14. The parameters and their meanings in this case are similar to those described in the $SNMF$ case.

Algorithm 14 iANMF for community detection

Input: Directed, unweighted (weighted) adjacent matrix X, K, n_0, ϵ, α;
Output: Matrices H and S and the inferred community labels;

Initialize H and S to be a random nongnegative matrices;
$iter \leftarrow 0$
while $(iter \leq n_0)$ and $(D_X(HH^\top) > \gamma)$ **do**
    Update H_{ab} based on equation (4–8);
    Update S_{ab} based on equation (4–9);
    $iter \leftarrow iter + 1$;
end while
% Inferring community labels from H%
$C_b \leftarrow \emptyset\ \forall b = 1...K$;
$P \leftarrow normalized(H)$;
for $b \leftarrow 1...K$ **do**
    if $P(a, b) \geq \alpha * max(P(a, :))$ **then**
        $C_b \leftarrow C_b \cup \{a\}$;
    end if
end for

Summary

With the multiplicative update rules (4–8) and (4–9), we give the complete steps for iteratively solving iANMF problem with respect to the I-divergence. These rules are different from what have been discovered in prior studies and, to our knowledge, have not yet been derived in the literature. Thus, they are our contributions in this paper.

Analysis

We first show the following result

Theorem 4.2. *At the stationary point* (H, S) *of* $D_X(HSH^\top)$*, KKT conditions imply that*

$$\sum_{st} X_{st} = \sum_{st} [HSH^\top]_{st}$$

Proof. We show that the condition $S_{ab}\frac{\partial D_X(HSH^\top)}{\partial S_{ab}} = 0$ imply the above equality. In particular, the KKT condition equals to

$$S_{ab}\sum_{st} H_{sa}H_{tb} = S_{ab}\sum_{st} X_{st}\frac{H_{sa}H_{tb}}{[HSH^\top]_{st}}.$$

Summing over all a and b of the LHS gives

$$\sum_{ab} S_{ab}\sum_{st} H_{sa}H_{tb} = \sum_{st}\sum_{ab} H_{sa}S_{ab}H_{tb} = \sum_{st}[HSH^\top]_{st}.$$

Similarly, summing over all a and b of the RHS gives

$$\sum_{ab}\sum_{st} X_{st}\frac{H_{sa}Htb}{[HSH^\top]_{st}} = \sum_{st} X_{st}\frac{[HSH^\top]_{st}}{[HSH^\top]_{st}} = \sum_{st} X_{st}.$$

Therefore, the equality follows. □

We next analyze the convergence analysis of our proposed rules (4–8) and (4–9). By using appropriate auxiliary functions $G(S, \widetilde{S})$ and $G(H, \widetilde{H})$, one can show the following

Theorem 4.3. *The divergence* $D(X||HSH^\top)$ *is nonincreasing under the update rules (4–8) and (4–9) and is invariant if and only if* S *and* H *are at their stationary points in the divergence.*

Proof. The proof of convergence for the two update rules (4–8) and (4–9) is similar to Theorem 4.1. Let us first define two functions

$$G(S, \widetilde{S}) = \sum_{ij} X_{ij}(\log X_{ij} - 1) + \sum_{ij}[HSH^{\top}]_{ij} - \sum_{ij} X_{ij}\beta_{ijuv}(\log H_{iv}S_{vu}H_{ju} - \log \beta_{ijuv}),$$

$$\text{and } G(H, \widetilde{H}) = \sum_{ij} X_{ij}(\log X_{ij} - 1) + \sum_{ij}[HSH^{\top}]_{ij} - \sum_{ij} X_{ij}\xi_{ijuv}(\log H_{iv}S_{vu}H_{ju} - \log \xi_{ijuv}).$$

$$\text{where } \beta_{ijuv} = \frac{H_{iv}\widetilde{S}_{vu}H_{ju}}{\sum_{st} H_{it}\widetilde{S}_{ts}H_{js}}, \xi_{ijuv} = \frac{H_{iv}S_{vu}\widetilde{H}_{ju}}{\sum_{st} H_{it}S_{ts}\widetilde{H}_{js}}.$$

It is clear that each β_{ijuv}'s and ξ_{ijuv}'s are nonzero and sum up to unity. We now prove the convergence of rule (4–9) for S when matrix H is fixed. Let $F(S) = D_X(HSH^{\top})$. We show that $G(S, \widetilde{S})$ defined above is an auxiliary function for $F(S)$. When $\widetilde{S} = S$, one can verify that $G(S, \widetilde{S}) = F(S)$, thus we need to check $G(S, \widetilde{S}) \geq F(H)$. This inequality equals to

$$-\sum_{ij} X_{ij} \log [HSH^{\top}]_{ij} \leq -\sum_{ij} X_{ij}\beta_{ijuv}(\log H_{iv}S_{vu}H_{ju} - \log \beta_{ijuv})$$

By the definition of β_{ijuv}, one can rewrite the above inequality as

$$-\log \sum_{ij} \beta_{ijuv} \frac{H_{iv}S_{uv}H_{ju}}{\beta_{ijuv}} \leq -\sum_{ij} \beta_{ijuv} \log \frac{H_{iv}S_{vu}H_{ju}}{\beta_{ijuv}}$$

which generally holds true due to Jensen's inequality and the convexity of $-\log()$ function. Now, taking the derivative of $G(S, \widetilde{S})$ with respect to S gives the update rule (4–9). The proof for H can be obtained in a very similar manner with and thus, is omitted here. □

4.4 Experimental Results

In this section, we first validate our approaches on different synthesized networks with known ground-truths, and then present our findings on real-world traces including the Enron email [31] and Facebook social network [75]. To certify our performance, we

compare the results to two NMF methods proposed in [10] (i.e., wSNMF and wANMF), and the recently suggested Bayesian NMF [14] (i.e., Bayesian method).

Our methods require the number of communities K as an input parameter. We stress that determining this quantity is not the main focus of NMF-based detection methods since almost all of them rely on a predefined K to discover the network communities, as commonly observed in [10][11][36]. Thefore, this quantity K is predetermined using a procedure suggested in [76], which has been shown to well-predict the number of network communities in a timely manner. We also use this value as input for wSNMF and wANMF. For the Bayesian method, we keep the default settings as provided in its deliverable.

4.4.1 Empirical Results on Synthesized Networks

Of course, the best way to evaluate our approaches is to validate them on real-world networks with known community structures. Unfortunately, we often do not know that structures beforehand, or such structures cannot be easily mined from the network topologies. Although synthesized networks might not reflect all the statistical properties of real ones, they can provide us the known ground-truths via planted communities and the ability to vary other network parameters such as sizes, densities and overlapping levels, etc. Testing community detection methods on generated data has becomes a usual practice that is widely accepted in the field [3]. Therefore, running iSNMF and iANMF on synthesized networks not only certifies their performance but also provides us the confidence to their behaviors when applied to real-world traces.

Set up: We use the well-known LFR overlapping benchmark [77] to generate 22 weighted directed and undirected testbeds. Generated networks follow the power-law degree distribution and contain embedded overlapping communities of varying sizes that capture the internal characteristics of real-world networks. Parameters are: the number of nodes $N = 1000$, the mixing parameter $\mu = 0.1$ and 0.3 controlling the overall sharpness of the community structure, the weight mixing $\mu_w = 0.1$ and 0.3, the minimum

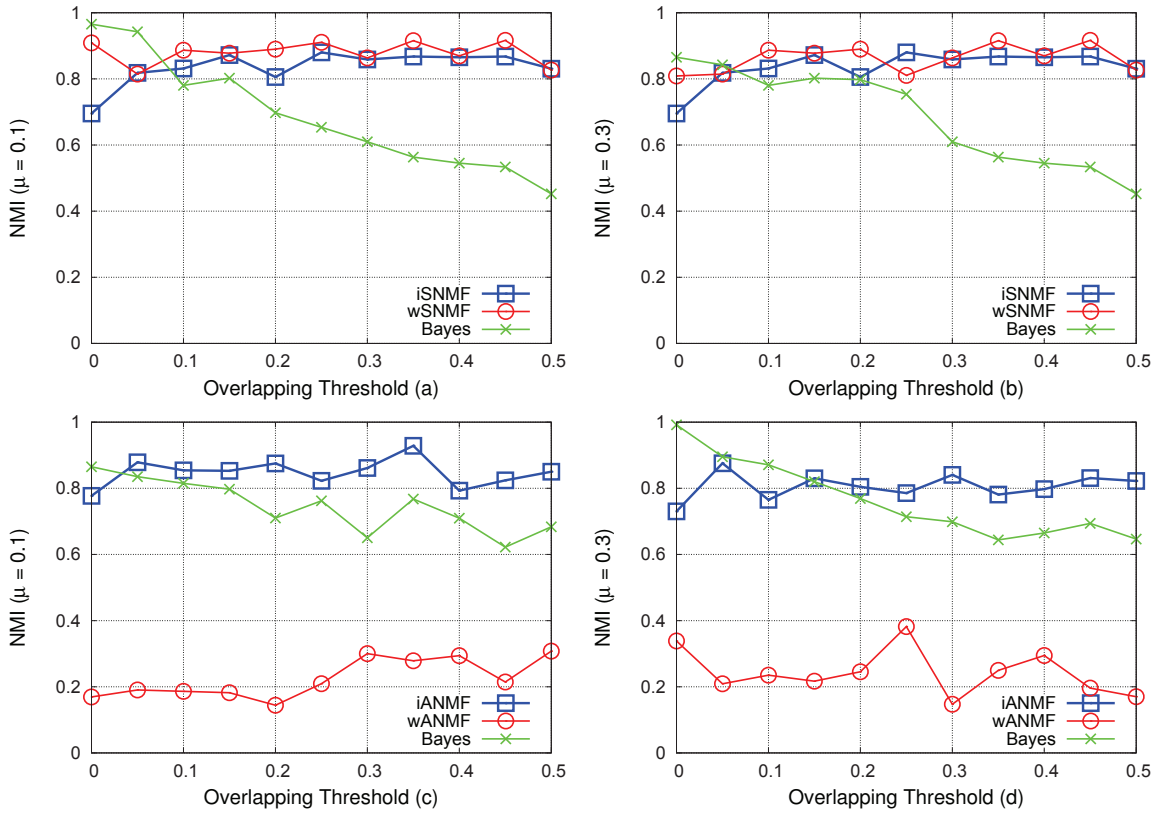


Figure 4-4. Normalized Mutual Information scores on synthesized networks

and maximum community sizes $c_{min} = 10$ and $c_{max} = 50$, the maximum memberships of a node $o_m = 2$, and the overlapping fraction $\gamma \in [0, 0.5]$ measuring the fraction of nodes with memberships in more than communities. We set the number of iterations to 400 in all methods and run 22 tests 100 times for consistency.

Metric: To measure the similarity between detected communities and the embedded ground-truth, we evaluate Generalized Normalized Mutual Information (NMI) [46]. $NMI(U, V)$ is 1 if structures U and V are identical and is 0 if they are totally separated. This is the most important metric for a community detection algorithm because it indicates how good the algorithm is in comparison with the true communities. The higher the NMI value to the ground-truth, the better.

Detection quality: As depicted in Figure 4-4, our approaches iSNMF and iANMF achieve the most stable and competitive (if not to say the best) NMI scores on both

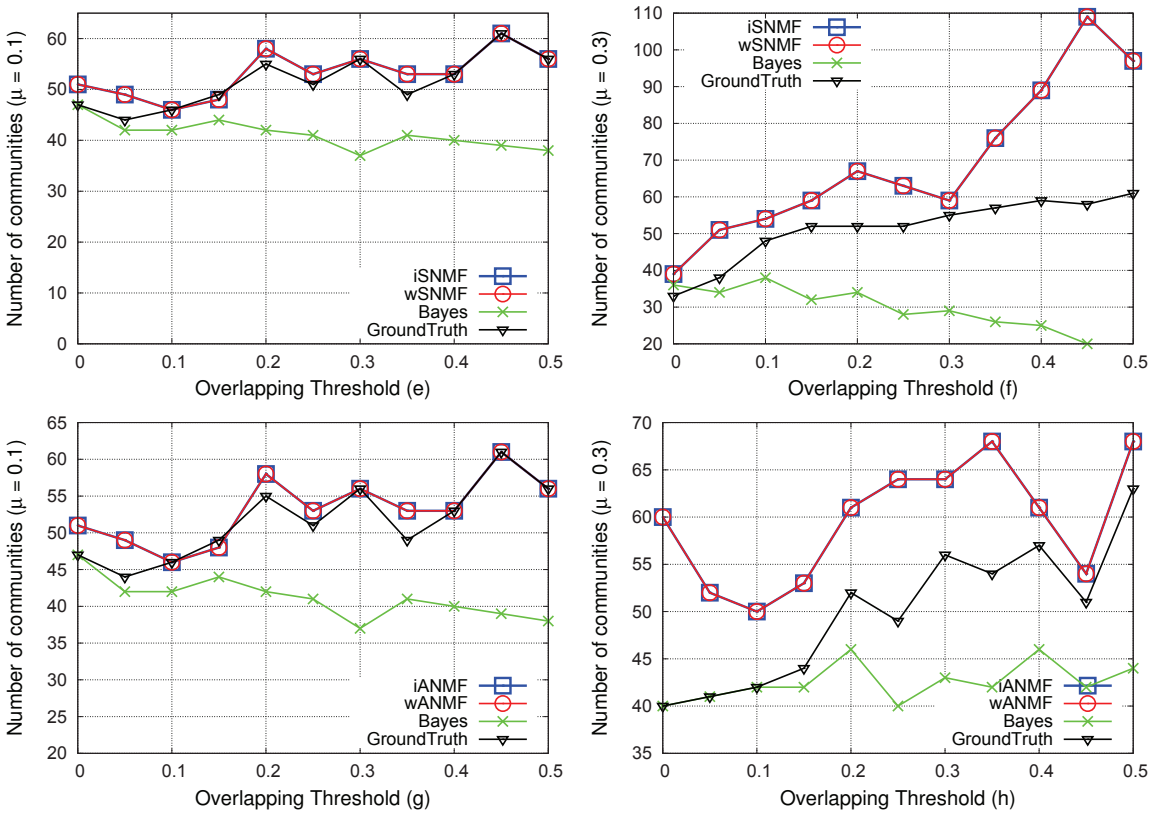


Figure 4-5. Number of communities on synthesized networks

weighted directed and undirected networks. In particular, on undirected networks (top 2 figures), NMI scores produced by iSNMF are highly competitive to those of wSNMF and are up to 84% better than those returned by the Bayesian method. Moreover, its NMI scores still remains high and balance as the mixing overlapping ratio γ increases. This means the communities discovered by iSNMF are consistently of high similarity to the ground-truth even when more and more network communities are overlapped with each other. wSNMF also displays these properties on undirected networks; however, its performance degrades significantly on directed weighted networks, as we will discuss shortly. The Bayesian method, on the other hand, produces very low NMI values that tend to decrease quickly as γ increases. This implies communities detected by this method are not ideally coincident with the embedded ones, especially when they highly overlap with each other.

There is a close relationship between the number of communities and the identification capacity that we observed in the case of undirected networks in Figure 4-5. As revealed in its top figures, the input numbers of communities for iSNMF and iSNMF are almost identical to the ground-truth when $\mu = 0.1$ and slightly deviate from them when $\mu = 0.3$, while those of the Bayesian method are far away from the baseline. This close relationship, as a result, helps iSNMF and wSNMF to determine a proper number of basic features and consequently, indicate more appropriate community labels. However, this observation does not appear to hold for wANMF on directed networks since it performs poorly whereas our approach iANMF still performs excellently on this type of networks (Figure 4-4, bottom figures). The big gap between the Bayesian method and the ground-truth implies its built-in estimate of the number of communities could potentially mislead the factorization, thus results in its low NMI scores.

The superiority of our iANMF approach becomes more visible on directed weighted networks (Figure 4-4, bottom figures). In these figures, iANMF returns the best stable NMI values and they remain wealthy even when γ evolves, i.e., when strongly overlapped communities appear. In particular, the NMI scores returned by iANMF are more than twice those of wANMF and are up to 10% those of Bayesian method. The performance of wANMF, surprisingly, reduces to no more than half of its prior achievement even when fed with the relatively close number of true communities (bottom figures of Figure 4-5). This in turn indicates the communities discovered by wANMF are heavily deviated from and are of very low similarity to the ground-truth. Bayesian method's performance is somehow the same on these directed networks with average NMI scores tend to quickly decrease in the long run. This comparison among three NMF factorizations reveals that iSNMF and iANMF are the best ideal methods for effectively recovering the overlapped network community structures, especially on weighted and directed networks.

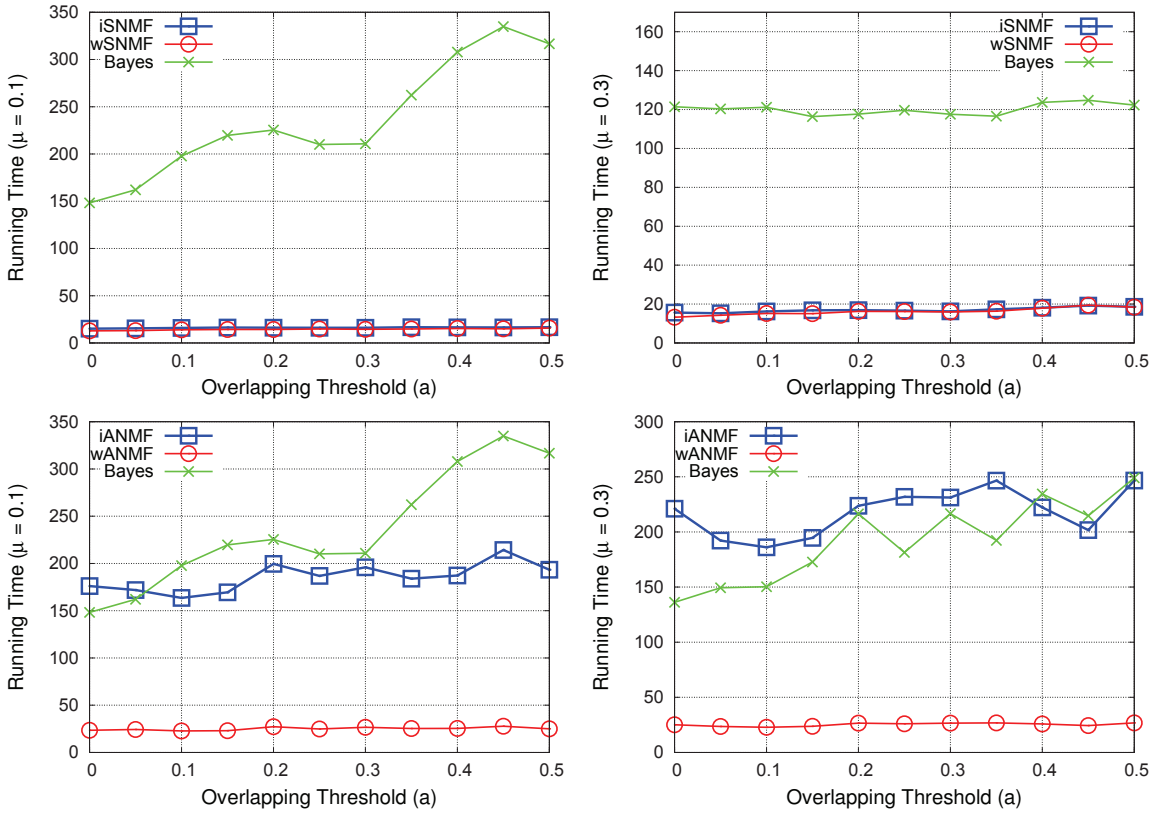


Figure 4-6. Running Time on synthesized networks

We next compare the running time of three methods. As reported in Figure 4-6, the running times of iSNMF and wSNMF on undirected networks are fairly similar to each other (at most 2s difference) and are much less than the huge time requirement of the Bayesian method. In average, the Bayesian method requires almost 200s in order to finish the test whereas iSNMF and wSNMF only ask for roughly 16s and 14s, respectively. On directed networks, iANMF requires nearly the same amount of time of the Bayesian method and much more time than wANMF. Note that this time consumption of iANMF is quite understandable because each update for S_{ab} in equation (4–9) based on the I-divergence already took $O(n^2)$ time. However, the superiority of its produced NMI scores to other competitors makes iANMF a promising approach, especially suited for those who strive to discover excellent network community structures.

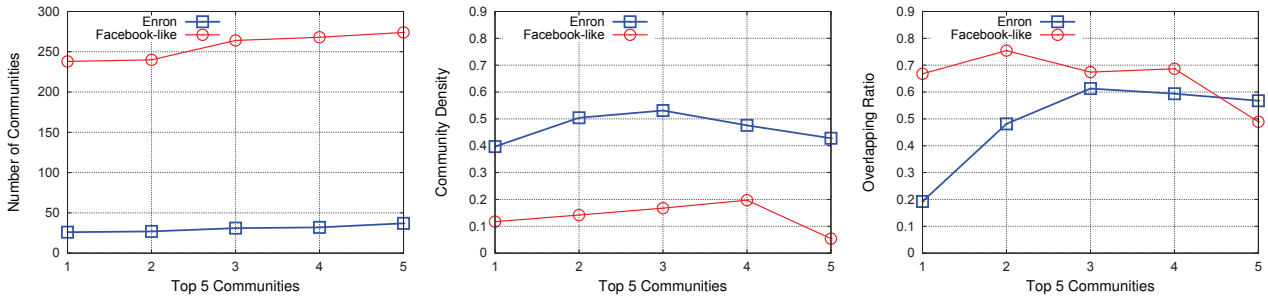


Figure 4-7. The number of communities, Internal density and Overlapping ratio of Enron email and Facebook-like datasets

In summary, comparisons among three algorithms on generated networks show that (1) iSNMF is among the best NMF methods for efficiently identify high quality overlapping communities in weighted and unweighted undirected networks (2) iANMF is the best among three methods for analyzing weighted and directed networks containing highly overlapped communities, despite it long running time. More importantly, the performance of both approaches remains healthily stable even when more and more overlapping communities are introduced. These results provide us the strong confidence when applying iSNMF and iANMF to analyze the real-world traces.

4.4.2 Results on Real Networks

We next utilize iANMF and iSNMF to analyze the real network datasets and present our findings on their overlapping structures. In particular, we choose the Enron email dataset and the Facebook-like social network [75]. The Enron email network contains email messages data from about 150 users, mostly senior management of Enron Inc., from Jan 1999 to July 2002 [31]. Each email address is represented by an unique identification number in the dataset and each link corresponds to a message sent between the sender and the receiver. The Facebook-like social network is collected from students of University of California, Irvine. The dataset contains 20296 messages sent and received among 1899 users. The number of communities inputed for Enron email and Facebook-like datasets are set to 8 and 18, respectively.

We are interested in understanding their overlapping structures and what the overlapping nodes really mean to them, particularly in the top 5 biggest communities. As revealed in Figure 4-7, the numbers of members in top 5 communities of Facebook-like network are, not surprisingly, much bigger are those of the Enron email network. However, the internal density, i.e., the inner structures of those top 5 communities in Enron emails are much stronger than those of Facebook networks. Indeed, the density values of Enron email communities are more than twice of Facebook networks. This can be explained as email communication in a work place among managers occurs much more frequently than messages on a social environment like the Facebook network.

We next investigate on the overlapping substructures of these real networks, i.e., we want to know how much they are overlapped and what the overlapped nodes mean to the communities. As described in Figure 4-7, all 5 top communities of Facebook network are highly overlapped whereas just 3 top communities of Enron email network appear to have this properties. Moreover, overlapped nodes on Facebook network tend to be active users who eagerly participate in multiple communities at the same time, i.e., they send messages to multiple friends in different groups. Overlapped nodes on Enron email network, though fewer, suggest that they potentially play vital roles in the company since most of them communicate frequently many other members in all of the communities.

