Give a man a fish and you feed him for a day.
Teach a man to fish and you feed him for a lifetime.

– Chinese Proverb.

University of Alberta

COMMUNITY MINING
DISCOVERING COMMUNITIES IN SOCIAL NETWORKS

by

Jiyang Chen

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

Spring 2010
Edmonton, Alberta

Examining Committee

Dr. Randy Goebel, Department of Computing Science

Dr. Osmar R. Zaïane, Department of Computing Science

Dr. Martin Ester, Simon Fraser University

Dr. Ali Shiri, School of Library and Information Studies

Dr. Guohui Lin, Department of Computing Science

Dr. Eleni Stroulia, Department of Computing Science

Abstract

Much structured data of scientific interest can be represented as networks, where sets of nodes or vertices are joined together in pairs by links or edges. Although these networks may belong to different research areas, there is one property that many of them do have in common: the network community structure, which means that there exists densely connected groups of vertices, with only sparser connections between groups. The main goal of community mining is to discover these communities in social networks or other similar information network environments.

We face many deficiencies in current community structure discovery methods. First, one similarity metric is typically applied in all networks, without considering the differences in network and application characteristics. Second, many existing methods assume the network information is fully available, and one node only belongs to one cluster. However, in reality, a social network can be huge thus it is hard to access the complete network. It is also common for social entities to belong to multiple communities. Finally, relations between entities are hard to understand in heterogeneous social networks, where multiple types of relations and entities exist. Therefore, the thesis of this research is to tackle these community mining problems, in order to discover and evaluate community structures in social networks from various aspects.

Acknowledgements

I would like to thank many people for making this dissertation possible. First, I would like to thank my supervisors, Dr. Randy Goebel and Dr. Osmar Zaïane for their enthusiasm, patience, advice and guidance during all these years. I would like to thank my committee members, Dr. Guohui Lin, Dr. Eleni Stroulia, Dr. Ali Shiri, and Dr. Martin Ester, for carefully reading my thesis and providing many valuable comments. Thanks to my family for their love and support in all these years. I would also like to thank my colleagues, Tong Zheng, William Thorne, Daniel Huntley, Mojdeh Jalali-Heravi, Seyed-Vahid Jazayeri and Deng Kang, for their help in this research.

Our research has been supported by the Canadian Natural Sciences and Engineering Research Council (NSERC), by the Alberta Ingenuity Centre for Machine Learning (AICML), and by the Alberta Informatics Circle of Research Excellence (iCORE).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many structured data of scientific interest can be represented as networks, where sets of nodes or vertices are joined together in pairs by links or edges. Examples include social networks [210] such as researcher collaboration [148, 149], friendship network [13], the World Wide Web [6] (e.g., the web page hyperlink network [82, 117, 120]) (WWW), and biological networks (e.g., neural networks [211] and food webs [219]). A common property of these networks is their *community structure* [150], which notes the existence of densely connected groups of vertices, with only sparser connections between groups. From that perspective, identifying communities can be seen as finding node clusters in a graph. Indeed, this research task is highly related with clustering, although by somewhat different means (See Chapter 3.2). As a new research field, *Community Mining* [85], which focuses on the detection and characterization of such network structure, has received considerable attention over the past few years. A community can be defined as *a group of entities that share similar properties or connect to each other via selected relations* [226]. Identifying these connections and locating entities in different communities is the main goal of the community mining research.

The ability to detect communities could be of significant practical importance. For example, groups of web pages that link to more web pages in the community than to pages outside might correspond to set of web pages on related topics, which can enable search engines and portals to increase the precision and recall of search results by focusing on narrow but topically related subsets of the web [72]; groups

within social networks might correspond to social communities, which can be used to understand organizational structures [203], academic collaboration [143, 188], criminal group detection [68] and even political election [2]. Merely finding that a biochemical network contains tightly-connected groups at all can convey useful information by providing evidence that different groups of nodes performing different functions with some degree of independence [102]. Moreover, the community structure influence may reach further than these: a number of recent results suggest that networks can have properties at the community level that are quite different from their properties at the level of the entire network, so that analysis that focus on whole networks and ignore community structure may miss many interesting features [152]. For example, we may find that individuals within different community groups have a different mean number of contacts in some social networks: the individuals in one group might have many contacts with others while the others in another group might be more reticent. Example networks are reported in [12, 79] as sexual contact networks. Therefore, characterizing such networks by only quoting a single figure for the average number of contacts an individual has, and without considering the community structure, will definitely miss important features of the network, which is directly relevant to questions of scientific interest such as epidemiological dynamics [92].

Most of the existing community mining approaches assume that there is only one network representing one single relationship, such as the web linkage. We call such a network a homogeneous network. They also assume the network information is available and that each entity belongs to only one community. However, in real networks, many of these assumptions do not hold. For example, there exist multiple relations and entity types, each of which can be treated as a relation network. We call such a network *a heterogeneous network*, e.g., in a protein interaction network, proteins are related to each other via interactions in different environment. Some social networks, e.g. the WWW, can be dynamic or too huge to be accessed completely for community mining. Also, one entity could usually belong to multiple communities, e.g., one researcher active in the data mining community and the vi-

sualization community of the same time. We call such communities *overlapping communities*. In this research, we are particularly interested in *finding and evaluating community structures in a social network from different aspects*. Below we show several scenarios that illustrate the usefulness and necessities of the community mining research, and of the unsolved challenges that we face.

1.1 Scenarios and Challenges

Scenario 1 (Predict the possible disease infector). *Suppose an infectious disease just breaks out. What we have on hand is a typical human community network. We have labeled patients and information from the experts, which summarizes the possibility of infection between certain people. The health institution tries to find those people who are more likely to be infected next, based on the implicit community information provided.*

Challenge 1. *Discover communities in a social network for a certain purpose. Domain knowledge such as classified examples are provided.*

There are quite a few community mining approaches that can be used to generate communities from this network. However, they usually apply a fixed metric to all networks, despite the unique characteristics of each one. These characteristics can be obtained as domain knowledge from experts. Thus, we may identify useful communities for some networks by one approach, but could fail elsewhere. We state this problem as Challenge 1.

Scenario 2 (Refine query terms for search engine). *Mark is a new graduate student who just started his Ph.D program. He chose to work on data mining after he talked to a few professors. However, he basically has no idea of this field, which topic is popular, which paper is required to read, who is playing an important role, etc. He is trying to find related information via a search engine, with the query term "Data Mining". Some recommendations of appropriate query terms*

or name entities regarding different aspects of this field can absolutely help him out.

Challenge 2. *Given one entity, find categorized topics that are closely related to the given one in the form of various communities.*

As we have mentioned, one entity does not typically belong to only one community in real world networks. On the contrary, in most of the cases it is related to multiple groups. For example, as illustrated in Scenario 2, the name entity "Data Mining" may appear in various web pages and is related to many other entities: researchers, conferences, papers, even industrial companies that have related projects; and these name entities may not be related to each other at all, i.e., in different communities. An important challenge for community mining approaches is to find multiple communities in which an entity is involved, here referred to as Challenge 2.

Scenario 3 (Recommend future collaborators for researchers). *Prof. Jones is editing a book focusing on his research area. He wishes to find some researchers who are also in this area, to contribute to the work. He has a candidate list based on previous research experience and publication list, but still has no idea of which ones are the best to invite. He also prefers those who are more closely related to him to make the cooperation easier.*

Challenge 3. *Measure the asymmetric closeness between entity pairs within communities in the network.*

An ideal community mining approach should not only discover the community structure, but also somehow measure the relationship between any two entities to rank their closeness. However, in social networks, it often happens that the relation between two entities are not necessarily symmetric. For example, Prof. Simons may be the most related researcher to Prof. Jones regarding research interests, but Prof. Jones may only be one of the many researchers that share similar interests with Prof. Simons. Another good example is the friendship between children:

while Sandy claims George to be his best friend, George only treats Sandy as a normal classmate. Therefore, measuring the asymmetric closeness between two entities that are in the same community provides additional interesting features of the community network.

Scenario 4 (Locate communities for friend network website). *Friend network sites, such as FaceBook and MySpace, maintain a large and dynamic social network between their users. Every hour there are people joining or leaving the network, new relations are connected in minutes. Researchers working for these sites want to create user communities so that more accurate recommendation is possible.*

Challenge 4. *Identify social communities for real world networks, where the network could be huge and dynamic, also one node can belong to multiple communities.*

In practical situations, community mining algorithms are normally used on networks for which the communities are not known ahead of the time. These networks are too large and dynamic to be accessed frequently and completely. Nodes in these networks could join one or more communities. The problem is labeled as Challenge 4.

1.2 Thesis Statement

In this work, we investigate the possible issues that prevent existing state-of-the-art algorithms from discovering accurate community knowledge for particular social networks. We also demonstrate how we can leverage these community relational patterns to address the aforementioned challenges. The central thesis statement of this research is presented as follows:

Approaches of community mining can be implemented to discover implicit community information on social networks with various characteristics.

This research work justifies empirically and theoretically the practicality and feasibility of finding communities in social networks. The major issues addressed to support the thesis statements of this research are as follows.

- Investigate taking domain knowledge into consideration for discovering communities for networks in different domains.
- Investigate identifying communities with only limited network information.
- Investigate finding overlapping communities by locating accurate parameters.
- Investigate the practicality of applying community mining to improve search engine performance.
- Investigate ranking entities by computing relevance measures based on the relations.

1.3 Thesis Contributions

The major contributions of this dissertation can be summarized as follows:

1. We propose a method to include domain knowledge as guiding criteria in the community detection process by either rewarding or penalizing the metric that evaluates the discovered structure, without increasing algorithm complexity. Our new measure and algorithm improve the accuracy for community detection over previous algorithms when applied to real world networks for which the community structures are already known, and also give promising results when applied to randomly generated networks for which we only have approximate information about communities. This shows the robustness of the algorithm against noise.
2. For community mining with only local information, our method proposed in Chapter 6 requires no parameters and our metric L is robust against outliers.

The proposed algorithm not only discovers local communities without an arbitrary threshold, but also determines whether a local community exists or not for certain nodes.

3. For overlapping community identification, our visual data mining approach could assist the user in finding appropriate parameters to describe the communities they are looking for. The new metric R can effectively quantify the relations between entities. The method is scalable and is able to discover communities, hubs, and outliers in social networks.

4. Our community mining technique, when applied to search engine results to identify query sense communities on a network of extracted keywords, gives good results. The unsupervised approach, which categorizes web pages into meaningful categories with information of the query and the result page content only, does not require additional information about the query in question and is feasible for real time page clustering for search engine results. The use of a community mining metric of the discovered query sense community structure to assess whether page clustering is required for search results is new. While previous methods overlook this issue, experimental results show that our method is accurate.

5. While previous ranking methods focus on homogeneous networks, our entity ranking approach is useful for heterogeneous networks where there exist multiple types of entities and relations. We also provide a convenient interface for users to navigate the ranking and relations.

Organization: This dissertation is organized as follows. Chapter 2 briefly surveys the background of social network analysis. Chapter 3 introduces the state-of-the-art community mining approaches, evaluations and other related work. Chapter 4 defines the problems and overviews our solution to address the challenges, which is presented in detail in Chapter 5, 6, 7, 8 and Chapter 9. Finally, Chapter 10 concludes.

Chapter 2

A Review of Social Network Analysis

While emerged as an important technique in modern sociology, Social Network Analysis (SNA) has been a widely used approach in many fields, including anthropology, biology, communication studies, economics, geography, information science, organizational studies, social psychology, and sociolinguistics. The peculiarity of this perspective is that it focuses not only on individuals or other social units, but also on the relationship between them [135]. In this chapter, our goal is to give a general review of this aspect and a description of resources and principle topics covered by Social Network Analysis and Data Mining. In the first section, we concentrate on the historical and methodological context of SNA. In the second section, by introducing some recent studies about Social Network and its relationship with Computer Science, in particular Data Mining, we present the research background of this dissertation.

2.1 Social Network Analysis: A Brief Survey

Many social aggregations can be represented in terms of units composing the aggregation and relations between these units. This kind of representation is called "Social Network", which can be defined as a social structure made of nodes that are tied by one or more specific types of relations. Every node (or unit) is usually called a "social actor", which may represent a person, a group, a document, an organization, a nation and so on. A relation is represented as a linkage or a flow between these nodes. The set of possible relations is probably infinite: friendship,

kinship, dislike, conflict, trade, acquaintance, financial exchange, values, contents, physical connection, WWW hyperlink, the presence in the same place, and so on. Therefore, in contrast to traditional social scientific studies, which concentrate on attributes of individual social actors, Social Network Analysis provides an alternative view, where the attributes of units are less important than their relationships and ties with other units within a network. The advantage of such a representation is that it permits the analysis of social processes as a product of the relationships among social entities [135].

2.1.1 History

According to Scott [185], there are three main research lines for Social Network Analysis, led respectively by

- Sociometric Analysts, who mainly use and develop graph theory.
- Harvard researchers, who first studied models of interpersonal relationship and clique formation, and after 1970 developed many useful ideas in SNA.
- Anthropologists from the University of Manchester, who studied relational structures characterizing tribal society communities.

Social Network Analysis originated in 1930s, led by three scientists: Jacob Moreno, Kurt Lewin and Fritz Heider. Moreno developed sociometry [141]. He started asking people who their friends were and explored how their relations with others served as both limitations and opportunities for their psychological behavior and action. He believed that large scale social phenomena, such as the economy and state, were sustained and reproduced over time by the small scale configurations formed by peoples patterns of friendship, dislike and other relations. Moreno's chief innovation was to devise the "sociogram" (see Figure 2.1) as a way of representing the formal properties of social configurations [185]. One of the configurations he observed was the sociometric star: an individual was chosen by many others as a friend. Another early work by Lewin on group behaviour proposed that the field of social forces where the group was located determines the behaviour of

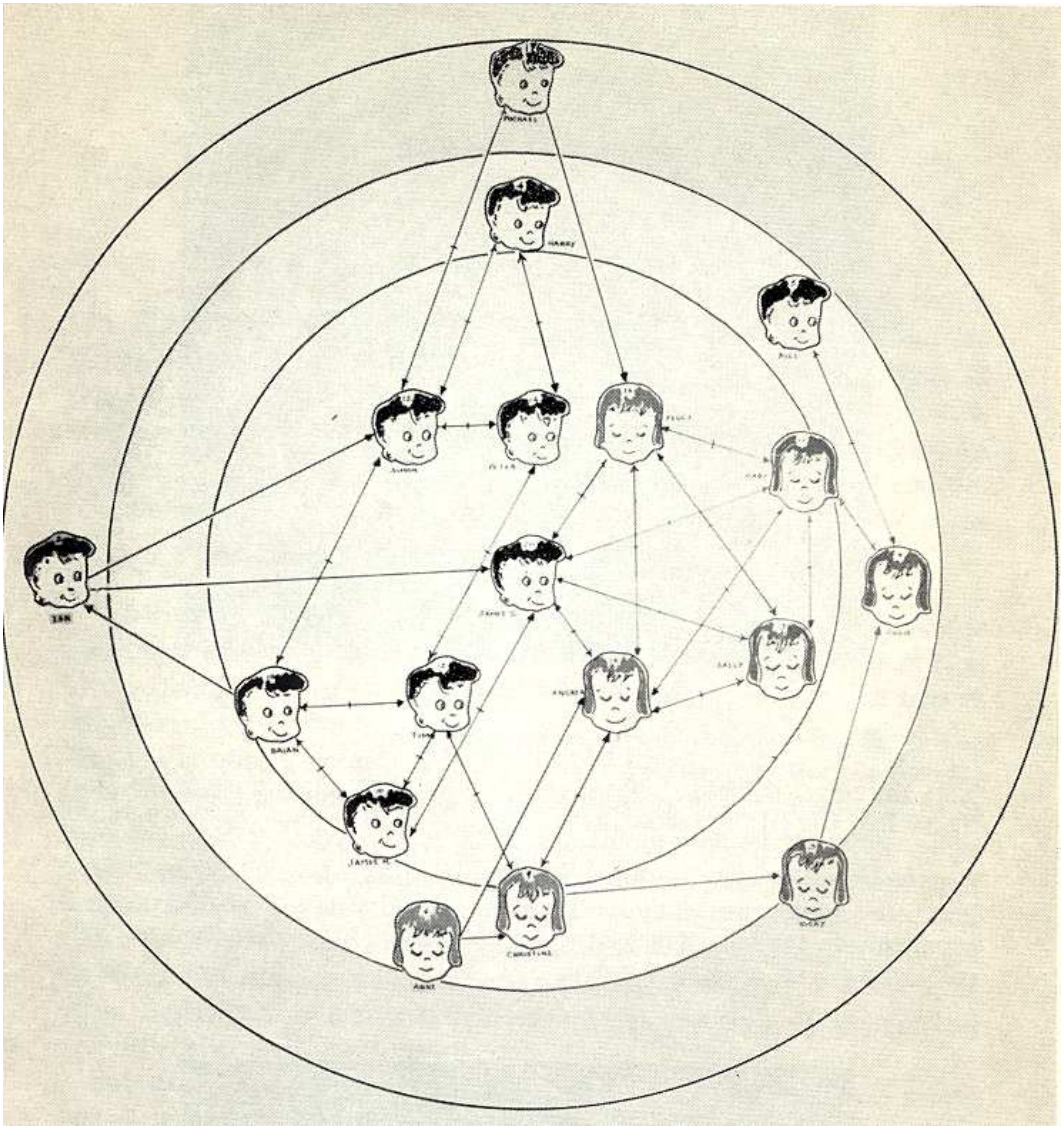


Figure 2.1: An example of original social network diagrams [74]

the group [124]. Lewin argued that the structural properties of this social space could be investigated mathematically using vector theory and topology. Finally, Heider pioneered the area of social perception and attitudes. He developed what is known as balance theory [185]. He believed that the mind seeks balance by trying to hold ideas that are not in conflict with one another, which also applies to attitudes towards other people [101]. These ideas were soon developed by Frank Harary and Dorwin Cartwright, who used graph theory to build a powerful formal tool for social structure analysis [35].

Around the same time, Social Network Analysis developed with the studies of some Harvard researchers, who dedicated their attention to the search for decomposition and exploration methods of structures composing a graph. In particular, the research of W. Lloyd Warner and Eltan Mayo, who are the leaders of this movement, about Chicago's central plant "Hawthorne" was a milestone. The originality of their studies consisted of the large usage of sociograms [136], and the introduction of the concept of "clique".

In the 1950s, researchers from the Department of Social Anthropology at Manchester University pointed their attention at the effective configuration of relationships deriving from power and conflict between individuals, instead of set up norms and institutions of a society [185]. This group, led by John Barnes, Elizabeth Bott and later J. Clyde Mitchell, began investigating how the structure of relations among people affected not only the individuals but the society as a whole. The term "Social Network" was first introduced in 1954 by John Barnes [17], who gave life to a remarkable formal development of the analysis of social structures.

Based on Barnes et al.'s work, a group at Harvard led by Harrison C. White in the 1960s and 1970s further developed the mathematical aspects of social network analysis, translating many important concepts from social science, such as the notion of "social role", into mathematical form which allowed them to be measured and modeled. *The central idea is that the search of structure in a network should not be based on a-priori defined and well-known categories, but on the real relations among the network nodes and on how these relations structure it, with the aim to describe the concrete and emergent role structures.* This important assumption makes most modern data mining methods inappropriate for SNA tasks, as we review in Chapter 3. In this context, they introduced the concept of "block model", which was intended as corresponding to the role of a group of components of a social network [132, 215]. Furthermore, Mark Granovetter proposed the important "weak tie hypothesis" [87], which was shown to be useful in many researches. The "weak tie hypothesis" argues that if node A is strongly linked to nodes B and C, then there is a great probability that B and C are linked to each other.

Among the many ideas developed by White and his students, one original theory, known as "small world phenomenon" and proposed by Stanley Milgram [139], revealed to be very interesting. Milgram's approach concerned the empirical effort to determine how many steps (or ties) are necessary, in a well-defined population, so that two different individuals can meet each other [135]. In order to learn more about the probability that two randomly selected people would "know" each other, Milgram chose individuals in the U.S. cities of Omaha, Nebraska and Wichita, Kansas to be the starting points and Boston, Massachusetts to be the end point. Letters that detailed contact information about the target person were given to the starting individual. Upon receiving the letter, the recipient was asked to forward the letter directly to the target person if he personally knew that person. Otherwise, the recipient was asked to sign their name on the letter and forward the letter to a friend or relative they know personally that is more likely to know the target. As a result, 64 of the 296 letters eventually did reach the destination. The average path length fell around 5.5 or 6. Hence, the researchers concluded that people in the United States are separated by about six people on average. This is the famous "Small World Experiment".

To summarize, Social Network Analysis was born from the joint activity of social psychologists, anthropologists, sociologists, mathematicians, physicians and economists. Nowadays it is used with profit for research in many fields including behavioral, social, economical and political disciplines. However, in the era of information explosion with the rapidly increasing amount and size of social networks, e.g., the ever-growing World Wide Web (WWW) and Facebook community, the problem of managing the social information becomes more challenging. Therefore, it is necessary to apply computer science techniques to social networks to analyze the structure more efficiently and accurately.

2.1.2 Types of Data

According to Scott [185], there are two principle types of social science data.

- Attribute Data represent the attitudes, opinions and behaviour of individuals, properties, qualities or characteristics of units or groups. This information

are often regarded simply as attributes of particular individuals that can be quantified and analyzed by the many available statistical procedures.

- Relational Data are the ties, connections and contacts, which relate one unit to another. Relations cannot be reduced to a property of individuals, but are properties of the structure of units.

For attribute data, the appropriate methods are variable analysis, whereby attributes are measured as values of particular variables, such as income, profession, education, etc. For relational data, on the other hand, the appropriate methods are network analysis, whereby relations are treated as linkages which connect units. While it is still possible to conduct quantitative and statistical counts of relations, network analysis methods usually apply qualitative measures of the network structure. Note that although there are distinct types of data each with their own appropriate methods of analysis, there is nothing specific about data collection methods which are used to produce them. Traditionally, questionnaires, interviews, participant observation or documentary sources can be consulted in order to generate the data. Also note that, in this dissertation we are more interested in relational data and network analysis than attributes and variable analysis.

2.1.3 Concepts

In this section, we first introduce the relation definition and network representation, which are the basis of Social Network Analysis. Then we describe several important notions originated from graph theory.

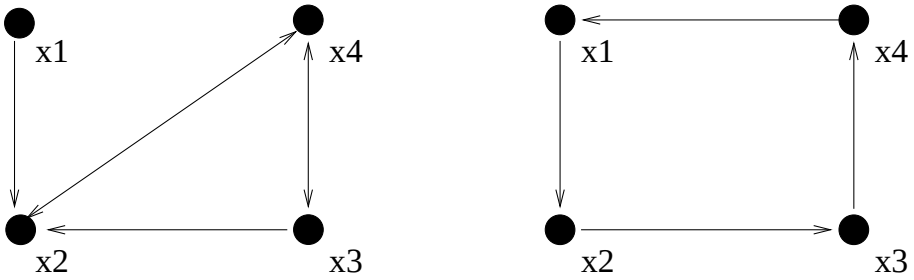
Relation Definition and Network Description

The fundamental concept that all SNA approaches share is the notion of relation. Let us consider two actor sets, labeled X and Y:

A relation between X and Y is every set of ordered couples where the first element belongs to X, and the second element belongs to Y [135].

A = {(x1, x2), (x2, x4), (x4, x2), (x3, x2), (x3, x4), (x4, x3)}
B = {(x1, x2), (x2, x3), (x3, x4), (x4, x1)}

List Representation of Relation A and B



Network Representation of Relation A and B

	x1	x2	x3	x4
x1	0	1	0	0
x2	0	0	0	1
x3	0	1	0	1
x4	0	1	1	0

	x1	x2	x3	x4
x1	0	1	0	0
x2	0	0	1	0
x3	0	0	0	1
x4	1	0	0	0

Matrix Representation of Relation A and B

Figure 2.2: Different representations of relations A and B on the set $X = \{x_1, x_2, x_3, x_4\}$

For simplicity, we now consider the simplest network form, which is a binary network on the set of actors X. There are three kinds of representation of this social network (see Figure 2.2 for relation representation on set $X = \{x_1, x_2, x_3, x_4\}$):

- The first description is the simple list of all the elements taken from the actor set, and the list of the pairs of elements which are linked by a social relation.
- The second description comes from the graph theory: every actor is represented as a node, the relations defined by pairs of individuals are represented by edges between two nodes.
- The third description is in a matrix form.