CHAPTER 5
SOCIAL-AWARE ROUTING STRATEGIES IN MOBILE AD-HOC NETWORKS

In this chapter, we demonstrate the applicability of our proposed detection algorithms QCA and AFOCS as the community identification cores in forwarding and routing strategies in mobile dynamic networks. In the following paragraphs, we first present the application of QCA and then describe how AFOCS can help to improve the performance of this practical applications.

5.1 A Message Forwarding and Routing Strategy Employing QCA

In a broad view, a MANET is a dynamic wireless network with or without the underlying infrastructure, in which each node can move freely in any direction and organize itself in an arbitrary manner. Due to nodes mobility and unstable links nature of a MANET, designing an efficient routing scheme has become one of the most important and challenging problems on MANETs. Recent researches have shown that MANETs exhibit the properties of social networks [78][79][80] and social-aware algorithms for network routing are of great potential. This is due to the fact that people have a natural tendency to form groups or communities in communication networks, where individuals inside each community communicate with each other more frequent than with people outside. This social property is nicely reflected to the underlying MANETs by the existence of groups of nodes where each group is densely connected inside than outside. This resembles the idea of community structure in Mobile Ad hoc Networks.

Multiple routing strategies [79]-[81] based on the discovery of network community structures have provided significant enhancement over traditional methods. However, the community detection methods utilized in those strategies are not applicable for dynamic MANETs since they have to recompute the network structure whenever changes to the network topology are introduced, which results in significant computational costs and processing time. Therefore, employing an adaptive community structure

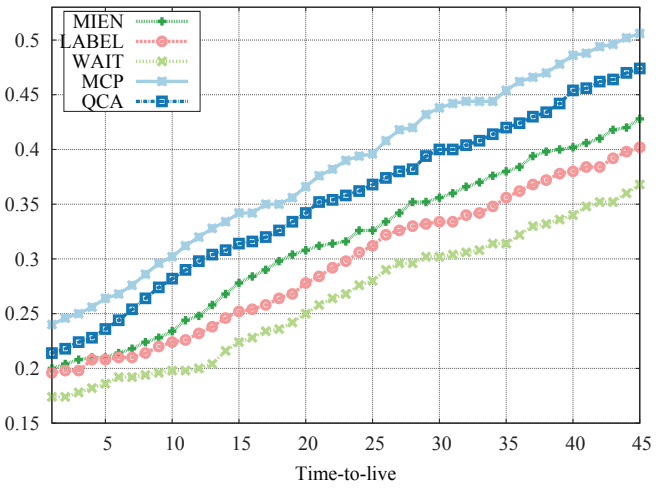
detection algorithm as a core will provide a speedup as well as robust to routing strategies in MANETs.

We evaluate five routing strategies (1) WAIT: the source node waits until it meets the destination node (2) MCP: A node keeps forwarding the messages until they reach the maximum number of hops (3) LABEL: A node forwards or sends the messages to all members in the destination community [78] (4) QCA: A Label version utilizing QCA as the dynamic community detection method and lastly, (5) MIEN: A social-aware routing strategy on MANETs [35].

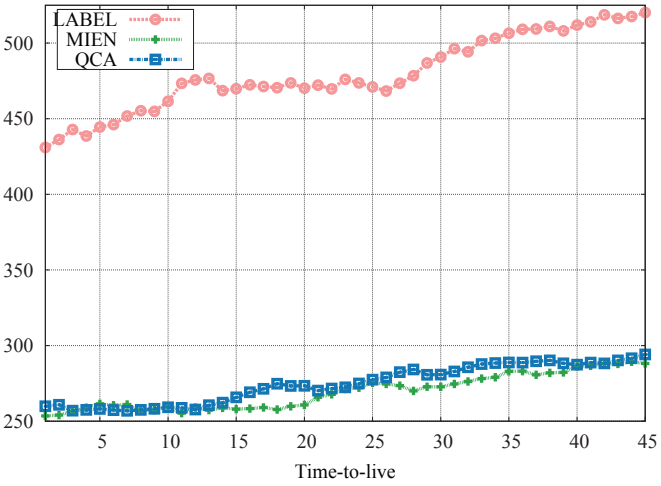
Even though WAIT and MCP algorithms are very simple and straightforward to understand, they provide us helpful information about the lower and upper bounds on the message delivery ratio, time redundancy as well as message redundancy. The LABEL forwarding strategy works as follow: it first finds the community structure of the underlying MANET, assigns each community with the same label and then exclusively forwards messages to destinations, or to next-hop nodes having the same labels as the destinations. MIEN forwarding method utilizes MIEN algorithm as a subroutine. QCA routing strategy, instead of using a static community detection method, employs QCA algorithm for adaptively updating the network community structure and then uses the newly updated structure to inform the routing strategy for forwarding messages.

5.1.1 Setup

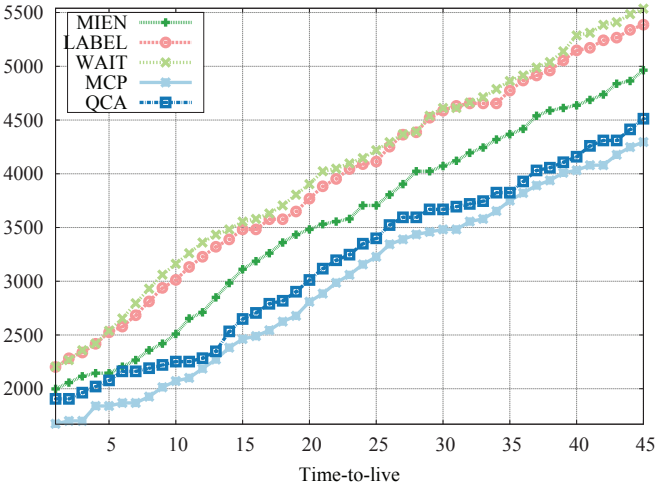
We choose Reality Mining data set [82] provided by the MIT Media Lab to test our proposed algorithm. The Reality Mining data set contains communication, proximity, location, and activity information from 100 students at MIT over the course of the 2004-2005 academic year. In particular, the data set includes call logs, Bluetooth devices in proximity, cell tower IDs, application usage, and phone status (such as charging and idle) of the participated students of over 350,000 hours ($\tilde{4}$0 years). In this paper, we take into account the Bluetooth information to form the underlying MANET and evaluate the performance of the above five routing strategies.



A Delivery Ratio



B Average Duplicate Message



C Average Delivery Time

Figure 5-1. Experimental results on the Reality Mining data set

5.1.2 Results

For each routing method, we evaluate the followings (1) Delivery ratio: The portion of successfully delivered over the total number of messages (2) Average delivery time: Average time for a message to be delivered. (3) Average number of duplicated messages for each sent message. In particular, a total of 1000 messages are created and uniformly distributed during the experiment duration and each message can not exist longer than a threshold time-to-live. The experimental results are shown in Figure $5-1A$, $5-1B$ and $5-1C$.

Figure $5-1A$ describes the delivery ratio as a function of time-to-live. As revealed by this figure, QCA achieves much better delivery ratio than MIEN as well as LABEL and far better than WAIT. This means that QCA routing strategy successfully delivers many more messages from the source nodes to the destinations than the others. Moreover, as time-to-live increases, the delivery ratio of QCA tends to approximate the ratio of MCP, the strategy with highest delivery ratio.

Comparison on delivery time shows that QCA requires less time and gets messages delivered successfully faster than LABEL, as depicted in Figure $5-1C$.
It even requires less delivery time in comparison with the social-aware method MIEN. This can be explained as the static community structures in LABEL can possibly get message forwarded to a wrong community when the destinations eventually change their communities during the experiment. Both QCA and MIEN, on the other hand, captures and updates the community structures on-the-fly as changes occur, thus achieves better results.

The numbers of duplicate messages presented in Figure $5-1B$ indicate that both QCA and MIEN achieves the best results. The numbers of duplicated messages of MCP method are substantially higher than those of the others and are not plotted. In fact, the results of QCA and MIEN are relatively close and tend to approximate each other as time-to-live increases.

In conclusion, QCA is the best social-aware routing algorithm among five routing strategies since its delivery ratio, delivery time, and redundancy outperform those of the other methods and are only below MCP while the number of duplicate messages is much lower. QCA also shows a significant improvement over the naive LABEL method which uses a static community detection method and thus, confirms the applicability of our adaptive algorithm to routing strategies in MANETs.

5.2 A Message Forwarding and Routing Strategy Employing AFOCS

We present a practical application where the detection of overlapping network communities plays a vital role in forwarding strategies in communication networks. With the helpful knowledge of the network community structure discovered by AFOCS, we propose a new community-based forwarding algorithm that significantly reduces the number of duplicate messages while maintaining competitive delivery times and ratios, which are essential factors of a forwarding strategy.

5.2.1 Message Forwarding Strategy

Let us first discuss how our new forwarding algorithm works in practice and then how AFOCS helps it to overcome the above limitations. We use AFOCS to detect overlapping communities and keep it up-to-date as the network changes. Each node in a community is assigned the same label and each overlapped node u has a set of corresponding labels $Com(u)$. During the network operation, if a devices u carrying the message meets another device v who indeed shares more common community labels with the destination than u, i.e., $|Com(v) \cap Com(dest)| > |Com(u) \cap Com(dest)|$, then u will forward the message to v. The same actions then apply to v as well as to devices that v meets.

The intuition behinds this strategy is that if v shares more communities with the destination nodes, it is likely that v will have more chances to deliver the message to the destination. By doing in this way, we not only have higher chances to correctly forward the messages but also generate much less duplicate messages. Due to its

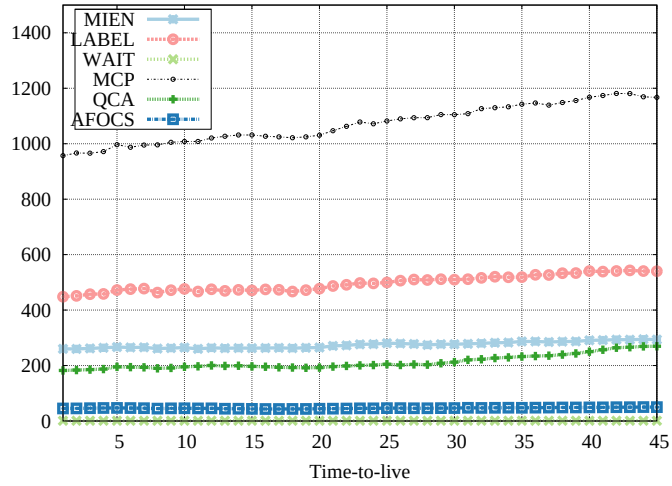
adaptive nature and the ability of identifying overlapping communities, AFOCS helps our algorithm to overcome the above shortcomings naturally. This explains why our forwarding algorithm can significantly reduce the number of duplicate messages while maintaining very competitive delivery times and ratios.

5.2.2 Setup

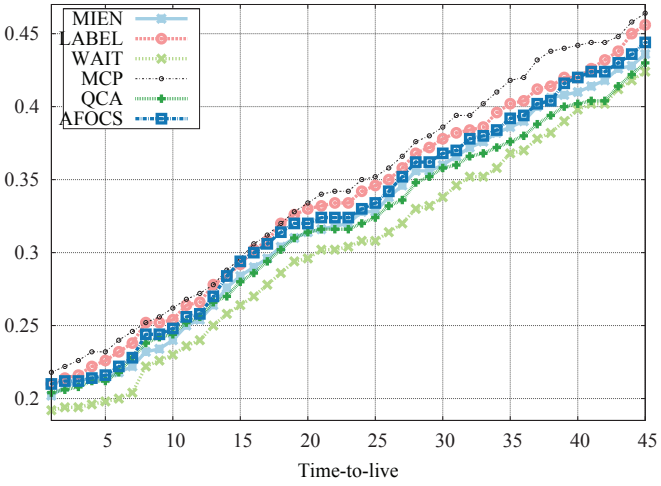
We compare six forwarding strategies (1) MIEN: A recently proposed social-aware routing strategy on MANETs [35] (2) LABEL: A node will forward the messages to another node if it is in the same community as the destination [78] (3) WAIT: The source node waits and keeps forwarding the message until it meets the destination (4) MCP: A node keeps forwarding the messages until they reach the maximum number of hops (5) QCA: A LABEL version utilizing QCA [17] as the adaptive disjoint community detection method and lastly (6) AFOCS: Our newly proposed forwarding algorithm equipped with AFOCS as an community detection and update core.

Results of WAIT and MCP algorithms provide us the lower and upper bounds of important factors: message delivery ratio, time redundancy and message redundancy. Our experiments are performed on the Reality Mining dataset provided by the MIT Media Lab [82]. This dataset contains communication, proximity, location, and activity information from 100 students at MIT over the course of the 2004-2005 academic year. In particular, we take into account the Bluetooth information to construct the underlying communication network and evaluate the performance of the above six routing strategies.

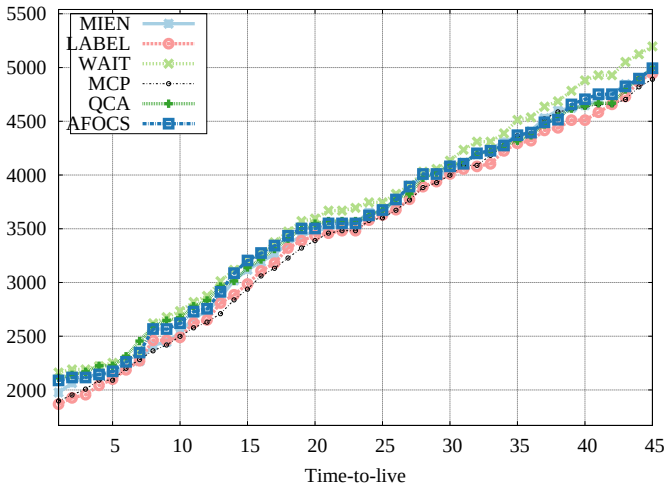
In each experiment, 500 message sending requests are randomly generated and distributed in different time points. To control the forwarding process, we use hop-limit, time-to-live, and max-copies parameters. A message cannot be forwarded more than hop-limit hops in the network or exist in the process longer than time-to-live, otherwise it will be automatically discarded. Moreover, the maximum number of same messages



A Average Duplicate Message



B Delivery Ratio



C Average Delivery Time

Figure 5-2. Experimental results on the Reality Mining data set

a device can forward to the others is restricted by max-copies. Experiments results are repeated and results are averaged for consistency.

5.2.3 Results

Our results are presented in Figures 5-2A, 5-2B, 5-2C. The first observation reveals that our proposed forwarding algorithm achieves the lowest number of duplicate messages as depicted in Figure 5-2A, and even far better than the second best method QCA. On average, only 46.5 duplicate messages are generated by AFOCS during evaluation process in contrast with 212.2 of QCA, 274.2 of MIEN, 496.4 of LABEL and the huge 1071.0 overhead messages of MCP. Thus, on the number of duplicate messages, AFOCS strikingly achieves improvement factors of 4.5x, 5x, 11x and 23x over these mentioned strategies, respectively. These extremely low overhead strongly imply the efficiency of AFOCS in communication networks.

Figures 5-2B and 5-2C present our results on the other two important factors, the message delivery ratios and delivery times. These figures supportively indicate that AFOCS achieves competitive results on both of these vital factors. In general, AFOCS is the second best strategy with almost no noticeable different between itself and the leader method LABEL. On average, AFOCS gets 33% of the total messages delivered in 3569.2s and only a little bit lags over MCP (34% in 3465.3s) and LABEL (slightly over 33% in 3462.7s), and is far better than MIEN (32% in 3537.6s) and QCA (32% in 3572.2s). This can be explained by the advantages of knowing the overlapping community structure: the disjoint network communities in QCA and MIEN can possibly have messages forwarded to the wrong communities when the destination changes its membership. With the ability of quickly updating the network structure, AFOCS can efficiently cope with this change and thus, can still provide the most updated forwarding information.

In summary, AFOCS helps our forwarding strategy to reduce up to 11x the number of duplicate messages while keeping good average delivery ratio and time. These

experimental results are highly competitive and supportively confirm the effectiveness of AFOCS and our new routing algorithm on communication networks.

CHAPTER 6
SOLUTIONS FOR WORM CONTAINMENT IN ONLINE SOCIAL NETWORKS

In this section, we present another practical application of our proposed algorithms in worm containment problem in OSNs. We first suggest a solution based on QCA, and then describe how AFOCS can help to improve the performance of this solution for this practical problem in complex networks. Since their introduction, popular social network sites such as Facebook, Twitter, Bebo, and MySpace have attracted millions of users worldwide, many of whom have integrated those sites into their everyday lives. On the bright side, OSNs are ideal places for people to keep in touch with friends and colleagues, to share their common interests, or just simply to socialize online. However, on the other side, social networks are also fertile grounds for the rapid propagation of malicious softwares (such as viruses or worms) and false information.

Facebook, one of the most famous social sites, experienced a wide propagation of a trojan worm named “Koobface” in late 2008. Koobface made its way not only through Facebook but also Bebo, MySpace and Friendster social networks [83][84]. Once a user’s machine is infected, this worm scans through the current user’s profile and sends out fake messages or wall posts to everyone in the user’s friend list with titles or comments to appeal to people’s curiosity. If one of the user’s friends, attracted by the comments without a shadow of doubt, clicks on the link and installs the fake “flash player”, his computer will be infected and Koobface’s life will then cycle on this newly infected machine.

Worm containment problem becomes more and more pressing in OSNs as this kind of networks evolves and changes rapidly over time. The dynamics of social networks thus gives worms more chances to spread out faster and wider as they can flexibly switch between existing and new users in order to propagate. Therefore, containing worm propagation on social networks is extremely challenging in the sense that a good solution at the previous time step might not be sufficient or effective at the next time

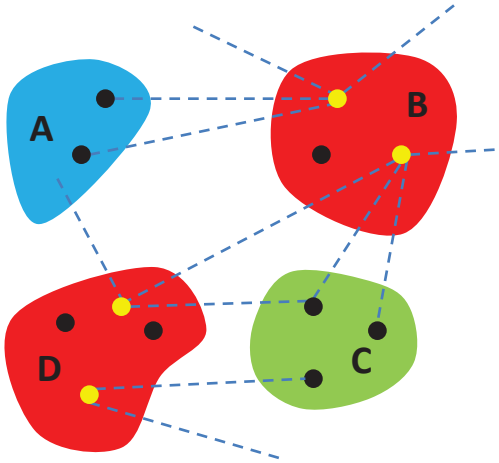


Figure 6-1. A general worm containment strategy.

step. Although one can recompute a new solution at each time the network changes, doing so would result in heavy computational costs and be time consuming as well as allowing worms spreading out wider during the recomputing process. A better solution should quickly and adaptively update the current containing strategy based on changes in network topology, and thus can avoid the hassle of recomputation.

There are many proposed methods for worm containment on computer networks by either using a multi-resolution approach [85], or using a simplification of the Threshold Random Walk scan detector [86], or using fast and efficient worm signature generation [87]. There are also several methods proposed for cellular and mobile networks [88][89]. However, these approaches fail to take into account the community structure as well as the dynamics of social networks, and thus might not be appropriate for our problem. A recent work [16] proposed a social-based patching scheme for worm containment on cellular networks. However, this method encounters the following limitations on a real social network (1) its clustered partitions do not necessarily reflect the natural network communities, (2) it requires the number of clusters k (which is generally unknown for social networks) must be specified beforehand, and (3) it exposes weaknesses when dealing with the network's dynamics.

6.1 An Application of QCA in Containing Worms in OSNs

6.1.1 Setup

To overcome these limitations, our approach first utilizes QCA to identify the network community structure, and adaptively keeps this structure updated as the network evolves. Once network communities are detected, our patch distribution procedure will select the most influential users from different communities in order to send patches. These users, as soon as they receive patches, will apply them to first disinfect the worm and then redistribute them to all friends in their communities. These actions will contain worm propagation to only some communities and prevent it from spreading out to a larger population. To this end, a quick and precise community detection method will definitely help the network administrator to select a more sufficient set of critical users to send patches, thus lowering down the number of sent patches as well as overhead information over the social network.

Algorithm 15 Patch Distribution

Input: $G = (V, E)$ and its community structure $\mathcal{C} = \{C_1, C_2, .., C_p\}$
Output: The set of influential users $\mathcal{P}$.

1: $\mathcal{P} = \emptyset$;
2: **for** $C_i \in \mathcal{C}$ **do**
3: **while** $\exists u$ unvisited in C_i satisfying $\max_{u \in C_i}\{e_{out}^{C_i}(u)\} > 0$ **do**
4: Let $v \leftarrow \arg\max_{u \in C_i}\{e_{out}^{C_i}(u)\}$;
5: $\mathcal{P} = \mathcal{P} \cup v$;
6: Mark v as visited in C_i;
7: **end while**
8: **end for**
9: Send patches to users in $\mathcal{P}$;

We next describe our patch distribution. This procedure takes into account the identified network communities and selects a set of influential users from each community in order to distribute patches. Influential users of a community are ones having the most relationships or connections to other communities. In an adversary point of view, these influential users are potentially vulnerable since they not only interact actively within their communities but also with people outside, and thus, they

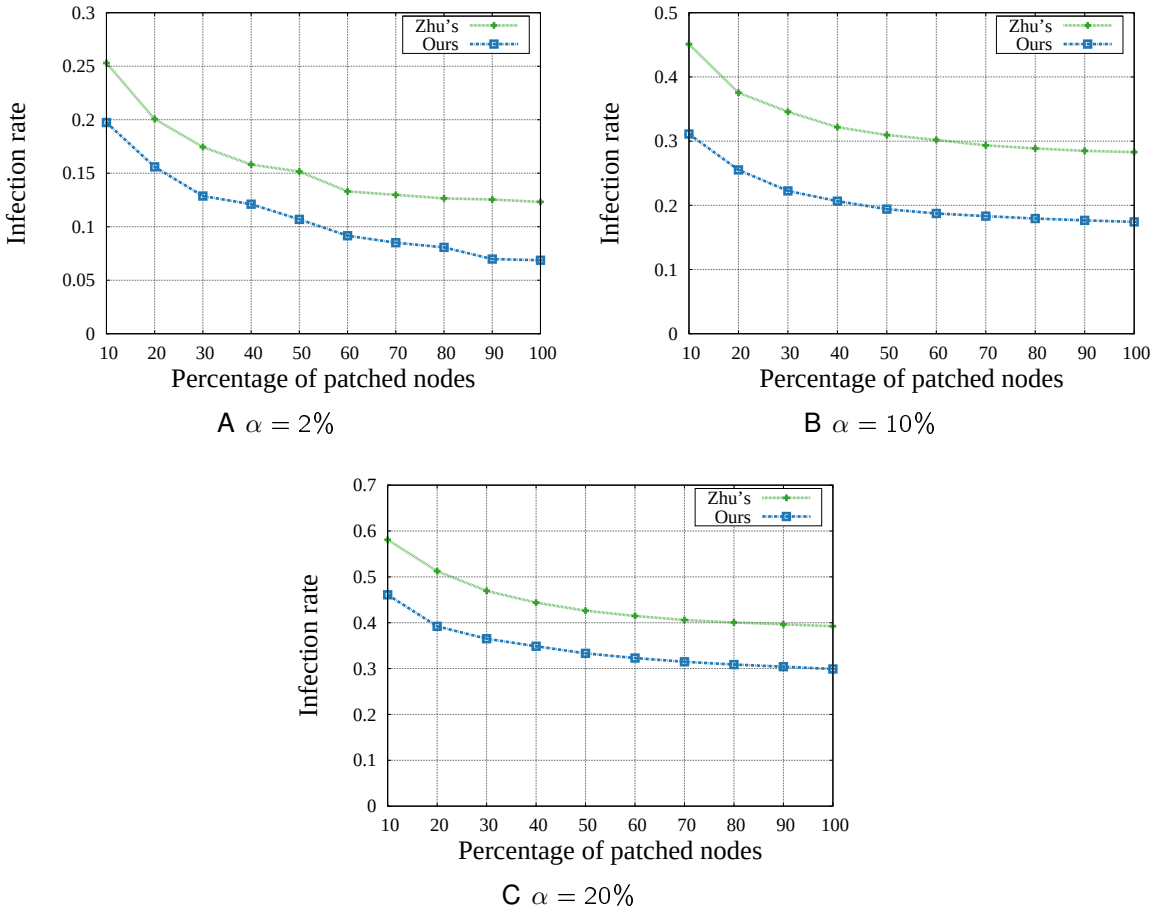


Figure 6-2. Infection rates on static network with $k = 150$ clusters

can easily fool (or be fooled by) people both inside and outside of their communities. On the other point of view, these users are also the best candidates for the network defender to distribute patches since they can easily announce and forward patches to other members and non-members.