There are other kinds of networks, for example, there can be more than one relation, relations can be weighted or directed, etc. It is also possible to use relations with more than 2 actors, e.g., ternary relations. However, these relations are not common in Social Network Analysis: "(binary relational representations) include the vast majority of the models that have been proposed and applied in the social network literature to date" [173].

Graph Theory

Graph Theory has been used in several scientific fields such as anthropology, communication, geography and many others. In particular, it plays a crucial role in Social Network Analysis since it has been used to denote the structural properties of a network. More specifically, Graph Theory provides to Social Network Analysis a tool to quantify the network properties, and to formally analyze different characteristics of a network. Therefore, Graph Theory, and more generally a graph, is one of the best way to represent and detect the social structure which is defined as a relation (or some relations) connects different individuals of a social group. In the following we introduce some basic concepts of Graph Theory in SNA, which are used frequently in this dissertation.

The aim of SNA is to detect and to investigate the structure identified by a set of relations connecting a set of individuals. We have specified that we only investigate binary relations between a couple of individuals. However, a binary relation can be *directed* or *undirected*: relations like marriage, neighbour or friendship are undirected while relations such as "to be parent of" or "to be the best friend of" are directed relations. Assume in any graph we have n nodes and m edges, an undirected relation can be represented by graph G, where there is no difference between line (n_i, n_j) and line (n_j, n_i) between two nodes n_i and n_j. Thus an undirected relation is symmetric: if n_i is a friend to n_j, n_j is also a friend to n_i. On the other hand, for a directed relation where the two ties are different, we need to use a directed graph, where a relation is represented as a set of arcs connecting nodes in the network. Every couple of nodes connected by an arc is ordered and the order is not invertible. Note that most of the proposed approaches in this dissertation focus on

undirected networks, the directed-network-extensions are possible, even though we do not cover them herein.

Now we introduce the important notion of *subgraph*. A graph G_s is a subgraph of a graph G if the nodes of G_s are a subset of the nodes of G and the edges of G_s represent a subset of the edges of G [135]. The notion of subgraph is fundamental for several definitions and metrics in Social Network Analysis. For example, a *clique* is a maximal complete subgraph composed by at least three nodes. A clique is complete because each node in it is adjacent to any other node in the clique. A clique is maximal because there are no nodes that are not included in the clique and are also adjacent to all the nodes in the clique [96]. The clique notion is the basis of many research works that aim to discover groups or communities in the social network.

We now introduce several quantitative metrics to represent social structure properties using Graph Theory.

- *Node Degree* is the simplest and most commonly used metric. The degree of a node in a graph is defined as the number of edges connecting to that node. The mean node degree $\bar{d} = \frac{2m}{n}$, since each edge counts for both ending nodes. It is more complicated for directed graphs. There are two kinds of node degree for each node: the in-degree, which is the number of arcs connect *to* a node, and out-degree, which is the number of arcs connect *from* a node. The mean in-degree and out degree are equivalent: $\bar{d}_i = \bar{d}_o = \frac{m}{n}$.

- *Density* measures the proportion of edges in the graph over the total number of possible edges. The density $D = \frac{2m}{n(n-1)}$, where m is the edge number and the denominator is the edge number for a completely connected graph. For directed graphs, $D = \frac{m}{n(n-1)}$.

- *Diameter* of a connected graph is the maximum geodesic distance [95] between any pair of nodes. The geodesic distance is defined as the shortest path between two nodes in the network.

- *Neighbourhood* of the node n_i represents all nodes that connect to n_i. The neighbourhood size is the number of these nodes.

There are many other important metrics and indexes for SNA, however, the listed ones are most related to our research and are used frequently in this dissertation.

2.2 Social Networks and Data Mining

After the invention of the computer and the Internet in late 20^{th} century, the amount of published information has increased to a tremendous size. For example, in the World Wide Web, which can be seen as the world's largest database, there are at least 20 billion pages in the indexed web[1] and 115,658,964 registered domains[2] as of February 23, 2010. The amount is still increasing in a faster speed. Many of these datasets are best described as a linked connection of interrelated objects, i.e., a social network, for example, a collection of linked web pages, medical data describing patients, disease, treatments and contacts, or bibliographic data describing publications, authors and conferences. Therefore, on one hand, computer scientists apply data mining techniques to extract hidden patterns from the data; on the other hand, sociologists develop SNA methods to discover properties of the same network. Both research fields aim at the same problem, albeit by somewhat different means. In this section, we introduce a list of data mining tasks that are also covered by Social Network Analysis before we define the research area of Community Mining.

2.2.1 Data Mining Tasks for Social Network Analysis

"Links", or more generically relationships, among data instances are ubiquitous. These links often exhibit patterns that can indicate properties of the data instances such as the importance, rank, or category of the object [81]. By taking links into account, more complex situations arise. For example, not all links might be observed at the same time, therefore predicting the existence of links between objects may be necessary. Mining with links also leads to other derivative challenges focused on discovering substructures, such as groups or common subgraphs. Here we briefly

[1] http://www.worldwidewebsize.com/
[2] http://www.domaintools.com/internet-statistics/

introduce several recent data mining tasks that can be broadly categorized as Social Network Analysis tasks, which focus on social actors, relations, and graphs.

Group Detection

The goal of group detection is to cluster the nodes in the graph into groups that share common characteristics. A range of techniques have been presented in different fields to address this general problem, e.g., hierarchical clustering [151], stochastic blockmodeling [163], spectral clustering [60, 218], edge betweenness [155], etc. We will discuss related works for this problem in more detail in Chapter 3.

Relational Object Ranking

Ranking is probably one of the most well known data mining tasks, thanks to Google's success as a web search engine. The objective of relational object ranking is to exploit the network relational structure to order a set of nodes within the graph. The idea has been widely used in the web information retrieval domain [117, 167]. In SNA, measures of *centrality* [28, 29, 76, 78] are developed to characterize the network structure to order individuals. See Chapter 3 for more detail.

Relational Object Classification

Relational object classification recently received considerable attention [38, 123, 164]. In this problem, the task is to label the members of a social network from a finite set of categorical values. The discerning feature of relational object classification that makes it different from traditional classification is that in many cases, the labels of related objects tend to be correlated, thus the challenge is to design algorithms that exploit such correlations and jointly infer the categorical values associated with the entities in the graph [81].

Entity Resolution

Nodes in a social network might refer to the same real-world entity, thus identifying and determining the references become a problem. Examples of this problem arise especially in Natural Language Processing (NLP) [20, 160, 176] and Database

Management [24, 62]. Traditionally, entity resolution has been considered as a pairwise resolution problem, where each pair is independently resolved as being coreferent or not, solely based on the similarity of their attributes. In a social network, however, nodes that connect to the node in question should also be considered. For example, co-authors of an author in bibliographic data, or co-occurrence links between name entities in documents could be used to verify the reference of the entity in question.

Link Prediction

As we have mentioned, some links in the network may be unobserved while others are observed. Moreover, a network can be dynamic, i.e. relations in this network may change over time. Examples include predicting links among actors in social networks, such as predicting friendships; predicting the participation of actors in events [165], such as email, telephone calls and co-authorship; and predicting semantic relationships such as "adviser-of" based on web page links and content [55, 200]. Therefore, link prediction is the problem of predicting the existence of a connection between two entities, based on other observed relations and attribute similarity.

Subgraph Discovery

The work on subgraph discovery attempts to find interesting or commonly appearing subgraphs in a set of networks. The discovery of these patterns may be useful for graph classification or other data mining tasks [81]. Subgraph discovery is analogous to the frequent item set problem in Data Mining, thus many approaches [107, 122] naturally exploit the Apriori property [4].

Among all these data mining and SNA tasks, we are particularly interested in group detection and entity ranking. We believe that group detection is the basis of other SNA tasks and investigating the relations between grouped entities by ranking would be the next important step.

2.2.2 Community Mining

In social networks, which are composed of nodes representing individuals and edges indicating relationships, a common property is the *community structure*, which notes the existence of densely connected groups of vertices, with only sparser connections between groups. For example, WWW pages are more likely to connect to or to be connected from pages that cover the same topic; employees that work on related projects in an organization may communicate frequently via email; researchers that have co-authored several papers may work closely on the same research problem; people that call each other more often in the tele-communication network are usually friends, and so on. The discovery of these groups, clusters or communities is the basis for many other SNA tasks, e.g., entity ranking, classification and resolution.

We define *a community to be a social network partition such that entities within the same community share some common trait or proximity, judged by some defined entity similarity or relationship metric*. Thus, *"Community Mining" is the process of analyzing attributes and relations from different perspectives to discover communities from social networks and extract hidden patterns from communities*. In particular, "community patterns" in this dissertation are discovered and presented in the form of rankings among related entities in the social network.

In recent years, *Community Mining*, which focuses on the detection and characterization of social network structure, has received considerable attention in sociology and lately data mining (see Chapter 3). While there are many possible fields that can benefit from the community mining research, we list a few of them as follows.

- *Search Engines.* Identifying highly related web page communities may affect page rankings of a search engine. Pages that are categorized into the same word sense community as the target query should be ranked higher than those that are not. See Chapter 8 for details.

- *Customer Services.* Detecting customer groups in a tele-community network may be useful to evaluate the importance of a particular customer. If one is

evaluated as a core member of a large community, the service provider may give more bonus to make him or her stay.

- *Academic Collaboration.* Locating people that work on the same topic may help a researcher find future collaborations and possible projects.

- *e-Business.* Grouping related products can improve the recommendation list that is shown to the user and accelerate potential sales.

In the next Chapter, we first categorize topics of Community Mining based on constraints on the social network in question, then present the state-of-the-art solutions for each problem respectively.

Chapter 3

State-of-the-Art in Community Mining

Traditional data mining algorithms, such as association rule mining, supervised classification and clustering analysis, commonly attempt to find patterns in a data set characterized by a collection of independent instances of a single relation, which is consistent with the classical statistical inference problem of trying to identify a model given an independent, identically distributed (IID) sample [81]. However, a new emerging challenge that data mining researchers face is solving the problem of mining richly structured, heterogeneous data sets. Such data sets are usually modeled as networks or graphs and contain multiple object types, which can be related to each other in various ways, e.g., commercial data describing relations between customers, movies and actors or bibliographic data describing relations between conferences, authors and topics. Naïvely applying traditional statistical inference procedures, which assume that instances are independent, can lead to inappropriate conclusions about the data [110]. For example, for a search engine, indexing and clustering web pages based on the text content without considering their linking structure would lead to bad results for queries. The relations between objects should be taken into consideration and can be important for understanding the data structure and knowledge patterns.

The newly emerging research area for these problems refers to data mining techniques that explicitly consider the links or relations between objects when building predictive or descriptive models of the linked data [81]. Common data mining and

SNA tasks include entity ranking, object classification, group detection, entity resolution, link prediction and subgraph discovery. In particular, we are interested in the tasks of group detection and entity ranking. We call this process *community mining*. In this chapter, we review related works on these emerging themes.

3.1 Community Mining Categorization

Before introducing details of related works in community mining, we classify the main research works in this field into seven categories.

- *Community discovery with global network information.* In a perfect world, we have full access to complete information of the social network in question, e.g., number of communities, size of the node set and edge set. A global community is a community defined based on global information about the entire network. That is, one needs to access and see the whole network information. The majority of current research works on community mining rely on this assumption. They are reviewed in Section 3.2.

- *Community discovery with local network information.* For huge or dynamic networks, where global information is no longer available, e.g., WWW, we have to discover communities with only local information, such as neighbourhoods and node degrees. A local community is a community defined based on local information without having access to the entire network. Research works with this constraint are reviewed in Section 3.3.

- *Community evaluation.* For many social networks, the ground truth of communities usually does not exist. Thus, evaluation metrics are necessary to verify the results of various algorithms. These works are reviewed in Section 3.4.

- *Overlapping community discovery.* While most approaches assume that one node can belong to only one community, it is usually not the case in the real world. For example, an author could publish in several different areas, or a

person could be an active member of multiple social groups. Research that focuses on overlapping community discovery is reviewed in Section 3.5.

- *Dynamic community discovery.* Social network communities usually evolve over time. One node may belong to one community and change to another after a certain amount of time. Related works on discovering dynamic communities are reviewed in Section 3.6.

- *Community discovery with multiple relations or attributes.* In heterogeneous networks, there are more than one relation connecting social actors. Identifying the importance of different relations before attempting to find communities is required on such a network. Moreover, in many applications more informative graphs with attributes are given with their network structure. Related works are reviewed in Section 3.7.

- *Entity ranking.* The relations between entities among different communities, which can be represented as relevance rankings, are interesting to investigate after the communities in the network are discovered. Related works on entity ranking are reviewed in Section 3.8.

3.2 Mining on Global Network

Many datasets can be described in the form of graphs or networks where nodes in the graph represent entities and edges represent relationships between pairs of entities. For example, the WWW can be viewed as a very large graph where nodes represent web pages and edges represent hyperlinks between pages. In social networks, the nodes typically represent individuals and edges indicate relationships, e.g., citation graphs can be constructed with papers as nodes and references as edges. A number of community mining approaches, as we review in the following, studied ways to effectively discover community information from these network structures. Most of these attempts are based on the simplest kind of network, with a single type of undirected, unweighted edge connecting unweighted vertices of a single type. An important common assumption for these approaches is that the complete global net-

work information is always available. We now survey several main methods, and where they fall short.

Past work on the problem of finding groups in networks divides into two main principal lines of research. The first, which is referred to as *graph partitioning* with applications in parallel computing and VLSI (Very Large Scale Integration) design [65, 71], is described as dividing the vertices of a network into some number k of groups with roughly equal size, while minimizing the number of edges that run between vertices in different groups. The second, identified by names such as *block modeling*, *hierarchical clustering*, or *community structure detection*, has been pursued by sociologists, physicists and applied mathematicians, with applications especially to social and biological networks [150, 210, 216]. These two lines of research are really addressing the same question, albeit by somewhat different means. There are, however, important differences between the goals of the two camps that make quite different technical approaches desirable [153]. For example, *graph partitioning* approaches usually know in advance the number and size of the groups into which the network is to be split, while *community structure detection* methods normally assume that the network of interests may divide naturally into some subgroups, which is determined by the network itself but not by the user.

3.2.1 Graph Partitioning

There is a long tradition of research by computer scientists on graph partitioning. Generally, finding an exact solution to a partitioning task is believed to be an NP-hard problem, making it prohibitively difficult to solve for large graphs. However, a wide variety of heuristic algorithms have been developed that give acceptable good solutions in many cases: METIS [115], flow-based methods [72, 183], information-theoretic methods [58] and may be the best known Kernighan-Lin algorithm [116], which improves on an initial division of the network by optimization of the number of within- and between-community edges using a greedy algorithm, and achieves $O(n^3)$ running time on sparse graphs. The main issue of these methods is that input parameters such as the number of the partitions and their sizes are usually required, but we do not typically know how many communities there are, and there is no

reason that they should be of roughly the same size. The number and sizes of the groups should be determined by the network topology but not by the user. Moreover, the number of inter-community edges need not be strictly minimized either, since more such edges are admissible between large communities than between small ones [155].

In practice, most approaches to graph partitioning have been based on iterative bisection: find the best division of the graph into two groups, and then further subdivide those into two until one finds the required number of groups. An important family of algorithms following this strategy is the spectral bisection method [70, 156, 177], which is based on the eigenvectors of the graph Laplacian. The Laplacian of a n-vertex undirected graph G is the $n \times n$ symmetric matrix L whose diagonal element L_{ii} is the degree of vertex i, and whose off-diagonal element L_{ij} is -1 if vertices i and j are connected by an edge and zero otherwise [150]:

$$L = D - A \tag{3.1}$$

where D is the diagonal matrix of vertex degrees and A is the adjacency matrix. For example, the vector $(1, 1, 1...)$ is always an eigenvector with eigenvalue zero since all rows and columns of the Laplacian sum to zero ($D_{ii} = \sum_j A_{ij}$).

If the network can be perfectly divided into communities, i.e. there are k non-overlapping groups of vertices ($G_1, ...G_k$) such that there are only intra-community edges and no inter-community ones, the Laplacian will be block diagonal. Each diagonal block will form the Laplacian of its own, thus there will be k eigenvectors $v_1, v_2, ...v_k$ with eigenvalue 0. If the network separates well but not perfectly, i.e., there are a few edges that do not fit the block-diagonal pattern, there will in general be one eigenvector with eigenvalue zero, and $k-1$ eigenvalues slightly greater than zero, since all eigenvalues of the graph Laplacian are non-negative. These eigenvectors, which correspond to non-negative eigenvalues of graph Laplacian of the network, are approximately linear combinations of the eigenvectors $v_1, ...v_k$ of the Laplacian of perfect communities, which are non-overlapping groups of vertices in the network. Hence, by looking for eigenvalues of the graph Laplacian only slightly greater than zero and taking linear combinations of the corresponding eigenvectors,

one should in theory be able to find the community blocks themselves, at least approximately [150]. In a particular special case, when there are only two blocks, since all eigenvectors corresponding to non-degenerate eigenvalues of a symmetric matrix are orthogonal, it is obvious that all eigenvectors other than the one corresponding to the lowest eigenvalue must have both positive and negative elements. Therefore, we can find one eigenvector with eigenvalue slightly greater than zero and elements all positive for one community and all negative for the other. Thus, the spectral bisection method divides the network into its two communities by looking at the eigenvector corresponding to the second lowest eigenvalue and separating the vertices by the signs of their corresponding elements.

Despite its evident success in the graph partitioning arena, spectral partitioning suffers from the same issue as the other partitioning methods: the sizes of the groups into which the network is divided need to be fixed but are not usually known in advance. Moreover, if we set the group sizes to be free, the spectral partitioning method (and other methods that minimize the cut size without constraints on the group sizes) breaks down: the minimum cut size is always achieved by the trivial division which puts all vertices in one group and none in the others. Various constraints have been proposed to resolve the issue, such as the *ratio cut* [39, 47, 93, 213], which minimizes not the simple cut size R ($R = \sum_{i \in group_1, j \in group_2} A_{ij}$) but the ratio $\frac{R}{n_1} + \frac{R}{n_2}$ where A is the adjacency matrix, n_1 and n_2 are again the sizes of the two groups of vertices. Still, however, this approach is biased in favor of divisions into equal-sized parts and thus still suffers from the same drawbacks that make standard spectral partitioning inappropriate for community mining. Researchers soon have realized that cut sizes are simply not the right thing to optimize because they don't accurately reflect the intuitive concept of network communities. Therefore, similarity, or association based methods have been proposed: the *normalized cut* [187], the *min-max cut* [60] and the *modularity-based method* [152]. In more detail, the normalized cut measures the cut cost as a fraction of the total edge connections to all the nodes in the graph; the min-max method minimizes the association between two cut sets and maximizes that within each subgraphs; the

modularity-based method uses a benefit function Q defined by

$$\begin{aligned} Q &= (number\ of\ edges\ within\ communities) \\ &\quad -(expected\ number\ of\ such\ edges)\ [75] \end{aligned}$$

This benefit function Q is called *modularity* [147, 155]. It is a function of the particular division of the network into groups, with larger values indicating stronger community structure (more details on the modularity are provided in Section 3.4).

The major disadvantage of the spectral bisection method is that it only bisects graphs. Division into a larger number of communities is usually achieved by repeated bisection, but this does not always give satisfactory results given the community ground truth.

3.2.2 Hierarchical Clustering

The approaches developed by sociologists in their study of social networks for finding communities, which have been directed almost exclusively at the analysis of empirically derived network data, are perhaps better suited for our current purpose than the aforementioned spectral clustering methods, whose performance highly depends on input parameters.

The principal and most popular technique in use is *hierarchical clustering* [185]. The main idea of this technique is to discover natural divisions of social networks into groups, based on various metrics of similarity (usually represented as similarity x_{ij} between pairs (i, j) of vertices, based on the given network information.) There are a variety of ways of defining the similarity between vertices, e.g. the *Euclidean distance* [32, 210] compares the neighbours that two vertices share:

$$x_{i,j} = \sqrt{\sum_{k \neq i,j} (A_{ik} - A_{jk})^2} \tag{3.2}$$

where x is the similarity measure and A is the adjacency matrix. Another commonly used similarity measure is the Pearson correlation between columns or rows of the adjacency matrix [210]. If we define means and variances of the columns as:

$$\mu_i = \frac{1}{n} \sum_j A_{ij} \tag{3.3}$$

$$\sigma_i = \sqrt{\frac{1}{n}\sum_j (A_{ij} - \mu_i)^2} \tag{3.4}$$

The correlation coefficient is:

$$x_{ij} = \frac{\frac{1}{n}\sum_k (A_{ik} - \mu_i)(A_{jk} - \mu_i)}{\sigma_i \sigma_j} \tag{3.5}$$

Related work on similarity measure is also reviewed in Section 3.8.

Hierarchical clustering methods fall into two broad classes: *agglomerative* and *divisive* [185], depending on whether they focus on the addition or removal of edges to or from the network. Once we have a measure for vertex similarity, in an agglomerative method, edges are added to an initially empty network, which has n vertices and no edges, starting with the vertex pair with the highest similarity. Agglomerative methods have their problems: they tend to find only the cores of communities and leave out the periphery. The core nodes in a community usually have high similarity, and thus are found early in the clustering process, but peripheral nodes that have weak similarity to others often get neglected. On the other hand, in a divisive method, we start with the whole network of interest and attempt to find the *least* similar connected pairs of vertices and then remove the edges between them. By doing this repeatedly, we divide the network into smaller and smaller components. While agglomerative methods may ignore peripheral nodes, divisive methods usually classify all of them, including outliers, into communities, which would lower the accuracy of community detection. The procedure of both hierarchical clustering methods can be halted at any stage and the result components in the network are taken to be the communities. It has been applied to various social networks with natural or predefined similarity metrics [7, 52, 85, 134, 151, 155, 211].

The hierarchical clustering method has the advantage that it does not require the size or number of groups we want to find beforehand. However, they tend to find only the cores of communities correctly and leave out the periphery, which has weak similarity and is hard to be placed accurately.

3.2.3 Recent Approaches

After discussing traditional approaches to find communities in networks, we now describe some more recent community mining algorithms.

Girvan and Newman proposed a community finding algorithm based on an interesting measure, called "edge betweenness" [85]. Their method belongs to divisive hierarchical clustering, which progressively removes edges from a network. *The betweenness of an edge is defined to be the number of shortest paths between vertex pairs that run along the edge in question, summed over all vertex pairs.* This quantity can be calculated for all edges in time $O(mn)$ on a graph with m edges and n vertices [30]. The algorithm then simply calculates the betweenness of all edges in the network and removes the one with the highest betweenness, and repeats until no edges remain or a stopping criterion is met.The main disadvantage of this algorithm is that it is slow, since the betweenness score must be recalculated in each iteration. Tyler et al. have suggested modifications based on Monte Carlo estimation to address this problem, at the cost of a reduction in accuracy [203]. To evaluate the result of their algorithm, Newman and Girvan proposed a measure called *Modularity* (reviewed in Section 3.4), which is a numerical index of how good a particular division is [155]. Later Newman proposed an alternative agglomerative hierarchical clustering approach to discover community structure based on a greedy optimization of modularity Q over possible divisions [151]. It starts with each vertex in a separate community on its own and amalgamates communities in pairs, choosing at each step the pair whose amalgamation gives the greatest increase in Q. The main advantage of this algorithm is its speed, which is even faster after Clauset et al.'s modification [52]. In later work, inspired by spectral clustering theory, Newman proposed a framework to maximize the modularity based on the eigenspectrum of a modularity matrix [152], similar to the graph Laplacian in graph partitioning algorithms. By building such a matrix and its eigenspectrum, the problem of community structure detection is equivalent to choosing a division of those eigenvector values so as to maximize the result modularity. A similar approach has been proposed by White and Smyth [218] in the computer science community.

Li et al. focused on community structure of name entities extracted from web pages and blogs [125, 126]. At first, they collect data from Google search result pages, given query string with specific person names. Then the name entity graph is generated based on names and relations extracted from these pages. A hierarchical

clustering algorithm is proposed to find and merge triangles of name entities as community cores. Similar to the hierarchical clustering method used in [126], Zhou et al. proposed a concentric-circle model to locate the cores of communities first, and then merge them based on appropriate similarity thresholds [234]. However, both Li and Zhou et al.'s work did not provide a convincing evaluation method to verify the value of their results. The fact that they focus on locating the community cores makes their algorithm hard to find community periphery entities.