In Alg. 15, we present a quick algorithm for selecting the set of most influential users in each community. This algorithm starts by picking the user whose number of social connections to outside communities is the highest, and temporarily disregards this user from the considering community. This process repeats until no connections crossing among communities exists. This set of influential users is the candidate for the network defender for distributing patches.

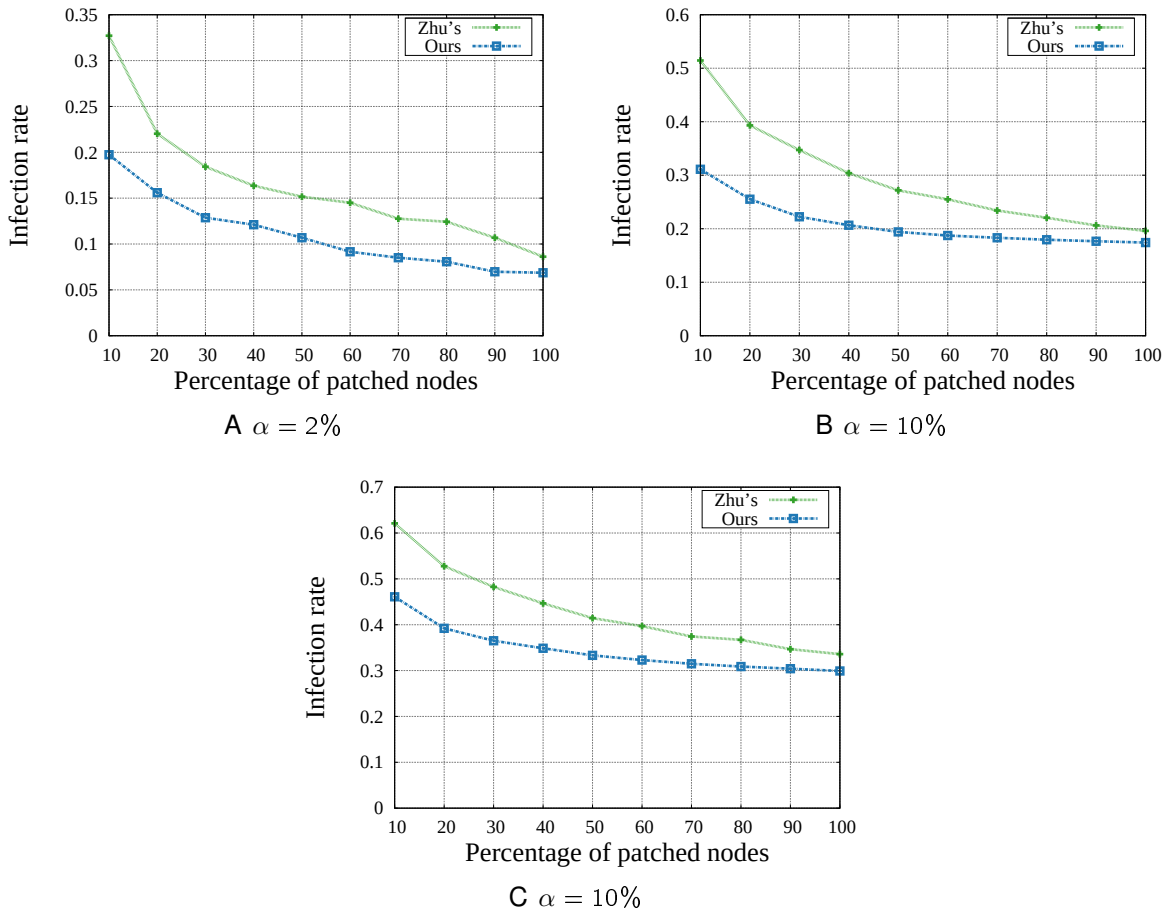


Figure 6-3. Infection rates on dynamic network with $k = 200$ clusters

6.1.2 Results

We present the results of our QCA method on the Facebook network dataset [61] and compare the results with the social based method (Zhu's method [16]) via a weighted version of our algorithms.

The worm propagation model in our experiments mimics the behavior of the famous "Koobface" worm. The probabilities of activating the worm is proportional to communication frequency between the victim and his friends. The time taken for worms to spread out from one user to another is inversely proportional to the communication frequency between this user and his particular friend. Finally, when a worm has successfully infected a user's computer, it will start propagating as soon as

this computer connects to a specific social network (Facebook in this case). When the fraction of infected users reaches a threshold α, the detection system raises an alarm and patches will automatically be sent to most influential users selected by Alg. 15. Once a user receives the patch, he will first apply it to disinfect the worm and then will have an option to forward it to all friends in his community. Each experiment is seeded with 0.02% of users to be initially infected by worms.

We compare infection rates of the social-based method of Zhu's and ours. The infection rate is computed as the fraction of the remaining infected users over all infected ones. The number of clusters k in Zhu's method is set to be 150 in static and 200 in dynamic networks, and for each value of k, the alarming threshold α is set to be 2%, 10%, and 20%, respectively. Each experiment is repeated 1000 times for consistency.

Figure 6-2, 6-3 show the results of our experiments for three different values of k and α. We first observe that the longer we wait (the higher the alarm threshold is), the higher number of users we need to send patches to in order to achieve the desired infection rate. For example, with $k = 150$ clusters and an expected infection rate of 0.3, we need to send patches to less than 10% number of users when $\alpha = 2\%$, to more than 15% number of users when $\alpha = 10\%$ and to nearly 90% of total influential users when $\alpha = 20\%$.

A second observation reveals that our approach achieves better infection rates than the social-based method of Zhu's in a static version of the social network as depicted in Figure 6-2. In particular, the infection rates obtained in our method are from 5% to 10% better than those of Zhu's. When the network evolves as new users join in and new social relationships are introduced, we resize the number of cluster k and recompute the infection rates of the social based method with the number of cluster $k = 200$, and the alarm threshold $\alpha = 2\%$ and 10% respectively. As depicted in Figures 6-3, our method, with the power of quickly and adaptively updating the network community structure, achieves better infection rates than Zhu's method while the computational costs and

running time is significantly reduced. As discussed, detecting and updating the network community is the crucial part of a social based patching scheme: a good and up-to-date network community structure will provide the network defender a tighter set of vulnerable users, and thus, will help to achieve lower infection rates. Our adaptive algorithm, instead of recomputing the network structure every time changes are introduced, quickly and adaptively updates the network communities on-the-fly. Thanks to this frequently updated community structure, our patch distribution procedure is able to select a better set of influential users, and thus, helps in reducing the number of infected users.

We further look more into the behavior of Zhu's method when the number of clusters k varies. We compute and compare the infection rates on Facebook dataset for various k ranging from 1K to 2.5K with our approach. We first hope that the more predefined clusters, the better infection rates clustered partitioning method will achieve. However, the experimental results reveal the opposite. In particular, with a fixed alarming threshold $\alpha = 10\%$ and 60% patched nodes, the infection rates achived by Zhu's method do not decrease but ranging near 28% while ours are far better (20%) with much less computational time.

Finally, a comparison on running time on the two approaches shows that time taken for Zhu's method is much more than our community updating procedure, and hence, may prevent this method to complete in a timely manner. In particular, our approach takes only 3 seconds for obtaining the basic community structure and at most 30 seconds to complete all the tasks whereas [16] requires more than 5 minutes to divide the communication network into modules and selecting the vertex separators. In that delay, worm propagation may spread out to a larger population, and thus, the solution may not be effective. These experimental results confirm the robustness and efficiency of our approach on social networks.

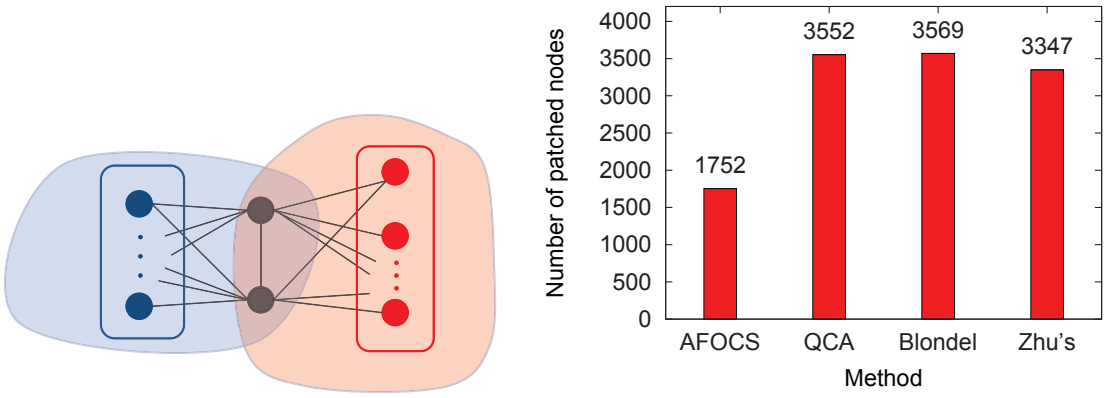
6.2 Containing Worms with Overlapping Communities Detected by AFOCS

We show another application of AFOCS in worm containment problem on OSNs. OSNs are good places for people to socialize online or to stay in touch with friends and colleagues. However, when some of the users are infected with malicious software, such as viruses or worms, OSNs are also fertile grounds for their rapid propagations. Since mobile devices are able to access online social applications nowadays, worms and viruses now can target computers [17] and mobile devices [16].

Recently, community structure-based methods have been proven to be effective solutions to prevent worms from spreading out wider on not only social networks [17][18] but also cellular networks [16]. Due to the high and low frequencies of interactions inside and between communities, worms spread out quicker within a community than between communities. Therefore, an appropriate reaction should first contain worms into only infected communities, and then prevent them from getting outside. This strategy can be accomplished by patching the most influential members who are well-connected not only to members of their community but also to people in other communities.

6.2.1 Setup

In our experiments, we use Facebook network dataset collected in [61]. This data set contains friendship information and wall posts among New Orleans regional network, spanning from Sep 2006 to Jan 2009. The data set contains more than 63.7K nodes (users) connected by more than 1.5 million friendship links. We keep other parameters as well as the "Koobface" worm propagation model the same as [18] for comparison convenience. With the advantages of knowledge overlapping communities, we are able to develop a better and more efficient patching scheme. In particular, we enhance the patching scheme presented in in [18] to take the advantage of the overlap regions: nodes in the boundary of overlapped regions are selected for patching (Figure 6-4A). Alg 16 details the adjusted scheme.



A Influential users selection

B Number of patched nodes

Figure 6-4. *OverCom* patching scheme.

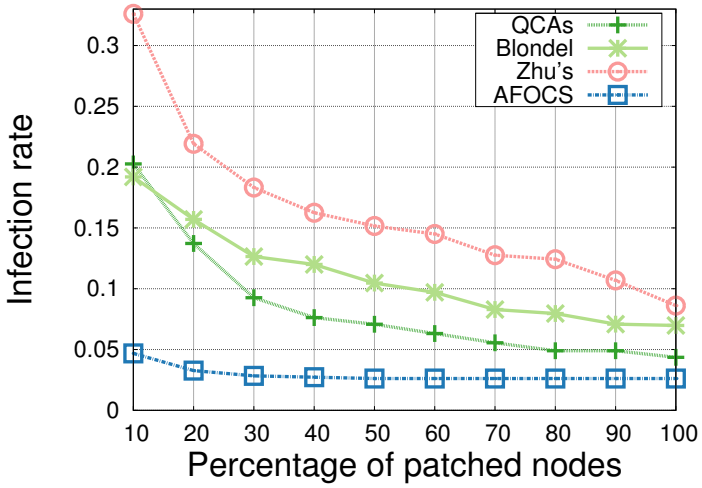
Algorithm 16 *OverCom* Patching Scheme

Input: $G = (V, E)$ and $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ detected by AFOCS
Output: A set of patched nodes IS.

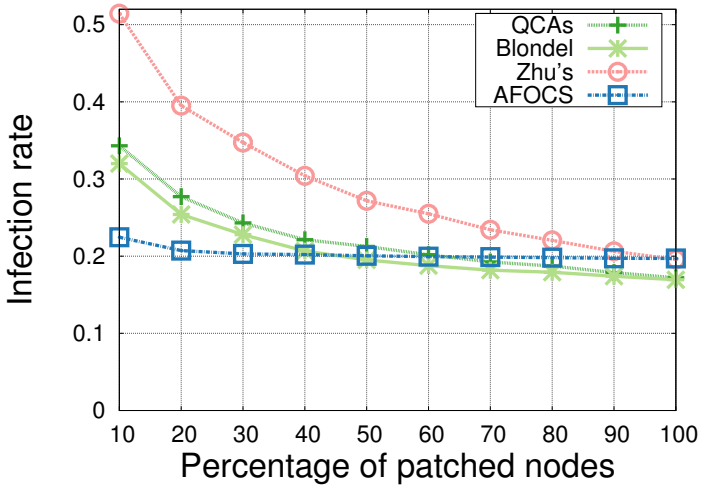
$IS \leftarrow \emptyset$;
for $(C_i, C_j \in \mathcal{C})$ **do**
    if $(C_i \cap C_j \neq \emptyset)$ **then**
        %Choose the neighbors of overlapped nodes as influential ones%
        $IS \leftarrow IS \cup N(u) \quad \forall u \in C_i \cap C_j$;
    end if
end for
%Patch distribution procedure%
for $(u \in IS)$ **do**
    Send patches to u;
    Let u redistribute patches to $w \in IS \backslash N(u)$;
end for

6.2.2 Results

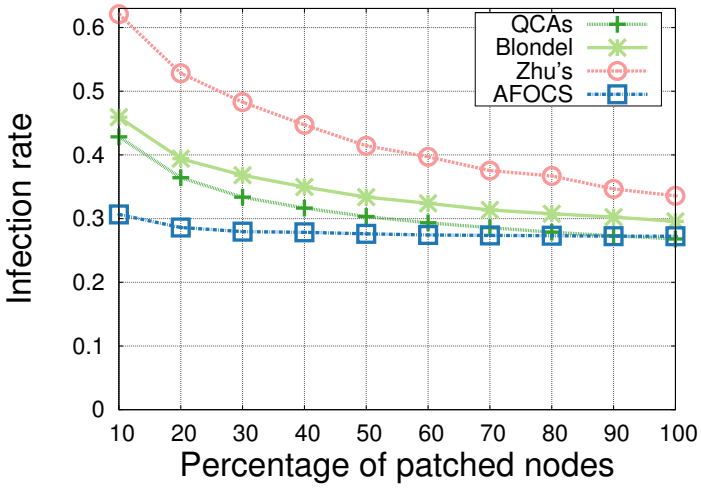
We compare the *OverCom* patching scheme and overlapping communities found by AFOCS to those using disjoint communities proposed by Blondel et al. [5], QCA by Nguyen et al. [17] and Clustering based method suggested by Zhu et al. [16]. The number of patched nodes is shown in Figure 6-4B. Both the number of patched nodes and the infection rates decline remarkably. In particular, the number of nodes to send patch in AFOCS is substantially smaller by half of those required by Blondel, QCA as well as Zhu's methods: only 1725 nodes over 63K nodes in the networks are needed



A $\alpha = 2\%$



B $\alpha = 10\%$



C $\alpha = 20\%$

Figure 6-5. Infection rates between four methods.

to be patched by $OverCom$ patching scheme, while the other schemes require nearly twice ($\geq$3,300 nodes). The reason behind this improvement is due to the nature of our AFOCS framework, the neighbors of the overlapped nodes should not be to far away from the center of each community, thus they can easily redistribute the patches once received.

We next present the achieved infection rates with alarming thresholds (the fraction of infected nodes over all nodes) α = 2%, 10% and 20%, respectively. This threshold alarms the distribution process as soon as the infected rate goes beyond α. The results are reported in Figures 6-5A, 6-5B, 6-5C, respectively. In general, the higher α (i.e., the longer we wait), the more nodes we have to send patches and the higher infection rate. OverCom with AFOCS achieves the lowest infection rates in almost all the experiments and just a little bit lag behind when α = 10%. In particular, when $\alpha = 2\%$, AFOCS helps $OverCom$ to remarkably reduce from 1.6x up to 4.3x the infection rates of QCA, from 2.6x up to 4x the infection rates of Blondel and 3.2x to 7x those of Zhu's method. When α = 10%, AFOCS $+OverCom$ achieves average improved rates of 9% over QCA, 5% over $Blondel$ and 43% over Zhu's methods. As α = 20%, the average improvements are 12%, 23% and 53%, respectively. Due to the nature of the event handling processes, the neighbors of overlapped nodes are not located far away from the rest of their communities. As a result, they can help to distribute patches to more users in the communities, hence help to lower the infection rates of AFOCS. These improvement factors, again, confirm the effectiveness of our proposed method.

CHAPTER 7
STABLE COMMUNITY DETECTION IN ONLINE SOCIAL NETWORKS

A large body of work has been devoted to find general communities (i.e., without the concept of stability) on both directed and undirected networks in the literature [9]. On the contrary, only a very few approaches are suggested to identify stable communities [50][51], especially on directed and weighted networks. The main source of difficulty is due to the inconsistency of community members in a general structure: while they might appear to be in a community at one time, they may not commit to that particular community in a long run. One possible approach, therefore, is to find a consensus of a specific algorithm after multiple runs and use this core as stable communities [50]. However, doing in this way would result in expensive computational cost and time consuming as well as lack of convergence guarantees. In [51], the authors estimate the mutual links between pairs of users and suggest a detection method that optimizes the total mutual connection on the whole network. While the idea of mutual connection is quite interesting, we find that it might not be sufficient because some estimated mutual links are of low magnitudes, and thus, may not reflect the correct concept of stability at the community level.

In general, a stable community is often characterized either by its tight and strong internal relationships represented by the mutual connections among its users [51], or by its internal links who possess a high tendency to remain within the community over a long period of time [49]. In other words, stable communities in the network are commonly characterized by stable connections among their members. Motivated by these observations, we suggest SCD (short for Stable Community Detection), a framework to effectively identify stable communities in directed OSNs that facilitates both of the above intuitions. In a big picture, SCD works by first enriching the input network with the stability estimation of all links in the network, and then discovering communities via stable connections using the lumped Markov chain model. Our approach is

mathematically supported by a key connection between the persistence probability of a community at the stationary distribution and its local topology. One notable advantage of SCD is that it requires only a single iteration, which shall significantly reduce the running time. Furthermore, since our method intrinsically accounts for stability, the discovered communities should be stable as opposed to doing a statistical analysis.

In summary, we suggest an estimation which provides helpful insights into the stability of links in the input network. Based on that, we propose SCD - a framework to identify community structure in directional OSNs with the advantage of community stability. We next explore an essential connection between the persistence probability of a community at the stationary distribution and its local topology, which is the fundamental mathematical theory to support the SCD framework. To certify the efficiency of our approach, we extensively test SCD on both synthesized datasets with embedded communities and real-world social traces, including NetHEPT and NetHEPT_WC collaboration networks as well as Facebook social networks, in reference to the consensus of other state-of-the-art detection methods. Highly competitive empirical results confirm the quality and efficiency of SCD on identifying stable communities in OSNs.

7.1 Basic Notations

We introduce the basic notations representing the underlying social network that we will use throughout this paper.

(Graph notation) Let $G = (V, E, w)$ be a directed and weighted graph representing a social network with V is the set of n network users (or nodes), E is the set of m directed relationships (or edges), and w (or precisely w_{uv}) is the weight function on each edge $(u, v) \in E$ representing the communication frequency between user u and v in the social network. Without loss of generality, we assume that all edge weights are normalized, i.e., $\sum_{(u,v)\in E} w_{uv} = 1$ and $w_{uv} \geq 0$. For each edge $(u, v) \in E$ which $(v, u) \notin E$, we say that the backwards edge (v, u) is missing, we will use the notation

$(v; u)$ and $w_{v;u}$ to denote the mutual link of edge (u, v) if (v, u) should indeed exist in E and its weight, respectively. Furthermore, we will use the notation $st(u, v, t)$ to denote the stability estimate of edge (u, v) at time step (or hop) t. These notations will be described in detail in next section.

(Community notation) Denote by $\mathcal{C} = \{C_1, C_2, ..., C_q\}$ the network community structure, i.e., a collection of q subsets of V satisfying $\cup_{i=1}^{q} C_i = V$ and $C_i \cap C_j = \emptyset \; \forall i, j$. We say that each $C_i \in \mathcal{C}$ and its induced subgraph form a community of G. For a node $u \in V$, let N_u^+, N_u^- and N_u denote the set of outgoing, the set of incoming, and the set of all neighbor nodes adjacent to u, respectively. Furthermore, let k_u^+(or w_u^+), k_u^-(or w_u^-) and k_u(or w_u) be the corresponding cardinalities (or total weights) of these sets. For any $C \subseteq V$, let C^{in} and C^{out} denote the set of links having both endpoints in C and the set of links heading out from C, respectively. In addition, let $m_C = |C^{in}|$ (rsp. $w_C = w(C^{in})$) and $k_C^+ = \sum_{u \in C} k_u^+$ (rsp. $w_C^+ = \sum_{u \in C} w_u^+$). Finally, the terms node-vertex as well as edge-link-connection are used interchangeably.

7.2 Link Stability Estimation

We describe our first step towards the identification of stable communities in the network: the link stability estimation process. Intuitively, a stable community is often characterized either by its tight and strong internal relationships represented by the mutual connections among its users [51], or by its internal links who possess a high tendency to remain within the community over a long period of time [49]. In other words, stable communities in the network are commonly characterized by stable connections among their members. Motivated by these observations, in this section, we suggest a procedure for estimating the stability of each link in the network that facilitates both of the above intuitions. Our estimation procedure consists of two stages: In the early stage, the reciprocity of each link in the network is first predicted, and based on that, its stability is consequently evaluated in the later stage.

7.2.1 Link Reciprocity Prediction

When dealing with large scale OSNs, it is possible that some backwards edges between individuals are missing. This lack of information may due to the imperfect data collection process, or because these backwards edges are not yet reflected in the underlying network but should due to the strong relationships between local network users. For instance, Leskovec et al. [90] observe that friends of friends in social networks tend to be friend of each other in the near future, i.e., there should be dual connections between friends of friends with high chance even if they are not yet friend of each other. Therefore, predicting the existence of these backwards edges will allow a more complete and comprehensive detection of stable communities by increasing the internal density of strongly connected components, which are potential candidates for network communities.

Link reciprocity prediction problem is a well-studied field and many methods are proposed in the literature [91][92][93]. In this paper, we utilize a method called "friends-measure" suggested in [93]. The intuition behind this measure is that when looking at two users in the social network, one can assume that the more connections their neighbors have with each other, the higher the chance the two users are actually connected. Originally, this friends-measure between two users u and v is formulated as:

$$\text{friends-measure}(u, v) = \sum_{x \in N_u} \sum_{y \in N_v} \delta(x, y)$$

where $\delta(x, y) = 1$ if either $x = y$ or $(x, y) \in E$ or $(y, x) \in E$. This measure has been extensively verified among other topological features and has been shown to be a promising one in comparison with other metrics [93]. However, in the case of directed networks, there are possibilities that different link topologies can share a common friends-measure value. Therefore, we need to modify the above formula so that it reflects the true relationship between the network users, and furthermore copes with edge weights in the network.

In order to better handle directed and weighted graphs, we will attempt to predict the existence of backwards edges of unidirectional links. For example, if $(u, v) \in E$ and $(v, u) \notin E$, we will try to find the possibility whether we should enrich the network by inserting (v, u) into E. To this end, we first relax the direction of the edge between u and v, and next compute the likelihood that a backwards edge should exist between u and v by using the modified formula

$$\Delta(u, v) = \frac{\sum_{x \in N_v} \sum_{y \in N_u} \tau(x, y)}{W_v W_u} \tag{7–1}$$

Where $\tau(x, y) = w_{vx} w_{xy} w_{yu}$ is the total possibility of the backwards path starting from v, passing through neighbor nodes x and y, and ending at u. When the network is unweighted $W_u = d_u$, $W_v = d_v$ and thus, $\Delta(u, v)$ counts the (normalized) number of paths of lengths two and three joining two users u and v, which intuitively agrees with the aforementioned friends-measure formula. By Proposition 7.1, we show that $\Delta(u, v)$ is indeed the generalization of weighted friend-measure(u, v) and depends only on the nodes' topology. Hence, $\Delta(u, v)$ can be regarded as the estimated probability that the backwards connection $< v, u >$ indeed exists, i.e., we set $w_{<vu>} = \Delta(u, v)$.

Proposition 7.1. *For any* $(u, v) \in E$ *which* $(v, u) \notin E$*,* $0 \leq \Delta(u, v) \leq 1$.

Proof. We first prove this for unweighted graphs. The proof for weighted graphs can be extended straightforwardly. It is obvious that $0 \leq \Delta(u, v)$. Now we show $\Delta(u, v) \leq 1$. For any x, y such that $\delta(x, y) = 1$, if $x = y$, they can make just one connection counted towards the summation. Otherwise, they can make at most d_u (or d_v) dual-connections at each vertex. Taking these facts into account, we have $\Delta(u, v) = \sum_{x \in N_v} \sum_{y \in N_u} \delta(x, y) \leq d_v d_u$. Thus, the inequalities follows. The left equality holds when there are no connections from u to v and vice versa. The right equality holds when every path of length 2 from u to v (or from v to u) are contained in the corresponding path of length 3. □

Proposition 7.2. *Let n_0 be the number of unidirectional links in the input network. The time complexity for estimating the mutual connections for these links is $O(n_0 M)$.*

Proof. The total time required for estimating the possibility for a backward connection at an edge (u, v) is $d_u + d_v + \min \sum_{x \in N_v^+} \sum_{y \in N_y^-} \{d_x^+, d_y^-\}$. Thus, for all n_0 links, the total time complexity is upper bounded by $n_0(2M) + n_0 M = O(n_0 M)$. □

7.2.2 Link Stability Estimation

After the reciprocity of each link in the network has been estimated, the input network is now enriched with more information of the backwards edges. While the presence of these dual edges is helpful in characterizing the mutual relationships between pairs of network users, it might not be sufficient to evaluate the stability of all network connections as some of the backwards edges may be of low magnitudes, and thus, may not be able to hint the stability of the connection. Therefore, we need to further estimate the stability of a network link given its predicted reciprocity. In order to do so, we define the stability of an edge $(u, v) \in E$ at t time steps (or t hops) as follow

$$st(u, v, t) = \sum_{|P|=t} w(P)$$

where P is a path going from v to u (v and u are excluded) of length $|P| = t$, and $w(P) = \prod_{(a,b)\in P} w_{ab}$ is the total weight of path P. Finally, we define the stability $st(u, v)$ of a link $(u, v) \in E$ as the total stability of up to T_0 time steps, where T_0 is a predefined parameter (or the upper bound on the number of hops)

$$st(u, v) = \sum_{t=1}^{T_0} st(u, v, t). \tag{7–2}$$

The intuition behind our stability function $st(u, v)$ is as follow: since stable communities are commonly recognized by a high density of stable edges, it is reasonable to expect that such edges form a cycles. In the senses of directed and weighted networks, the stronger the strength of cycles an edge (u, v) is on, the more stable it is believed to be.

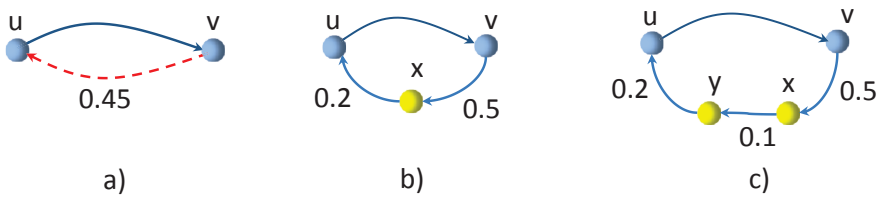


Figure 7-1. Illustrations of stability function.

On the contrary, edges that connecting or joining between communities shall hardly be part of many cycles, and eventually result in low stability. Figure 7-1 illustrates the stability estimates for link (u, v) at 0, 1 and 2 hops: a) $st(u, v, 0) = 0.45 = w_{<vu>}$, b) $st(u, v, 1) = 0.5 \times 0.2 = 0.1$, c) $st(u, v, 2) = 0.5 \times 0.1 \times 0.2 = 0.05$.

As a local measure, our suggested stability function has the following advantages (1) it puts more focus on the existence of the mutual link of any link (u, v) by reserving the original strength of the backwards edge $< v, u >$. This intuitively agrees with the findings that stable clusters are usually made of bidirectional links in [51]. Moreover, our formula further takes into account the strength of cycles containing the current link; (2) the more time (or, number of hops) we allow, the more stability a link would be. Nevertheless, links that really belong to a stable community are more likely to have strong stability whereas those connecting communities are of low stability. These advantages support the intuitions of stable communities that we discussed above. The performance of our stability estimation is evaluated in more detail in section 7.4.

In summary, our link stability estimation first predicts the potential of the dual link of any link $(u, v) \in E$ such that $(v, u) \notin E$ by using the modified measure in equation (7–1). Next, it evaluates the stability of the every link in the given network enriched from the first stage by using equation (7–2), and utilizes these stability values as new weights for links in the network. This resulting network will be consequently passed as the input network to our main process: the identification of stable communities.

7.3 Stable Community Detection

In this section, we present our main contribution: the stable community identification process. Given the input network enriched with link stability information, we discover the stable communities by exploring an important connection between the persistence probability of each community and its local network topology. In the following paragraphs, we first review the concept of Lumped Markov chain [94][95], and then establish our key connection between this Markov chain and the local network topology. Finally, we describe in detail our last but most important process: stable community detection.

7.3.1 Lumped Markov Chain

A Markov chain [96] is a mathematical system representing transitions from one system's state to another, between a finite number of predefined states. In terms of social networks, a state can be either a user (a node in the graph) or a group of tightly connected users (a community) in the networks, whereas transitions can be regarded as the user-to-user or group-to-group communication tendencies. An n-state Markov chain corresponding to an n-node network is commonly represented by the transition $\pi_{t+1} = \pi_t P$, where $\pi_t = (\pi_{1,t}, \pi_{2,t}, \ldots, \pi_{n,t})$ with $\pi_{u,t}$ is the probability of being at node u at time t, and $P = (p_{uv})$ is the transition matrix. In particular, this n-state Markov chain can be associated to input network by letting the probability of transiting from a node u to a neighbor node v as

$$p_{uv} = \frac{w_{uv}}{\sum_j w_{uj}} = \frac{w_{uv}}{w_u^+}.$$

Basically, p_{uv} is the probability of a random walker jumps from node u to node v given the network topology. A Markov chain is said to be at its stationary state distribution π if π satisfies the equation $\pi = \pi P$. As shown in [97], when the network is originally connected P would be irreducible, and thus, the equation $\pi = \pi P$ has a unique solution which is strictly positive ($\pi_u > 0 \; \forall u \in V$) which corresponds to the stationary Markov chain state distribution. When the network is undirected, π can be exactly computed as $\pi = \frac{1}{2W_0}(w_1, w_2, \ldots, w_n)$ with W_0 is the total edge weights. However, we do not have an

exact form for the stationary distribution π in general for directed network, and thus, π has to be computed numerically.

As our ultimate goal is to detect the stable network community structure, we sought to find a good partitioning of V where each partition will remain wealthy over time. In the light of Markovian chain method, this corresponds to finding a collection of communities $\mathcal{C} = \{C_1, C_2, \ldots, C_q\}$ where a random walker would spend most of the time walking inside a community and less time wandering among communities. By defining this partition $\mathcal{C}$ of q communities, we introduce a so called q-state meta-network where each community in the network becomes a meta-state. However, at this aggregate level, a in general dynamics Markovian description of a random walker walking among communities is not possible because the Markovian property may not be well-preserved [94]. Nevertheless, this q-state community-to-community transition can still be defined using the lumped Markov chain, which correctly describes the random walker at this scale given the stochastic process is started at the stationary distribution π [97]. This lumped Markov chain is defined via the $q \times q$ matrix as U in [95]

$$U = [diag(\pi H)]^{-1} H^{\mathsf{T}} diag(\pi) P H$$

where H is a $n \times q$ binary matrix representing the partitioning $\mathcal{C}$.

One of the notable advantages of the lumped Markov chain $\Pi_{t+1} = \Pi_t U$ defined on U is that it shares the same stationary distribution with the original Markov chain, i.e., the new stationary distribution defined by $\Pi = \pi H$ satisfies the equation $\Pi = \Pi U$. Moreover, the difference between $\Pi_{t+1} = \Pi_t U$, starting at $\Pi_0 = \pi U$, and the original $\pi_t H$ tends exponentially to zero if the two chains are regular. These advantages make the community-based lumped Markov chain defined by $\Pi_t U$ a very good approximation of the original n-node network. We stress that the ability of the lumped Markov chain to describe the random walk dynamics only at stationary is not a limitation for the detection of stable communities. Indeed, this stationary requirement evaluates the random walk

dynamics of all nodes at their stable states, and hence perfectly supports the concept of stable communities.

In terms of interpretation, each entry u_{cd} of U denotes the chance that a random walker, at time t, wanders from community c to another community d in time $t+1$. As a result, the diagonal elements u_{CC}'s (or u_C's in short) of U indicate the persistence probabilities that a random walker just walking within a particular community C. Of course, large values of u_C's are expected for meaningful communities. It is also shown in [95] that in directed and weighted graphs, u_C can be computed as

$$u_C = \frac{\sum_{i,j\in C} \pi_i p_{ij}}{\sum_{i\in C} \pi_i} \tag{7–3}$$

Note that $\sum_{i,j\in C} \pi_i p_{ij}$ is the fraction of time a random walker spends on the links inside a community C. Hence, u_C is indeed the ratio between the amount of time a random walker spends on links and that it spends on nodes in C. In undirected networks, one can verify that

$$u_C = \frac{\sum_{i,j\in C} \pi_i w_{ij}}{\sum_{i\in C} w_i} = \frac{2w_C}{2w_C + w(C^{out})}.$$

7.3.2 The Connection to Network Topology

At this stage, one might try to optimize u_C for all communities $C \in \mathcal{C}$ in order to maximize theie persistence probabilities. However, doing in this way requires solving for the stationary distribution π_i's (as in equation (7–3)) which may be extremely costly, especially in large scale directed networks. So, how can we effectively optimize the persistence probability u_C for each community without solving for that costly exact stationary distribution? As an answer for this challenging question, we present in Proposition 7.3 a connection between the persistence probability of a community C and its local topology. In particular, we show that the minimum value of u_C can be represented by quantities that only involve C's local topology. Therefore, optimizing u_C

can be shifted as the optimization of these local components, which are inexpensive and easy to derive.

Proposition 7.3. *For any community $C \in \mathcal{C}$, at the stationary distribution π, we have the following inequality*

$$u_C = \frac{\sum_{i,j\in C} \pi_i p_{ij}}{\sum_{i\in C} \pi_i} \geq \frac{w_C}{w_C^+}.$$

Proof. It is easy to see that

$$u_C = \frac{\sum_{i,j\in C} \pi_i p_{ij}}{\sum_{i\in C} \pi_i} = \frac{\sum_{i\in C} \pi_i \frac{w_{i,C}}{w_i^+}}{\sum_{i\in C} \pi_i}.$$

where $w_{i,C} = \sum_{j\in C} w_{ij}$. Next, we rewrite $\sum_{i\in C} \pi_i$ in the form $\sum_{i\in C} \pi_i = \pi^\top e_C$ where $e_C = (e_i)_{N\times 1}$ and $e_i = 1$ if $i \in C$ and 0 otherwise. Since π is the stationary distribution of the Markov chain, we have $\pi = \pi P$. Thus

$$\pi^\top e_C = \pi^\top P e_C = \sum_{i\in C} \pi_i \Big(\sum_{j:(i,j)\in E} \frac{1}{w_i^+} \Big)$$

Now we have,

$$\begin{aligned} \sum_{i\in C} \pi_i \times w_C &= \sum_{i\in C} \pi_i \Big(\sum_{j:(i,j)\in E} \frac{1}{w_i^+} \Big) w_C \\ &\leq \sum_{i\in C} \pi_i \frac{w_{i,C}}{w_i^+} \Big(\sum_{t\in C} w_t^+ \Big) = \sum_{i,j\in C} \pi_i \frac{w_{i,C}}{w_i^+} \times w_C^+ \end{aligned}$$

Hence, the conclusion follows. The quality holds when all π_i equals to each other and $w_C = w_C^+$. This happens when C is a full dually connected clique and is disconnected from the rest of the network. □

7.3.3 Detecting Communities

7.3.3.1 Formulation

Proposition 7.3 discussed in the above paragraph establishes the connection between the persistence probability of a random walker staying within a community C and the local network topology. As a result, if we can maximize the later quantity, we can provide some insurance to the desired optimization with high confidence. Taking into

account this intuition, we propose Stable Community Detection (SCD) as an optimization problem defined as follow: Given a directed, weighted network $G = (V, E, w)$, find a community structure $\mathcal{C} = \{C_1, C_2, ..., C_q\}$ such that the overall total persistence probability is maximized:

$$\max \mathcal{R} = \sum_{C \in \mathcal{C}} \frac{w_C}{w_C^+}$$

subject to

$$C_i \cap C_j = \emptyset \qquad \forall i, j \in \{1, 2, ..., q\}$$

$$\bigcup_{i=1}^{q} C_i = V$$

Note that in our SCD formulation, the number of communities q will be determined by optimizing the objective function $\mathcal{R}$ and is not an input parameter. Indeed, optimizing $\mathcal{R}$ provides us q a very good estimate for the actual number of communities, as we will show in section 7.4.

7.3.3.2 Resolution limit analysis

Perhaps one of the most important properties a metric suggested for identifying community structure should satisfy is the ability of overcoming the resolution limit [58], i.e., the metric should be able to detect network communities even at different scaling levels. In this subsection, we analyze the resistance to resolution limit of our proposed function $\mathcal{R}$ by looking particularly at the condition in which two communities should be merged together. In what following, we simplify the situation by considering undirected networks.

Let us consider two communities C_1 and C_2. Let m_{12} be the number of edges connecting C_1 and C_2. In order to merge C_1 and C_2 into a bigger community, m_{12} should satisfy:

$$\frac{m_{C_1}}{d_{C_1}^+} + \frac{m_{C_2}}{d_{C_2}^+} \leq \frac{m_{C_1} + m_{C_2} + m_{12}}{d_{C_1}^+ + d_{C_2}^+}$$

The above condition is equivalent to:

$$\frac{m_{C_1}}{d^+_{C_2}} d^+_{C_1} + \frac{m_{C_2}}{d^+_{C_1}} d^+_{C_2} \leq m_{12}$$

which in turn implies $2\sqrt{m_{C_1} m_{c_2}} \leq m_{12}$. Without loss of generality, we can assume that $m_{C_1} \leq m_{C_2}$, thus $2m_{C_1} \leq m_{12}$. This violates the condition of even a weak community. Moreover, this inequality implies the sufficient condition to merge two adjacent communities depends on the local structure of two communities only, regardless of the rest of the network. This observation indicates that our proposed metric $\mathcal{R}$ is strongly against the resolution limit.

7.3.3.3 Connection to stability estimation

We next verify the following properties of network communities identified by optimizing our suggested metric $\mathcal{R}$: (1) links within a communities are of high stability and (2) links connecting communities are of low stability values. These two observations are shown in Propositon 7.4.

Proposition 7.4. *Let $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ be a community structure detected by optimizing $\mathcal{R}$, links within each C_i are of strong stability and those connecting communities are of weak stability values.*

Proof. For any node $p \in V$ and subset $A \subseteq V$, let $w_{p,A}$ be the total weight of all links that p has towards A and vice versa. By this definition, we obtain $w_p = w_{p,A} + w_{p,V\setminus A}$. For any community $C \in \mathcal{C}$, $s \in C$ and $p \notin C$, since p is not a member of C, we have

$$\frac{w_C}{w^+_C} > \frac{w_C + w_{p,C}}{w^+_C + w_p} = \frac{w_C + w_{p,C}}{w^+_C + w_{p,C} + w_{p,V\setminus C}},$$

because otherwise joining p to C will give a better value of $\mathcal{R}$. This equality equals

$$\frac{w_{p,C}}{w_p} < \frac{w_C}{w^+_C},$$

which in turn implies that the stability contribution of links joining p to C are insignificant in comparison to C as a whole.

Similarly, for any node $s \in C$, we have

$$\frac{w_C}{w_C^+} > \frac{w_C - w_{p,C}}{w_C^+ - w_p} = \frac{w_C - w_{p,C}}{w_C^+ - w_{p,C} - w_{p,V\setminus C}},$$

because otherwise excluding s from C will give a better $\mathcal{R}$. This inequality equals to

$$\frac{w_{s,C}}{w_s} > \frac{w_C}{w_C^+},$$

which in turn implies that the stability contribution of internal links of C are significant in comparison to C as a whole. □

7.3.3.4 A greedy algorithm for SCD problem

Analyzing the theoretical hardness of the SCD problem is an aspect beyond the scope of this paper. In fact, the NP-hardness of the SCD problem can be shown by a similar reduction to MODULARITY as in [7] (see also [98] and [99] for a comprehensive survey on similar graph clustering problems). Given its NP-hardness, a heuristic approach that can provide a good solution in a timely manner is therefore more desirable. In this section, we describe a greedy algorithm for the SCD problem consisting of community growing, strengthening and refinement phases described as follow.

Growing phase. This phase is responsible for discovering raw communities in the input network. Initially, all nodes are unassigned and do not belong to any community. Next, a random node is selected as the first member (or the seed) of a new community C, and consequently, new members who help to maximize C's persistence probability are gradually admitted into C. When there is no more node that can improve this objective of the current community, another new community is formed and the whole process is then cycled in the very same manner on this newly formed community.

Strengthening phase. We further rearrange nodes into more appropriate communities. Since new members are admitted into a community C in a random order, C's objective value could be further improve with the absence of some of it members as they can be

Algorithm 17 SCD Algorithm

Input: A directed weighted graph $G = (V, E, w)$
Output: Community structure $\mathcal{C}$

Growing Phase:
$\mathcal{C} \leftarrow \emptyset$
$A \leftarrow V$
while $\exists$ unassigned node $u \in A$ **do**
  $C \leftarrow \{u\}$
  $A \leftarrow A \setminus \{u\}$
  while $\exists\ v \in A$ such that $u_{C\cup\{v\}} > u_C$ **do**
    $v \leftarrow \arg\max_{v\in A}\{u_{C\cup\{v\}}\}$
    $C \leftarrow C \cup \{v\}$
    $A \leftarrow A \setminus \{v\}$
  end while
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$
end while

Strengthening Phase:
for $C \in \mathcal{C}$ **do**
  while $\exists u \in C$ such that $u_C < u_{C\setminus\{u\}}$ **do**
    $C \leftarrow C \setminus \{u\}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{u\}$
  end while
end for

Refining Phase:
while $\exists C_1, C_2$ such that $u_{C_1\cup C_2} > u_{C_1} + u_{C_2}$ **do**
  $(C_1, C_2) \leftarrow \arg\max_{C_1,C_2\in\mathcal{C}}\{u_{C_1\cup C_2} - u_{C_1} - u_{C_2}\}$
  $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C_1, C_2\}) \cup \{C_1 \cup C_2\}$
end while
Return $\mathcal{C}$

obstacles for the total stability. This requires the reevaluation of all C's members as a result. Therefore, in this phase, we exclude any node which reduces the persistence probability of a community and let them be singleton communities. The removal of such nodes creates more cohesive communities, i.e., communities with higher internal stability.

Refining phase. In the last phase, the global stability of the whole network is reevaluated. In particular, this last refinement phase looks at the merging of

two adjacent communities in order to improve the overall objective function. If two communities have a great number of mutual connections between them, it is thus more stable to merge them into one community. The final algorithm, which we call SCD algorithm, is presented in Alg. 17.

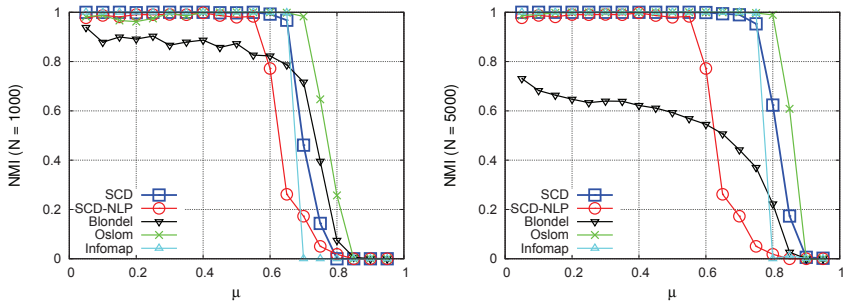
7.4 Experimental Results

In this section, we present our results on the discovery of network communities on both synthesized networks with known groundtruths and real-world social traces including NetHEPT and NetHEPT_WC collaboration and Facebook networks. We evaluate the following aspects of our proposed SCD framework (1) the effectiveness of our link stability estimation process, (2) the ability of identifying the general network community structure without the concept of community stability, i.e., how similar our detected communities are in comparison with the groundtruths, and (3) the ability of identifying stable communities in reference to the consensus of other state-of-the-art methods, including Blondel's [5], Infomap [100] and OSLOM [101] methods, after their multiple executions.

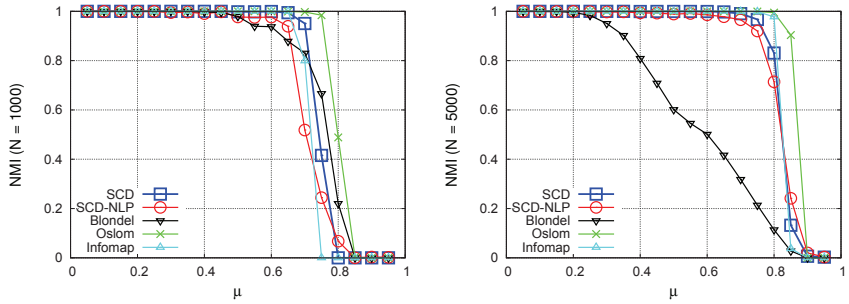
7.4.1 Datasets

(Synthesized networks) Of course, the best way to evaluate our approaches is to validate them on real-world networks with known community structures. Unfortunately, we often do not know that structures beforehand, or such structures cannot be easily mined from the network topologies. Although synthesized networks might not reflect all the statistical properties of real ones, they can provide us the known groundtruths via planted communities and the ability to vary other network parameters such as sizes, densities and overlapping levels, etc. Testing community detection methods on generated data has also becomes a usual practice that is widely accepted in the field [102].