In contrast to the above methods, where community assignments are deterministic, approaches for group detection have been introduced based on the concept of *stochastic blockmodeling* [163] from SNA. In that model, positions for the entities are treated as IID random variables, and relations of a given type between two entities are random variables dependent solely on the positions of the individuals they link. Nowicki and Snijders proposed a general stochastic blockmodeling approach using Gibbs sampling to infer the posterior distribution for positions [163]. Wolfe and Jensen [220] extended the general stochastic blockmodeling approach by allowing an individual to have multiple position types in order to provide the flexibility to model multiple roles that one individual may have in different context. Wang et al. proposed a generalization of the broad stochastic blockmodeling approach that allows joint inference of groups and topics based on observed relationships and their attributes [207, 208]. Similarly, Zhang et al. proposed several probabilistic community discovery models for large scale social networks [231, 232]. The network topology is used to compute the weights of the connections, denoted as social interaction profiles of the nodes. On the other hand, visualization methods are applied to mining research communities based on bibliography data [104, 105]. Interactive visualization is provided to explore the paper network model, also the local and global research communities. While appearing to give good results in many cases, probabilistic modeling and visualization methods share the same disadvantage: the lacking of evaluation metric limits their value as effective community mining approaches.

Recently, there are increasing interests in finding communities on the web, each focusing on a specific set of topics. Aside from the community discovery algo-

rithm, this task is challenging in several ways: efficiently retrieving the raw, largely unstructured data from multiple disparate web resources, e.g., home pages, mailing lists, newsletters; accurately extracting name entities and topics from the documents; and finally solving the co-reference problem of entities, that two entities with different names are actually referring to the same object. There are several ongoing projects, including the DBLife [61, 197], Microsoft Libra [198], and ArnetMiner [196]. All of the systems discover communities of connected authors, conferences and journals. However, in our own experience of exploring the systems, there are still some incorrect instances of these related entities. A related contribution in the context of recommending future collaborators based on their communities is W-RECMAS, which is an academic recommendation system developed by Cazella et al. [36]. The approach is based on collaborative filtering on user profile data of the Brazilian e-government's database, and can aid scientists by identifying people in the same research field or with similar interests in order to help exchange ideas and create academic communities. However, researchers need to post and update their research interests and personal information in the database before they can be recognized and recommended by the system, which makes the approach impractical.

3.3 Mining on Local Network

The problem of finding communities in social networks has been studied for decades. However, most of the proposed approaches require knowledge of the entire graph structure. This constraint is problematic for networks which are either too large to know completely or dynamic, e.g., the WWW. In spite of these limitations, finding local community structure would still be useful, albeit confined by the little accessible information of the huge network in question. For example, we might like to quantify the local communities of either a particular webpage given its link structure in WWW, or a person given his friend network in Facebook. In this section, we first define the research problem of finding local communities in a network, then focus our efforts on reviewing existing algorithms.

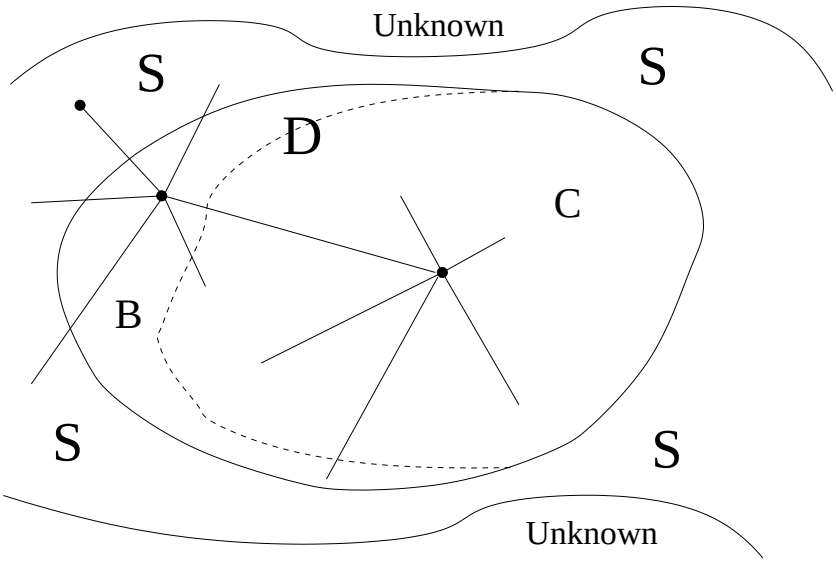


Figure 3.1: Local Community Definition

3.3.1 Problem Definition

Generally speaking, local communities are densely-connected node sets that are discovered and evaluated based only on local information. Suppose that in an undirected network G (directed networks are usually transformed to undirected ones first), we start with perfect knowledge of the connectivity of some set of nodes, i.e., the known local portion of the graph, which we denote as D. This necessarily implies that we also have limited information for another shell node set S, which contains nodes that are adjacent to nodes in D but do not belong to D (note "limited" means that the complete connectivity information of any node in S is unknown). In such circumstances, the only way to gain additional information about the network G is to visit some neighbour nodes s_i of D (where $s_i \in S$) and obtain a list of adjacencies of s_i. As a result, s_i is removed from S and becomes a member of D while additional nodes may be added to S as neighbours of s_i. This typical one-node-at-one-step discovery process for local community detection is analogous to the method that is used by web crawling systems to explore the WWW. Furthermore, we define two subsets of D: the core node set C, where any node $c_i \in C$ have no outward links, i.e., all neighbours of c_i belong to D; and the boundary node set B, where any node $b_i \in B$ have at least one neighbour in S. Figure 3.1 shows

node sets D, S, C and B in a network. Similar problem settings can be found in [14, 15, 51, 133], however, the metrics used to discover and evaluate the local community are different, as reviewed below.

3.3.2 Local Community Mining Approaches

Clauset has proposed the local modularity R [51] for the local community detection problem. R focuses on the boundary node set B to evaluate the quality of the discovered local community D.

$$R = \frac{B_{in\_edge}}{B_{out\_edge} + B_{in\_edge}} \tag{3.6}$$

where B_{in_edge} is the number of edges that connect boundary nodes and other nodes in D, while B_{out_edge} is the number of edges that connect boundary nodes and nodes in S. In other words, R measures the fraction of those "inside-community" edges in all edges with one or more endpoints in B. Intuitively, a community with a sharp boundary would have few connections from its boundary to the unknown portion of the network, while having a greater proportion of connections from the boundary back into the local community. Therefore, the community D is measured by the sharpness of the boundary given by B. Additionally, this measure is independent of the size of the enclosed community.

Similarly, Luo et al. later proposed the modularity M [133] for local community discovery. Instead of measuring the internal edge fraction of boundary nodes, they directly compare the ratio of internal and external edges.

$$M = \frac{number\ of\ internal\ edges}{number\ of\ external\ edges} \tag{3.7}$$

where "internal" means two endpoints are both in D and "external" means only one of them belongs to D. An arbitrary threshold is set for M so that only node sets that have $M \geq 1$ are considered to be qualified local communities. M is strongly related to R. Consider a candidate node set D where every node in D has external neighbours, thus we have $|C| = 0$ and $B = D$, which means $B_{in\_edge} = internal\ edges$ and $B_{out\_edge} = external\ edges$. The threshold $M \geq 1$ is equivalent to $R \geq 0.5$.

It is straight-forward to identify local communities with the R or M metric. Given a starting set D, in every step we merge one node into D from S, which

increases the metric score the most, breaking ties randomly, then we update D, B and S. This process is repeated until all nodes in S give negative value if merged in D, i.e., $\Delta R < 0$ or $\Delta M < 0$. For large networks, the running time of this method will be dominated by the time-consuming network structure information retrieval. Therefore, the algorithm complexity is linear in the size of the explored subgraph, $O(k)$, if k nodes have been merged into the local community. In the local community discovery process, it is possible that one node is qualified to be in the community when it is small at the beginning, but after the community grows to a certain size, this node may be no longer a qualified member. To solve this problem, Luo et al. [133] added a "deletion" step in the discovery process to remove any node that would decrease the M measure after the merging stops. If the starting node is removed by this step, they consider no local community exists for this node.

Bagrow et al. proposed an alternative method to detect local communities [15], which spreads an l-shell outward from the starting node n, where l is the distance from n to all shell nodes. The intuition of their metric is similar to Equation 3.7: for each l level, the number of edges connecting from the nodes on the l^{th} shell to the nodes on the $(l+1)^{th}$ shell K_{l+1}^{l} is recorded. The algorithm keeps merging the l shell nodes into the community until $K_{l+1}^{l} < \alpha$, where α is a given parameter. The performance of this approach highly depends on the parameter l and the starting node because the result communities could be very different if the algorithm starts from border nodes instead of cores. The other parameter α is also critical, e.g, if α is set to a small number, the algorithm may merge many more nodes than necessary before it stops.

Bagrow later proposed the "outwardness" metric Ω to measure local structure [14]. The "outwardness" $\Omega_v(D)$ of node $v \in B$ from community D:

$$\Omega_v(D) = \frac{1}{degree_v}(degree_v^{out} - degree_v^{in}) \tag{3.8}$$

In other words, the outwardness of a node is the number of neighbours outside the community minus the number inside, normalized by its degree. Thus, $\Omega_v(D)$ has a minimum value of -1 if all neighbours of v are inside D, and a maximum value of $1 - \frac{2}{degree_v}$, because any v has at least one neighbour in D. Since finding a

community corresponds to maximizing its internal edges while minimizing external ones, at each step, the node with the smallest Ω is merged. This metric works fine to find the most related node for the current community D, but it lacks appropriate stopping criteria since the algorithm can always find a node with the smallest Ω. The author bypassed this problem by using arbitrary thresholds, however, the threshold parameter would be extremely difficult to be accurate when it comes to real world dynamic and huge networks.

Recently, Xu et al. proposed another similarity metric between a node pair (i, j) for their approach SCAN [221], we call this metric S:

$$S_{ij} = \frac{|N_i \cap N_j|}{\sqrt{|N_i| * |N_j|}} \tag{3.9}$$

where N_i is the neighbourhood of node i, including i itself and all nodes connecting to i. Therefore, metric S normalizes the number of common neighbours by the geometric mean of sizes of two neighbourhoods in order to compare the neighbourhood structure of the two vertices in question. Since this metric only requires local information such as the neighbourhoods of the nodes, we classify it as a local community detection metric, although it was originally proposed to find global communities in networks.

3.4 Community Evaluation

As we have reviewed, many community mining algorithms do excellent jobs of recovering known communities both in artificially generated random networks and in real-world examples. However, in practical situations these algorithms will normally be used on networks in which the communities are not known, i.e., there is no ground truth for the mining problem. This leads to a new problem: how do we know whether the communities found by the algorithms are good ones? Community mining algorithms can produce communities even for completely random networks which have no meaningful community structure at all, how to measure the structure that is found for these "structureless" networks? How can we compare between different community results to find the "best"ones for a given network?

These questions show the necessity of an evaluation method for community mining approaches.

3.4.1 Global Community Evaluation

To address the evaluation problem for community mining approaches, researchers have typically resorted to human knowledge and ad hoc assessment [33, 34, 125, 126, 193]. They applied their approaches on real world databases and used common sense to validate whether the extracted communities are accurate or not. For example, several researchers [33, 34, 193] applied their algorithms on the DBLP data [25], which provides bibliographic information on major computer science journals and conference proceedings, then evaluated the performance by showing that the result is making sense, e.g., ICDE is found in the same community with VLDB and SIGMOD. Li et al. [125, 126] focused on finding name entity communities extracted from Google search result pages with the key name entity as the query term, then evaluated them by common knowledge about these entities, e.g., Hillary Clinton, Bill Clinton and Monica Lewinsky are in the same community in their result, as expected. Evaluating community mining results on real world data by common knowledge is interesting, but this method is not convincing enough to show the generality of the proposed approaches. Moreover, it is nearly impossible to compare the results between different algorithms, if they are close to each other.

To deal with this problem, Newman and Girvan [155] proposed a measure of the quality of a particular division of a network based on a previous measure of associative mixing [147]. They called it the *modularity*. The modularity measure is based on the intrinsic linking structure of the graph. Consider a particular division of a network into k communities, we define a $k \times k$ symmetric matrix e, of which each element e_{ij} is the fraction of all edges in the network that link vertices in community i to vertices in community j. The matrix trace $Tr(e) = \sum_i e_{ii}$ gives the fraction of edges in the network that connect vertices in the same community i, and clearly a good community division should have a high value of this trace. However, the trace alone is not a good indicator of the quality, since placing all vertices in one single community would give the maximal value 1 while giving no information of

the community structure at all. The authors further defined the row sum $a_i = \sum_j e_{ij}$ to represent the fraction of edges that has at least one end in community i. So, a_i^2 is the expected fraction of edges within community i if the edges were distributed randomly on the network. Thus, the modularity measure is defined by:

$$Q = \sum(e_{ii} - a_i^2) = Tr(e) - \|e^2\| \tag{3.10}$$

where $\|x\|$ indicates the sum of the elements of the matrix x. In other words, the modularity measures the fraction of the edges in the network that connect vertices of the same type, i.e., within-community edges, minus the expected value of the same quantity in a network with the same community division but random connections between the vertices [85]. If the number of with-in community edges is no better than random, we get $Q = 0$. Values of Q that are close to 1, which is the maximum, indicates strong community structure. Q typically falls in the range from 0.3 to 0.7 [85], and high values are rare.

The modularity approach, first proposed in [151, 155] and since pursued by a number of authors [52, 63, 90, 91, 152, 153], has been proved highly effective in practice for community evaluation [57]. However, there are three major problems for the Q measure. First, the modularity requires information of the entire structure of the graph, which is problematic for huge networks like the WWW. To solve this problem, Clauset [51] proposed a measure of local community structure, called *local modularity*, for graphs which lack global knowledge. Secondly, recent research showed that modularity-based methods have a resolution limit and may fail to identify communities smaller than a certain scale [73]. Possible solutions include recursive algorithms based on modularity optimization [180]. Finally, as pointed out by Scripps et al. [186], the modularity only measures existing links on the network, but does not explicitly consider the absent links between two nodes in the same community. In other words, the modularity only measures how good the discovered community structure fits the existing links (connected vertices should be in the same community), but fails to measure how good the structure fits the absent links (disconnected vertices might not be necessarily in the same community). Therefore, it is adequate to compare algorithms on the same network, but

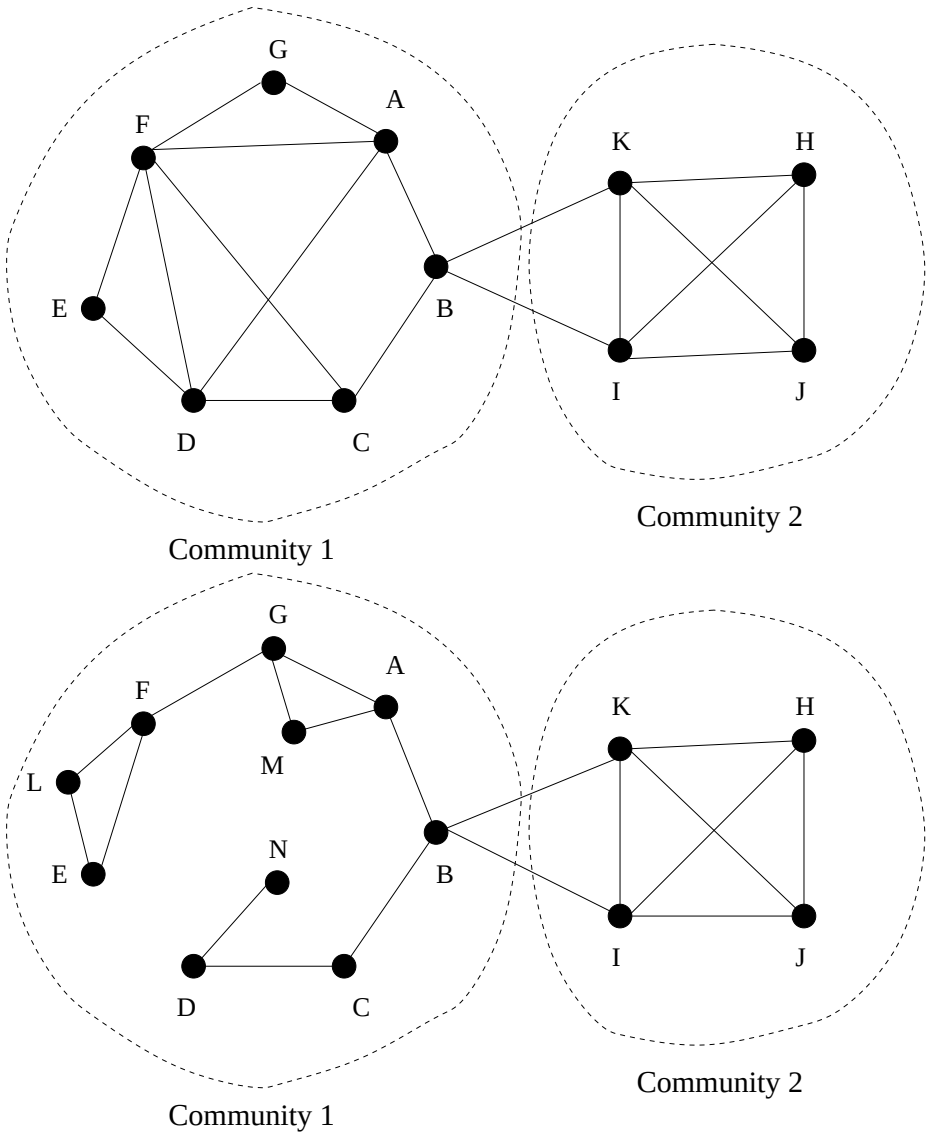


Figure 3.2: Network Community Example for Modularity Measure

not for comparing one network to another. For example, the two networks shown in Figure 3.2 have the same number of edges and the same Q, which is 0.360, but the intuition is that the community division in the second network is worse, since it has more disconnected node pairs within community 1, which is not considered by the Q measure. Therefore, modularity fails to compare the community structure between different graphs. To solve this problem, Scripps [186] proposed two ratios p and q, measuring the fraction of links within communities and absent links between communities, respectively. The drawback of their method is that the inter-

pretation is not clear since there are two measures: a mining result can have higher p but lower q than the other, which makes it hard to compare the quality of different community structures. In Chapter 5, we propose a new evaluation measure to increase the accuracy of community detection, which not only solves this particular problem by taking unrelated node pairs (defined by domain knowledge) into consideration, but also makes it possible to compare the community structure quality between different graphs.

3.4.2 Local Community Evaluation

As we have discussed in Section 3.3.1, the network information that can be used to discover and evaluate local communities is very limited. More specifically, for one local community, the only information we have are the network structures within the community and the connections from community nodes to other parts of the network. Therefore, local community evaluation metrics are usually simple. We have introduced several metrics in Section 3.3.2, including R [51], M [133], Ω [14] and S [221]. Among them, M measures the "outwardness" of the whole community, while R measures that of the boundary node set and Ω focuses on that of single nodes on the boundary. S evaluates the local community from a different perspective by quantifying the relationships between every pair in the same community, which is calculated based on the similarity between their neighbourhoods. Existing local community algorithms usually greedily maximize one of these evaluation metrics in order to discover a local community.

Another popular method for local community evaluation is by human knowledge or synthetic "ground truth". Researchers either extract communities from real world networks and manually verify the results, or apply mining algorithms on computer generated data and compare the results with the generated "ground truth". For example, a network among products of Amazon.com, where each node represents an item and each connection indicates the two items have been frequently purchased together, has been used for local community discovery [51, 133] and the generated communities consist of DVDs, CDs and books, thus relations are easy to justify. Synthetic data are also heavily used [14, 51] since researchers could just

create “ground truth” to evaluate their experimental results.

3.5 Mining Overlapping Communities

The goal of overlapping community mining is to find communities in the network with the assumption that one actor can belong to one or more communities. This situation is common in the real world, but most community mining algorithms would fail under this circumstance, e.g., spectral clustering or hierarchical clustering based on greedy modularity maximization. Note that mining overlapping communities is a problem for both global or local social networks, however, it is more common to identify overlaps in global networks, since local networks usually provide limited structural information which is only adequate for identifying one community.

3.5.1 Common Approaches

Generally speaking, there are two common ways to detect overlapping communities in a network. One natural idea is to first globally partition the network and then locally expand the discovered communities to locate overlapping components. Wei et al. [212] first partition the network into seed groups of the overlapping community structure using existing spectral clustering method. A locally optimal expansion process is then applied to greedily optimize Newman’s Modularity Q measure. Li et al. [125] generate overlapping communities of named entities, i.e. people names, organizations, from web contents and text documents. They first collect a set of relevant documents on the named entity by submitting the entity as a query to Google and extracting the text from search result pages. A named entity graph can be generated by connecting all entity pairs that have co-occurred in the sentences. They then locate cores of communities by merging triangle, thus a community expands predominantly by triangles sharing one common edge. Since one node can be a part of multiple different triangles, it may belong to different communities of corresponding triangle groups. Similarly, Baumes et al. [19] initialize community cores using the Link Aggregate (LA) Algorithm and then refine the peripheries by an Iterative Scan (IS) procedure.

The other main research direction for this problem is based on fuzzy clustering. Zhang et al. [233] combine modularity and a fuzzy c-means clustering algorithm to identify overlapping communities. Their method combines a generalized modularity function, spectral mapping, and fuzzy clustering technique. The nodes of the network are projected into a d-dimensional Euclidean space which is obtained by computing the top d nontrivial eigenvectors of the generalized eigensystem $Ax = tDx$. Then the fuzzy c-means clustering method is applied onto this space based on the general Euclidean distance to cluster the data points. By maximizing one generalized modular function for varying d, the number of clusters can be obtained. The final overlapping community assignment matrix determines the final clusters with a designated threshold. Nepusz et al. [144] argued that every node is allowed to belong to multiple communities with different membership degrees, represented by a single real value $u_{ki} \in [0, 1]$ for each node i and community k. Since the $U = [u_{ki}]$ matrix encodes the membership values in a compact form, they define the similarities of the vertices as $S = U^T U$. The similarities are then optimized using gradient-based constrained optimization methods in order to make connected vertices similar and disconnected nodes dissimilar. Thus, the fuzzy community detection problem becomes a constrained optimization problem, given parameters such as the number of the clusters. Nicosia et al. [162] extend the modularity definition to evaluate the overlapping community structure in the more general case of directed networks, by redefining Newman's modularity function for directed graphs using out-degree and in-degree of each node. Similar to Nepusz et al.'s work, they also model overlapping community structures by using an array of "belonging factors" $[\alpha_{i,1}, \alpha_{i,1}, ...\alpha_{i,k}]$ to represent how strong one node i belongs to community k with the requirement $\sum_{k=1} \alpha_{i,k} = 1$. Thus the modularity of having a connection between i and j in community k is weighted by the belonging factors of i and j to that community.

3.5.2 Recent Approaches

Recently, Palla et al. [170] proposed the CFinder system to partition complex networks into overlapping communities. Since a typical community usually con-

sists of several complete subgraphs that tend to share many of their nodes, they define a community, or more precisely a k-clique community, as a union of all k-cliques (complete subgraphs (cliques) of size k) that can be reached from each other through a series of adjacent k-cliques, where "adjacent" means two k-cliques sharing $k-1$ nodes and k is a given parameter as clique size. There are some parts of the whole network that are not reachable from a particular k-clique, but they potentially contain their own k-clique communities. Therefore, a single node can belong to several communities. They also claim that in most cases, relaxing this definition, e.g., by allowing incomplete k-cliques, is practically equivalent to decreasing k. A heuristic algorithm is then proposed to first locating all cliques of the network and then identifying the communities by carrying out a standard component analysis of the clique-clique overlap matrix. They later apply their method on multiple datasets to capture patterns in networks of the collaboration between scientists and the calls between mobile phone users [169].

Gregory proposes the CONGA algorithm [88] based on the betweenness centrality measure [155]. The "betweenness" of edge e is defined as the number of shortest paths, between all pairs of vertices, that pass along e. A high betweenness means that the edge acts as a bottleneck between a large number of vertex pairs and suggests that it is an inter-cluster edge. Gregory extends the approach for overlapping community detection by adding a second operation: splitting a node. A best split is found for each node with the maximum betweenness after split. Gregory later improves his approach on running speed and proposes a heuristic algorithm named CONGO [89]. In this approach, an additional parameter h, which is the parameter of the local region of the node in question, is taken when calculating the best split option. This changes tremendously in reducing the running time, from $O(m^2n)$ to $O(n \log n)$, where m is the number of edges and n is the number of vertices. He also shows that the second algorithm CONGO ($h = 2$) provides the same level of performance as CFinder on synthetic networks, and is ideally suited for finding overlapping communities in very large networks. For smaller networks, h can be increased for more accurate results.