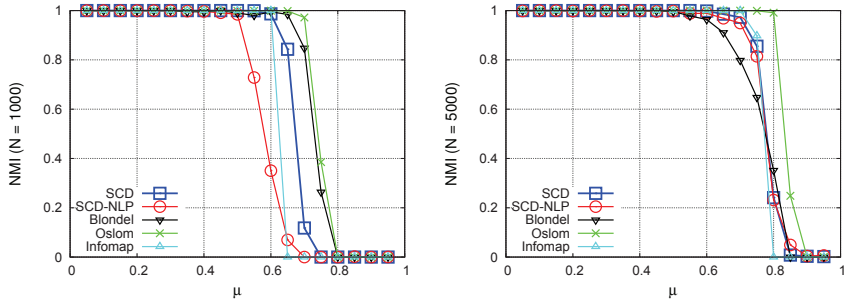
We use the well-known LFR benchmark [102] to generate 190 weighted and directed testbeds. Generated data follow power-law degree distribution and contain



A Networks with $minC$, $maxC$ unconstrained.



B Networks with $minC = 25$, $maxC = 50$ (small-size).



C Networks with $minC = 50$, $maxC = 100$ (big-size).

Figure 7-2. Results on synthesized networks with different community criteria.

embedded communities of varying sizes that capture characteristics of real-world networks. Parameters are: the number of nodes $N = 1000$ and 5000, the mixing parameter $\mu = [0.1...1]$ controlling the overall sharpness of the community structure, the minimum ($minC$) and maximum ($maxC$) of community sizes are set to (25, 50) for small-size and (50, 100) for big-size communities as in the standard settings. Each test is averaged over 100 runs for consistency.

(NetHEPT and NetHEPT_WC) The NetHEPT traces are widely-used datasets for testing social-aware detection methods [103][104]. These traces contain information, mostly the academic collaboration from arXiv's "High Energy Physics - Theory" section where nodes stand for authors and links represent coauthorships. In their deliverable, the NetHEPT networks contain 15233 nodes and 31398 links, and weights on edges are assigned by either uniformly at random (for NetHEPT data) or by weighted cascade (for NetHEPT_WC data) where $w_{uv} = 1/d_{in}(v)$ with $d_{in}(v)$ is the indegree of a node v.

(Facebook) This dataset contains friendship information among New Orleans regional network on Facebook, spanning from September 2006 to January 2009 [61]. The data contains more than 63K nodes (users) connected by more than 1.5 million friendship links with an average node degree of 23.5. In our experiments, the weight for each link between users u and v is proportional to the communication frequency between them, normalized on the whole network.

7.4.2 Metric

To measure the quality of the detected communities in comparison with the embedded groundtruths, we evaluate Generalized Normalized Mutual Information (NMI) [102]. Basically, the NMI(U,V) value of two structures U and V is 1 if U and V are identical and is 0 if they are totally separated. This is the most important metric for a community detection algorithm because it indicates how good the algorithm is in comparison with the planned communities. Higher NMI values are expected for a better community detection algorithm.

7.4.3 Effect of Link Stability Estimation

We first evaluate the effect of our link stability estimation on the detection of network communities by comparing NMI values of SCD and its version with No Link stability Prediction (SCD-NLP). Due to space limit, results of SCD and SCD-NLP are also reported in Figure 7-2, where those on general community structure detection are also presented.

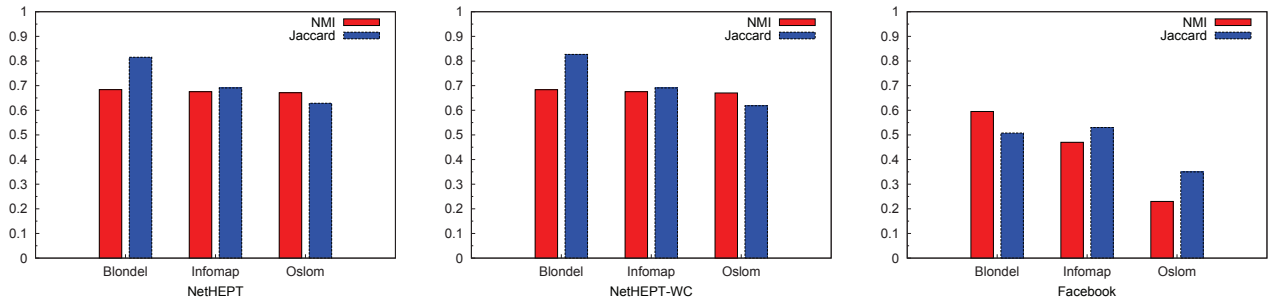


Figure 7-3. Performance of SCD in detecting stable communities on real social traces.

In general, SCD-NLP performs very competitively even without being preprocessed: on synthesized networks with no community size constraint (Figure 7-2A), its discovered communities are almost of perfect similarity to the embedded ones (NMI values approximately 1) on $\mu = [0...0.5]$ whereas the quality drops down quickly when μ is above 0.5. We note that this drop of detection quality is controversial and does not necessary imply a bad performance since networks with $\mu > 0.5$ is consider very stochastic, and thus, may not contain a clear community structure. Nevertheless, with the help of the stability estimation, the performance is now boosted up significantly on SCD as the detection qualities are very high even for $\mu > 0.65$ ($N = 1000$) and $\mu > 0.75$ ($N = 5000$), and only drop down when the networks are extremely stochastic ($\mu > 0.8$).

We next take a look at the cases where networks are constrained with small-sized (Figure 7-2B) and big-sized communities (Figure 7-2C). We observe that, when community sizes are constrained, SCD-NLP performs much better than before and even overcome its prior limit $\mu = 0.5$. In particular, the performance of SCD-NLP closely approaches that of SCD, especially in large networks ($N = 5000$). However, SCD-NLP appears to be sensitive to big-size communities in small networks as its quality drops down quickly in Figure 7-2C (left), and seems to favor small-size communities as its plots tend to tangle with those of SCD (Figure 7-2B). SCD detection quality, thanks to the stability estimation, stays wealthy in all test cases.

In summary, these results indicate that (1) without the stability estimation process, our suggested metric $\mathcal{R}$ appears to be a very good one to detect community structure in general directed and weighted networks, and (2) when the community size is constrained, link stability estimation has a little effect on the community detection quality. However, in real-world social network settings where community sizes are typically unknown, and therefore unconstrained, the stability estimation has a significant effect on the detection of network communities. These experiments also confirm the efficacy of our proposed stability estimation procedure.

7.4.4 General Community Structure Detection

We next investigate on SCD's ability to identify general network community structure, i.e., without community stability, in comparison with the aforementioned state-of-the-art detection methods. Results are reported in Figure 7-2.

In general, the performance of our SCD frameworks on synthesized networks appears to be better than those of Blondel and Infomap methods, and only lags behind Oslom's when the networks are heavily stochastic. When the community size is unconstrained, the detection quality of SCD and other methods, except for Blondel's, retain at nearly perfect on $\mu = [0...0.65]$ ($N = 1000$) and $\mu = [0...0.8]$ ($N = 5000$) and then all degrade quickly. Among the three methods, Infomap's performance appears to be sensitive to some certain mixing threshold μ as it NMI values tend to drop directly to 0, whereas Oslom and ours tend to drop down slower. On average, the NMI values of SCD are about 8% and 3% better than those of Blondel and Informap methods, and are about 2% lag behind those of Oslom method. Blondel's method, on the other hand, does not attain a good performance through due to low NMI values even at a low range of mixing value μ. An possible explanation for this behavior of Blondel's method is due to the effect of resolution limit, as we shall discuss below.

When the embedded communities are constrained with small and large community sizes, we observe the nearly same behavior of SCD, Oslom and Infomap methods as

depicted in Figures 7-2B and 7-2C. Blondel's method gets a significant improvement in these cases where its performance is closely related to the others. As we discussed above, one possible reason for the bad behavior of Blondel's method is due to the resolution limit of modularity objective function [58]. As the community size is unconstrained, this resolution limit can mislead Blondel method to merge some communities whose are of small sizes in comparison to the rest of the network, thus results in the low NMI values. On the other hand, this resolution limit does not take effect when size constraints are imposed and thus the significant improvement. Our SCD framework, as shown in section 7.3.3.2, can withstand this scaling limit as its obtains highly competitively results. Moreover, the difference between our SCD and other methods are insignificant on average which indicates that all methods are able to detect network communities with high quality. This is not a surprising result since Blondel, Oslom and Informap are currently state-of-the-art methods but a great motivation and award for our SCD framework.

7.4.5 Results on Stable Community Detection

In order to compare our results to the consensus of other detection methods, we will adopt a strategy recently proposed in [50]. In particular, given a specific community detection method $\mathcal{A}$, its consensus (or stable) communities can be determined by: (i) execute $\mathcal{A}$ on G n_p times to have n_p partitions (ii) find the matrix $D = (D_{ij})$ where D_{ij} is the probability which vertices i and j of G are assigned to the same cluster among n_p partitions (iii) all D_{ij}'s that are below a threshold τ will be disregarded (iv) Apply $\mathcal{A}$ on D n_p times, so to create n_p partitions and (v) if all partitions are equal, stop (the result matrix would be block diagonal). Otherwise go to step (ii). As suggested in [50], the resulted communities are ideal candidates for stable structures as members commit to their communities. We also compute the Jaccard index $J(U, V) = \frac{|A \cap B|}{|A \cup B|}$ to better evaluate the quality of the detected stable communities. Results are represented in Figure 7-3.

As illustrated by the subfigures, even a single run of SCD is able to obtain very high NMI scores and Jaccard indicies in comparison with the consensus of other methods after multiple runs. In particular, community structures discovered by SCD on NetHEPT and NetHEPT_WC obtain nearly 70% similarity in comparison with Blondel, Infomap and Oslom methods, meanwhile the Jaccard indicies indicate that, in average, almost 66% number of nodes are found in common between SCD and the core structure of other competitors. This show that communities discovered by SCD are indeed highly overlap with core community structures identified by other detection methods, which in turns implies that those clusters found by SCD are stable with high confidence. Surprising, in both NetHEPT and NetHEPT_WC networks, we observe the high similarity among the consensus of Blondel, Infomap and Oslom methods even with difference in edge weight distribution. This observation indicate those identified communities by SCD are, in fact, stable in these networks.

Even in Facebook, a large network with real social interactions, the similarity between consensus communities discovered by other methods and by SCD are still of high similarity with nearly 60%, 50% similarity to those found by Blondel and Infomap methods with over 50% overlap in the stable partitions as indicated by the Jaccard indices. The achieved NMI values in comparison with Oslom method are relatively low as their core communities do not appear to highly overlap (Jaccard index of only 35%). We note that this low similarity does not indicate the unstability community structure of our SCD framework since communities detected by Oslom can be overlapped with each other, while SCD works towards disjoint community structure. Nevertheless, as just a single run, the above competitively results in reference to other state-of-the-art methods confirm the efficay and quality of our method in detecting stable network communities in OSNs.

7.5 Conclusion

In this work, we investigate community structures in directed OSNs with more focus on community stability. As an effort towards the understanding of stable communities, we suggest an estimation procedure which provides helpful insights into the stability of links in the input network. Based on that, we propose SCD, a framework to identify community structure in directed OSNs with the advantage of community stability. We explore an essential connection between the persistence probability of a community at the stationary distribution and its local topology, which is the fundamental point to back our SCD framework. Finally, we certify the efficiency of our approach on both synthesized datasets with embedded communities and real-world social traces, including NetHEPT collaboration and Facebook social networks, in reference to the consensus of other state-of-the-art detection methods. Highly competitive empirical results confirm the quality and efficiency of SCD on identifying stable communities in OSNs.

CHAPTER 8
ASSESSING NETWORK COMMUNITY STRUCTURE VULNERABILITY

8.1 Introduction

As a first study on assessing the vulneraibility of the network community structure, in this paper, we take the first step on understanding how the failures of crucial nodes in the network will affect its community structure. Particularly, we are interested in identifying network nodes whose removals trigger a significant restruction of the current community structure. Formally, given the input network and a positive number k, we introduce the Community Structure Vulnerability (CSV) which aims to find out a set S of k nodes whose removal maximally transforms the current network community structure to a totally different one, i.e., the new community structure resulted from the removal of S is of least similarity to the original one, evaluated via the Normalized Mutual Information [105] measure.

Knowledge about this crucial vulnerability of network community structure is of considerable usage, especially for social-aware methods in mobile ad-hoc and online social networks (OSNs). To give a sense of its effects, consider message forwarding in DTNs. Since social-based forwarding strategies in DTNs rely on the highest ranked nodes in each community to forward the message [106][76], the knowledge of this vulnerability can help to either design routing algorithms that do not overload those crucial devices, if they are those highly ranked ones in a community, or to design an effective backup plan when some of them may fail at the same time. In worm containment application in OSNs [18][16], this knowledge can provide helpful insights into the protection of those sensitive nodes, if they are indeed high influential users, once worms spread out in the network. As a result, the identification of nodes whose removal triggers a massive restruction of the community structure is extremely important for the network's regular operation. However, under a minor structural change when a node is excluded from a community, this particular community can either stay intact if the

removed node is less important, or can be broken down into smaller subcommunities which can further be merged to other communities if the current node is of great important to the community. This unpredictable transformation of network communities together with their large scales in reality make the assessment of community structure vulnerability a fundamental yet challenging problem.

8.2 Problem Definition

In this section, we first define the graph notations that will be used thoroughly in this paper. We then describe Normalized Mutual Information (NMI) [105], a concept in Information Theory, as a metric to assess the difference between community structures before and after the removal of important nodes. Finally, we formally define the Community Structure Vulnerability problem - our main focus in this paper.

(Notations) Let $G = (V, E)$ be an undirected unweighted graph representing a network where V is the set of $|V| = N$ nodes (e.g., users), and E is the set of $|E| = M$ links. For any node $u \in V$ and a set $C \subseteq V$, let $N(u)$, d_u and d_u^C be the set of all neighbors of u, its degree in G and its degree in C, respectively. Furthermore, let $n_C = |C|$ be the number of nodes and m_C be the number of internal edges in C.

(Community structure) Denote by $\mathcal{A}$ the specific community detection algorithm that will be applied on G, and by $X = \{X_1, X_2, ..., X_{c_X}\}$, $Y = \{Y_1, Y_2, ..., Y_{c_Y}\}$ the two (possibly overlapped) community structures of c_X and c_Y communities detected by $\mathcal{A}$ before and after the removal of a set S of k nodes in G, respectively. Mathematically, X and Y are represented as $X = \mathcal{A}(G)$ and $Y = \mathcal{A}(G[V \backslash S])$, where $G[V \backslash S]$ is the subgraph induced by $V \backslash S$ on G. For any index $i = 1, ..., c_X$ and $j = 1, ..., c_Y$, let $x_i = |X_i|$, $y_j = |Y_j|$, and $n_{ij} = |X_i \cap Y_j|$. Finally, let $\bar{x} = \sum_{i=1}^{c_X} x_i$, $\bar{y} = \sum_{j=1}^{c_Y} y_j$ and $\bar{n} = \sum_{i=1}^{c_X} \sum_{j=1}^{c_Y} n_{ij}$ be the total size of communities in X and Y, and the total number of common nodes shared between X and Y, respectively.

(Normalized Mutual Information) In order to evaluate how much the network community structure changes before and after the removal of important nodes, we

utilize the concept of Normalized Mutual Information suggested in [105]. Basically, given two structures X and Y, $NMI(X, Y)$ is 1 if X and Y are identical and is 0 if X and Y are totally separated, and the higher the NMI score, the more similarity between X and Y. As a result, NMI is a well-suited metric dedicated for certifying the quality of community structures discovered by different detection algorithms. The effectiveness of this widely-accepted measure has also been extensively verified in the literature [102]. Formally, $NMI(X, Y)$ is defined as

$$NMI(X, Y) = \frac{2I(X, Y)}{H(X) + H(Y)},$$

where $H(X)$, $H(Y)$ and $I(X, Y)$ are the entropy of structures X and Y, and the Mutual Information conveyed between them, respectively. More details about NMI formulation will be elaborated in our analysis.

(Problem definition) Finally, the Community Structure Vulnerability (CSV) problem is formulated as follow.

Definition 1. *Given a network represented by an undirected and unweighted graph G, a specific community detection algorithm $\mathcal{A}$, and a positive integer $k \leq N$, we seek for a subset $S \subseteq V$ such that*

$$S = \underset{T \subseteq V, |T|=k}{\arg\min} \{NMI(\mathcal{A}(G), \mathcal{A}(G[V \setminus T]))\}.$$

In other words, CSV problem seeks for a subset $S \subseteq V$ of k nodes whose removal results in the maximum difference between the initial community structure X and the new community structure Y detected by $\mathcal{A}$ on $G[V \setminus S]$. We call S the Node-Vulnerability set of G since its removal maximally transforms network communities of G to different structures.

Remark. The formulation of CSV requires the community detection algorithm $\mathcal{A}$ as an input parameter. Because there is not yet an universal agreement or accepted definition of a network community, this input is necessary in the sense that different

algorithms with different objective functions might favor different sets of nodes, and thus, a good solution set for one community detection algorithm may not be good for the others. However, when there is a clear objective function for finding community structure, such as maximizing Modularity Q [102] or the total internal density [76], this requirement can be lifted. Nevertheless, the node selection strategy that relies more on the input network and less on the community detection algorithm is always of desire.

8.3 Analysis of NMI Measure

In this section, we investigate the possible conditions on sizes and the number of communities that can potentially lead to either the global or local minimization of $NMI(X, Y)$. We stress that these conditions are by no means universal or exhaustive since some of them might not hold true simultaneously, given the input parameters. Indeed, what we hope for is these conditions would provide us key insights into the selection of important nodes to maximally separate X and Y. In the coming paragraphs, we first discuss the NMI formulation in a greater detail, and then analyze it in terms of both disjoint and overlapping community structures.

8.3.1 NMI Formulation

To evaluate $NMI(X, Y)$ [105] where $X = \{X_1, X_2, \ldots, X_{c_X}\}$ and $Y = \{Y_1, Y_2, \ldots, Y_{c_Y}\}$, we start out by considering community assignments X_i and Y_j, where X_i and Y_j indicate the community labels of a node t in X and Y, respectively. Without loss of generality, we can aslo assume that the labels X_i and Y_j are also values of two random "variables" X and Y (here we reuse notations X and Y to denote the two random variables), with joint distribution

$$P(X_i, Y_j) = P(X = X_i; Y = Y_i) = n_{ij}/(N - k),$$

and individual distribution

$$P(X_i) = P(X = X_i) = x_i/N,$$

$$P(Y_j) = P(Y = Y_j) = y_j/(N - k).$$

The entropy (or uncertainty) of X and Y is defined as [107]

$$H(X) = -\sum_{i=1}^{c_X} P(X_i)\log P(X_i) = -\sum_{i=1}^{c_X} \frac{x_i}{N}\log\frac{x_i}{N},$$

$$\begin{aligned} H(Y) &= -\sum_{j=1}^{c_Y} P(Y_j)\log P(Y_j) = -\sum_{j=1}^{c_Y} \frac{y_j}{N-k}\log\frac{y_j}{N-k} \\ &= \frac{1}{N-k}\big(\bar{y}\log(N-k) - \sum_{j=1}^{c_Y} y_j \log y_j\big). \end{aligned}$$

Note that in CSV problem, X can be derived straightforwardly based on $\mathcal{A}$ and G, and thus, quantities x_i's can also be inferred from these input parameters. Therefore, we simply consider x_i's and $H(X)$ as constants in this paper.

The Mutual Information $I(X, Y)$ [107] of two random variables X and Y is defined as

$$\begin{aligned} I(X, Y) &= \sum_{i=1}^{c_X}\sum_{j=1}^{c_Y} P(X_i, Y_j)\log\frac{P(X_i, Y_j)}{P(X_i)P(Y_j)} \\ &= \sum_{i=1}^{c_X}\sum_{j=1}^{c_Y} \frac{n_{ij}}{(N-k)}\log\frac{Nn_{ij}}{x_i y_j}. \end{aligned}$$

This measure is symmetric and it tells us how much we know about variable (or structure) Y if we already know about variable X, and vice versa. However, as indicated in [105][102], Mutual Information itself is not ideal as a global similarity metric since any subpartition of a given community structure X would result in the same mutual information with X, even though they can possibly be very different from each other. As a result, [105] introduces the Normalized Mutual Information which can overcome that limitation. Formally, NMI of two random variables X and Y is defined as

$$NMI(X, Y) = \frac{2I(X, Y)}{H(X) + H(Y)} \tag{8–1}$$

In term of notations, $NMI(X, Y)$ can be written as

$$\frac{2\sum_{i=1}^{c_X}\sum_{j=1}^{c_Y} n_{ij}\log\frac{Nn_{ij}}{x_i y_j}}{(N-k)H(X)+\bar{y}\log(N-k)-\sum_{j=1}^{c_Y} y_j\log y_j} \tag{8–2}$$

8.3.2 Minimizing NMI in a Disjoint Community Structure

When network communities are disjoint from each other, we have $X_i \cap X_s = \emptyset$, $\cup_{i=1}^{c_X} X_i = V$, $Y_j \cap Y_t = \emptyset$, and $\cup_{j=1}^{c_Y} Y_j = V \backslash S$ for all $i, s = 1, \ldots, c_X$ and all $j, t = 1, \ldots, c_Y$. As a result, the following equalities hold true: $\bar{x} = \sum_{i=1}^{c_X} x_i = N$, $\bar{y} = \sum_{j=1}^{c_Y} y_j = N-k$ and $\bar{n} = \sum_{ij} n_{ij} = N-k \quad (*)$.

8.3.2.1 Minimizing NMI within a community

We first investigate the behavior of $NMI(X, Y)$ in a special case where only one specific community of X is affected by the removal of set S of k nodes while other communities stay intact. We can assume that X_1 is the targeted community which is further divided into p smaller subcommunities of sizes $s_1, s_2, \ldots, s_p$ satisfying $\sum_{j=1}^{p} s_j = x_1 - k$. In this case

$$\begin{aligned} H(Y) &= \sum_{j=1}^{p}\frac{s_j}{N-k}\log\frac{N-k}{s_j}+\sum_{i=2}^{c_x}\frac{x_i}{N-k}\log\frac{N-k}{x_i} \\ &= \frac{(x_1-k)\log(N-k)+\sum_{i=2}^{c_X} x_i\log\frac{N-k}{x_i}-\sum_{j=1}^{p} s_j\log s_j}{N-k}, \end{aligned}$$

and

$$\begin{aligned} I(X, Y) &= \sum_{j=1}^{p}\frac{s_j}{N-k}\log\frac{N}{x_1}+\sum_{i=2}^{c_x}\frac{x_i}{N-k}\log\frac{N}{x_i} \\ &= \frac{x_1-k}{N-k}\log\frac{N}{x_1}+\sum_{i=2}^{c_X}\frac{x_i}{N-k}\log\frac{N}{x_i}. \end{aligned}$$

Thus, $NMI(X, Y)$ is minimized when $\sum_{j=1}^{p} s_j\log s_j$ is minimized. Since function $s\log s$ is strictly convex for any $s > 0$, we apply Jensen's inequality [107] to this summation and get

$$\frac{1}{p}\sum_{j=1}^{p} s_j\log s_j \geq \frac{\sum_{j=1}^{p} s_j}{p}\log\frac{\sum_{j=1}^{p} s_j}{p} = \frac{x_1}{p}\log\frac{x_1}{p},$$

with the equality holds when all s_j's are equal to each other. It reveals from this inequality that, in order to further minimize the RHS quantity, one can try to break X_1 into as many smaller communities of the relatively same size as possible (i.e., to enlarge p as much as possible while ensuring s_i's are all equal). This intuition makes senses since a new structure of X_1 with all singleton communities will incur $\sum_{j=1}^{p} s_j \log s_j = 0$, and hence, will maximize $H(Y)$ and in turn will minimize $NMI(X, Y)$. However, since the new structure of X_1 depends on the community detection algorithm $\mathcal{A}$, the all-singleton communities scenario might not always be the case. Furthermore, will this crucial observation hold true in a general disjoint and overlapping community structure? We tend to lean over the affirmative answer through our analysis in the coming subsections.

8.3.2.2 Minimizing NMI in a general disjoint community structure

In general disjoint community structure, the equalities $(*)$ help to simplify $NMI(X, Y)$ (eq. 8–2) to

$$\frac{2\sum_{i=1}^{c_X}\sum_{j=1}^{c_Y} n_{ij}\log\frac{Nn_{ij}}{x_i y_j}}{(N-k)H(X) + (N-k)\log(N-k) - \sum_{j=1}^{c_Y} y_j \log y_j}.$$

In order to minimize the above ratio, one would seek for the conditions in which the numerator of $NMI(X, Y)$ is minimized while its denominator is also maximized. To maximize the latter quantity, we need to minimize $\sum_{j=1}^{c_Y} y_j \log y_j$. Applying Jensen's inequality to this summand gives

$$\frac{1}{c_Y}\sum_{j=1}^{c_Y} y_j \log y_j \geq \frac{\bar{y}}{c_Y}\log\frac{\bar{y}}{c_Y} = \frac{N-k}{c_Y}\log\frac{N-k}{c_Y},$$

and thus $\sum_{j=1}^{c_Y} y_j \log y_j$ can attain it minimum at $(N-k)\log\frac{N-k}{c_Y}$ with equality holds when all y_j's are equal to each other. As N and k are input parameters, $\log\frac{N-k}{c_Y}$ can further be minimized when c_Y is as large as possible, while requiring y_j's to be equal to each other. Mathematically, this can be achieved when Y contains exactly $c_Y \equiv (N-k)$ singleton communities. However, since our problem depends on the detection algorithm, this inequality advises that the newly community structure Y should contain as many communities of relatively the same size as possible. We take into account

this observation as it will play a key role in our important-node selection process. This observation is also coincident with what inferred in the prior special case, and intuitively agrees with the concept of Critical Node Detection (CND) [21] and Balanced Graph Partitioning (BGP) [108] whose goals aim to delete nodes and cut the input graph into p connected components of relatively the same size. However, CSV fundamentally differs from these problems in the senses that connected components in BGP and CND do not necessarily reflex network communities.

In order to minimize the numerator, we rewrite it as

$$I(X, Y) = \frac{1}{N-k}(\sum_{ij} n_{ij} \log \frac{N n_{ij}}{y_j} - \sum_{ij} n_{ij} \log x_i).$$

Applying Log Sum Theorem [107] to the first summand gives

$$\begin{aligned} I(X, Y) &\geq \frac{1}{N-k}\left(\bar{n} \log \frac{N\bar{n}}{c_X \bar{y}} - \sum_{ij} n_{ij} \log x_i\right) \\ &= \log \frac{N}{c_X} - \frac{1}{N-k}\sum_i (x_i - l_i) \log x_i, \end{aligned}$$

because $\bar{n} = \bar{y} = N - k$ and $\sum_{j=1}^{c_Y} n_{ij} = x_i - l_i, \forall i = 1, \ldots, c_X$, where l_i is the number of deleted (or lost) nodes in community X_i, and l_i's satisfy $\sum_{i=1}^{c_X} l_i = k$. The equality holds when n_{ij}/y_j is a constant, say $\gamma \geq 0$, for all $i = 1, \ldots, c_X, j = 1, \ldots, c_Y$. If we assume that this is the case, then $\sum_{j=1}^{c_Y} n_{ij} = \gamma \sum_{j=1}^{c_Y} y_j = \gamma(N-k)$, which in turn implies $N - k = \sum_{ij} n_{ij} = c_X \gamma (N-k)$. Hence, $\gamma = 1/c_X$ and thus, $l_i = x_i - (N-k)/c_X$. Therefore, to minimize the second summand, the equation $l_i = x_i - (N-k)/c_X$ advises that we should put more focus on (i.e., remove more nodes in) big-sized communities X_i of X to break it into smaller modules. This breaking down of big-sized communities partially supports the prior observation that communities of Y should have relatively the same size. Note that in this analysis, we have assumed that n_{ij}/y_j is a constant for all pair of i and j. In practice, this might not always be the case since real communities can be distributed differently based on the underlying detection algorithm. Nevertheless, we find

this observation helpful as it suggests a general instruction for selecting important nodes in the network.

8.3.3 Minimizing NMI in an Overlapped Community Structure

The minimization of $NMI(X, Y)$ measure is much more complicated when network communities can overlap with each other. In particular, the conditions $\cup_{i=1}^{c_X} X_i = V$ and $\cup_{j=1}^{c_Y} Y_j = V \backslash S$ still hold in this case; however, $X_i \cap X_s$ and $Y_j \cap Y_t$ might not be empty for some $i, s = 1, \ldots, c_X$ and $j, t = 1, \ldots, c_Y$. These facts indicate that $\bar{x} = \sum_{i=1}^{c_X} x_i \geq N$, $\bar{y} = \sum_{j=1}^{c_Y} y_j \geq N - k$ and $\bar{n} = \sum_{ij} n_{ij} \geq N - k$.

Our analysis strategy in this case is similar to the prior one as we also strive for maximizing the denominator while minimizing the numerator of $NMI(X, Y)$ (eq. 8–2). Because $\bar{n} \geq N - k$, the minimization of the top term $I(X, Y)$ no longer depends only on x_i's anymore. One way to work around this issue is to investigate the relative correlation between the total community size $\bar{y}$ and the number of communities c_Y. Let $\alpha_{\mathcal{A}} = \frac{\bar{y}}{c_Y}$ be the ratio between these two quantities, or in other words, the averaged community size. The denominator of $NMI(X, Y)$ is evaluated as

$$\begin{aligned} \bar{y} \log(N-k) - \sum_{j=1}^{c_Y} y_j \log y_j &\leq \bar{y} \left(\log(N-k) - \frac{\log(\bar{y}/c_Y)}{c_Y} \right) \\ &= \bar{y} \log (N-k) - \alpha_{\mathcal{A}} \log \alpha_{\mathcal{A}}. \end{aligned}$$

with equality holds when all y_j's are equal to each other. To further maximize this denominator, we need $\bar{y}$ to be as large as possible while keeping $\alpha_{\mathcal{A}}$ as small as possible, i.e., the new community structure Y should contain more and more communities as to increase c_Y as well as to lower down $\alpha_{\mathcal{A}}$.

Due to the dependence on the specific detection algorithm $\mathcal{A}$, this optimization on the correlation between $\bar{y}$ and c_Y might not be globally achieved. However, a coarse analysis between $\bar{y}$ and c_Y can relatively be conducted in the following senses: if we assume that $\bar{y}$ is within a constant factor of the total number of actual nodes $(N - k)$, i.e., $\bar{y} \leq a_0(N - k)$ for some constant $a_0 > 1$, we can then increase the value of the

RHS by breaking as many communities as possible while keeping them having the size (i.e., enlarge c_Y and keep y_j's are all the same), which helps to reduce the impact of $\alpha_{\mathcal{A}} \log \alpha_{\mathcal{A}}$. This observation, though relative, agrees with what we achieved in the case of disjoint community structure. In an unfortunate case where $\bar{y}$ is not known to be within any constant factor of $(N - k)$, the observation might not hold since both $\bar{y}$ and c_Y can be arbitrary large and thus, $\alpha_{\mathcal{A}} \log \alpha_{\mathcal{A}}$ could still be relatively small.

Next, applying Log Sum Theorem on the numerator yields

$$I(X, Y) = \sum_{ij} n_{ij} \log \frac{N n_{ij}}{x_i y_j} \geq \bar{n} \log \frac{N \bar{n}}{\bar{x} \bar{y}},$$

with equality holds when $\frac{N n_{ij}}{x_i y_j}$ is a constant for all $i = 1, \ldots, c_X$ and $j = 1, \ldots, c_Y$. Thus, one can try to minimize $I(X, Y)$ by deleting nodes in such a way that $\bar{n}$ is maximized and $\bar{y}$ is minimized while making sure that $\frac{N n_{ij}}{x_i y_j}$ is a constant. As a result, this minimization of $I(X, Y)$ is a multiple-objective optimizations problem which may not have a feasible solution. However, if we assume that the later condition is imposed, i.e., $\frac{N n_{ij}}{x_i y_j} = \beta_{\mathcal{A}}$ for some constant $\beta_{\mathcal{A}} > 0$, then $n_{ij} = \frac{\beta_{\mathcal{A}} x_i y_j}{N}$, and thus $\bar{n} = \frac{\beta_{\mathcal{A}}}{N} \bar{x} \bar{y}$. This reduces the above inequality to

$$I(X, Y) \geq \frac{\bar{x}}{N} \beta_{\mathcal{A}} \bar{y} \log \beta_{\mathcal{A}} N.$$

The RHS of the inequality advises that, in order to minimized $I(X, Y)$, the total size of network communities should not be too large while the overlapping ratio of every community should be equal to each other and be as small as possible. This is a different criterion from the disjoint community structure point of view.

8.4 A Solution to CSV Problem

In the following paragraphs, we consider the scenario when maximizing the internal density [76] is the objective function for finding network communities, i.e., communities of G are assumed to have optimized internal densities. In this manner, we present genEdeg, an algorithm for solving CSV problem that is independent of the underlying community detection algorithm $\mathcal{A}$. Our solution strategy will try to break

larger communities to as many small ones as possible while looking for them to have the relatively same size with small overlapping ratios. The idea of our strategy is based on the following intuition: since communities in X are optimized for their internal density, they are likely to contain strong substructures that are tightly connected which form the cores of these communities. As a result, the removal of crucial nodes in a core might potentially break the community into smaller modules. Moreover, as nodes in a core are tightly connected, there should be some edge that generate them, i.e., all nodes in the core are incident to both endpoints of this edge. Inspired by this intuition, our strategy works towards the identification of these generating edges of a community, and then seek for the minimum set of generating edges that composes the original communities.

Let D be a subset of V. Denote by $\Psi(D) = \frac{2m_D}{n_D(n_D-1)}$ the internal density of D and by $\tau(D) = \frac{n_D(n_D-1)}{2} - \frac{2}{n_D(n_D-1)}$ the threshold function on the internal density of D, respectively. For any nodes $u, v \in D$, if edge (u, v) is not in E, we call it a missing edge in D. In addition, we call an edge in D "negative" if it is incident to a missing edge in D, and "positive" otherwise. We define the concept of generating edges of D as follow

Definition 2. *(Generating edge) For any edge (u, v) in D, if $D = (D \cap N(u) \cap N(v)) \cup \{u, v\}$ and $\Psi(D) \geq \tau(D)$, we call (u, v) a generating edge of D. We further call D a local core generated by (u, v) and write $gen(u, v) = D$.*

For any community C of G, a set $L \subseteq E$ is called a "generating edge set" of a C if $\cup_{(u,v)\in L} gen(u, v) = C$. Since C can be generated by different generating edge sets and we are constrained on the node budget, we would intuitively seek for the generating edge set of minimal cardinality.

Definition 3. *(Minimum Generating Edge Set) Given a community C of G, the MGES problem seeks for a generating edge set L^* of C with the smallest cardinality.*

The cores generated by edges in a MGES of a community C of G are tightly connected and they all together compose C. As a result, if we delete an endpoint of every edge in a MGES, C will be broken into smaller modules with the number of

modules is at least the number of edges in a MGES (Lemma 16). Since our goal is to break the current community structure X into as many new communities as possible, the removal of crucial nodes defined by edges in a MGES will be a good heuristic for this purpose. But first and foremost, we need to characterize all MGESs in the current community structure X based only on the input network G. Lemma 17 realizes the location of the generating edge(s) of a local core in a community C: they have to adjacent to nodes with the highest degree in C. Based on this result, we present in Alg. 18 a procedure that can correctly find the MGES of a given community C (Theorem 8.1).

Algorithm 18 An optimal algorithm for finding the MGES

Input: Network $G = (V, E)$ and a community $C \in X$;
Output: Minimum generating edge set L^* of C;

0. Mark all nodes as "unassigned" and $L^* = \emptyset$.
1. Remove all negative edges in C. If any edge(s) survive, they are candidate for generating edges in their corresponding communities, including them to L^*, go to step 2. Else, go to step 3.
2. Reconstruct local cores based on generating edges found in step 1. Mark all nodes in those communities as "assigned". Discard generating edges in L^* that fall into any newly constructed communities. Return if all edges are assigned.
3. Find the set U as in Lemma 17. Find the edge in $NE(U)$ that can generate a local community having the largest size. Include this edge to L^* and mark all nodes in the new local community as "assigned". Ties are broken randomly. Return if all edges are assigned.
4. If there are still unassigned nodes, say the set $I \subseteq C$, construct $G_1 = G[(I \cup N(I)) \cap C]$. Go to back to step 1.

Lemma 16. *Let L^* be a MSGE of a community C. The removal of an endpoint in every edge of L^* will break C into at least $|L^*|$ subcommunities.*

Proof. Clearly, the removal of an endpoint of every edge in L^* will degrade the internal density of each core since the endpoint of the generating edge is of full degree in its core. Now, if the number of subcommunities resulted in the node removal is less than $|L^*|$, it means there are at least two cores that are merged together. That is there are cores c_1 and c_2 are merged together even with less internal density. This should not be

the case since otherwise, they have to be identified as a single core at the first place. Their combination, as a result, implies that C has a MGES of size less than $|L^*|$, which raises a contradiction to the assumption that L^* is a MGES of C. □

Lemma 17. *Let C be a subset of V, $U = \{u \in C | d_u^C$ is the highest in $C\}$ and $NE(U) = \{(u, v) | u \in U$ or $v \in U$ but not both $\}$. Then, $|NE(U) \cap L^*| \geq 1$.*

Proof. After each refreshment in step 2, let u be the node with the highest indegree in C. After step 1 of Alg. 18, all negative edges are deleted since they do not contribute to the actual generating set L^*. As such, edges incident to u are not negative. This in turn implies that they are candidates for generating edges. Now, iterate through all edges incident to u and choose the one that generates the biggest-sized core. This edge should be in the list L^*. □

Theorem 8.1. *Let d_C be the maximum in-degree of a node in C. Alg. 18 takes $O(d_C|C|)$ time in the worst case scenarios and returns an optimal solution for MGES problem.*

Proof. Since every time Lemma 17 makes sure that at least one edge should be added to L^* and the procedure terminates when no edges left, the Alg. 18 should terminate. Moreover, it is verifiable that Alg. 18 take at time as most the number of edges in C, which is $O(d_C|C|)$. Also, due to the intense internal density of a core, every time an edge is added into L^*, that edge actually generates the largest core possible. The proof follows from this fact, Lemma 17 and the exhaustive property of Alg. 18. □

Algorithm 19 *genEdge* - A node selection strategy for CSV based on generating edges

Input: Network $G = (V, E)$, $X = \mathcal{A}(G)$;
Output: A set $S \subseteq V$ of k nodes;

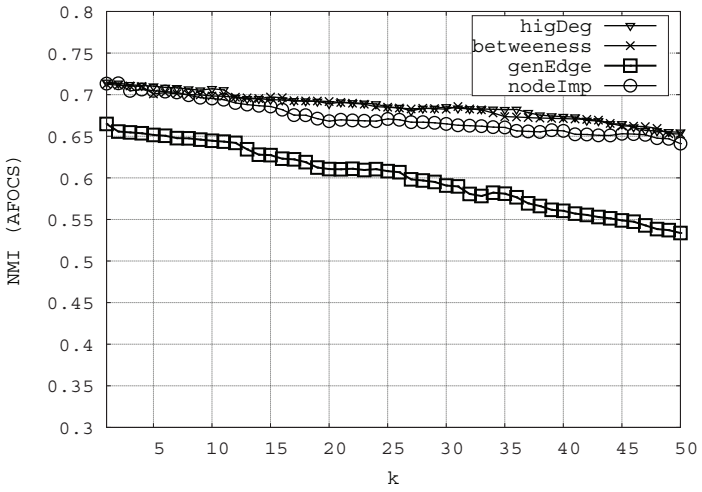
1. Use Alg. 18 to find $L^*_{X_i}$ for all communities X_i's in X.
2. Sort all communities X_i's in X by their sizes of MGSEs.
3. Sort all nodes in G by the number of generating edges that they are incident to in X_i. If there is a tie, sort them by their degrees in G.
4. Return top k nodes in step 3.

With the optimal solution of MGES taken into account, we next suggest a heuristic for selecting important nodes following the guidelines suggested in the previous. In particular, our heuristic selects nodes in a greedy manner, starting from communities that have large-size MGESs. Moreover, in the MGES of each community C, we give priority to nodes that are incident to more generating edges since their removals will break C into more subcommunities.

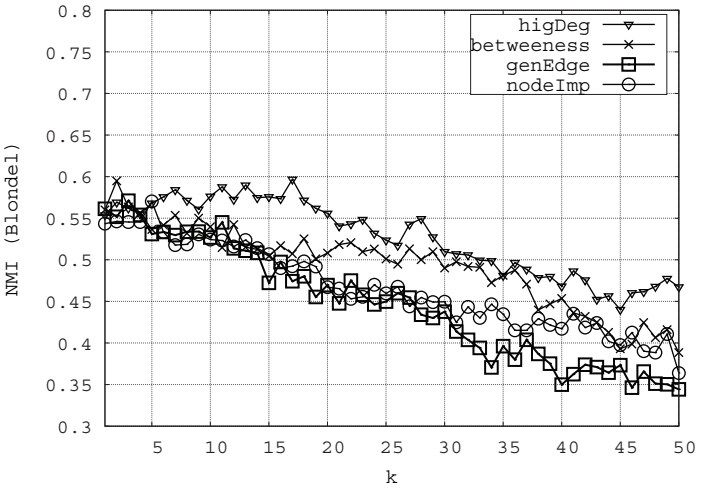
8.5 Experimental Results

In this section, we show the empirical results of our node selection strategy for CSV on both synthesized networks with known community structures and real-world social traces including the Reality mining cellular dataset [82], Facebook [61] and Foursquare [109] social networks. In order to certify the performance of our approach, we compare the results obtained by the following methods: High degree centrality ($highDeg$) selects top k nodes in G with the highest degrees, $betweeness$ centrality ($betweeness$) selects top k nodes in G with the highest $betweeness$es (where the $betweeness$ of a node u is the number of shortest paths in G that pass through u), Generating edges ($genEdge$) - our strategy described in Alg. 19, and finally, Node Importance ($nodeImp$) [54] selects top k nodes by their importance to the community structure.

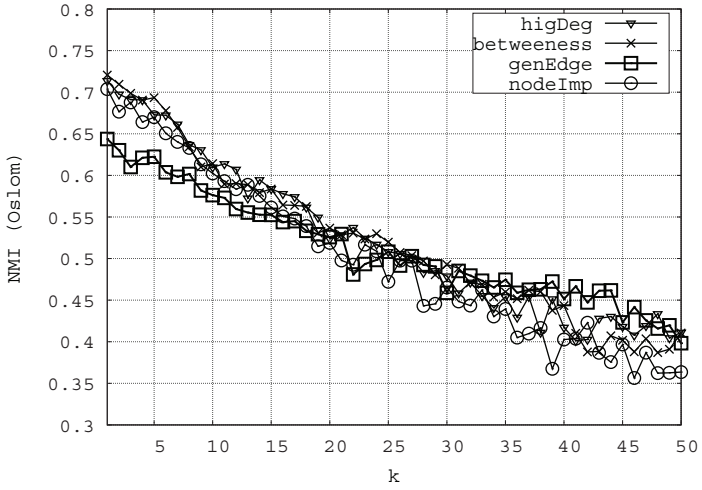
We first examine the effect of the underlying community detection methods by comparing results obtained by AFOCS [76], Blondel [5] and Oslom [101] algorithms to the embedded groundtruths. In particular, we set X to be the groundtruth community structure and when S is removed from the network $NMI(X, Y)$ is reported, where $Y = AFOCS(G[V \backslash S])$, $Y = Blondel(G[V \backslash S])$ and $Y = Oslom(G[V \backslash S])$, respectively. These methods have been empirically certified in the literature to the best algorithms for finding non-overlapping and overlapping community structure [102]. Verifying our strategy on synthesized networks not only certifies its performance but also provides us the confidence to its behaviors when applied to real-word traces. We next demonstrate the following quantities (1) the NMI differences between community structures before



A NMI scores by AFOCS

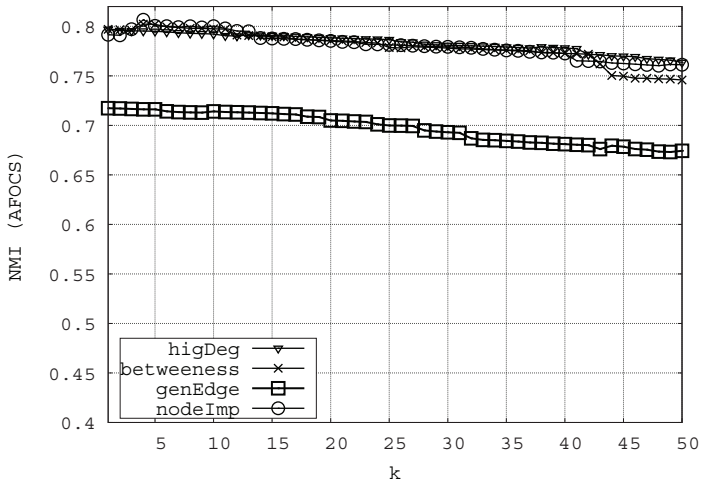


B NMI scores by Blondel

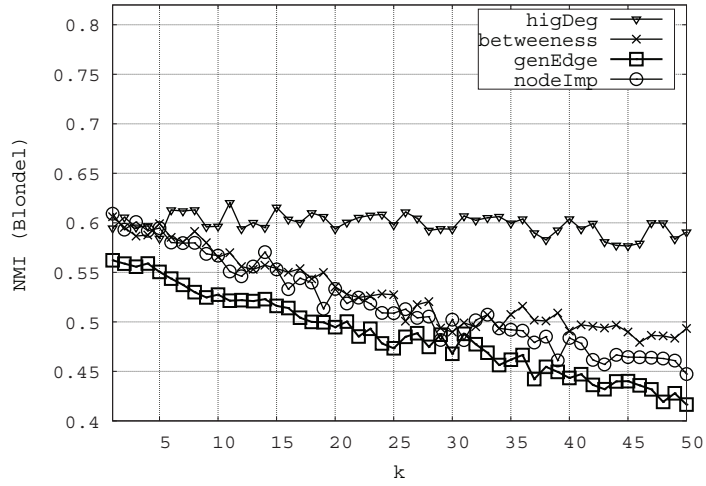


C NMI scores by Oslom

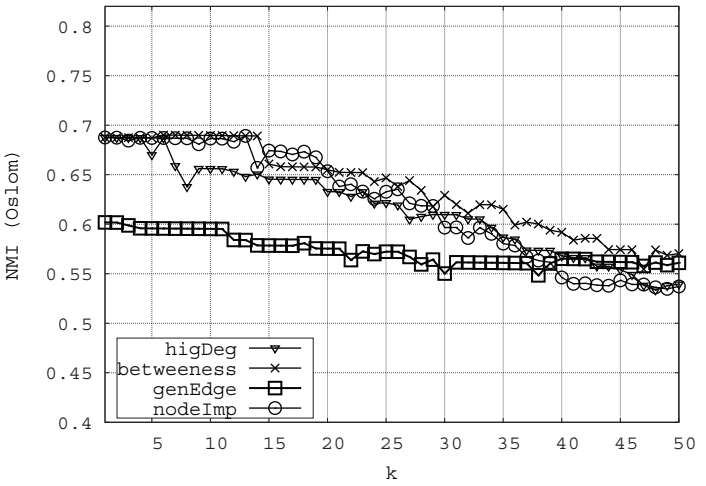
Figure 8-1. Comparison among different node selection strategies on synthesized networks with $N = 2500$ nodes



A NMI scores on AFOCS



B NMI scores on Blondel



C NMI scores on Oslom

Figure 8-2. Comparison among different node selection strategies on synthesized networks with $N = 5000$ nodes

and after the node removal, which is our main objective function, (2) the number of communities in the new structure, and (3) the average size of the network communities in the new structure.