While all of the above methods successfully detect overlapping communities,

some major problems exist. Most methods do not consider outliers, which are nodes that are weakly connected to the community and belong to no communities, thus many outliers would be classified as community members, i.e., they force outliers into existing clusters. Additionally, the fact that they intentionally focus on overlapping communities makes them find or force overlap even for data without such structure. More importantly, many approaches require parameters that are not only difficult to determine but also very sensitive, e.g., number of communities [88, 233], community density [125, 170], or size of a local community region [89].

3.6 Dynamic Community Discovery

In analyzing social networks, one property has been largely ignored, which is the fact that these networks tend to change dynamically. Indeed, the ongoing changes of a network and its possible causes may be among the most interesting properties to observe [199]. Moreover, the rapid growing electronic networks, such as emails, blogs, friendship sites and mobile networks, provide an abundance of dynamic social network data to support such explicitly dynamic analysis. When dealing with dynamic social networks, most studies would either analyze a snapshot of a time point, or an aggregation of all the network information over a large time window. Both approaches may miss important tendencies of these dynamic networks.

Typically, a dynamic community mining problem is defined as follows [22, 199]: A social network is first modeled as an undirected graph $G = (V, E)$. To model dynamic interactions in G, there is a set $X = \{i_1, ..., i_n\}$ of individuals (nodes), and a sequence $H =< P_1, P_2, ..., P_T >$ of observations. Each P_t is a collection of non-empty and pairwise disjoint sets. The interpretation is that, for a time step t, the individuals were observed interacting with each other. Some individuals may not be observed at all at certain times. In other words, all nodes were assumed to be in the graph from the beginning, even though they may only be observed in the very end of the time window. The dynamic community structure thus is determined by the evolving connections in the network.

The evolution of communities in large social networks, such as membership,

growth and disbandment, is becoming increasingly prominent especially for social networking sites such as MySpace and Facebook. By monitoring friendship links and community membership on a social network site, and co-authorship and conference publication in a bibliography database, Backstrom et al. [13] studied how the evolution of these communities related to properties such as the structure of the underlying social networks. They find that the propensity of individuals to join communities, and of communities to grow, depends in subtle ways on the underlying network. For example, the tendency of an individual to join a community is influenced not just by the number of friends he or she has within the community, but also crucially by how those friends are connected to one another [13]. Falkowski et al. proposed an interactive visualization tool to analyze the dynamics of subgroups in an online student network [67]. Decision-tree techniques are also used to identify the most significant structural determinants of candidate properties. Sarkar et al. [182] proposed to associate each entity with a point in a Euclidean latent space for modeling relationships that change over time and then optimize the global likelihood to generate the communities. Berger-Wolf et al. [22, 199] proposed a framework for identifying dynamic communities. They proved that finding optimal grouping is NP-hard and proposed heuristics trying to maximize the amount of "similarity" preserved from one time step to the next. Sun et al. [192] proposed an alternative method, GraphScope, to discover communities in large time-evolving graphs without any parameter. They focus on bipartite graphs, e.g., senders and receivers in an email dataset, and treat source and destination nodes separately. Each graph in the time stream is encoded based on the Minimum Description Length (MDL) principle and find the best partition by minimizing the encoding cost. In particular, their approach adapts to the dynamic environment by automatically finding the communities and determining good change points. More specifically, if the partition encodings do not change much over time, consecutive snapshots have similar descriptions and can be grouped together into a time segment. Whenever a new social network snapshot that cannot fit well into the old segment appears, GraphScope introduces a change point, and starts a new segment at that time-stamp. Those change points automatically detect drastic discontinuities in time [192].

3.7 Community Discovery with Multiple Relations or Attributes

Most of the existing methods on community mining and social network analysis assume that there is only one single social network, representing a relatively homogeneous relationship, such as the page linkage in WWW. However, in reality, there are always a variety of relations that co-exist and each relation plays a different role for a particular case. Therefore, mining such multi-relational social networks by assuming only one relation may miss much valuable hidden information. We need to quantify the significance of different relations in order to discover true communities in such circumstances.

The relation identification problem here for community mining is fundamentally related to the feature selection problem [64] in Machine Learning, which has received much attention as part of the process of classification and clustering. Feature selection is a process that selects and optimizes a subset of original features, measured by an evaluation criterion [131]. It aims at finding a linear combination of the original features that can better represent the intrinsic structure of a data set based on given criterion. Unfortunately, it requires explicit vector representation of the objects, which is hardly available for relational networks. There are many notable feature selection methods in Machine Learning, such as Principal Component Analysis (PCA) [112] and Linear Discriminant Analysis (LDA) [138].

Although feature selection for machine learning and data mining has been well studied, little work on relation selection for community mining has been done. Cai et al.'s work [33, 34] is among the earliest research efforts on this topic. Given some examples with different community labels, Cai et al. proposed a regression-based algorithm, which tried to find a combined relation which makes the relationship between intra-community example as tight as possible and, at the same time, the relationship between inter-community examples as loose as possible. A possibility matrix is built as target relation matrix based on the information an domain expert provides. (The value can be 0 or 1 if the expert is certain, and can be a percentage number if the user is uncertain whether the two objects belong to the same

community or not.)

$$\tilde{M}_{ij} = Prob(entity\ i\ and\ j\ belong\ to\ the\ same\ community) \tag{3.11}$$

The algorithm aims at finding a linear combination of the existing relations to optimally approximate the target relation matrix. Let $a = [a_1, a_2, ..., a_n]^T \in R^n$ denote the combination coefficients for different relations. Since M is symmetric, ans $m(m-1)/2$ dimensional vector v can be used to represent it. Thus, the approximation problem can be characterized as the following optimization problem:

$$a^{opt} = argmin_a \|\tilde{v} - \sum_{i=1}^{n} a_i v_i\|^2 \tag{3.12}$$

Since this unconstrained least squares solution might not be a satisfactory solution, the author normalized all the weights on the edges in the range $[0, 1]$ and put a constraint $\sum_{i=1}^{n} a_i^2 \leq 1$ on the above objective function. Such a constrained regression problem is called Ridge Regression [97] and can be solved by some numerical methods [26]. If the examples provided by the expert belong to only one community, the authors propose a similar algorithm based on the minimum cut on the graph [33].

Cai et al.'s work is among the earliest attempts to discover communities in a multi-relation social network. Such networks are quite common in the real world and need new mining algorithms to dynamically combine influences from multiple relations to form communities. However, it is difficult for experts to provide high quality examples for Cai's algorithm. Even they do, the learning performance of the algorithm is limited by the size and makeup of the examples.

While most proposed community mining methods only use network structure data, in many applications more informative graphs with attributes, which are represented as feature vectors of vertices, are provided. To detect communities in such networks, Moser et al. proposed the CoPaM algorithm to find all maximal dense and connected subgraphs with similar feature values [142]. Compared to their work, this thesis focuses more on the network structure and relations, rather than attributes of vertices in the network.

3.8 Entity Ranking

Thanks to Google's big success as a search engine, the most famous and profitable community mining task may be that of entity ranking, whose main objective is to order or prioritize sets of objects within the network based on the relations among them and the overall linking structure. The problem of ranking these entities has been studied mainly in two fields, in particular, computer science and sociology, which have developed quite different approaches as we now describe.

3.8.1 Sociological Approaches

In the domain of Social Network Analysis (SNA), entity ranking tasks focus on ranking individuals in a given social network in terms of a measure of the importance, referred to as *centrality*. Measures of centrality have been the subject of research in the SNA community for decades [210], and focus on aspects of the local or global network structures as seen from the location of the given object in the graph. Centrality measures on the graph vertices are first introduced in the analysis of social networks [28, 76], but at present they are commonly used in the analysis of many real world networks. Many centrality measures have been studied in the literature [28, 78]; among them the degree centrality [76] and the eigenvector centrality [28, 29] have been most deeply investigated.

Degree Centrality

The simplest and perhaps the most intuitively obvious conception of a measure of the node centrality in a network is some function of the degree of a point, which is simply the count of the number of other nodes that are adjacent and in direct contact or connection to the current one. Unfortunately, many proposed degree-based measures are often unnecessarily complicated [76] and they only measure the local network structure.

Eigenvector Centrality

Eigenvector centrality is based on the intuition that connections to high-scoring nodes should contribute more to the score of the node in question, than equal con-

nections to low-scoring nodes. Let A be the adjacency matrix of the network and $A_{i,j} = 1$ if node i and node j are connected and 0 otherwise. Let x_i denote the score of the i^{th} node. For the i^{th} node, the centrality score is defined to be proportional to the sum of the scores of all nodes that connect to it:

$$x_i = \frac{1}{\lambda} \sum_{j=1}^{N} A_{i,j} x_j \tag{3.13}$$

where N is the total number of nodes and λ is a constant. The equation can be rewritten as

$$X = \frac{1}{\lambda} AX$$

or

$$AX = \lambda X$$

which is the well known eigenvector equation. The i^{th} component of this eigenvector gives the centrality score of the i^{th} node in the network.

In general, there will be many different eigenvalues λ and eigenvector solutions. However, the additional requirement that all the entries in the eigenvectors be positive implies that only the greatest eigenvalue results in the desired centrality measure (proved by the Perron-Frobenius Theorem) [154]. In the link mining research field, there are many variants and applications of the eigenvector centrality measure, among which Google's PageRank perhaps is the most famous one.

3.8.2 Computer Science Approaches

In the context of web information retrieval, entity ranking leads to a crucial problem for search engines: how to rank documents that are connected to each other by hyperlinks. At first, the occurrence of a search phrase within a document is one major factor within ranking techniques of any search engine. To achieve a more accurate ranking, the concept of link popularity was developed to take into consider the structural context. Following this concept, the number of links that connect to a document measures its overall importance. Therefore, a web page is generally more important if many other web pages link to it. However, these measures are vulnerable to noise, for example, webmasters can easily bias them by creating masses

of inbound links or repeatedly adding keywords for their doorway pages. Search engines apply a variety of different ranking technique to solve these problems; the PageRank [167] and HITS [117] algorithms are the most notable approaches.

PageRank

The intuition of the PageRank algorithm is that a document is considered more important the more other documents link to it, however, those incoming links do not count equally: a document is ranked higher if other high ranking documents link to it. In other words, the PageRank score of any document is always determined recursively by the PageRank score of other documents, which is based on the linking structure of the whole network. PageRank can be loosely interpreted as a probability distribution used to represent the likelihood that a web surfer randomly clicking on links to visit any particular page and occasionally jumps to a new page to start another traversal of the link structure. The PageRank computations usually require several iterations until the approximate values reflect closely enough the theoretical true value. Although this approach seems to be very complex, Page and Brin managed to put it into practice by a relatively trivial algorithm as the following.

$$PR(u) = (1-d) + d(\frac{PR(p_1)}{L(p_1)} + ... + \frac{PR(p_n)}{L(p_n)})$$

or

$$PR(u) = (1-d)/N + d(\frac{PR(p_1)}{L(p_1)} + ... + \frac{PR(p_n)}{L(p_n)})$$

where $PR(u)$ is the PageRank of page u, $PR(p_i)$ is the PageRank of pages p_i that link to page u, $L(p_i)$ is the number of outbound links on page p_i, N is the number of the documents in collection and d is a damping factor, which is set between 0 and 1, to control convergence of the computation.

PageRank can be described as a model of user behaviour, where a web surfer clicks on links at random with no regard towards the page's content [31]. At first, the page's PageRank is the probability that this random surfer visits it. The chance that the surfer clicks on one link is solely decided by the number of links on that page. Thus, the probability for the random surfer reaching one page is the sum of probabilities of following the links that connect to this page. The surfer can get

bored sometimes and stop following the link, and jumps to another page at random. Therefore, the probability is reduced by the damping factor d, which is the probability that the surfer will keep clicking and is generally set around 0.85 [31]. Lawrence Page and Sergey Brin have published the two different equations of their PageRank algorithm in different papers [31, 167]. They do not differ fundamentally from each other. However, the second version's Pagerank of a page is the exact probability for a surfer reaching that page after clicking on many links. The algorithm then forms a probability distribution over all web pages so that the sum of all pages' PageRank equals one. On the other hand, the first version of PageRank is an expected value for the random surfer to visit a page, i.e., when we restart this procedure as often as the web has pages. For example, if the web has 1000 pages and a page had a PageRank score of 3, then we would expect the surfer to reach that page for three times on average if he restarts for 1000 times.

The main disadvantage of PageRank is that it favors older pages, because a new page, even a very popular one or very related to the query, will not have many inbound links unless it is part of an existing site. It also suffers from link spamming and other similar *spamdexing* [1] methods, which deliberately modify HTML pages to increase the chances of these pages being placed close to the beginning of search engine results. Since PageRank focuses on page ranking on WWW, which is a single relation network, it is also not appropriate for our community mining needs.

The PageRank algorithm has many variations and extensions. For example, linguists have used PageRank to automatically rank WordNet synonym sets according to how strongly they possess a given semantic property, such as positivity or negativity [66]; A web crawler can also use PageRank as one of the importance metrics to determine which URL to visit next during a crawl of the web [49].

HITS

Jon Kleinberg's Hypertext Induced Topic Selection (HITS) algorithm [117] assumes a slightly more complex process than PageRank, by modeling the web as being composed of two types of web pages: *hubs* and *authorities*. Hubs are pages

[1] http://en.wikipedia.org/wiki/Spamdexing

that link to many authoritative pages while authorities are pages that are linked to by many hubs. Each page is assigned hub and authority scores. An authority score estimates the value of the content of the page, computed as the sum of the scaled hub values of the pages that point to that page. A hub score estimates the value of its links to other pages, computed as the sum of the scaled authority values of the pages it points to. These scores are, like PageRank, calculated by an iterative algorithm that updates the scores of a page based purely on the linkage of the documents of the web, thus, the hub and authorities scores are also steady-state distributions of the respective random processes. However, different from PageRank, HITS has two separate random walks: one with hub transitions and one with authority transitions, on a corresponding bipartite graph of hubs and authorities [158, 178].

After HITS and PageRank were introduced, a number of algorithms have been proposed based on them. In order to yield more accurate PageRank results for search engines, Haveliwala [98] and Jeh and Widom [109] proposed Topic-Sensitive PageRank to precompute a set of biased PageRank vectors, which emphasize the effect of particular representative topic keywords to increase the importance of certain web pages. Those vectors are used to generate query-specific importance scores for pages at query time. Bharat and Henzinger [23] and Chakrabarti et al. [37] improved HITS by exploiting web page content to weight pages and links based on their relevance. Cohn and Chang [53] propose an analogue to HITS based on maximizing the likelihood of probabilistic models. To compare and combine the two algorithms, Ding et al. [59] introduced a unified framework to include both PageRank and HITS; Ng et al. [157, 158] analyzed the stability of PageRank and HITS to small perturbations in the linking structure; Cohn and Hofmann [54], and Richardson and Domingos [179] proposed probabilistic models incorporating both content and link structure, inspired by HITS and PageRank, respectively.

The common goal of PageRank, HITS and all other related algorithms discussed above is to achieve a global ranking of entities in a static graph connected via a specified relation. Alternatively, other variations of entity ranking include approaches that rank entities relative to one or more relevant entities in the graph and methods that rank objects over time in dynamic graphs. Jeh and Widom's SimRank [108]

computes a purely structural score that is independent of domain-specific information. The SimRank score is a structure similarity measure between pairs of pages in the web graph with the intuition that two pages are similar if they are related by similar pages. Unfortunately, SimRank is very expensive in computation since it needs to calculate similarities between many pairs of objects. According to the authors, a pruning technique is possible to approximate SimRank by only computing a small part of the object pairs. However, it is very hard to identify the right pairs to compute at the beginning, because the similarity between objects may only be recognized after the score between them is calculated. Similar random walk approaches have been used in other domains for the same problem. For example, the Mixed Media Graph [171] applies a random walk on multimedia collection to assign keywords to the multimedia object, such as images and video clips, but a similarity function for each type of the involved media is required from domain experts. He et al. [99] propose a framework named MRBIR using a random walk on a weighted graph from images to rank related images given an image query. Sun et al. [193] detect anomaly data for datasets that can be modeled as bipartite graphs using the random walk with restart algorithm. Several other systems use random walk and related methods for keyword search and tuple ranking in databases: ObjectRank [103] applies random walk algorithm on the tuples following links in the citation network and the database schema; RelationalRank [80] builds a data attribute graph based on tuples in the query result and applies random walk on these attributes to rank them; BANKS system [3] first creates trees of tuples that are found as query results, then ranks these trees based on the global ranking of tuples and weights of connections between these tuples. Recent work by Tong et al. [202] proposed a fast solution for applying random walk with restart on large graphs, to save pre-computation cost and reduce query time. White and Smyth [217] define and evaluate a host of metrics to compute the similarity between a given entity and one or more reference entities in a graph. On the other hand, dynamic graphs such as email networks and telephone call records that capture event data and change over time bring new challenges for ranking entities. The time ordering of events and link structure change over a given time interval limit the use of static ranking

methods. For this problem, O'Madadhain et al. [165, 166] discussed a series of algorithmic properties for dynamic entity ranking and proposed a ranking algorithm based on potential flow that satisfies specific requirements for dynamic networks.

In our research, we are particularly interested in ranking entities for a given object in a multiple relational network, and use the result to understand the explicit community structure of the graph.

Chapter 4

Research Problem Statement

In Chapter 3, we have reviewed the state-of-the-art works on Community Mining, which is an emerging research area with many interesting problems. In this chapter, we define the questions this dissertation tackles, why they are unanswered and why they are worthwhile to answer. For the topics (see Figure 4.1), we are particularly interested in identifying and detecting different kinds of communities (overlap or non-overlap) in homogeneous networks with various network assumptions (global network or local network), while most of the previous approaches only focus on finding non-overlap communities in a global network. Moreover, we are also interested in ranking entities in heterogeneous networks to understand relations between social actors.

4.1 Community Mining with Domain Knowledge

To solve the problem of mining communities in a social network, where global network information can be fully accessed, various relation-based methods have been developed, such as the spectral clustering approaches [60, 187, 218] and modularity-based algorithms [52, 155]. The identification of such community detection relies on some notion of clustering or density measure, which defines the communities that can be found. However, previous methods always apply the same structural measure on all kinds of networks, e.g., biological network, bibliographical network, social network, hyperlink network and so on, despite the distinct dissimilar characteristics of these networks. These significant network character-

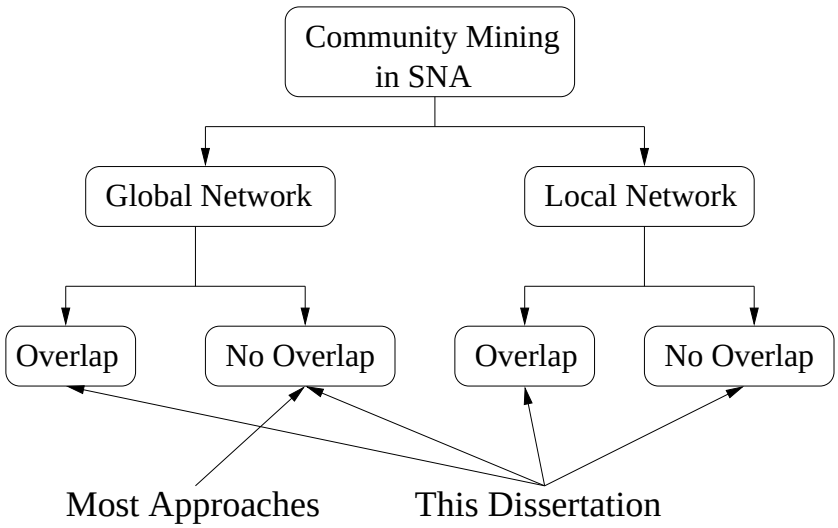


Figure 4.1: The Topic Hierarchy of Community Mining in SNA

istics may be hard for community mining algorithms to observe directly, but can be summarized and explained by experts in the corresponding field and provided to the mining process as domain knowledge. The accuracy of community mining approaches would definitely improve by taking extra domain knowledge into consideration.

To solve this problem, in Chapter 5 we present a new community mining measure, Max-Min Modularity, which considers both connected pairs and criteria defined by domain experts in finding communities. We use it to estimate the compatibility between the discovered communities and the link structure. Generally speaking, our Max-Min Modularity compares the difference between the fraction of links that occur within communities to the fraction that would be expected to occur if the links were randomly distributed; in addition, it also considers the fraction of user-defined related node pairs within communities to the expected fraction of such pairs. Based on this measure, we specify a hierarchical clustering algorithm to detect communities in networks. More details of the algorithm and experiment results can be found in Chapter 5.

4.2 Local Community Mining

While domain knowledge can be used to improve the performance of community mining algorithms, such knowledge is not always available, even the connection structure would be impossible to access completely for some huge networks. Recall that we have defined the problem of local community mining in Chapter 3: local communities are densely-connected node sets that are discovered and evaluated based only on local network information, and the only way to acquire more structural knowledge of the network is to include one candidate node into the local community at each step and explore its neighbourhood (see Figure 3.1).

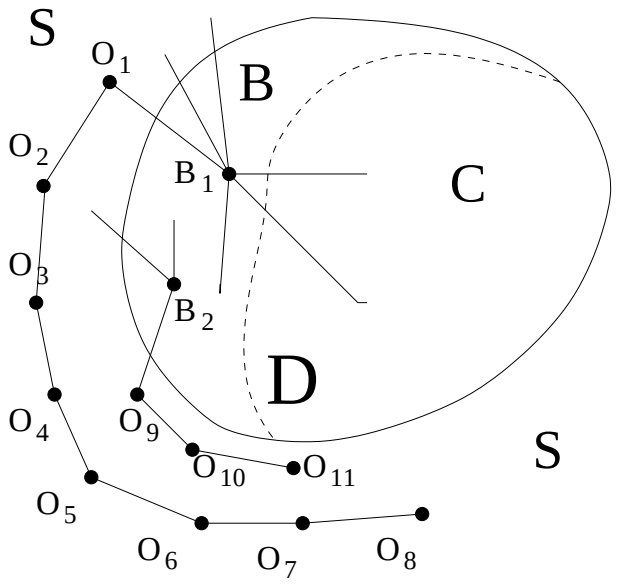


Figure 4.2: Problem of Previous Local Community Metrics

We have reviewed several metrics to discover and evaluate local communities. (See Equation (3.6) and Equation (3.7).) Indeed algorithms using these metrics are able to detect interesting communities in complex networks, however, their results usually include many outliers, i.e., the discovered communities have a high recall but low accuracy, which reduces the overall community quality. Figure 4.2 illustrates the problem for state-of-the-art local community metrics R and M. In the figure, we have a local community D, its boundary B and nodes $O_1, ..., O_{11}$, which are outliers since they are barely related to nodes in D. Without loss of generality, let us assume that all nodes in S, except O_1 and O_9, will decrease the metric score

if included in D. Now if we try to greedily maximize the metric M (Equation 3.7), all outliers (O_1 to O_8 and O_9 to O_{11}) will be merged into D one by one. The reason is that every merge of node O_i does not affect the external edge number but will increase the internal edge number by one. R is marginally better. It stops the merging on O_2 and O_{10} since further expansion would not change the in and out degrees of *boundary* nodes. However, algorithms using R would still merge any other nodes in to D as long as it connects to an equal number of nodes inside the boundary and outside the local community node set. Therefore, in addition to real members, the resulting community by R and M algorithms would contain many weak-linked outliers, whose number can be huge for some networks, e.g., the WWW.

Several techniques [14, 15, 51, 133] have been proposed to identify local community structures given limited information of a network. However, parameters that are hard to obtain are usually required. Moreover, communities discovered by these algorithms include many outliers and thus suffer from low accuracy, as we have shown in Figure 4.2. In Chapter 6, we propose a new metric, which we call L, to evaluate the local community structure for networks in which we lack global information. We then define a two-phase algorithm based on L to find the local community of given starting nodes and compare the algorithm's performance with previous methods on several real world networks. We also propose a community discovery process for large networks that iteratively finds communities based on our metric.

4.3 Overlapping Community Mining

For problems presented above, we have assumed that one node belongs only to one community, however, recent studies have revealed that network models of many real world phenomena exhibit an overlapping community structure, i.e., a node can belong to more than one community, which is hard to take into account with classical graph clustering methods where every vertex of the graph belongs to exactly one community [170]. This is especially true for social networks, where individuals can connect to several groups in the network as *hubs*. Furthermore, in real networks

we also have another node category, which belongs to no community, i.e., *outliers*. Therefore, a typical social network consists of communities, hubs and outliers. It is essential for community discovery methods to identify nodes in these three categories, since the isolation of hubs and outliers can be crucial for many applications. Unfortunately, a precise description of what *community, hub and outlier* really are would be different across various domains, or even across different networks of the same domain. Therefore, most proposed approaches [88, 89, 144, 170, 221] for overlapping community detection require the user to describe the communities they are looking for by giving parameters, e.g., community size, density range, the number of communities, etc. However, appropriate settings of these parameters are usually extremely hard to determine without tedious and repeated beforehand testing. Moreover, arbitrary parameters may over-restrain the space in which communities are found and lead to inaccurate results.

In Chapter 7, we first propose a list of requirements for an appropriate overlapping community mining metric, based on our observations of social network characteristics. After reviewing the advantages and disadvantages of existing metrics, we propose the R *(Relation)* metric to measure the similarity between any pair of entities in a social network, then show its advantages by comparison with existing metrics. We then propose our visual data mining approach ONDOCS (Ordering Nodes to Detect Overlapping Community Structure), which generates preliminary visualizations of the network in question by ordering nodes based on their reachability scores (RS) to help the user understand the network structure in order to choose appropriate parameters. Selected parameters are then used to extract communities, hubs and outliers from the network. More details of our approach and experiment results can be found in Chapter 7.

4.4 A Community Mining Application in Web Context

As we have introduced in Chapter 2, there has been a surge of interests on community mining in social networks, because accurately discovering groups in networks

can benefit myriad research fields. As an example, here we apply our understanding of community mining and adopt proposed metrics and algorithms to solve a real problem in the web information retrieval domain. By giving a solution to this problem using community mining metrics and algorithms, we show that community mining, as a promising research area, is important for many applications.

As we know, existing search engines often return a long list of search results, ranked by their relevance to the given query. Since web pages, on different aspects (meanings) of the same query, are usually mixed together, users have to go through the long list and examine titles and content sequentially to locate pages of interest to their information need. For example, when the query "jaguar" is submitted to a search engine looking for information about *the Mac system*, a user might have to sift through a large number of pages about *automobile* or *animals*. The sought for pages might be buried very deep. While the underlying retrieval model and ranking function is vital for search engines, organization and presentation of search results is also capital, and could significantly affect the utility of a search engine. However, compared with the vast literature on page ranking and retrieval, there is relatively very little research on how to improve the effectiveness of search result organization [209].

Popular search engines such as Google, Yahoo! and Bing deal with this problem by recommending refined queries to the user to focus on specific topics. For instance, after the query "jaguar" is submitted to the search engine, several queries including "jaguar animal", "jaguar cars" etc. are listed along search results as candidates for further query refinement (See Figures 4.3, 4.4, 4.5). The refinement strategy works generally well, however, its effectiveness is limited for two reasons: First, candidate queries are usually acquired from a list of most frequent queries that contain the submitted query word, according to previous search records, but some topics of interest might not be popular enough to be included in the recommended queries. For example, as shown in the figures, none of the three search engines suggest "jaguar mac" or anything similar because the Jaguar Mac Operating System is less queried compared to the search history of jaguar the car and jaguar the animal. Second, the search result for refined queries might still contain mixed information

Figure 4.3: Query Refinement by Google Search

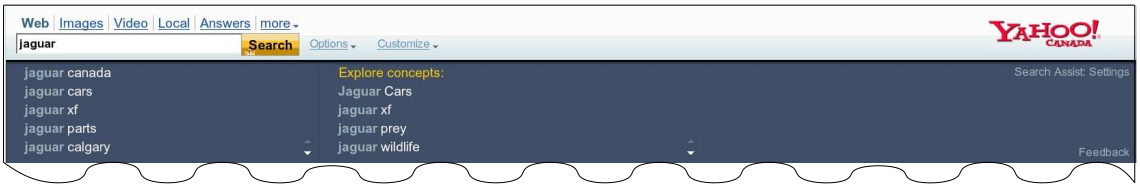


Figure 4.4: Query Refinement by Yahoo! Search

and pages on different topics. For example, the ranking list for "jaguar car" might contain pages about car sales, company business, manufacturing or new model presentations. It is possible that the result will be finally clear after several refinement attempts though, if the user does not run out of patience before that.

Although we believe that an appropriate solution to this problem is to (online) cluster search results into different groups so that users can select their required group at a glance. Previous document grouping approaches usually require high quality training data to build a classifier, which is infeasible due to the dynamic nature of the web and the need of a realtime answer. Some unsupervised clustering approaches for search engine result organization exist. However, since clustering methods always generate groups of documents, even when unnecessary. A metric is required to indicate whether page clustering is helpful for the user. Otherwise, the user would be confused when document clusters are reported for a list of search result pages which are on the same topic.

We apply community mining ideas to solve this problem. In Chapter 8, we reformulate the search result organization problem as a query sense disambiguation problem and apply a community detection algorithm to cluster pages based on discovered senses of a given query. We first parse the crawled documents and transform the documents into lists of related keywords. Given a query and a list of documents returned by a search engine, our method generates a keyword net-

Figure 4.5: Query Refinement by Bing (previously known as MSN Search)

work based on the corresponding document keyword lists, and detects query sense communities by an unsupervised hierarchical algorithm. The documents are then assigned to different refined query sense communities based on Term-Frequency-Inverse-Document-Frequency (TF-IDF) scores to form clusters. More importantly, we propose to use the modularity score of the discovered query sense community structure to measure the page clustering necessity. More details of our approach and experiment results can be found in Chapter 8.

4.5 Entity Ranking

Ranking, i.e., a process of positioning entities on an ordinal scale in relation to others, is an important task that has a significant role in community mining and many other applications. In large databases, users would prefer the top-k partial tuples that are most related to their queries rather than a long list of tuples in a random order. In social networks, the ordering task is also challenging. One typically measures closeness of related entities in the network by a relevance score, which is computed with certain similarity metrics based on selected relationships between nodes in a graph. However, most of the existing approaches focus on ranking for homogeneous networks, which only contains one type of social relation, e.g., PageRank and HITS for web hyperlinks, or ObjectRank and RelationalRank for database tuples. While these methods are indeed successful, ranking for heterogeneous networks, which consists of multiple types of social actors and relations,

Club	Mark	David	Zhang	Rebecca
Golf	x	x	x	
Tennis	x			x
Basketball	x		x	
Swimming		x	x	x

Table 4.1: Membership for sports clubs

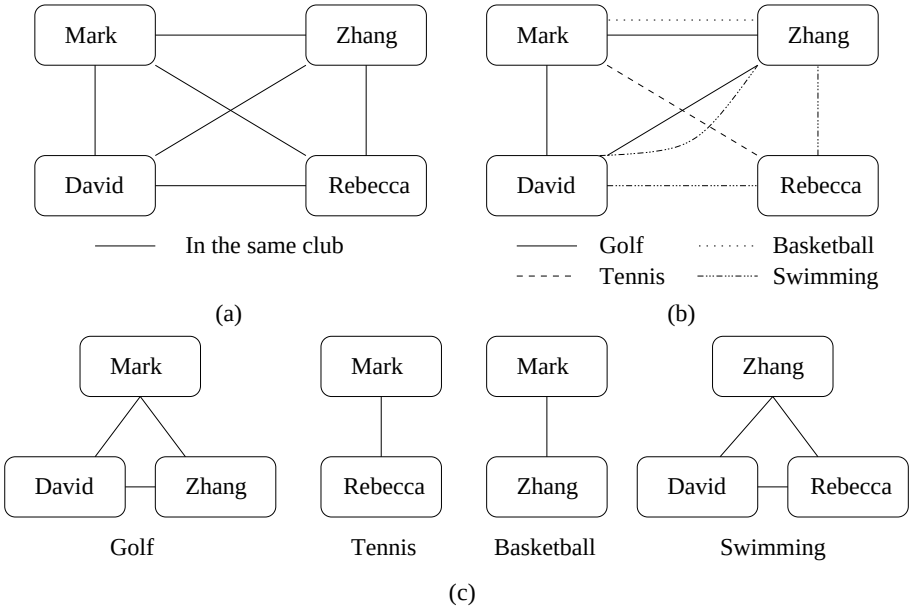


Figure 4.6: Traditional Models for Social Networks

are also interesting to investigate. There are many real world applications that can take advantage of this ranking. For example, there are seeds, peers and file types (multimedia, text, etc.) for P2P system data; conferences, authors and research topics for research publication data; books, readers and categories for a library system; customers, movies, genre and actors for a movie purchase database. Quantifying the relations between these entities by ranking could be of significant importance for tasks such as academic collaboration, customer service, etc.

Traditionally, richly structured datasets are naturally represented as networks, where each node represents an entity and the edges between nodes indicate the relations that connect two entities. The term *entity* refers to any single instance in the dataset, e.g., authors in a bibliography database. An example of such data model for a small social network between four people (Table 4.1) is shown in Figure 4.6 (a): nodes are connected to each other if their corresponding people are in the same

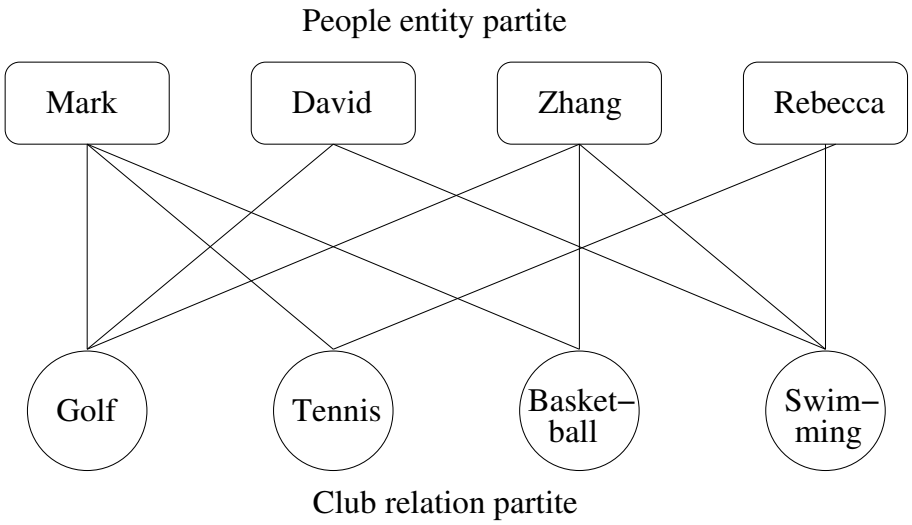


Figure 4.7: Bipartite Models for Social Networks

sports club. However, heterogeneous networks have multiple types of relations and entities, thus the traditional model cannot represent such network. Although we can use multi-edges between nodes to represent different kinds of relations (Figure 4.6 (b)), such model would become too complex to analyze when there are a number of relations. We can also model this network by using several single relation graphs (Figure 4.6 (c)), one for each relation. However, the increase in the number of relations may generate too many graphs, and nodes in different graphs are usually redundant. In oder to rank entities as well as various relations between them, we separate them as different graph partites. For example, in the social network shown in Figure 4.6, we have one partite for people, and another partite for sports clubs. The fact that one person is in a certain club is represented by an edge connecting an entity node and a relation node. Therefore there are no connections within a partite, i.e., people are only related to each other via sports club membership in this social network. We represent this model for heterogeneous social network as a bipartite graph, as shown in Figure 4.7. Compared to the traditional model, our model is able to represent a large number of social relations and how they connect different entities.

In order to solve the ranking problem in heterogeneous networks, we assign relevance values to entities in different partites. We then generate useful ranked lists of

entities based on this relevance assignment. More specifically, we first construct a k-partite graph based on the social relations, then apply a random walk approach on the graph to obtain rank values for nodes. Simply put, a random walk is a sequence of nodes in a graph such that when moving from one node N to the subsequent one in the sequence, one of N's neighbours is selected at random but with consideration for an edge weight. The relevance of a node, or the importance of a node B with respect to a node A, is the static steady state probability that the sequence of nodes would pass by B when the random walk starts in A. This probability, often estimated by a relevance score, is computed iteratively until some convergence (i.e. when no further changes in the probabilities are observed). A variation of this idea, which we advocate in our approach, is the random walk with restart. Given a graph and a starting point A, if at each step, we select a neighbour of the current point at random proportionally to the edge weights and move to this neighbour, or with probability $P_{restart}$ go back to the initial node A, the sequence of points selected is a random walk with restart (RWR) on this graph. Note that since the random walk algorithm has a start point, the ranking we obtain is only for one specific entity: the starting point in the graph.

In Chapter 9, we propose our approach to model heterogeneous social networks and present an iterative random walk algorithm on these models to compute the relevance score between entities. We also investigate the influence of different random walk directions on a k-partite graph model. To illustrate the effectiveness of our approach we used the DBLP bibliography database[1] to generate bipartite (author-conference) and tripartite database graphs (author-conference-topics) for tuple ranking. Topics are frequent n-grams extracted from paper titles and abstracts.

[1] http://www.informatik.uni-trier.de/∼ley/db/

Chapter 5

Discovering Communities with Domain Knowledge

Many community mining approaches have been developed to discover communities in networks. However, none of them distinguish the intrinsic features of the domain of the network in question. In other words, the same structural measure has been applied to all kinds of networks, despite their different characteristics, which can be achieved from domain experts. In this chapter, we present a new metric, Max-Min Modularity [43], to include domain knowledge as guiding criteria in the community discovery process. Then we propose a hierarchical algorithm based on this metric to find communities. More specifically, the domain knowledge in our approach is represented as a set of pairs of nodes, i.e, our approach takes a pair of nodes as "related nodes" or "unrelated nodes" to help the mining process. The research problem of this chapter is defined in Chapter 4.1.

5.1 Our Elaboration

Communities are defined to be densely connected groups of entities in a relational network, i.e., nodes in a strong community should be related to all other nodes in the same community. However, the original modularity measure (See Chapter 3) does not consider absent links. In other words, it only checks whether connected vertices are placed in the same community, but ignores disconnected vertices that share the same community. Therefore, to more thoroughly evaluate a network division, we should not only reward the evaluation score if connected vertices are put

in the same community, but also penalize the score if disconnected vertices are in the same community. However, a "disconnection" could possibly be an unobserved connection, which is very common in biological and social networks, so it is dangerous to assume disconnection to be a negative sign of the community structure. Therefore, while connected pairs remain as positive signs of a strong community structure, we separate the disconnected pair set into two categories based on knowledge provided by domain experts: the *related pair set*, in which pairs of nodes are *possibly* related, and *unrelated pair set*, in which pairs of nodes are *certainly* unrelated. We only penalize our measure score if we see unrelated pairs share the same community. Based on this criterion, we proposed a user-defined community structure measure, we call it *Max-Min (MM) Modularity*.

The idea of our Max-Min Modularity is based on the intuition that a good division of a network into communities is not merely one in which the number of edges between groups is smaller than expected, but also one in which *the number of unrelated pairs within groups is smaller than expected*. Only if both the numbers of between-group edges and within-group *unrelated pairs* are significantly lower than would be expected purely by chance, can we justifiably claim to have found significant community structure. Equivalently, we can examine the number of edges within communities and *unrelated pairs* between communities and look for divisions of the network in which this number is higher than expected. These two approaches are equivalent since the total number of edges/pairs is fixed, therefore any edges/pairs that do not lie between communities must necessarily lie inside one of them [152].

Generally speaking, our evaluation attempts to maximize the number of edges within groups and minimize the number of *unrelated pairs* from the user-defined unrelated pair set within groups at the same time, therefore we named it *Max-Min Modularity*. Note that maximizing the edge number within groups does not automatically minimize the unrelated pair number, e.g., if we have no network knowledge, thus have no *related pairs*, and *unrelated pairs* as disconnected node pairs, consider a node that only connects one member of a community with size n, maximizing the within-group edge number by including that node in this community

would increase the unrelated pair number by $n-1$.

5.1.1 Generalizing the Max-Min Modularity

The modularity Q can be transformed from its original form, which is community-based, to a node-based form. Given an unweighted and undirected network $G = (V, E)$, $|V| = n, |E| = m$, let A_{xy} be an element of the adjacency matrix of G:

$$A_{xy} = \begin{cases} 1 & \textit{if vertices x and y are connected} \\ 0 & otherwise \end{cases} \tag{5.1}$$

Assume the network is divided into k communities and node x belongs to community C_x, the fraction of edges that fall between community i and community j is defined as follows:

$$e_{ij} = \frac{1}{2m} \sum_{xy} A_{xy} \phi(C_x, i) \phi(C_y, j) \tag{5.2}$$

where the ϕ function $\phi(i, j)$ is 1 if i and j are the same community and 0 otherwise. The degree d_x of a vertex x is the number of edges that connect to it: $d_x = \sum_y A_{xy}$. Therefore, the fraction of edges that have at least one end in community i is:

$$\begin{aligned} a_i &= \tfrac{1}{2m} \textstyle\sum_{xy} A_{xy} \phi(C_x, i) \\ &= \tfrac{1}{2m} \textstyle\sum_{x} d_x \phi(C_x, i) \end{aligned}$$

From all the equations above:

$$\begin{aligned} Q &= \textstyle\sum_i (e_{ii} - a_i^2) \\ &= \textstyle\sum_i [\tfrac{1}{2m} \sum_{xy} A_{xy} \phi(C_x, i) \phi(C_y, i) \\ &\quad - \tfrac{1}{2m} \textstyle\sum_x d_x \phi(C_x, i) \tfrac{1}{2m} \sum_y d_y \phi(C_y, i)] \\ &= \tfrac{1}{2m} \textstyle\sum_{xy} [A_{xy} - \tfrac{d_x d_y}{2m}] \sum_i \phi(C_x, i) \phi(C_y, i) \end{aligned}$$

Define $P_{xy} = \frac{d_x d_y}{2m}$, we have the modularity Q as follows:

$$Q = \frac{1}{2m} \sum_{xy} [A_{xy} - P_{xy}] \phi(C_x, C_y) \tag{5.3}$$

We see that the original modularity has already measured the first part of MM Modularity:

$$Q_{max} = \frac{1}{2m} \sum_{xy} [A_{xy} - P_{xy}] \phi(C_x, C_y) \tag{5.4}$$

thus we only need to measure the other part, which is minimizing the *unrelated pair* fraction within communities. It is obvious that, if a division contains very few unrelated node pairs within communities for a graph, the same division will have equivalently few connected node pairs within communities for the complement graph,

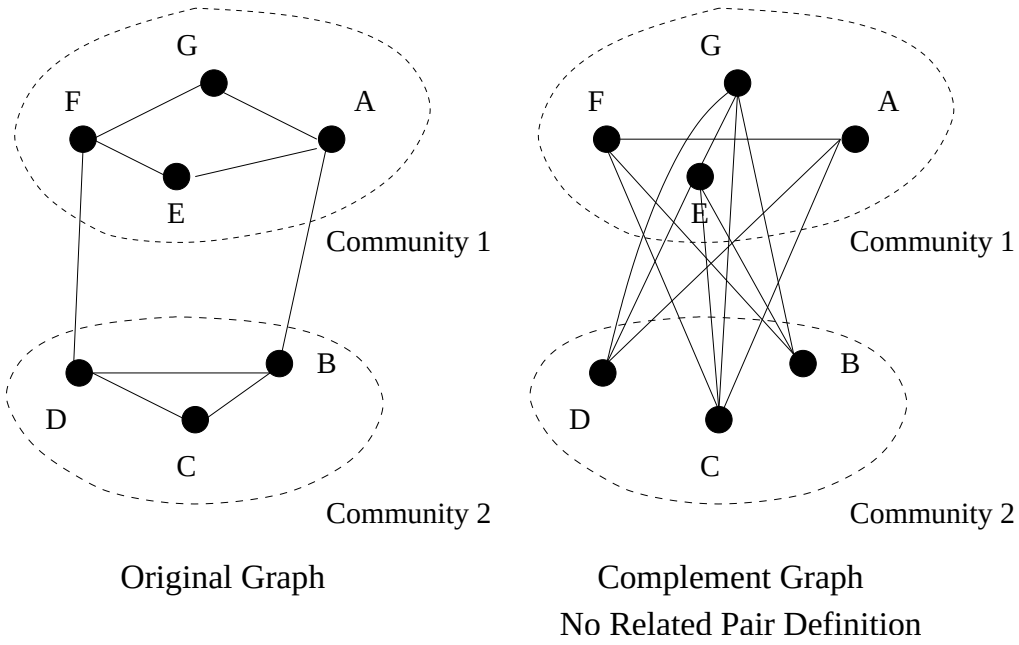


Figure 5.1: A Graph Division and its Complement

which is a graph on the same vertices as the original network such that two nodes are connected if and only if they are defined as an *unrelated pair* in the original graph. In other words, the better this division is for the original network regarding *containing few unrelated node pairs within communities*, the worse it is for the complement graph as a community structure since *there are equally few connected node pairs within communities* (See Figure 5.1). Therefore, we can compute the modularity score for a division on the complement graph of the network. The lower Q score we get, the better community division we have for the original network. Note that, although building a complement graph can be very expensive especially when the original graph is sparse, our method does not require extra computation or materialization of this complement graph, since we only need to maintain and update the *related pair* set defined by domain experts. Other information we need, which is the node degree and number of edges, can be easily achieved from the structure of the original graph.

More precisely, given an unweighted graph $G = (V, E)$, $V = \{v_x | 1 \leq x \leq n\}$, $E = \{e_x | 1 \leq x \leq m\}$ and the user-defined criteria U to define whether two disconnected nodes i, j are *related* $(i, j) \in U$ or *unrelated* $(i, j) \notin U$, we create

$G' = (V, E')$, such that if and only if $(i, j) \notin E$ and $(i, j) \notin U, (i, j) \in E'$, i.e., G' is G's complement graph. A' is the adjacency matrix of G'': $A'_{ij} = 1$ iff $A_{ij} = 0$ and $(i, j) \notin U$. We define Q_{min} as

$$Q_{min} = \frac{1}{n(n-1) - 2m - 2|U|} \sum_{xy} [A'_{xy} - P'_{xy}] \phi(C_x, C_y) \tag{5.5}$$

$\phi(C_x, C_y) = 1$ if C_x and C_y are the same community, 0 otherwise. $|U|$ is the number of pairs in U. Similarly, P'_{xy} is the expected probability of an edge between vertices x and y in a random graph:

$$P'_{xy} = \frac{(d'_x)(d'_y)}{2m'} \tag{5.6}$$

where d'_x is the degree of node x in G', and $m' = \frac{n(n-1)-2m-2|U|}{2}$. Since $d'_x = n - d_x - u_x$, we have

$$P'_{xy} = \frac{(n - d_x - u_x)(n - d_y - u_y)}{n(n-1) - 2m - 2|U|} \tag{5.7}$$

where u_x and u_y are number of nodes that are disconnected from but defined as *related* nodes with x and y. Now we want to maximize Q_{max} and minimize Q_{min} at the same time. Fortunately, it can be achieved by maximizing the following term Q_{Max_Min}:

$$\begin{aligned} Q_{Max\_Min} &= Q_{max} - Q_{min} \\ &= \textstyle\sum_{xy} [\frac{1}{2m}(A_{xy} - P_{xy}) \\ &\quad \textstyle -\frac{1}{2m'}(A'_{xy} - P'_{xy})] \phi(C_x, C_y) \end{aligned}$$

The higher Q_{Max_Min} is, the better community division we get. Note that we choose to use $Q_{max} - Q_{min}$ instead of $\frac{Q_{max}}{Q_{min}}$ for Q_{Max_Min}, since the former equation allows us to compute the Δ modularity between every pair of nodes. Also note that it is easy to extend the MM Modularity for weighted graphs by using the weight in degree computation for Q_{max} and Q_{min}.