8.5.1 Results on Synthesized Networks

Set up: We use the well-known LFR overlapping benchmark [102] to generate test networks. The number of nodes are $N = 2500$ and 5000, the mixing parameter $\mu = 0.15$, the community sizes $c_{min} = 10$ and $c_{max} = 50$ for $N = 2500$ and $c_{min} = 30$ and $c_{max} = 100$ for $N = 5000$. At every k nodes are removed from the network, the network community structure is reidentified and compared to the original embedded one (or the ground-truth). The overlapping threshold β in AFOCS is set at 0.7 and all tests are averaged on 100 runs for consistency.

8.5.1.1 Solution quality

We first evaluate the performance of all aforementioned node selections strategies on different community detection algorithms $AFCOS$, Blondel and Oslom, respectively. Because the ground-truth communities on synthesized networks are given a priori, comparisons through NMI scores among these strategies as well as among detection algorithms are therefore valid, and the lower NMI scores a strategy obtains, the more effective it seems to be. In addition, the higher the remaining NMI measure a detection algorithm obtains after the node removal, the more resistant to node vulnerability it seems to be.

The quality of node selection solutions, are reported in figures 8-1 and 8-2. In a general trend, NMI scores tend to drop down quickly as more nodes are removed from the network when $N = 2500$; however, they degrade much slower in networks with $N = 5000$. The first observation revealed in those figures is that our approach $genEdge$ achieves the best (lowest) NMI scores on almost all test cases. In average, on networks with 2500 nodes, $genEdge$ is 14% better than both $highDeg$ and $betweeness$, and is 12% better than $nodeImp$ on AFOCS algorithm; and is 19%, 11% and 5% better than $highDeg$,

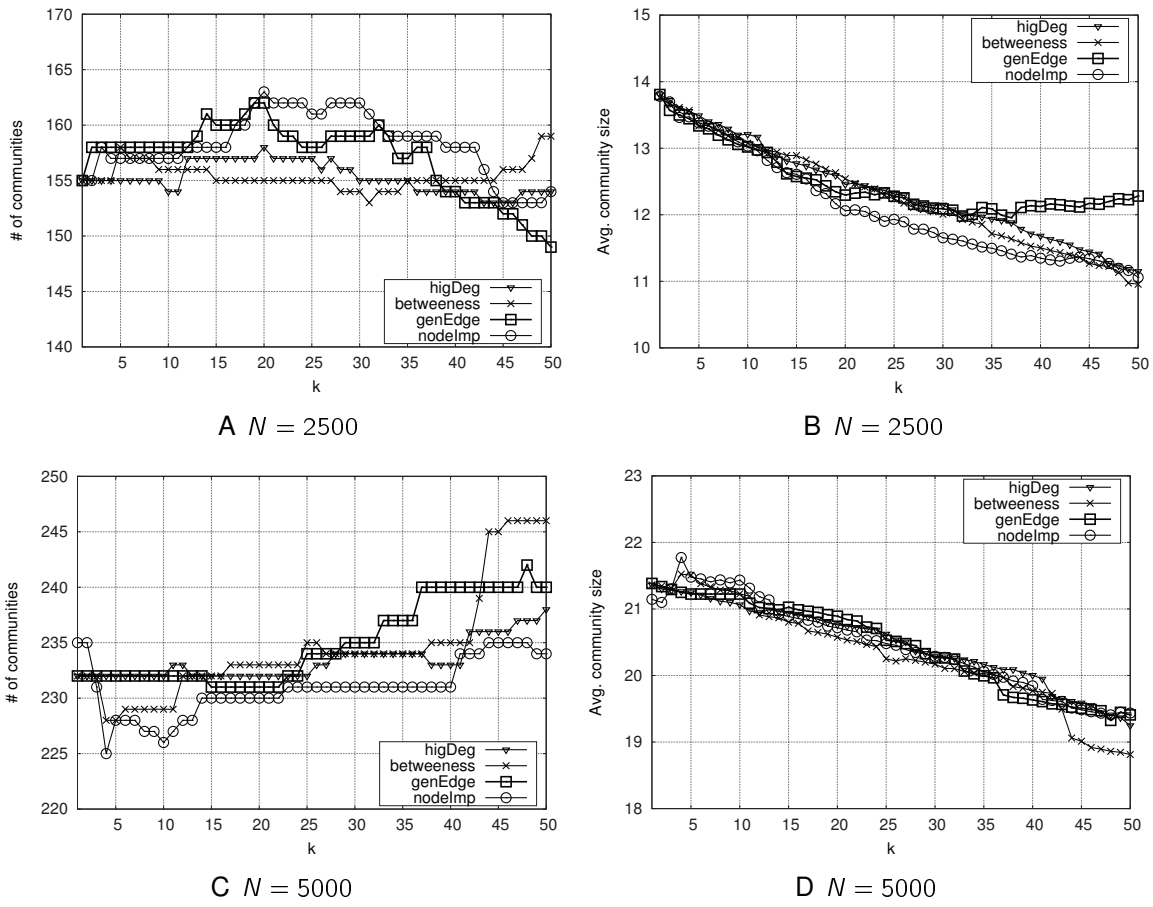


Figure 8-3. Results obtained by AFOCS on networks with $N = 2500$ nodes and $N = 2500$ nodes.

betweeness, and *nodeImp* on Blondel algorithm (figure 8-1A, 8-1B). On Oslom algorithm, *genEdge* differs insignificant with *highDeg* and *betweeness* with 1.5% and 1.4% better, and is only lagged behind *nodeImp* with 3% lower NMI scores. On network with 5000 nodes, *genEdge* still outperforms other strategies with 12% lower NMI scores than the others on AFOCS algorithm, and with 23%, 8% and 6% lower NMI scores than *highDeg*, *betweeness* and *nodeImp* on Blondel algorithm, and finally, with 7%, 10% and 8% better than the others on Oslom algorithm (figure 8-2). These results imply that *genEdge* node selection strategy performs excellently with competitive results on different community detection algorithm in comparison with other strategies.

The second observation we obtain from figures 8-1 and 8-2 is that the top-of-the-list node seems to be essential to the network community structure. The removal of only this node from the network brings the NMI scores to as low as 0.7 - 0.8 on AFOCS (figure 8-1A, 8-2A), to 0.58 - 0.6 on Blondel algorithm (figure 8-1B, 8-2B), and to 0.7 on Oslom algorithm. Furthermore, the top 15-20 nodes are also vital to the network community structure detected by Oslom and Blondel since their destruction brings the NMI scores down to 0.5, the threshold where the community structure become stochastic and fuzzy to recognize. The NMI values on AFOCS algorithm, on the other hand, do not suffer from this destruction as they only come close to 0.5 when almost $k = 50$ nodes are removed from the networks with $N = 2500$ nodes (figure 8-1A).

Finally, the last observation inferred from figures 8-1 and 8-2 is that, among the three community detection algorithms, AFOCS algorithm obtains the highest remaining NMI values when the same number of nodes is removed from the networks. In other words, AFOCS was able to detect the community structure which was of the most similarity to the ground-truth communities. As we discussed above, this observation implies that AFOCS seems to be the detection algorithm which is more resistant to node vulnerability than the other algorithms. Therefore, we employ AFOCS as the main community detection algorithm to further analyze network communities of real-world traces.

8.5.1.2 The Number of Communities and Their Sizes

We next examine the number of communities and their sizes when k important nodes are removed from the network. As discussed in subsection 8.4, our selection strategy gives priority to breaking the current community structure into more communities while looking for their sizes to be relatively the same in order to minimize NMI measure. The results are presented in figure 8-3.

As reported in these figures, the numbers of new communities generated by $genEdge$ tend to increase as more nodes are excluded; however, they differ insignificantly

Table 8-1. Statistic of social traces

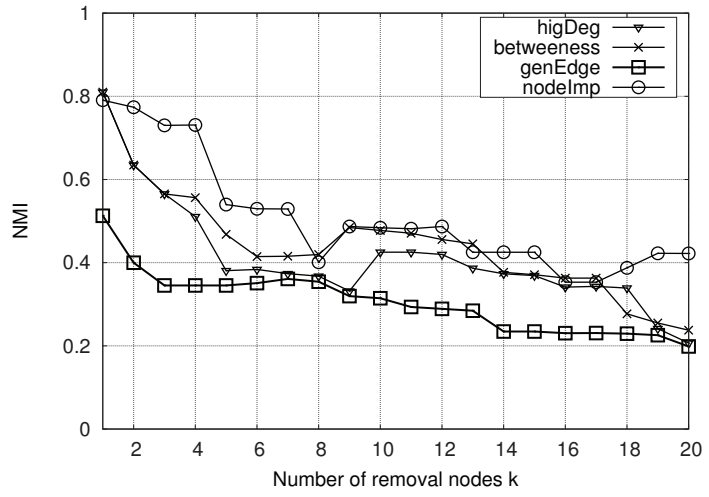
Data	N	M	Avg. Deg	Max. Com. Size
Reality	100	3100	62	35
Facebook	63731	1.5M	23.50	33425
Foursquare	47260	1.1M	49.13	30381

from other methods on small networks of 2500 nodes (figure 8-3A), but the differences become more visible on larger networks of 5000 nodes (figure 8-3C). In particular, the number of communities generated by *genEdge* is the second highest when $N = 5000$ (only below *betweeness* method) while the average sizes of communities are relatively equal to other methods (figure 8-3B and 8-3D). One might question why the NMI scores returned by *genEdge* is still high since its number of communities and average community size are relatively the same as the other. One possible reason is because new communities formed by other strategies might possibly be the subcommunities or parts of of the original structure, which in turn results in high similarity to the ground-truth. Our strategy, on the other hand, makes sure that once a node incident to the most generating edges is excluded, the subcommunity structure is broken and the new community structure has little similarity to the original one, and hence, the lower NMI measures.

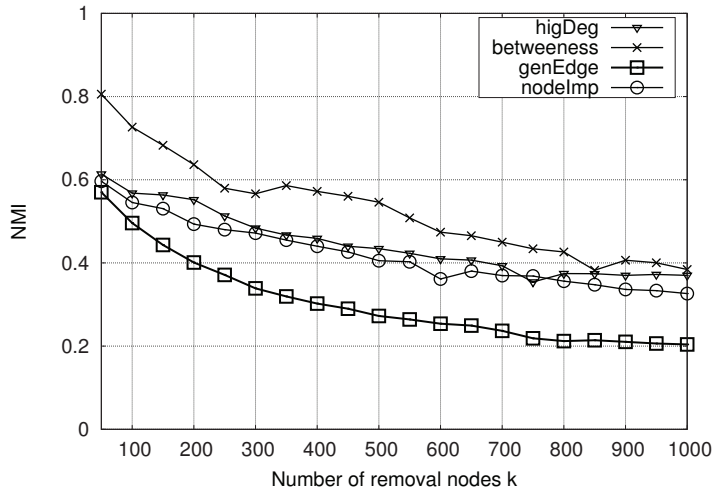
8.5.2 Results on Real World Traces

We further present the empirical results of CSV on real-world networks including Reality mobile phone data [82], Facebook [61] and Foursquare [109] datasets. The overview of these datasets is summarized in Table 8-1.

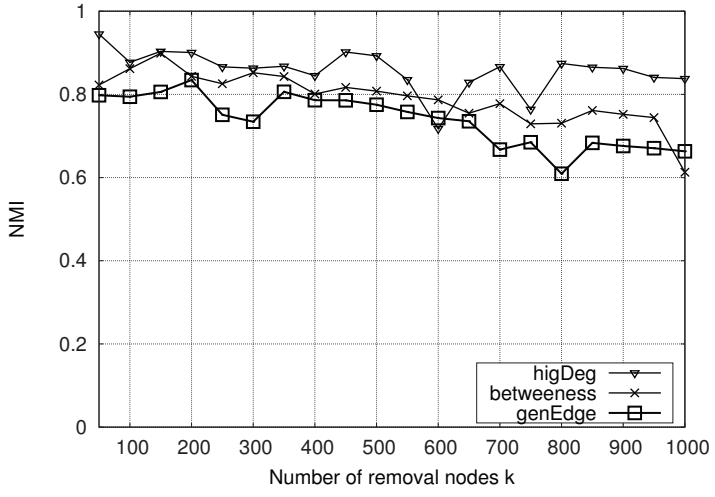
Reality Mining dataset provided by the MIT Media Lab. This dataset contains communication, proximity, location, call, and activity information from 100 students at MIT over the course of the 2004-2005 academic year. Facebook dataset contains friendship information (i.e., who is friend with whom and wall posts) among New



A Reality



B Foursquare



C Facebook

Figure 8-4. NMI scores on Reality mining data, Foursquare and Facebook networks obtained by AFOCS ($k = 50...1000$)

Orleans regional network on Facebook, spanning from Sep 2006 to Jan 2009. To collect the information, the authors created several Facebook accounts, joined each to the regional network, started crawling from a single user and visited all friends in a breath-first-search fashion. Foursquare dataset contains location and activities of 47260 users on Foursquare social network on May 2011 - Jul 2011. To collect the data, we created several Foursquare accounts, joined to the network, started crawling from a single user and visited all friends also in a breadth-first-search fashion.

On Reality Mining dataset, we set $k = 1...20$ and report result in figure 8-4A. It reveals from this figure that community structure in this dataset is extremely vulnerable to node attacks since the removal of only 2 nodes, found by $genEdge$ is enough to make the new community structure significantly differs from the original one as it brings down the NMI values to 0.4. In comparison with other node selection methods, $genEdge$ still perform excellently and is about 14% - 17% better than the others. We note that the first node identified by $genEdge$ is indeed crucial to the community structure of this network since it immediately brings down NMI score to 0.6 while the other does not seem to discover this important feature. Furthermore, when too many nodes are removed from the network, the network does seem to contain communities any more or the community structure become extremely fuzzy as NMI values converge down to around 0.2. This is understandable since this dataset is of small size with a very high average node degree.

On larger networks Facebook and Foursquare, we set k from 50 nodes to 1000 nodes (only 2.1% and 1.5% number of nodes of Foursquare and Facebook networks) with a 50-node increment at a time. The numerical results are reported in figure 8-4. In general, NMI values of all methods degrade quickly on Foursquare networks, and tend to decrease slower on Facebook networks. As more nodes are excluded from the network, $genEdge$ still achieves the best performance on both networks with significantly lower NMI values than the other methods. Specifically, on Foursquare with high average degree and internal community density, the removal of nodes incident to the most

generating edges in $genEdge$ significantly leads to the separation of network community structure as NMI scores drop down to 0.2 in $genEdge$. On Facebook network, the similarity between the original and new community structure seem to retain fairly high even all 1000 nodes are removed, whereas the new structure of ArXiv network is at the edge of stochastic threshold since the NMI measure is around 0.5. This implies that community structure in Foursquare network is also extremely vulnerable to node removal attacks, while the mature Facebook network does not seem to suffer this threat. One possible reason for this is since Facebook contains a giant community with low average degree, it therefore requires much more effort in order to break that giant community apart.

In summary, the experiments on both synthesized and real-work social network confirm the effectiveness of our proposed method based on generating edges. The empirical results also confirm that, $genEdge$ outperforms other heuristic methods on other community detection methods such as AFOCS, Blondel and Oslom algorithms.

8.6 An Application in DTNs

We present a practical application where the detection of overlapping network communities plays a vital role in forwarding strategies in communication networks. In order to evaluate the impact of community restructuring in complex networks, we compare the set of critical nodes identified by our community structure vulnerability algorithm to the set of nodes selected using aforementioned algorithms. Furthermore, in order to evaluate which one of the critical node set is the most critical, we study how the removal of the critical node set influences the performance of routing in Pocket Switched Networks (PSN), in terms of average message delivery ratio, and delivery-time.

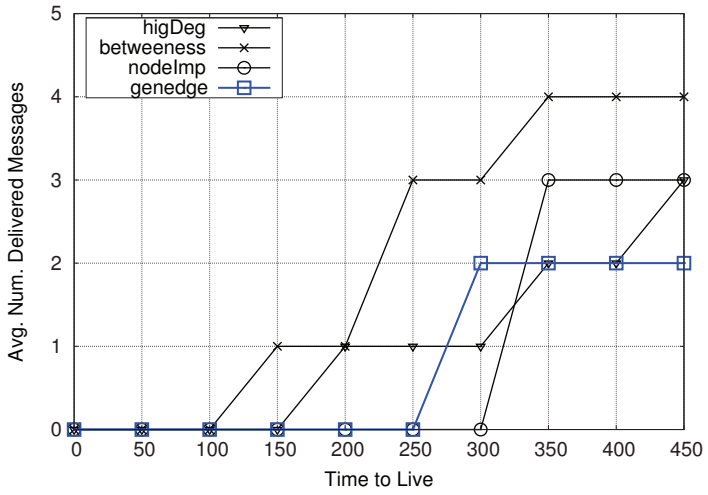
PSNs are a particular case of DTNs, where the nodes of the network correspond to actual people that are equipped with portable devices (i.e., mobile phones), and that use these portable devices to communicate. Because of the high degree of mobility of this type of networks, a path between a source and a destination seldom

exists, therefore most of the approaches to routing in this kind of environments adopt a store-carry-and-forward approach. In store-carry-and-forward approaches, messages are stored locally and, depending on the approach, they are forwarded or replicated to the encountered nodes when an opportunity occurs. In this manner, a node is important if it serves as a hub to forward the messages to other devices. As a result, the failures of these important nodes shall degrade the message delivery ratio while shall incur more duplicate messages and delivery time.

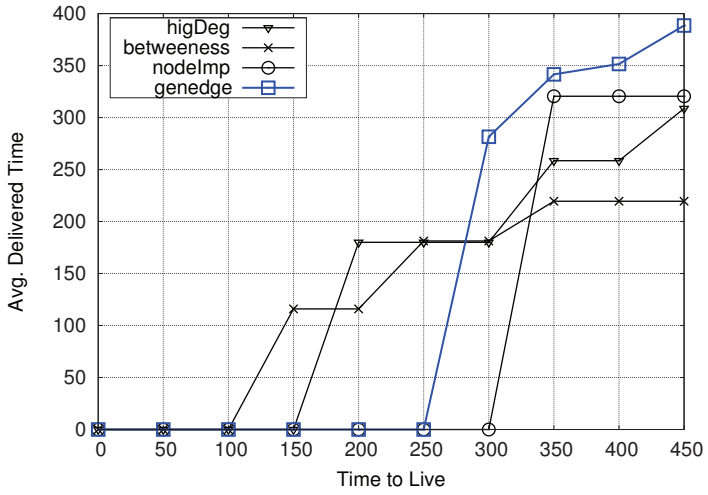
We use the HAGGLE dataset [110]. This trace was collected at the Infocom conference in 2006 in Barcelona. 70 students and researchers attending the workshop were equipped with iMote devices that registered they encounter for the duration of the conference (3 days). In addition to the 70 mobile partecipants, approximately 20 static, long range iMotes were deployed throughout the area of the conference. A total of 1000 messages are created and uniformly distributed during the experiment duration and each message can not exist longer than a threshold time-to-live. In our evaluation we will focus on the PSN routing algorithm inspired by BubbleRap [106]. While we expect the performance of this protocol to deteriorate upon the removal of important nodes, we expect the performances of BubbleRap to deteriorate more quickly, because of the reliance of the protocol on the community structure. Because BubbleRap relies on the knowledge of the community structure to route the messages, and because we realize that different algorithms that attempt to find the community structure use different objective functions that may be more susceptible to the removal of nodes, we consider evaluate the average delivery ratio, average delivery time and the average number of copied messages.

Results

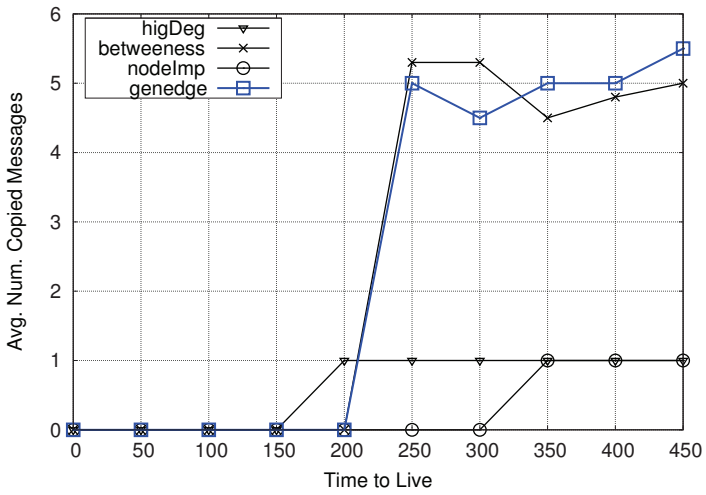
As the removal of 10 node in Haggle dataset is enough to make it original community structure to become stochastic (figure 8-6), we fix $k = 10$ and report the results as a function of $time - to - live$ (the amount of time a message can exist). The



A Avg. Delivered Messages



B Avg. Delivery Time



C Avg. Number of copied Messages

Figure 8-5. Simulation results on HAGGLE dataset.

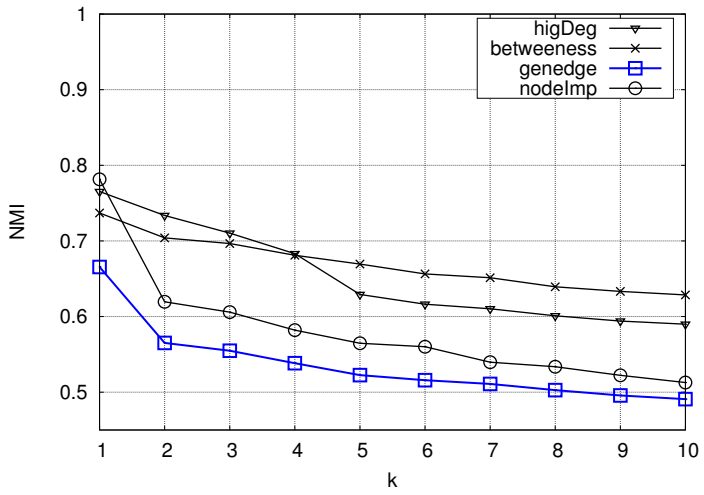


Figure 8-6. NMI measure on Haggle dataset.

performances of all methods are presented in figure 8-5. As reported in subfigures 8-5A, 8-5B and 8-5C, the removal of nodes selected by genEdge approach significantly degrades the performance of BubbleRap forwarding and routing system in terms of not only delivered messages and time but also the numbers of copied messages. As depicted in subfigure 8-5A, the averaged number of messages delivered by BubbleRap under genEdge and $time - to - live = 450s$ is only two, whereas those under highDeg, betweeness and $nodeImp$ are four, three and three, which implies 100% and 50% system downgrade when only 10 nodes are excluded from the networks. This also means nodes selected by genEdge are of important role in maintaining the normal operation of the whole network. Furthermore, when nodes are removed from the network, one expects that the delivery time should be increased as a consequnce because participants now have less chances to communicate with each other, and thus, it should take longer for participating devices to forward the carried messages. This intuition is nicely reflected in figure 8-5B. As reported in this subfigure, the average amount of time required to deliver carried messages increases significantly as $time - to - live$ increases (note that from 0-100s, there was no message delivered, and thus, the delivery time was 0). In terms of delivery time, the removal of nodes under

genEdge affects the system to requires a huge extra amount to deliver the messages in comparison with other methods. In particular, the system delivery time under genEdge is about 1.25x, 1.7x and 1.21x higher than that under betweeness, $nodeImp$ and highDeg when $time - to - live = 450$. Moreover, the number of copied messages, affected by genEdge approach, is also the highest one among other methods. This means that genEdge heuristic algorithm, indeed, selects appropriate nodes whose effects significantly reduce the system performance as reported by the three evaluated factors.