We present an example for user-defined criteria in the following. In social networks, the neighbourhood around nodes are usually as important as direct connections. Thus we naturally define disconnected people as *related pair* if they connect to the same intermediary person. Therefore, we define U as if $(i, j) \notin E$ and there is such k that $(i, k), (k, j) \in E$, we have $(i, j) \in U$. By applying this criterion, we reward the MM modularity for connected pairs in the same community, penalize it for pairs that are in same communities and have no shared neighbour, and do

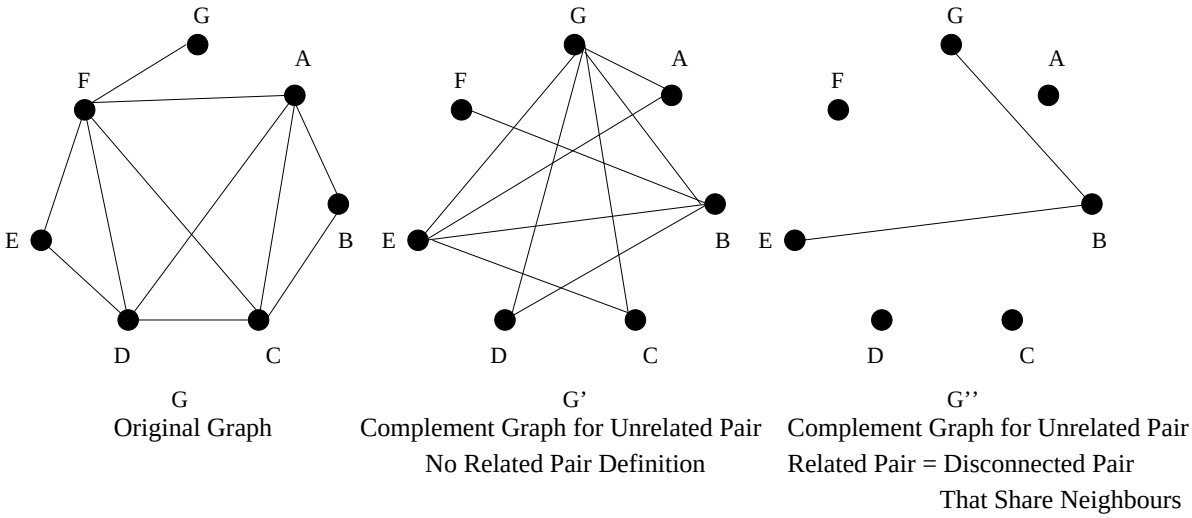


Figure 5.2: Building Complement Graphs

not reward or penalize pairs that are disconnected but share neighbour nodes. An example for building complement graphs for this criterion is shown in Figure 5.2. Other constraints can also be applied to define *related pairs*, as discussed in Section 5.3.

5.1.2 Algorithm for Community Detection

Here we propose a new method to evaluate the quality of the discovered community structure based on the MM Modularity. We may consider, if a high value of the metric represents a good community division, one can simply optimize Q_{Max_Min} over all possible divisions to find the best one. However, to find the optimal value of Q_{Max_Min} is very costly: to carry out a complete search of all possible divisions for the optimal value of Q_{Max_Min} would take at least an exponential amount of time, and is thus infeasible for large networks. Therefore, we propose a hierarchical clustering algorithm HMaxMin (see Algorithm 1) to greedily optimize the MM Modularity to find the approximate optimal value.

Recall that we reward the modularity for connected pairs, penalize it for unrelated pairs, and do nothing for disconnected but related pairs. Merging a pair of nodes between which there are no edges at all can never result in an increase in Q_{Max_Min}, thus in step 2 we only compute and store modularity scores for connected pairs in a sparse matrix. In step 3, we greedily merge the pair of commu-

nities which provides the highest modularity gain into one community and update the modularity matrix as well as the related pair matrix. For updating the modularity matrix, we first treat all pairs between community i and j as unrelated, i.e., $|i| * |j|$ pairs, then add the extra deducted value for related pairs (note that $Q_{i_r j_r} = \frac{1}{2m}(0-P_{ij}) - \frac{1}{2m'}(0-P'_{ij})$ if i and j are related and $Q_{ij} = \frac{1}{2m}(0-P_{ij}) - \frac{1}{2m'}(1-P'_{ij})$ if they are unrelated, thus $Q_{i_r j_r} = Q_{ij} + \frac{1}{2m'}$).

Algorithm 1 Hierarchical Clustering Algorithm HMaxMin to greedily optimize Q_{Max_Min}

Input: A social network $G = (V, E)$, $V = \{v_i | 1 \leq i \leq n\}$, $E = \{e_x | 1 \leq x \leq m\}$, Adjacency Matrix A, the user-defined criteria U.

Output: A division of V: $C_1, C_2, ...C_k$.

1. Assume each node is the sole member of one of the n communities. Build related pair matrix S, S_{ij} equals the number of related pairs between community i and j:

$S_{ij} = 1$ iff $(i, j) \in U$.

2. Compute the symmetric sparse matrix containing ΔQ_{xy} for each connected pair x, y:

$\Delta Q_{xy} = \frac{1}{2m}(1 - P_{xy}) - \frac{1}{2m'}(0 - P'_{xy})$.

Save each matrix row both as a balanced binary tree t_x and as a max-heap h_x. Save the largest element of each matrix row along with the x y label in a max-heap H. For each node x, we set: $a_x = \frac{d_x}{2m}$, $a'_x = \frac{d'_x}{2m'}$.

3. While ($pop\_heap(H) > 0$)

Select the largest element H and the corresponding x, y. Update ΔQ by merging y row (column) into x row (column) such that:

if community z connect to both x and y in G:

$\Delta Q_{xz} = \Delta Q_{xz} + \Delta Q_{yz}$

if z only connect to x in G and connect to y in G':

($|y|$ equals the number of nodes in community y)

$\Delta Q_{xz} = \Delta Q_{xz} + 2a'_y a'_z - 2a_y a_z - \frac{2|y||z|}{m'} + \frac{2S_{yz}}{m'}$

if z only connect to y in G and connect to x in G':

$\Delta Q_{xz} = 2a'_x a'_z - 2a_x a_z - \frac{2|x||z|}{m'} + \frac{2S_{xz}}{m'} + \Delta Q_{yz}$

Mark y row (column) in ΔQ and S as merged.

Update S by adding y row (column) into x column.

Update $new\_a_x = a_x + a_y, new\_a'_x = a'_x + a'_y$

Update t_x, update all h_z, update H.

4. Label merged nodes in the same unmarked row to be in the same community, as $C_1, C_2, ...C_k$.

5. Return $C_1, C_2, ...C_k$.

For algorithm complexity, consider we have n nodes and m edges. In step 2 we

only need to consider those pairs connected by edges, of which there will be at any time at most m. If we use n_x to represent the number of neighboring communities of community x, we have n_x elements for the x row in the sparse matrix. In step 3 of Algorithm 1, since we need to merge y row into x row, we will have $n_x + n_y$ insertions in the worse case. Since the rows are stored as balanced binary trees, each of the insertions take $O(\log n)$ in the worst case. Therefore, updating the matrix in step 2 takes $O((n_x + n_y) \log n)$ time. Similarly, we only need to update the heap for the k row if community k is adjacent to community x or y. We need to do at most $n_x + n_y$ updates of h_k, each of which takes $O(\log n)$ time, for a total of $O((n_i + n_j) \log n)$ time. Updating the related pair matrix S takes $O(|S_y|)$ (S_y is the number of elements in the y row of S), however, we can always choose y so that $|S_y| \leq |S_x|$, therefore, updating S takes $O(1)$ time. Since each merge takes $O((n_i + n_j) \log n)$ time, the total running time is the $O(\log n)$ times the sum over all degrees of the merged communities along the dendrogram. In the worst case, each node would contribute its degree to all of the communities it belongs to, along the path in the dendrogram from the node to the root, which makes the total degree $2m$, therefore, if the dendrogram has depth D, the algorithm runs in $O(mD \log n)$ time in a sparse graph. The approximate optimal Q value is also accumulated as the algorithm goes along. Thus, our algorithm includes domain knowledge but still runs in the same complexity as the similar algorithm proposed in [52].

5.2 Experiment Result

In this section, we apply our MM Modularity and the HMaxMin algorithm to detect communities on various social networks, including several data sets collected from real networks with ground truth as well as a synthetic data set which is randomly generated with given parameters, such as graph size and community numbers. The domain knowledge used is the example presented in Section 5.1.1: *if* $(i, j) \notin E$ *and there is such* k *that* $(i, k), (k, j) \in E$*, we have* $(i, j) \in U$. In other words, the generic domain knowledge given to the algorithm is represented in the form of pairs of disconnected nodes that could be related. All the experiments were conducted

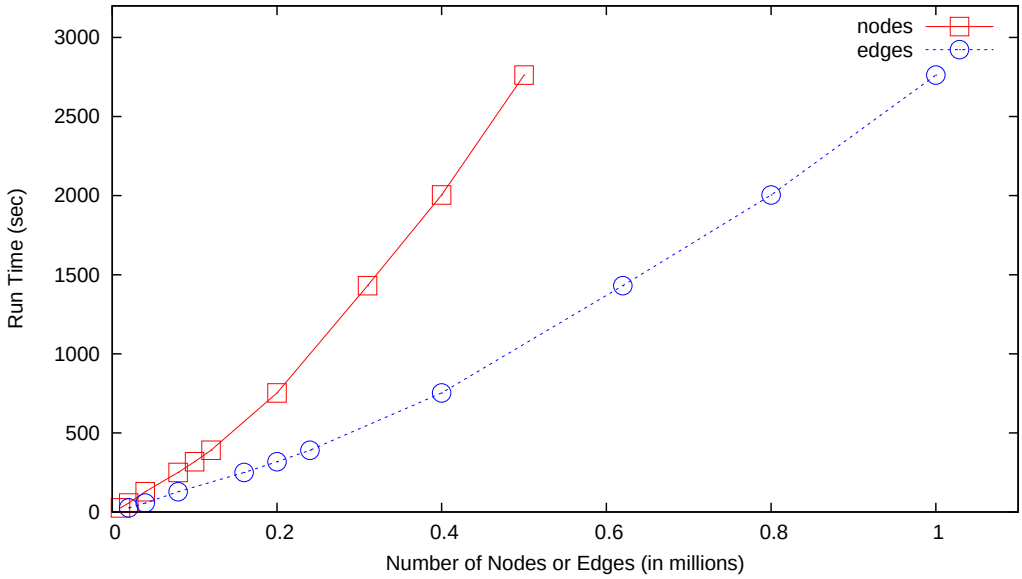


Figure 5.3: Algorithm Running Time

on a PC with a 3.0 GHz Xeon processor and 4GB of RAM.

5.2.1 Scalability

To test our algorithm on large datasets, we ran our algorithm on the largest component of the collaboration network of scientists posting preprints at www.arxiv.org [151], which has 27,519 nodes and 116,181 edges, and the IMDB network between actors who share involving movies [106], which has 47,436 nodes and 379,196 edges, within 376 and 1,037 seconds, respectively. To further evaluate algorithm efficiency, we generated ten random graphs of vertices ranging from 10,000 to 500,000 and the number of edges ranging from 20,000 to 1,000,000. Figure 5.3 shows the performance of our algorithm on those networks. It clearly reflects the $O(mD \log n)$ complexity of our approach.

However, we do not have ground truth to validate our result for such large datasets, thus we turn to synthetic data and real world datasets to evaluate the accuracy of our algorithm. Danon et al. [57] found that the modularity method outperformed all other methods for community detection of which they were aware, in most cases by an impressive margin. Thus maximization of the modularity to be perhaps the definitive state of the art method of community detection. Therefore, we

compared our HMaxMin algorithm with a similar hierarchical clustering algorithm FastModularity [52], which optimizes Newman's modularity to measure community structure, to show that our MM Modularity is more accurate for community detection tasks by including domain knowledge.

5.2.2 Evaluation Approach

To evaluate how closely each community in the result matches its corresponding community in ground truth, we adapt the Adjusted Rand Index (ARI) [223] as the performance metric for accuracy. The ARI measures how similar are the partition of objects according to the real communities (R) and the partition in an algorithm result (P). Denote a, b, c and d as the numbers of object pairs that are in the same community in both R and P, in the same community in R but not in P, in the same community in P but not in R, and in different communities in both R and P, respectively. ARI is defined as follows.

$$ARI(R, P) = \frac{2(a * d - b * c)}{(a + b) * (b + d) + (a + c) * (c + d)}$$

The more similar the two partitions (larger a and d, smaller b and c), the larger the ARI value. ARI will be 1 if R and P are identical and 0 if P is a random partition for the graph.

5.2.3 Synthetic Data

To test the performance of our algorithm on networks with varying degrees of community structure, we have applied it to a large set of randomly generated graphs. Each graph was constructed with 1000 vertices and 5 communities, each of which had 200 vertices. At first, each vertex was connected to 6 other randomly chosen nodes in the same community and had no connection to nodes in the different communities, thus we get 3000 within-community edges. Then we added a number of between-community edges, x, into the graph. Both ends were randomly chosen and each node could only connect up to 4 nodes in different communities so that within-community connections for all nodes were supposed to be stronger than between-community connections. This produces graphs which have a known

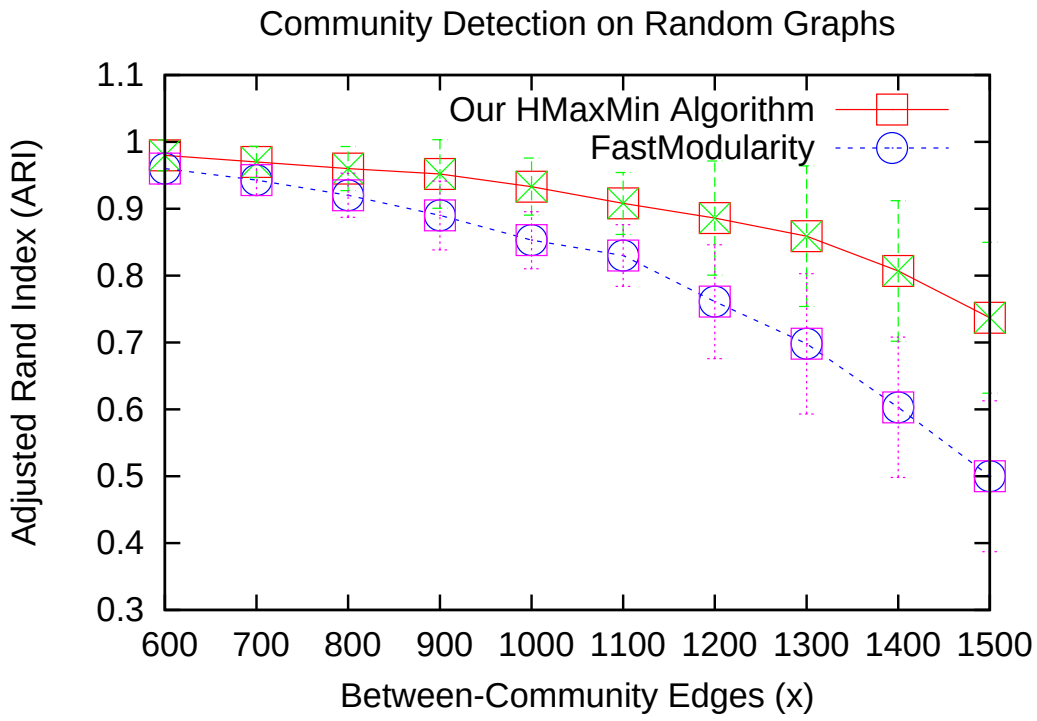


Figure 5.4: Synthetic Data Results (each point is an average over 50 1,000-node graphs.)

community structure, but are essentially random in other respects. Moreover, we can control the "noise", i.e., between-community edges, by adjusting x. Similar synthetic data generation methods have been used in [51, 85].

Using these graphs, we tested and compared the performance of HMaxMin algorithm and FastModularity with different x, as shown in Figure 5.4. As the figure shows, HMaxMin performs well, with a > 0.7 average ARI on graphs with less than 1500 between-community edges (50% of the within-community edges). HMaxMin begins to "fail" when x approaches more, however, there may not exist any communities in such circumstances. On the same plot, we also show the performance of the algorithm based on the original modularity Q (FastModularity) and, as we can see, that algorithm performs measurably worse than our algorithm. Interestingly, the performance of the two algorithms are about the same when the community structure is clear and strong, but when we increase the noise edge numbers, the accuracy of HMaxMin drops much slower than FastModularity. Therefore, it is reasonable to believe that our algorithm is more robust in finding community structure for data

	Ground Truth	FastModularity		HMaxMin		Improvement
	Communities	Comm.	ARI	Comm.	ARI	
Karate Club	2	3	0.680	2	1.00	47.1 %
Sawmill Network	3	4	0.664	3	1.00	50.6%
Mexican Politicians	2	3	0.255	3	0.359	40.7%

Table 5.1: Algorithm Comparison on Real World Networks.

with considerable noise.

5.2.4 The Karate Club

While random mid-size networks provide a reproducible and well-controlled testbed for community discovery evaluation, it is also desirable to test and compare the performance of our algorithm on real world networks. Since ground truth of large datasets is hard to come by, we have selected three network datasets, for which the community structure is already known from other sources. Table 5.1 shows the detected community number and the ARI score of result structures of both algorithms. The first network is drawn from the well-known "karate club" study of Zachary [224]. In this study, relations between 34 members of a karate club over a period of two years are observed. During the study, a disagreement developed between the administrator and the teacher of the club, which eventually made the club split into two smaller ones, centering around the administrator and the teacher, represented by node 34 and node 1. Then Zachary was able to construct a network of friendships, using a variety of measures to estimate the strength of ties between members of the club.

Note that the user-defined criteria we used for the karate network and all the following experiments is a heuristic rule for social networks where we believe the neighbourhood around people are as important as direct relations, thus disconnected people are treated as *related pairs* if they share the same intermediary friends. In other words, this criterion only penalizes the MM modularity for pairs that are in the same community and have no shared friends. We reward the score for connected pairs and do not reward or penalize sharing-neighbour pairs.

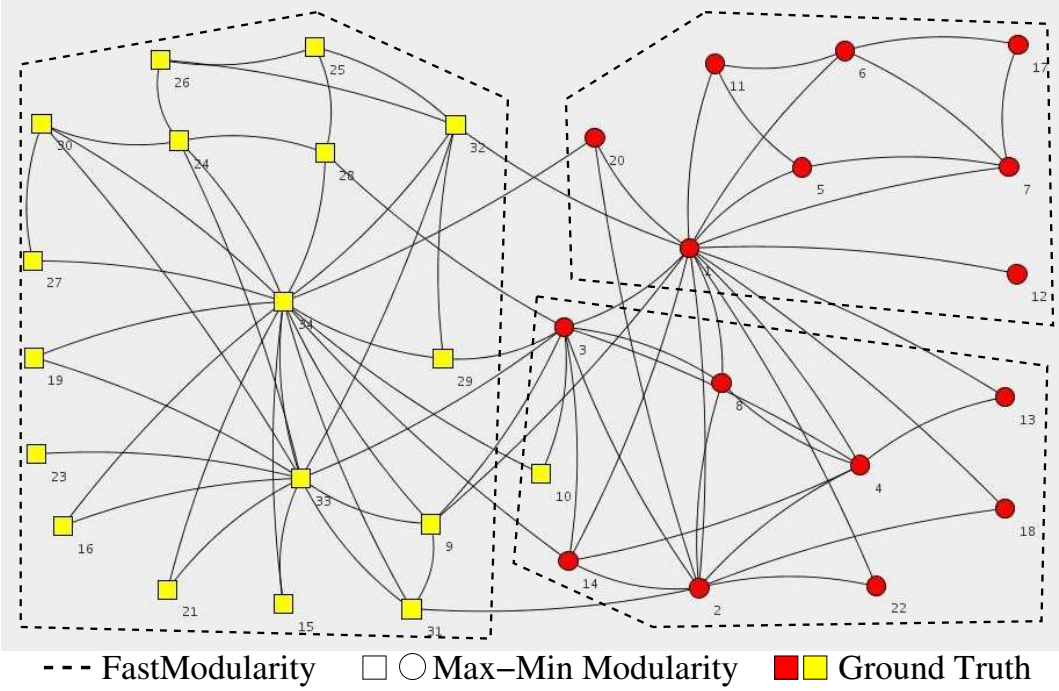


Figure 5.5: The Karate Club

In Figure 5.5[1], we show a unweighted network structure extracted from Zachary's observations. The actual divisions of the club following the break-up, as revealed by which club the members attended afterward, are indicated by node colours. We also show the results after feeding the network into HMaxMin (represented by node shape) and FastModularity (represented by closed area) in the figure. As we can see, FastModularity not only incorrectly generates one extra community, but also classifies node 10 into the wrong community[2], while results of our algorithm perfectly match the ground truth. In other words, the MM Modularity is better than the original as a predictor of subsequent social evolution of this friendship network.

5.2.5 Sawmill Communication Network

As a further test of our algorithm, we turn to the communication network of employees in a sawmill[3]. This data is collected in order to analyze the communication structure among the employees after a strike. An edge in the network means that

[1] All following network figures are generated using Meerkat [5].

[2] The result in [151], which showed that FastModularity find two communities with only one misclassified node, is incorrect, confirmed by M. Newman in private communication.

[3] This dataset is collected from the Pajek Project [168].

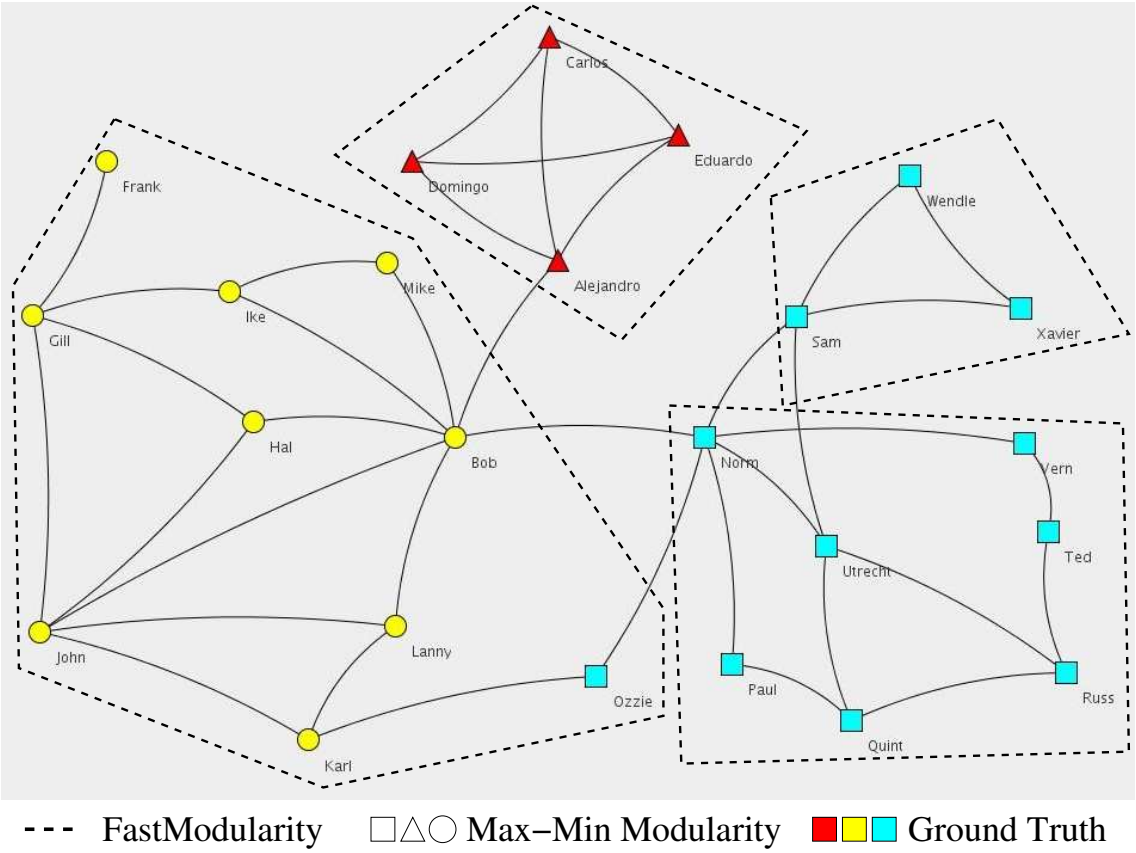


Figure 5.6: Social Network in a Sawmill

the two connected employees have discussed the strike with each other very often. As Figure 5.6 shows, there are three groups according to age and language. The Spanish-speaking young employees (the top group) are almost disconnected from the English-speaking young employees (the left group), who communicate with no more than two of the older English-speaking employees (the right group). All ties between groups have special backgrounds. For example, Alejandro is most proficient in English and Bob speaks some Spanish, which explains their connection. Bob owes Norm for getting his job, which may be the reason that they developed a friendship tie. Finally, Ozzie is Karl's father [168].

In Figure 5.6, we show the communities of the ground truth, our algorithm and FastModularity, indicated again by the node colour, shape and closed area. As we can see, both algorithms correctly identify the Spanish-speaking group, which is a clique. However, FastModularity inaccurately classifies Ozzie into the young

English-speaker group and generates an extra community for the older English-speaker group although Sam strongly connects the two separated groups. On the other hand, HMaxMin again perfectly detects the ground truth, as revealed by the sociology research.

5.2.6 Mexican Politician Network

For our next example, we look at a more complex relation network between politicians in Mexico (also collected from the Pajek Project [168]), which describes a social network between Mexican politicians in the 20th century. Edges represent significant social ties between the politicians, represented by nodes. Two groups within this network have been competing for power against each other (Figure 5.7), which are civilians (the top group) and members of the military force (the bottom group).

As we can see in Figure 5.7, this network has significantly more between-community connections than the previous two networks, which makes it harder for community mining algorithms to detect the communities correctly (as shown by the experiments on synthetic data). As the figure shows, FastModularity finds most members of the civilian group (red nodes), but it separates the military group (yellow nodes) into two communities and makes several mistakes around the periphery. On the other hand, HMaxMin also detects three communities, one for the civilian group, one for military group and one in the middle, mainly containing periphery nodes. Although it is hard to argue which partition is better simply by observation, Table 5.1 shows that the HMaxMin algorithm achieves a better 0.359 ARI score than 0.255 of FastModularity.

5.3 Discussion

Recently, community mining is increasingly attracting attention as an area of study and faces many challenges in developing community structures. Newman's Modularity has been proved to be effective and is thus pursued by many researchers, however, it has three main problems as we reviewed in Chapter 3. Our Max-Min

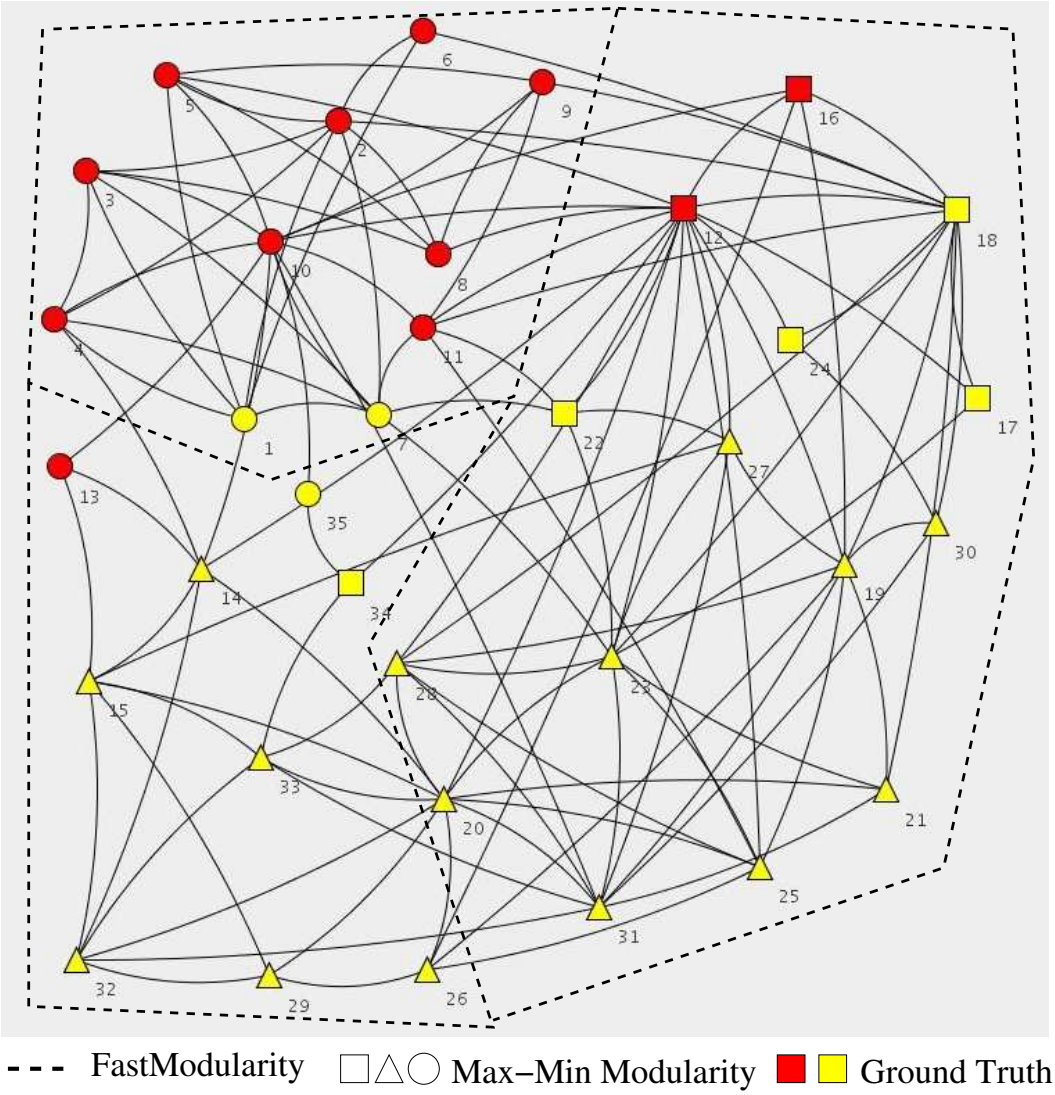


Figure 5.7: Mexican Politician Network

modularity solves the third problem of modularity shown in Figure 3.2, which does not consider absent links, by including the factor of user-defined related node pairs in the quality measure process, thus not only the detection accuracy is improved by taking advantage of domain knowledge, but community structure in different graphs can be compared. The Max-Min modularity idea can be easily extended to its local version, where global information about the graph is unavailable, and can be used in the recursive community detection to improve the community resolution. In networks such as biological and social networks, where connections can be unobserved, only considering the connected pairs might be inaccurate for commu-

nity structure detection. While other algorithms cannot handle such case, our MM modularity-based methods can achieve information from link prediction [128], and extract appropriate criteria for community detection. Additionally, our algorithm still runs in $O(mD \log n)$ time, which is the same as previous modularity-based algorithms.

5.4 Related Work

In the field of theoretical computer science, correlation clustering [16, 40, 195] considers a complete graph on n vertices, where each edge (u, v) is labeled either $+$ or $-$ depending on whether u and v have been deemed to be similar or different. Similar to our problem, the goal is to find a partition that agrees as much as possible with the edge labels, i.e., a clustering that maximizes the number of $+$ edges within clusters and minimize the number of $-$ edges inside clusters. However, while correlation clustering assumes the graph is complete and each connection is either positive or negative, such assumption is not true for community detection, where graphs are usually sparse and many of the edges are unobserved, i.e. labeled as 0. Moreover, while existing methods [83, 84] for correlation clustering require the user to specify parameters that are usually hard to determine [1], e.g., the number of clusters, our algorithm does not require parameters.

The idea of considering domain knowledge as related/unrelated pairs in this chapter is analogous to the notions of must/cannot links in semi-supervised clustering [18, 205, 206]. However, in semi-supervised clustering, the labeled data is used for cluster initialization [18] and the link constraints must be satisfied [205, 206]. Moreover, the number of clusters k is usually required as the starting parameter. On the other hand, our algorithm does not require parameters and the domain knowledge in our work is used to guide the bottom-up hierarchical clustering process, instead of generating initial communities. The given related/unrelated pairs are not enforced to be in the same/different communities, but contribute in calculating a metric score which evaluates the "closeness" of two communities.

5.5 Conclusions

We have described a new measure based on modularity for community structure and a hierarchical clustering algorithm HMaxMin for detecting communities from various networks. Different from other similar algorithms, which use one pre-defined similarity measure for all kinds of networks, our approach takes domain knowledge into consideration and thus improves the community detection accuracy. The proposed measure not only considers to maximize connected node pairs but also to minimize unrelated pairs in the same community, thus provides a considerable improvement over the original modularity, which only measures the existing connections within communities. While giving a penalty for all absent links might be too strict, domain knowledge is incorporated in our model to boost performance. Our change on the modularity has a big impact on community detection, and further improves the high accuracy that modularity-based method already achieves. We have applied the algorithm to randomly generated networks and a set of real world networks with ground truth for validation. We have also applied the algorithm on large networks to show its scalability. The experimental results confirm the accuracy and effectiveness of the proposed measure and algorithm for community structure detection.

Chapter 6

Discovering Local Communities

There has been much recent research on identifying global community structure in networks. However, most existing approaches require complete network information, which is impractical for some networks, e.g. the WWW or the cell phone telecommunication network. Local community detection algorithms have been proposed to solve the problem but their results usually contain many outliers, which are nodes that are weakly connected to the community and should not be included. In this chapter, we propose a new measure of local community structure [44], coupled with a two-phase algorithm that extracts all possible candidates first, and then optimizes the community hierarchy. We also propose a community discovery process for large networks that iteratively finds communities based on our measure [46]. We compare our results with previous methods on real world networks such as the co-purchase network from Amazon. Experimental results verify the feasibility and effectiveness of our approach. The research problem of this chapter is defined in Chapter 4.2.

6.1 Our Approach

Existing approaches discussed in Chapter 3 are simple. However, an effective local community detection method should be simple, not only because the accessible information of the network is restricted to merely a small portion of the whole graph, but also because the only means to learn more knowledge about the structure is by expanding the community, by one node at one step. With all these limitations

in mind, we present our L metric and the local community discovery algorithm in this section.

6.1.1 The Local Community Metric L

Intuitively, there are two factors one may consider to evaluate whether a node set in the network is a community or not, which are strong node relations within the set and weak relations between inside nodes and the rest of the graph. Therefore, almost all existing metrics directly use the internal and external degrees to represent these two significant factors, and identify local communities by maximizing the former while minimizing the latter. However, their community results might include many outliers and the overall community quality is questionable. (See Chapter 4.2 for explanation and Section 6.2.1 for experiment examples). The important missing aspect in these metrics is the *connection density*, not the absolute number of connections, that matters in community structure evaluation. For instance, even if there are one million edges within one node set N and no outward links at all, it is doubtful to identify N a strong community if every node in N only connects to one or two neighbours. Therefore, we propose to measure the community internal relation L_{in} by the average internal degree of nodes in D:

$$L_{in} = \frac{\sum_{i \in D} IK_i}{|D|} \tag{6.1}$$

where IK_i is the number of edges between node i and nodes in D. Similarly, we measure the community external relation L_{ex} by the average external degree of nodes in B:

$$L_{ex} = \frac{\sum_{j \in B} EK_j}{|B|} \tag{6.2}$$

where EK_j is the number of connections between node j and nodes in S. Note that L_{ex} only considers boundary nodes instead of the whole community D, i.e., the core nodes are not included since they do not contribute any outward connections. Now we want to maximize L_{in} and minimize L_{ex} at the same time. Fortunately, it can be achieved with the following equation:

$$L = \frac{L_{in}}{L_{ex}} \tag{6.3}$$

Note that it is possible to quantify the density L_{ex} by other means, e.g., average connections from the shell nodes to community nodes to measure L_{ex}. However, this method fails for the local community identification problem because the shell set is usually incomplete. For example, while the friend list of user A is available in Facebook, the list of the users that choose A as a friend is hard to obtain.

6.1.2 Local Community Structure Discovery

Using L to evaluate the community structure, one can identify a local community by greedily maximizing L and stopping when there is no remaining nodes in S that increases L if merged in D. However, this straight-forward method is not robust enough against outliers. Take Figure 6.1 as an example. Although L_{in} for O_1 would decrease because O_1 only connects to one node in D, the overall L might increase because the denominator L_{ex} decreases as well (O_1 only connects to one node outside D). Therefore, it is still possible to include outlier O_1 in the community. To deal with this problem, we look further into the metric instead of simply maximizing the score in a greedy manner. We note that, there are three situations in which we have an increasing L score. Assume i is the node in question and L'_{in}, L'_{ex} and L' are corresponding scores if we merge i into D, the three cases that may result in $L' > L$ are:

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

Obviously nodes in the first case belong to the community since they strengthen the internal relation and weaken the external relation. Nodes in the second case, e.g., O_1 in Figure 6.1, are outliers. They are weakly connected to the community as well as the rest of the graph. Finally, the roles of nodes in the third case cannot be decided yet, since they are strongly connected to both the community and the network outside the community. More specifically, when we meet a node i, which falls into this case during the local community discovery process, there are two

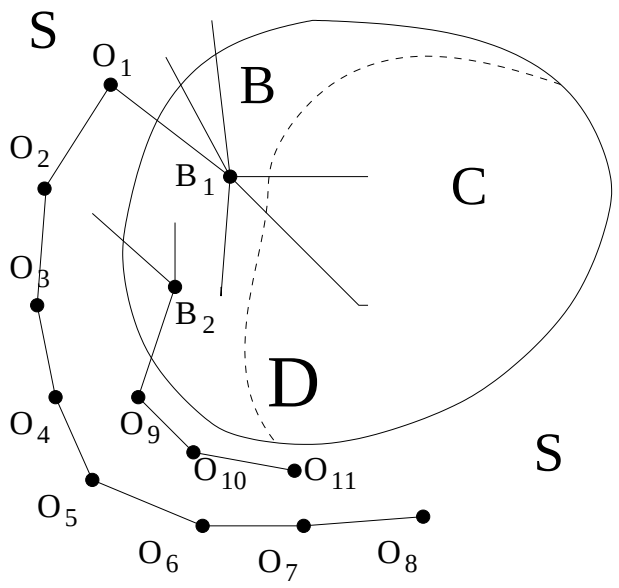


Figure 6.1: Problem of Previous Local Community Metrics

node types. First, node i can be the first node of an enclosing community group that is going to be merged one by one; Second, i connects to many nodes, inside or outside the community, and can be referred to as a "hub". We do not want hubs in the local community. However, it is too early to judge whether the incoming node is a hub or not in the exploration process. Therefore, we temporarily merge nodes in the first and third cases into the community. After all qualified nodes are included, we examine each node by removing it from D and check the metric value change of its merge again. Now we only keep nodes in the first case. If node i is a member of an enclosing group, L'_{ex} should decrease because all its neighbours are now in the community as well, while hub nodes would still belong to the third case (see Algorithm 2). Finally, the starting node should still be found in D, otherwise, we believe a local community does not exist. (see Algorithm 3.)

The computation of each L'_i can be done quickly using the following expression.

$$L'_i = \frac{\frac{Ind+2*Ind_i}{|D|+1}}{\frac{Outd-Ind_i+Outd_i}{|B'|}} \tag{6.4}$$

where Ind and $Outd$ are the number of within and outward edges of D before merging i, and should be updated after each merge; Ind_i and $Outd_i$ are the number of edges from node i to the community and the rest of network; B' is the new boundary set after examining all i's neighbour in D. In the discovery phase, L'_i need

Algorithm 2 General Local Community Identification

Input: A social network G and a start node n_0.
Output: A local community with its quality score L.
1. Discovery Phase:
Add n_0 to D and B, add all n_0's neighbours to S.
do
for each $n_i \in S$ **do**
compute L'_i
end for
Find n_i with the maximum L'_i, breaking ties randomly
Add n_i to D if it belongs to the first or third case
Otherwise remove n_i from S.
Update B, S, C, L.
While $(L' > L)$
2. Examination Phase:
for each $n_i \in D$ **do**
Compute L'_i, keep n_i only when it is the first case
end for

Algorithm 3 Single Local Community Identification

Input: A social network G and a start node n_0.
Output: A local community D for node n_0.
1. Apply algorithm 2 to find a local community D for n_0.
2. If $n_0 \in D$, return D, otherwise there is no local community for n_0.

to be recomputed for every node in S to find out the one with the maximum ΔL, thus the complexity of the algorithm is $O(kd|S|)$, where k is the number of nodes in the D, and d is the mean degree of the graph. However, in networks for which local community algorithms are applied, e.g., the WWW, and where adding a new node to D requires the algorithm to obtain the link structure, the running time will be dominated by this time-consuming network information retrieval. Therefore, for real world problems the running time of our algorithm is linear in the size of the local community, i.e., O(k). Note that in Algorithm 2 we begin with only one node n_0 while the same process could apply for multiple nodes to allow a larger starting D, C, B and S.

6.1.3 Iterative Local Expansion

Algorithm 3 is for identifying one local community for a specific set of starting nodes. However, we could apply this algorithm iteratively to cover the whole graph or a large section of the graph if the iterative process is terminated. In other words, instead of one-node-at-one-step, we expand as one-community-at-one-step to discover the community structure in the network. See Algorithm 4.

Algorithm 4 Iterative Expansion Algorithm

Input: A social network G, a start node n_0 and the community number m (optional).
Output: A list of local communities.
1. Apply algorithm 2 to find a local community l_0 for n_0.
2. Insert neighbours of l_0 into the shell node set S
3. **While** ($|S|! = 0$ && ($i \leq m$))
   Randomly pick one node $n_i \in S$.
   Apply algorithm 2 to find a local community l_i
   for n_i.
   Remove n_i and nodes in S that are covered by l_i.
   Update S by neighbours of l_i that are not covered
   yet.
4. Output m local communities $l_0, l_1, l_2..., l_m$, m could be given as a stop parameter if necessary.

In algorithm 4, we recursively apply the local community identification algorithm to expand the community structure. Every time we find a local community,

we update the shell node set, which is actually a set of nodes whose community information is still unclear. Note that here we accept identified local communities even if the starting node is not included. The shown algorithm stops when we have learned the whole structure of the network; however, we could also give parameters as stopping criteria if exploring the whole network is unnecessary or impractical, such as the number of discovered communities (m), or the number of nodes that has been visited (k). The algorithm could also be parallel and have multiple starting nodes, where several local community identification procedures start simultaneously from different locations of the network. Obviously, the complexity of the Algorithm 4 is still $O(kd|S|)$.

As previously discussed, in real world networks, one entity usually belongs to multiple communities. However, most of the existing approaches cannot identify such overlapping communities. Fortunately, even though we do not specifically focus on finding the overlapping property, our approach is able to discover overlapping communities, since in our algorithm nodes could be included in multiple local communities based on their connection structure.

6.2 Experiment Results

Since the ground truth of local communities in a large and dynamic network is hard to define, previous works usually apply their algorithms on real networks and analyze the results based on common sense, e.g., visualizing the community structure or manually evaluating the relationship between disclosed entities [15, 51, 133]. Here we adapt a different method to evaluate the discovered local communities. We provide a social network with absolute community ground truth to the algorithm, but limit its access of network information to local nodes only. The only way for the algorithm to obtain more network knowledge is to expand the community, one node at a time. Therefore, we can evaluate the result by its accuracy while satisfying limitations for local community identification. Based on our observations, the greedy algorithm based on metric R [51] (we refer to it as algorithm R) outperforms all other methods for local community detection. Furthermore, similar to

our approach, R does not require any parameters while other methods [14, 15, 133] rely on given parameters. Therefore, in this section we compare the results of our algorithm and algorithm R on different real world networks to show that our metric L is more accurate for local community detection. We evaluate our method by the traditional *Precision, Recall* and *F-measure* metric. Consider we have our result and the ground truth cluster, Precision and Recall are defined as

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

and F_measure

$$F\_measure = 2 * \frac{Precision * Recall}{Precision + Recall}$$

which is actually the harmonic mean of precision and recall,

6.2.1 The NCAA Football Network

The first dataset we examine is the schedule for 787 games of 2006 National Collegiate Athletic Association (NCAA) Football Bowl Subdivision (also known as Division 1-A) [221]. In the NCAA network, there are 115 universities divided into 11 conferences[1]. In addition, there are four independent schools, namely Navy, Army, Notre Dame and Temple, as well as 61 schools from lower divisions. Each school in a conference plays more often with schools in the same conference than schools outside. Independent schools do not belong to any conference and play with teams in different conferences, while lower division teams play only few games. In our network vocabulary, this network contains 180 vertices (115 nodes as 11 communities, 4 hubs and 61 outliers), connected by 787 edges.

We provide this network as input to Algorithm 3 and algorithm R. Every node in a community, which represents one of the 115 schools in an official conference, has been taken as the start node for the algorithms. Based on the ground truth posted online, the *precision*, *recall* and *f-measure* score of all the discovered local communities are calculated. We average the score for all schools in one conference

[1]The ground truth of communities (conferences) can be found at http://sports.espn.go.com/ncf/standings?stat=index&year=2006

2006 NCAA League		Algorithm Results						
		Greedy Algorithm R			Our Algorithm			
Conference	Size	P	R	F	No Comm.	P	R	F
Mountain West	9	0.505	0.728	0.588	0 node	0.944	1	0.963
Mid-American	12	0.392	0.570	0.463	1 nodes	0.923	1	0.96
Southeastern	12	0.331	0.541	0.410	3 nodes	1	1	1
Sun Belt	8	0.544	0.891	0.675	3 nodes	1	1	1
Western Athletic	9	0.421	0.716	0.510	4 nodes	0.6	1	0.733
Pacific-10	10	0.714	1	0.833	0 nodes	1	1	1
Big Ten	11	0.55	1	0.710	9 nodes	0.729	1	0.814
Big East	8	0.414	0.781	0.534	5 nodes	1	1	1
Atlantic Coast	12	0.524	0.924	0.668	3 nodes	1	1	1
Conference USA	12	0.661	1	0.796	1 nodes	1	1	1
Big 12	12	0.317	0.465	0.355	5 nodes	1	1	1
Total	115	0.488	0.783	0.595	34 nodes (29.6%)	0.927	1	0.952

Table 6.1: Algorithm Accuracy Comparison for the NCAA Network (Precision (P), Recall (R) and F-measure (F) score are all average values for all nodes in the community).

to evaluate the accuracy of the algorithm to detect that particular community. Finally, an overall average score of the precision, recall and f-measure score of all communities is calculated for comparison.

The experiment results are shown in Table 6.1. We first note the disadvantage of metric R we reviewed theoretically in Chapter 4.2, which is the vulnerability against outliers, has been confirmed by the results: for all communities, Algorithm R gets a higher recall but a much lower precision, which eventually leads to an unsatisfactory f-measure score. On the other hand, the accuracy of our algorithm is almost perfect, with a 0.952 f-measure score on average. Second, we see that our algorithm does not return local communities if starting with some nodes in the network, namely 34 of the 115 schools representing 29.6%. (Note that in these cases the local community is considered not existent and is not included in the average accuracy calculation even though the starting nodes are not outliers.) However, this result actually shows merit of our approach instead of weak points. Generally speaking, in one local community, nodes can be classified into cores and peripheries. It would be easier for an algorithm to identify the local community if starting from cores rather than peripheries. For example, if the algorithm starts

from a periphery node i in community c, the expansion step might fall into a different neighbour community d, which has some members connecting to i, due to lack of local information. The process would become more and more difficult to return to c as it goes along, because members of d are usually taken in one after another and finally the discovered local community would be d plus node i, instead of c. Fortunately, our algorithm detects such phenomena in the examination phase since i will be found as an outlier to d. Therefore we do not return the result as a local community for i since we already realize that it is misdirected in the beginning. As a possible solution for this problem, we can always start with multiple nodes, by which we provide more local information to avoid the starting misdirection. Note that while our algorithm handles such situation, algorithm R returns communities for every node without considering this problem, which is one reason for its low accuracy. Also note that another work [133] has a similar "deletion step", however, their approach depends on arbitrarily provided thresholds.

6.2.2 The Amazon Co-purchase Network

While mid-size networks with ground truth provide a well-controlled testbed for evaluation, it is also desirable to test the performance of our algorithm on large real world networks. However, since ground truth of such large networks is elusive, we have to justify the results by common sense. We applied our algorithm and algorithm R to the recommendation network of Amazon.com, collected in January 2006 [133]. The nodes in the network are items such as books, CDs and DVDs sold on the website. Edges connect items that are frequently purchased together by customers, as indicated by the "customers who bought this book also bought these items" feature on Amazon. There are 585,283 nodes and 3,448,754 undirected edges in this network with a mean degree of 5.89. Similar datasets have been used for testing in previous works [52, 133].