CHAPTER 9
CONCLUSIONS

In this dissertation, we establish the fundamental knowledge on the following aspects of the complex network science (1) the network organizational principals via the discovery of its dynamic community structure (2) the assessment of the community structure vulnerability, and (3) the social-based solutions for practical applications enabled by complex systems, such as in online social networks and mobile networks. We suggested two adaptive frameworks for discovering the dynamic network community structure and analyze theoretical results that guarantee their performances. In the execution perspective, our methods are adaptive, and thus, are scalable for very large networks with very competitive experimental results.

To investigate the assessment of the network community structure vulnerability, we introduce the new problem of identifying key nodes whose removal can maximally reform the current network communities. Those nodes are important in maintaining the normal functioning of the whole system, such as in the case of DTNs (in a mobile network) or lung cancer (in a biological network). Our work presents first and preliminary yet important insights, in terms of both theoretical results and heuristic algorithms, into the vulnerability assessment of the network community structure.

In an application perspective, our work in this dissertation focuses on proposing novel community structure-based solutions for the following emerging problems: the forwarding and routing strategy in mobile networks, the worm containment problem in social networks and the limiting misinformation spread in online social networks. Our suggested strategies provide a significant improvement in terms of the solution quality for those mentioned problems, and promise a wider range of applications enabled by dynamic complex networks.

REFERENCES

[1] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, June 2002.

[2] G. Palla, P. Pollner, A. Barabasi, and T. Vicsek. Social group dynamics in networks. *Adaptive Networks*, 2009.

[3] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: A comparative analysis. *Phys. Rev. E*, 80:056117, Nov 2009.

[4] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, Jun 2004.

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

[6] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review*, E 69(026113), 2004.

[7] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Trans. on Knowl. and Data Eng.*, 20(2):172–188, February 2008.

[8] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111+, December 2004.

[9] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75 – 174, 2010.

[10] Fei Wang, Tao Li, Xin Wang, Shenghuo Zhu, and Chris Ding. Community discovery using nonnegative matrix factorization. *Data Min. Knowl. Discov.*, 22(3):493–521, May 2011.

[11] Tao Li and Chris Ding. The relationships among various nonnegative matrix factorization methods for clustering. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 362–371, Washington, DC, USA, 2006. IEEE Computer Society.

[12] Chris Ding, Tao Li, and Wei Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Comput. Stat. Data Anal.*, 52(8):3913–3927, April 2008.

[13] Yu-Ru Lin, Jimeng Sun, Paul Castro, Ravi Konuru, Hari Sundaram, and Aisling Kelliher. Metafac: community discovery via relational hypergraph factorization. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge*

discovery and data mining, KDD '09, pages 527–536, New York, NY, USA, 2009. ACM.

[14] I. Psorakis, S. Roberts, and M. Ebden. Overlapping community detection using bayesian non-negative matrix factorization. *Phys. Rev. E. 83*, 11.

[15] Fei Wang, Tao Li, Xin Wang, Shenghuo Zhu, and Chris Ding. Community discovery using nonnegative matrix factorization. *Data Min. Knowl. Discov.*, 22(3):493–521, May 2011.

[16] Zhichao Zhu, Guohong Cao, Sencun Zhu, S. Ranjan, and A. Nucci. A social network based patching scheme for worm containment in cellular networks. In *INFOCOM 2009, IEEE*, pages 1476 –1484, april 2009.

[17] N.P. Nguyen, T.N. Dinh, Ying Xuan, and M.T. Thai. Adaptive algorithms for detecting community structure in dynamic social networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 2282 –2290, april 2011.

[18] N.P. Nguyen, Ying Xuan, and M.T. Thai. A novel method for worm containment on dynamic social networks. In *MILITARY COMMUNICATIONS CONFERENCE, 2010 - MILCOM 2010*, pages 2180 –2185, 31 2010-nov. 3 2010.

[19] Gergely Palla, Albert-Laszlo Barabasi, and Tamas Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, April 2007.

[20] Mohsen Jamali, Gholamreza Haffari, and Martin Ester. Modeling the temporal dynamics of social rating networks using bidirectional effects of social relations and rating patterns. In *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, ICDMW '10, pages 344–351, Washington, DC, USA, 2010. IEEE Computer Society.

[21] Thang N. Dinh, Ying Xuan, My T. Thai, Panos M. Pardalos, and Taieb Znati. On new approaches of assessing network vulnerability: hardness and approximation. *IEEE/ACM Trans. Netw.*, 20(2):609–619, April 2012.

[22] Karsten Peters, Lubos Buzna, and Dirk Helbing. Modelling of cascading effects and efficient response to disaster spreading in complex networks. *IJCIS*, 4(1/2):46–62, 2008.

[23] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99, 2002.

[24] M E J Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

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

[26] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.

[27] Joerg Reichardt and Stefan Bornholdt. Detecting fuzzy community structures in complex networks with a potts model. *Physical Review Letters*, 93(21):218701, 2004.

[28] M E J Newman and E A Leicht. Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 104(23):9564–9569, 2007.

[29] Weinan E, Tiejun Li, and Eric Vanden-Eijnden. Optimal partition and effective dynamics of complex networks. *Proceedings of the National Academy of Sciences of the United States of America*, 105(23):7907–7912, 2008.

[30] Yuzhou Zhang, Jianyong Wang, Yi Wang, and Lizhu Zhou. Parallel community detection on large networks with propinquity dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 997–1006, New York, NY, USA, 2009. ACM.

[31] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 687–696, New York, NY, USA, 2007. ACM.

[32] Pan Hui, Eiko Yoneki, Shu Yan Chan, and Jon Crowcroft. Distributed community detection in delay tolerant networks. In *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, MobiArch '07, pages 7:1–7:8, New York, NY, USA, 2007. ACM.

[33] John Hopcroft, Omar Khan, Brian Kulis, and Bart Selman. Tracking evolving communities in large linked networks. *Proceedings of the National Academy of Sciences*, 101:5249–5253, April 2004.

[34] Chayant Tantipathananandh and Tanya Berger-Wolf. Constant-factor approximation algorithms for identifying dynamic communities. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 827–836, New York, NY, USA, 2009. ACM.

[35] T.N. Dinh, Ying Xuan, and M.T. Thai. Towards social-aware routing in dynamic communication networks. In *Performance Computing and Communications Conference (IPCCC), 2009 IEEE 28th International*, pages 161 –168, dec. 2009.

[36] Yu-Ru Lin, Yun Chi, Shenghuo Zhu, Hari Sundaram, and Belle L. Tseng. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 685–694, New York, NY, USA, 2008. ACM.

[37] Dongsheng Duan, Yuhua Li, Yanan Jin, and Zhengding Lu. Community mining on dynamic weighted directed graphs. In *Proceedings of the 1st ACM international workshop on Complex networks meet information & knowledge management*, CNIKM '09, pages 11–18, New York, NY, USA, 2009. ACM.

[38] Min-Soo Kim and Jiawei Han. A particle-and-density based evolutionary clustering method for dynamic networks. *Proc. VLDB Endow.*, 2(1):622–633, August 2009.

[39] R. Cazabet, F. Amblard, and C. Hanachi. Detection of overlapping communities in dynamical social networks. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 309 –314, aug. 2010.

[40] Andrea Lancichinetti, Filippo Radicchi, Jos J. Ramasco, and Santo Fortunato. Finding statistically significant communities in networks. *PLoS ONE*, 6(4):e18961, 04 2011.

[41] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, 2006.

[42] G. Palla, I. Derenyi, I. Farkas1, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(10), 2005.

[43] C. Lee, F. Reid, A. McDaid, and N. Hurley. Detecting highly overlapping community structure by greedy clique expansion. *Proceedings of the 4th Workshop on Social Network Mining and Analysis*, Feb 2010.

[44] A. Lazar, D. Abel, and T. Vicsek. Modularity measure of networks with overlapping communities. *EPL (Europhysics Letters)*, 90(1):18001, 2010.

[45] V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri. Extending the definition of modularity to directed graphs with overlapping communities. *J. Stat. Mech.: Theory and Experiment*, 2009(03):P03024, 2009.

[46] A. Lancichinetti, S. Fortunato, and K. Jnos. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.

[47] Steve Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10):103018, 2010.

[48] Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann. Link communities reveal multi-scale complexity in networks. *Nature*, 466:761+, October 2010.

[49] J-C C. Delvenne, S. N. Yaliraki, and M. Barahona. Stability of graph communities across time scales. *Proceedings of the National Academy of Sciences of the United States of America*, 107(29):12755–12760, July 2010.

[50] Andrea Lancichinetti and Santo Fortunato. Consensus clustering in complex networks. *Scientific Reports*, 2, March 2012.

[51] Yanhua Li, Zhi-Li Zhang, and Jie Bao. Mutual or unrequited love: Identifying stable clusters in social networks with uni- and bi-directional links. In Anthony Bonato and Jeannette Janssen, editors, *Algorithms and Models for the Web Graph*, volume 7323 of *Lecture Notes in Computer Science*, pages 113–125. Springer Berlin Heidelberg, 2012.

[52] T. H. Grubesic, T. C. Matisziw, A. T. Murray, and D. Snediker. Comparative approaches for assessing network vulnerability. *Inter. Regional Sci. Review*, 31, 2008.

[53] Jerry Scripps, Pang-Ning Tan, and Abdol-Hossein Esfahanian. Node roles and community structure in networks. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, WebKDD/SNA-KDD '07, pages 26–35, New York, NY, USA, 2007. ACM.

[54] Yang Wang, Zengru Di, and Ying Fan. Identifying and characterizing nodes important to community structure using the spectrum of the graph. *PLoS ONE*, 6(11):e27418, 11 2011.

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

[56] Istvan A. Kovacs, Robin Palotai, Mate S. Szalay, and Peter Csermely. Community landscapes: An integrative approach to determine overlapping network module hierarchy, identify key nodes and predict network dynamics. *PLoS ONE*, 5(9):e12528, 09 2010.

[57] Timothy C. Matisziw and Alan T. Murray. Modeling s-t path availability to support disaster vulnerability assessment of network infrastructure. *Comput. Oper. Res.*, 36(1):16–26, January 2009.

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

[59] Zhenqing Ye, Songnian Hu, and Jun Yu. Adaptive clustering algorithm for community detection in complex networks. *Phys. Rev. E*, 78:046115, Oct 2008.

[60] ArXiv dataset. http://www.cs.cornell.edu/projects/kddcup/datasets.html. *KDD Cup 2003*, Feb 2003.

[61] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, WOSN '09, pages 37–42, New York, NY, USA, 2009. ACM.

[62] S. Fortunato and C. Castellano. Community structure in graphs. *eprint arXiv: 0712.2716*, 2007.

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

[64] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486:75–174, 2010.

[65] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 695–704, New York, NY, USA, 2008. ACM.

[66] M. Goldberg, S. Kelley, M. Magdon-Ismail, K. Mertsalov, and A. Wallace. Finding overlapping communities in social networks. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 104 –113, aug. 2010.

[67] Yu-Ru Lin, Yun Chi, Shenghuo Zhu, Hari Sundaram, and Belle L. Tseng. Analyzing communities and their evolutions in dynamic social networks. *ACM Trans. Knowl. Discov. Data*, 3(2):8:1–8:31, April 2009.

[68] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *In NIPS*, pages 556–562. MIT Press, 2000.

[69] A. Cichocki and R. Zdunek. Multilayer nonnegative matrix factorization using projected gradient approaches. *Proc. 13th International Conference on Neural Information Processing*, '07.

[70] Rafal Zdunek and Andrzej Cichocki. Non-negative matrix factorization with quasi-newton optimization. In *Proceedings of the 8th international conference on Artificial Intelligence and Soft Computing*, ICAISC'06, pages 870–879, Berlin, Heidelberg, 2006. Springer-Verlag.

[71] A. Cichocki, R. Zdunek, and S.-i. Amari. Nonnegative matrix and tensor factorization [lecture notes]. *Signal Processing Magazine, IEEE*, 25(1):142 –145, 2008.

[72] Michael W. Berry, Murray Browne, Amy N. Langville, V. Paul Pauca, and Robert J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. In *Computational Statistics and Data Analysis*, pages 155–173, 2006.

[73] A. Cichocki, H. Lee, Y-D Kim, and S. Choi. Non-negative matrix factorization with α-divergence. *Pattern Recognition Letters*, '08.

[74] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley Publishing, 2009.

[75] P. Panzarasa, T. Opsahl, and K. M. Carley. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *J. American Soc. of Info. Sci. Tech. 60(5)*, '09.

[76] Nam P. Nguyen, Thang N. Dinh, Sindhura Tokala, and My T. Thai. Overlapping communities in dynamic networks: their detection and mobile applications. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, MobiCom '11, pages 85–96, New York, NY, USA, 2011. ACM.

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

[78] Pan Hui and Jon Crowcroft. How small labels create big improvements. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 65 –70, march 2007.

[79] Elizabeth M. Daly and Mads Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '07, pages 32–40, New York, NY, USA, 2007. ACM.

[80] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, and James Scott. Impact of human mobility on opportunistic forwarding algorithms. *Mobile Computing, IEEE Transactions on*, 6(6):606 –620, june 2007.

[81] Pan Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. *Mobile Computing, IEEE Transactions on*, 10(11):1576 –1589, nov. 2011.

[82] Nathan Eagle and Alex (Sandy) Pentland. Reality mining: sensing complex social systems. *Personal Ubiquitous Comput.*, 10(4):255–268, March 2006.

[83] Foobface. facebook_virus_turns_your_computer_into_a_zombie.html, http://www.pcworld.com/article/155017/. In *PC World*, page 1, 2008.

[84] Koobface. http://news.cnet.com/koobface-virus-hits-facebook/. In *CNET*, page 1, 2008.

[85] Vyas Sekar, Yinglian Xie, Michael K. Reiter, and Hui Zhang. A multi-resolution approach forworm detection and containment. In *Proceedings of the International Conference on Dependable Systems and Networks*, DSN '06, pages 189–198, Washington, DC, USA, 2006. IEEE Computer Society.

[86] Nicholas Weaver, Stuart Staniford, and Vern Paxson. Very fast containment of scanning worms. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 3–3, Berkeley, CA, USA, 2004. USENIX Association.

[87] Hyang-Ah Kim and Brad Karp. Autograph: toward automated, distributed worm signature detection. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 19–19, Berkeley, CA, USA, 2004. USENIX Association.

[88] Pu Wang, Marta C. González, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. Understanding the Spreading Patterns of Mobile Phone Viruses. *Science*, 324(5930):1071–1076, May 2009.

[89] Abhijit Bose and Kang G. Shin. Proactive security for mobile messaging networks. In *Proceedings of the 5th ACM workshop on Wireless security*, WiSe '06, pages 95–104, New York, NY, USA, 2006. ACM.

[90] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 641–650, New York, NY, USA, 2010. ACM.

[91] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, CIKM '03, pages 556–559, New York, NY, USA, 2003. ACM.

[92] Lars Backstrom and Jure Leskovec. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 635–644, New York, NY, USA, 2011. ACM.

[93] Michael Fire, Lena Tenenboim, Ofrit Lesser, Rami Puzis, Lior Rokach, and Yuval Elovici. Link prediction in social networks using computationally efficient topological features. In *SocialCom/PASSAT*, pages 73–80. IEEE, 2011.

[94] John G. Kemeny, Mirkil Hazleton, Snell J. Laurie, and Thompson Gerald L. Finite mathematical structures. *1st edition. Englewood Cliffs, N.J.: Prentice-Hall, Inc*, 1959.

[95] Carlo Piccardi. Finding and testing network communities by lumped markov chains. *PLoS ONE*, 6(11):e27028, 11 2011.

[96] Meyn Sean P. and Tweedie Richard L. Markov chains and stochastic stability. *2nd edition. Cambridge University Press*, 2009.

[97] K H Hoffmann and P Salamon. Bounding the lumping error in markov chain dynamics. *Applied Mathematics Letters*, 22(9):1471–1475, 2009.

[98] Jiří Šíma and Satu Elisa Schaeffer. On the np-completeness of some graph cluster measures. In *Proceedings of the 32nd conference on Current Trends in Theory and Practice of Computer Science*, SOFSEM'06, pages 530–537, Berlin, Heidelberg, 2006. Springer-Verlag.

[99] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[100] Martin Rosvall and Carl T. Bergstrom. Mapping change in large networks. *PLoS ONE*, 5(1):e8694, 01 2010.

[101] Andrea Lancichinetti, Filippo Radicchi, Jos J. Ramasco, and Santo Fortunato. Finding statistically significant communities in networks. *PLoS ONE*, 6(4):e18961, 04 2011.

[102] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Phys. Rev. E*, 80(5):056117, Nov 2009.

[103] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 1029–1038, New York, NY, USA, 2010. ACM.

[104] Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 88–97, Washington, DC, USA, 2010. IEEE Computer Society.

[105] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, 2005.

[106] Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '08, pages 241–250, New York, NY, USA, 2008. ACM.

[107] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[108] K. Andreev and H. Räcke. Balanced graph partitioning. In *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, SPAA '04, pages 120–124, New York, NY, USA, 2004. ACM.

[109] Foursquare Data. sites.google.com/site/namnpuf/original_foursquare.7z. In *Collected data*, pages 0–0, 2012.

[110] James Scott, Richard Gass, Jon Crowcroft, Pan Hui, Christophe Diot, and Augustin Chaintreau. CRAWDAD trace cambridge/haggle/imote/infocom2006 (v. 2009-05-29). Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle /imote/infocom2006, May 2009.

[111] Y. Xuan, Y. Shen, N. P. Nguyen, and M. T. Thai. A graph-theoretic qos-aware vulnerability assessment for network topologies. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5, Dec 2010.

[112] Yilin Shen, Nam P. Nguyen, and My T. Thai. Exploiting the robustness on power-law networks. In *Proceedings of the 17th Annual International Conference on Computing and Combinatorics*, COCOON'11, pages 379–390, Berlin, Heidelberg, 2011. Springer-Verlag.

[113] D. T. Nguyen, N. P. Nguyen, M. T. Thai, and A. Helal. An optimal algorithm for coverage hole healing in hybrid sensor networks. In *2011 7th International Wireless Communications and Mobile Computing Conference*, pages 494–499, July 2011.

[114] N. P. Nguyen, T. N. Dinh, D. T. Nguyen, and M. T. Thai. Overlapping community structures and their detection on social networks. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 35–40, Oct 2011.

[115] Nam P. Nguyen, Guanhua Yan, My T. Thai, and Stephan Eidenbenz. Containment of misinformation spread in online social networks. In *Proceedings of the 4th Annual ACM Web Science Conference*, WebSci '12, pages 213–222, New York, NY, USA, 2012. ACM.

[116] D. T. Nguyen, N. P. Nguyen, and M. T. Thai. Sources of misinformation in online social networks: Who to suspect? In *MILCOM 2012 - 2012 IEEE Military Communications Conference*, pages 1–6, Oct 2012.

[117] N. P. Nguyen and M. T. Thai. Finding overlapped communities in online social networks with nonnegative matrix factorization. In *MILCOM 2012 - 2012 IEEE Military Communications Conference*, pages 1–6, Oct 2012.

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

[119] N. P. Nguyen, M. A. Alim, Y. Shen, and M. T. Thai. Assessing network vulnerability in a community structure point of view. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pages 231–235, Aug 2013.

[120] Md Abdul Alim, Nam P. Nguyen, Thang N. Dinh, and My T. Thai. Structural vulnerability analysis of overlapping communities in complex networks. In *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 01*, WI-IAT '14, pages 5–12, Washington, DC, USA, 2014. IEEE Computer Society.

[121] Dung T. Nguyen, Nam P. Nguyen, My T. Thai, and Abdelsalam Helal. Optimal and distributed algorithms for coverage hole healing in hybrid sensor networks. *Int. J. Sen. Netw.*, 11(4):228–240, June 2012.

[122] Y. Xuan, Y. Shen, N. P. Nguyen, and M. T. Thai. A trigger identification service for defending reactive jammers in wsn. *IEEE Transactions on Mobile Computing*, 11(5):793–806, May 2012.

[123] Y. Shen, N. P. Nguyen, Y. Xuan, and M. T. Thai. On the discovery of critical links and nodes for assessing network vulnerability. *IEEE/ACM Transactions on Networking*, 21(3):963–973, June 2013.

[124] Y. Xuan, Y. Shen, N. P. Nguyen, and M. T. Thai. Efficient multi-link failure localization schemes in all-optical networks. *IEEE Transactions on Communications*, 61(3):1144–1151, March 2013.

[125] Nam P. Nguyen, Guanhua Yan, and My T. Thai. Analysis of misinformation containment in online social networks. *Comput. Netw.*, 57(10):2133–2146, July 2013.

[126] Thang N. Dinh, Nam P. Nguyen, Md Abdul Alim, and My T. Thai. A near-optimal adaptive algorithm for maximizing modularity in dynamic scale-free networks. *J. Comb. Optim.*, 30(3):747–767, October 2015.

[127] Nam P. Nguyen, Thang N. Dinh, Yilin Shen, and My T. Thai. Dynamic social community detection and its applications. *PLOS ONE*, 9(4):1–18, 04 2014.

[128] Nam P. Nguyen, Md Abdul Alim, Thang N. Dinh, and My T. Thai. A method to detect communities with stability in social networks. *Social Network Analysis and Mining*, 4(1):224, 2014.

BIOGRAPHICAL SKETCH

Nam P. Nguyen is currently at his fourth year PhD student in Department of Computer and Information Science and Engineering (CISE), University of Florida and a member of Optima Network Science Lab under the guidance of Professor My T. Thai. Prior to his Ph.D. study, Nam received my B.S. and M.S. degrees both in applied mathematics from Vietnam National University (2007) and Ohio University (2009).

His research interests include dynamic complex network problems, such as non-overlapping and overlapping network community structure, worm and virus containment, social networks; cascading failures; combinatorial optimization and approximation algorithms. In particular, his current research focuses on designing adaptive algorithms for effectively identify communities in dynamic networks such as mobile or online social networks, as well as their applications in different aspects of networking problems. In addition, he is also interested in effective methods to stop the propagation of virus, worm and misinformation spread on large scale dynamic networks, in terms of both approximation and social-based algorithms.

During his Ph.D. study, Nam has published many papers in top-tier conferences and journals including INFOCOM, MOBICOM, WEBSCI and IEEE Transaction on Mobile Computing, IEEE Transaction on Networking, etc. In 2011, Nam spent his summer as an intern in CCS-3 division, Los Alamos National Laboratories, where he conducted research and published a paper on containing the spread of misinformation in large scale online social networks. Nam also the recipient of many awards such as the Student Travel Grants of MILCOM'10, MOBICOM'11 and SIGWEB'12 conferences, Travel Grant of The College of the Engineering in 2011, University of Florida.

The full list of publications that has been accomplished during his PhD is [[17, 18, 76, 111–128]]