We first show discovered local communities for one popular book (*The Lord of the Rings (LOR)* by J.R.R. Tolkien), which is used as the starting node. The results are shown in Table 6.2. While both algorithms find interesting communities, we see that our algorithm detects books by authors other than Tolkien but are strongly

Algorithm	Items (Books) in the Local Community
Both	Smith of Wootton Major*
	The Lord of the Rings: A Reader's Companion$^{\#}$
	The Lord of the Rings: 50th Anniversary, One Vol. Edition*
	(The starting node) The Lord of the Rings [BOX SET]*
L	On Tolkien: Interviews, Reminiscences, and Other Essays$^{\#}$
	Tolkien Studies: An Annual Scholarly Review, Vol. 2$^{\#}$
	Tolkien Studies: An Annual Scholarly Review, Vol. 1$^{\#}$
	A Gateway To Sindarin: A Grammar of an Elvish Language from J.R.R. Tolkien's Lord of the Rings$^{\#}$
	J.R.R. Tolkien Companion and Guide$^{\#}$
	The Rise of Tolkienian Fantasy$^{\#}$
	Perilous Realms: Celtic And Norse in Tolkien's Middle-Earth$^{\#}$
R	Farmer Giles of Ham & Other Stories*
	Smith of Wootton Major & Farmer Giles of Ham*
	Roverandom*
	Letters from Father Christmas, Revised Edition*
	Bilbo's Last Song*
	Farmer Giles of Ham : The Rise and Wonderful Adventures of Farmer Giles*
	Poems from The Hobbit*
	Father Christmas Letters Mini-Book*
	Tolkien: The Hobbit Calendar 2006*

Table 6.2: Algorithm Comparison for the Amazon Network. $*$ indicates the author is J.R.R. Tolkien while $\#$ is not.

related to the topic, i.e., they are all about the history and content of the fantasy world. On the other hand, more than 90% of the books in R's community are written by Tolkien. Moreover, after reading the reviews and descriptions on Amazon, we found that many of the books are for children, e.g, *Letters from Father Christmas* and *Farmer Giles of Ham & Other Stories*. These books are not related to dragons and magic at all, but are included in the local community because they weakly connect to the starting node due to the fact that they share the same author, as we have discussed in Chapter 4.2.

The next example is the local communities discovered by two algorithms for a classical book, which is *The Arden Shakespeare Complete Works* by William Shakespeare. The results are shown in Table 6.3. For this book, we can see that the result community of the L algorithm becomes a subset of the result community of the R algorithm. However, while books that are found by both algorithms seems to be related to the starting book, algorithm R again includes noises in its result. For example, *The Book of Classic Insults* is an obvious outlier. It is weakly linked to the book *Shakespeare's Insults: Educating Your Wit*, which again confirms the weaknesses of algorithm R. Note that the communities we find here are based on the network structure and the starting node, instead of the content. While there are much more Amazon books on the subject of Shakespeare, only books shown here are identified to be in the local community given the starting book.

In the above two examples, the local community identified by L is smaller than that of R, which typically includes more noise. However, this is not always the case. In this example, we show that our L algorithm not only avoids noises in its community result, but also locates more valid community members than R. Table 6.4 shows the experiment result for the starting book *Fairy Tales* by Hans Christian Andersen. Apparently, the local community discovered by L does not include outliers. Moreover, it contains more related books, which are all about children's fairy tales, than the result community of algorithm R.

Algorithm	Items (Books) in the Local Community
Both	All the Words on Stage: A Complete Pronunciation Dictionary for the Plays of William Shakespeare
	Shakespeare's Bawdy (Routledge Classics)
	Shakespeare A to Z: The Essential Reference to His Plays, His Poems, His Life and Times, and More
	Shakespeare's Metrical Art
	Shakespeare and the Arts of Language (Oxford Shakespeare Topics)
	(The starting node) The Arden Shakespeare Complete Works
Only in R	Shakespeare's Insults: Educating Your Wit
	The Book of Classic Insults
	A Dictionary of Shakespeare's Sexual Puns and Their Significance
	Coined by Shakespeare: Words and Meanings First Penned by the Bard
	Shakespeare and the Art of Verbal Seduction
	Brush Up Your Shakespeare!

Table 6.3: Shakespeare Example for the Amazon Network

6.2.3 Iteratively Finding Overlapping Communities

After evaluating the accuracy of the L metric and our algorithm for single community identification, here we apply Algorithm 4 on the Amazon network to find overlapping communities iteratively. Table 6.5 shows several local community examples of our result. Note that start nodes of some communities may be removed by our algorithm. Such communities are not included using Algorithm 3 for single local community identification in earlier experiments.

The first community has 19 nodes, originating at the book *Mozart: A Cultural Biography*. It naturally includes other books about the life and music of the legendary musician. Similarly, we have another 15-node-community about the famous Polish pianist Chopin. The third community is a book series, which is the *Cambridge Companions to Music*. Finally, the fourth community and fifth community contain books about English grammar and William Shakespeare. Note that many other global community detection algorithms, e.g., FastModularity [52], become slow for such huge networks. Moreover, they may not apply if the global network information is unavailable.

Aside from local communities of books in Amazon, our approach also finds overlaps between communities. For example, the two books *The Cambridge Com-*

Algorithm	Items (Books) in the Local Community
Both	Grimm: The Illustrated Fairy Tales of the Brothers Grimm
	(The starting node) Andersen's Fairy Tales
L	Aesop's Fables
	The Complete Hans Christian Andersen Fairy Tale
	The Complete Brothers Grimm Fairy Tales
	The Classic Treasury Of Aesop's Fable
	Aesop's Fables: A Classic Illustrated Edition
	The Golden Book of Fairy Tales
	The Random House Children's Treasury
	The Complete Fairy Tales of Charles Perrault
	The Classic Treasury of Hans Christian Andersen
	A Child's Book of Stories
	Great Big Treasury of Beatrix Potter
	Mother Goose: The Original Volland Edition
	Children's Stories from Dickens
	HarperCollins Treasury of Picture Book Classics: A Child's First Collection
	Original Mother Goose
	Classic Fairy Tales
	The Random House Book of Fairy Tales
	The Random House Book of Nursery Stories
	The Everything Fairy Tales Book: A Magical Collection of All-Time Favorites to Delight the Whole Family
R	Mushroom Girls Virus: A Guide to the Identification and Study of Our Commoner Fungi with Special Emphasis on the Edible Varieties
	Book Designed to Help
	The Illustrated Fairy Tales Of Hans Christian Andersen
	Life Of Buddha
	1001 Nights: Illustrated Fairy Tales from One Thousand And One Nights
	Grimm's Fairy Tales (Audio CD)

Table 6.4: Andersen Fairy Example for the Amazon Network

	Items (Books) in the Local Communities
1	Mozart: A Cultural Biography
2	The Cambridge Companion to Mozart (Cambridge Companions to Music)
3	The Mozart Compendium: A Guide to Mozart's Life and Music
4	Mozart: The Golden Years
...	...
19	The Complete Mozart: A Guide to the Musical Works of Wolfgang Amadeus Mozart
1	Chopin In Paris: The Life And Times Of The Romantic Composer
2	The Cambridge Companion to Chopin (Cambridge Companions to Music)
3	Chopin (Master Musicians Series)
4	Chopin: The Man and His Music
5	Chopin's Letters
...	...
15	The Parisian Worlds of Frederic Chopin
1	The Cambridge Companion to Schubert (Cambridge Companions to Music)
2	The Cambridge Companion to Mozart (Cambridge Companions to Music)
3	The Cambridge Companion to Chopin (Cambridge Companions to Music)
4	The Cambridge Companion to Stravinsky (Cambridge Companions to Music)
5	The Cambridge Companion to Ravel (Cambridge Companions to Music)
...	...
9	The Cambridge Companion to Beethoven (Cambridge Companions to Music)
1	The New Webster's Grammar Guide
2	Hardcover, Longman Grammar of Spoken and Written English
3	Editorial Freelancing: A Practical Guide
4	The Oxford Dictionary for Writers and Editors
...	...
52	Modern American Usage: A Guide
1	Shakespeare's Language
2	Imagining Shakespeare
3	Hamlet: Poem Unlimited
4	... A Complete Pronunciation Dictionary for the Plays of William Shakespeare
...	...
66	William Shakespeare: A Compact Documentary Life

Table 6.5: Overlapping Local Community Examples for the Amazon Network

panion to Mozart (Cambridge Companions to Music) and *The Cambridge Companion to Chopin (Cambridge Companions to Music)* are found both in the community of the book series and the community of the subject. One could easily justify there is indeed some overlap.

6.3 Conclusions

In this Chapter, we have reviewed problems of existing methods for constructing local communities, and propose a new metric L to evaluate local community structure when the global information of the network is unavailable. Based on the metric, we develop a two-phase algorithm to identify the local community of a set of given starting nodes. Our method does not require arbitrary initial parameters, and it can detect whether a local community exists or not for a particular node. Moreover, we extend the algorithm to an iterative local expansion approach to detect communities to cover large networks. We have tested our algorithm on real world networks and compared its performance with previous approaches. Experimental results confirm the accuracy and the effectiveness of our metric and algorithm.

Chapter 7

Discovering Overlapping Communities with Visual Data Mining

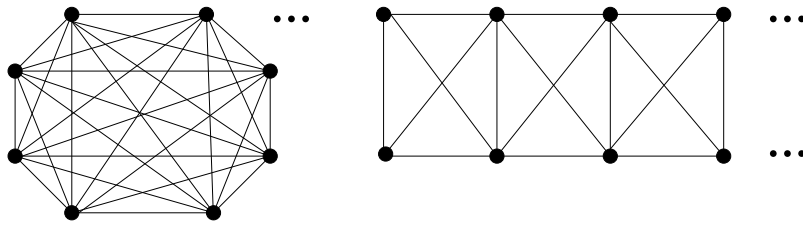
Finding global or local communities is an important task for the discovery of underlying structures in social networks. While existing approaches give interesting results, they typically neglect the fact that communities may overlap, with some hub nodes participating in multiple communities. Similarly, most methods cannot deal with outliers, which are nodes that belong to no germane communities. The definition of overlapping community is usually vague and the criterion to locate hubs or outliers vary. Existing approaches usually require guidance in this regard, specified as input parameters, e.g., the number of communities in the network, without much intuition. In this chapter we present a general list of requirements for a metric for overlapping community mining. We review advantages and disadvantages of existing metrics and propose our new metric to quantify the relation between nodes in a social network. We then use the new metric to build a visual data mining system [45], which first helps the user to achieve appropriate parameter selection by observing initial data visualizations, then detects and extracts overlapping communities from the network. Experiment results verify the scalability and accuracy of our approach on real data networks and show its advantages over existing methods. An empirical evaluation of our metric demonstrates superior performance over previous measures.

7.1 Visual Data Mining

Most community mining approaches apply data mining algorithms, e.g, agglomerative hierarchical clustering for a bottom-up merge, or partition clustering for a top-down split. Having noted that community mining is also a data mining process, we believe that the idea of visual data mining could be helpful in the mining process, both to guide the mining towards goals, and to better understand the results, since visualization and interaction capabilities enable the user to incorporate domain knowledge to finding communities in social networks. Generally speaking, the areas of data mining and information visualization offer various techniques which effectively complement one another supporting the discovery of patterns in data. Whereas traditional (algorithmic) techniques are analyzing the data automatically, information visualization techniques can leverage the data mining process from an orthogonal direction, by providing a platform for understanding the data and generating hypotheses about the data based on human capabilities such as domain knowledge, perception, and creativity [11]. In the past few years, visualization techniques have been specifically designed to support human involvement in the data mining process. For example, Ankerst et al. [9] propose an interactive decision tree classifier based on a multidimensional visualization of the training data. They later extend the work [10] to include categorical attributes to interactively build decision trees and thus support a much broader range of applications. Similar visual data mining ideas are also applied in [94, 201] to help users determine parameters for decision tree construction and classification rule discovery. In this chapter, we apply visual data mining on the problem of overlapping community discovery in order to help the parameter determination.

7.2 Preliminaries

In this section, we propose our observation for overlapping network communities and provide a list of requirements for a good measure for community detection. We discuss two existing measures based on those requirements.



(A) Clique Community (B) Transitive Community

Figure 7.1: Examples for Clique Community and Transitive Community

7.2.1 Community Definition

Recent research has proposed community detection methods in two different ways based on various motivations and similarity measures. First, hierarchical methods [155, 151] tend to find communities *globally* so that nodes, which are more densely connected to nodes in the same community than outside nodes, are grouped together; second, density-based approaches [221] classify nodes into communities based on their *local* structure, i.e., nodes are in the same community if they share many neighbours. In experiments, these two approaches typically yield noticeably different results on the same datasets. They actually target two different kinds of communities. On one hand, hierarchical methods partition networks by greedily maximizing an objective function, which increases for pairs of connected nodes that are in the same community and decrease for pairs of disconnected nodes also in the same community. Their methods favour communities where every node connects to everyone else in the same community, which we call *Clique Communities* (Fig 7.1 A). On the other hand, density-based approaches expand communities from nodes that are structurally dense, i.e., have enough neighbours, judged by appropriate parameters. Therefore, these approaches do not consider global properties but only the local network structure. They find communities where nodes may not directly connect to many others in the same community but are indirectly connected to every other node via some connections, which we call *Transitive Communities* (Fig 7.1 B). The difference between them is analogous to hierarchical-based and density-based methods in the data clustering field [227].

No matter how communities are defined, there are two major issues for overlapping community mining that remain to be addressed. First, each pair of nodes

should be measured by their similarity or relationship; second, pairs with high similarity or strong relationship should be put in the same community. Although it is the algorithm (hierarchical or density-based) that decides the community type to be found (clique or transitive), a good similarity metric is vital for both clique and transitive community structure detection. We present the requirements of a good metric in the following section.

7.2.2 Requirements for An Overlapping Community Mining Metric

It is easy to confuse graph partitioning with community mining since these two lines of research are really addressing the same question, which can be described as dividing vertices of a network into some number of groups. There are, however, important differences between network characteristics of the two camps that make quite distinct approaches and metrics desirable. For instance, in social network community mining, the relation between two nodes is asymmetric. (Take MySpace.com as an example: user A might list user B as one of his best friends while he is not even in the friend list of user B.) Thus, existing measures and approaches that are shown to be effective for some graph partitioning may not fit for community mining, since they do not take these differences into consideration. In the following, we propose a list of requirements, which we believe should be satisfied by a good metric for community mining.

1. ***A metric should measure the similarity between every pair of nodes.***
   A similarity score between two nodes is required for all algorithms to decide whether to put these two nodes into one community or not. The metric should be able to measure all pairs, connected or disconnected. Metrics, which do not consider disconnected pairs of nodes, may be able to find some community structure, but they naively assume that these nodes should not be in the same community.

2. ***A metric should reflect not only similarity but also dissimilarity.***
   In other words, the metric not only measures whether two nodes should be

in the same community but also measures whether they should not be in the same community. For instance, the metric should provide a means to solve a disagreement while merging a node n in a community when some existing nodes relate to n and others do not.

3. ***A metric should consider the asymmetric nature between pairs.***
   The pair asymmetry in social networks means that $Relation(i \rightarrow j) \neq Relation(j \rightarrow i)$, e.g., consider people pair (i, j) where i has many friends and is j's only friend, i is much more important to j than j is to i. For directed graphs, we have $Similarity(i \rightarrow j) \neq Similarity(j \rightarrow i)$. For undirected graphs, where the similarity measure are usually required to be symmetric, the asymmetric nature between the node pairs should still be considered.

4. ***An overlapping community metric should handle both hubs and outliers.***
   We think there are three kinds of nodes in a social network: hubs (nodes that have many connections and can be seen as community overlaps), outliers (nodes that have very few connections and do not belong to any community) and normal nodes (nodes that have some connections and belong to a community). The influences of hubs and outliers to community discovery have to be minimized by the metric.

7.2.3 Example Metrics for Community Detection

Newman et al. proposed the modularity Q as a quality measure of a particular division of a network [155]. For a social network with k communities, the modularity is defined as $Q = \sum_{c=1}^{k}[\frac{e_c}{m} - (\frac{d_c}{2m})^2]$ where m is the number of edges in the network, e_c is the number of edges between nodes within community c, and d_c is the sum of the degrees of the nodes in community c. The modularity Q measures the fraction of the edges in the network that connect vertices of the same community, i.e., within-community edges, minus the expected value of the same quantity in a network with the same community division but with random connections between the vertices. Q can be transformed as a sum of similarity scores for all node pairs

[52, 152]:

$$Q = \sum Q_{ij} = \sum_{i,j} (\frac{A_{ij}}{2m} - \frac{d_i}{2m} * \frac{d_j}{2m}) \tag{7.1}$$

where $A_{ij} = 1$ if nodes i and j are connected, 0 otherwise, d_i, d_j are the degree of node i and j, m is the edge number. Note that $Q_{ij} = 2 * (\frac{A_{ij}}{2m} - \frac{d_i}{2m} * \frac{d_j}{2m})$ since each pair (i, j) is calculated twice in the sum as (i, j) and (j, i). Also note that, Q_{ij} represents the difference between the probability of the event $i \leftrightarrow j$ *(node i and j are connected*) in the given graph structure ($P(i \leftrightarrow j) = \frac{A_{ij}}{m}$) and that in a random model with the same number of vertices, edges and degrees ($P(i \leftrightarrow j) = 2 * \frac{d_i}{2m} * \frac{d_j}{2m}$). (See [155, 152] for detail.)

The modularity Q provides a similarity score for all pairs of nodes. Whether the score is positive or negative depends on whether two nodes are connected or not, which reflects both similarity and dissimilarity. By taking the global information (the total edge number m) into consideration in the score calculation such that the higher degree the nodes have the lower score the pair gets, modularity handles the influence from hub nodes. However, the measure neglects the asymmetric nature between pairs in social networks by assuming $P(i \rightarrow j) = P(j \rightarrow i)$. Moreover, the method fails to handle outliers. Since outliers have small degrees and can achieve high scores given the formula, they are usually inaccurately merged first into a community by hierarchical algorithms.

Recently, Xu et al. [221] proposed another similarity measure S:

$$S_{ij} = \frac{|N_i \cap N_j|}{\sqrt{|N_i| * |N_j|}} \tag{7.2}$$

where N_i is the neighbourhood of node i, including i itself and all nodes connecting to i. This metric normalizes the number of common neighbours by the geometric mean of the two neighbourhoods' sizes in order to compare the neighbourhood structure of the two vertices in question.

The S metric considers the local structure of compared nodes (the common neighbour number) as well as their local attributes (the sizes of both neighbourhoods), thus it minimizes the similarity score from any nodes to both hubs and outliers. However, this metric does not measure dissimilarity, e.g., the score will always be zero if two nodes share no neighbours, disregarding the network structure,

and it fails to include pair asymmetry as well. Although this metric is easy to be extended for all pairs of nodes, it was originally proposed for connected pairs only. Additionally, even though the S metric considers the neighbourhood size of the two nodes in question, it neglects the degrees of other nodes in the neighbourhood, i.e., every node in the neighbourhood is weighted equally as 1 disregarding whether it is a hub, an outlier or a normal node.

We have summarized two state-of-the-art similarity metrics for community mining and analyzed their advantages and disadvantages (See Table 7.1). While they successfully find communities for some datasets, they are not particularly designed for overlapping community mining and thus do not meet all the requirements we list above.

7.3 Our ONDOCS Approach

In this section, we first present our characterization of the relation between nodes, then introduce the algorithm to generate network visualizations, and then show how to extract overlapping communities based on observed parameters.

7.3.1 Relationship Definition

Originally, ONDOCS is inspired by the OPTICS algorithm proposed by Ankerst et al. [8], where points are ordered for data clustering. However, unlike their clustering approach, we do not have a distance measure between nodes, so we need to define a new node relationship. The existing community metrics reviewed in Section 7.2 are designed to find optimal communities of a specific type, i.e., Q for clique communities and S for transitive communities, which means they focus only on partial aspects of network structure. We think that comparing the community structure to a random model, in which nodes are randomly connected in a network, is a better way to quantify node relations. The intuition is that community structure can be identified as that which is non-random; so developing a measure with a notion of random connections should help identify non-random structure. The neighborhood around any two nodes in question is also important in assessing their relationship.

Therefore we proposed a new measure R to combine these two aspects, defined as follows for undirected networks:

$$R(i,j) = \frac{Relation(i \rightarrow j) + Relation(j \rightarrow i)}{2} = \frac{\sum_{x \in N_j} r(i,x) + \sum_{x \in N_i} r(x,j)}{2} \tag{7.3}$$

where N_i is the neighbourhood of node i, including i itself and all nodes that connect to i. The similarity between node i and j is defined as the average of $R(i \rightarrow j)$, representing the relationship from i to j's neighbourhood, and $R(j \rightarrow i)$, representing relationship from j to i's neighbourhood. $R(i \rightarrow j)$ is defined as the sum of relation scores r between i and all nodes in j's neighbourhood, similarly for $R(j \rightarrow i)$ with respect to j and i's neighbourhood. Next, in order to quantify the relation $r(i,j)$ between node i and j, we compare the probability of the event that i *and* j *are connected* in the original graph G to a random model, where we only keep the same node number n and node degree $k_1, ..., k_n$ and leave the rest random. In such a random model, it is obvious that the probability of node i having a connection to any other node is $P(i) = \frac{k_i}{n-1}$ (similarly, $P(j) = \frac{k_j}{n-1}$). Here we assume G is undirected so that the event of i connecting to j and j connecting to i is equivalent, thus the probability of i and j being connected is the maximum of $P(i)$ and $P(j)$:

$$P(i \leftrightarrow j) = max(P(i \rightarrow j), P(j \rightarrow i)) = max(P(i), P(j)) = \frac{max(k_i, k_j)}{n-1} \tag{7.4}$$

Now we define the relation score $r(i,j)$ between node i and j:

$$r(i,j) = A_{ij} - \frac{max(k_i, k_j)}{n-1} \tag{7.5}$$

where $A_{ij} = 1$ if nodes i and j are connected in G, 0 otherwise. The extension for R on directed or weighted graphs is straightforward. The proposed metric R, r and the random model are justified in the next section.

Analyzing the R measure

We evaluate our R metric using the requirements listed in Section 7.2. First, R assesses similarity for both connected and disconnected pairs of nodes. Two nodes are measured by the relation between them and their neighbourhoods. Second, while

Metric	Metric Requirements			
	All Pairs	Similarity & Dissimilarity	Asymmetry	Hub & Outlier
Q	All	Yes	No	Only Hub
S	Connected	No	No	Both
R	All	Yes	Yes	Both

Table 7.1: Comparing Community Mining Metrics

the relation score r between each pair will be positive for connected pairs and negative for disconnected ones, R in Equation 3 considers all pairs within the local neighbourhood so that the R score represents an overall similarity, therefore $R(i, j)$ can be positive even if $r(i, j)$ is not. Similarly, $R(i, j)$ can be negative even if $r(i, j)$ is not. Third, the R metric is divided into two parts: $R(i \rightarrow j)$ and $R(j \rightarrow i)$, each of which represents the similarity between one node and the other's neighbourhood. The asymmetric characteristic of social networks is thus considered. Finally, the influence from hubs or outliers to other nodes are minimized. Hubs have big degrees which lead to large $\frac{max(k_i,k_j)}{n-1}$ and small r scores. Outliers have small neighbourhoods so R is small since there are few pairs to contribute in the sum. Therefore, as shown in Table 7.1, the R metric satisfies all requirements for a good community mining measure.

We now justify the formula for the relation score r and the random model presented in Section 7.3.1. Recall that the intuition behind the r score is to compare the probability of the event E, *that two nodes i and j are connected*, in the original graph structure with the probability of the same event in a random model, which has the same node number and degrees. Only if the probability of having these two nodes connected in the random model is low, does the fact that they are indeed connected show us strong relationship. Since the probability of E in the original graph is simply 1 or 0 given the network structure, we only need to answer the following question: *In an undirected graph G with n nodes, degrees $k_1, ..., k_n$ and the rest random, what is the probability of event E?* In this model, it is obvious that the probability of the event A: i connecting to j, equals to $\frac{k_i}{n-1}$ and the probability of the event B, j connecting to i, equals to $\frac{k_j}{n-1}$. However, either A or B confirms E, therefore we set $P(E) = max(P(A), P(B))$. In other words, with

respect to i, the probability of selecting j as one of i's neighbours is $\frac{k_i}{n-1}$. We cannot achieve a higher score unless $k_j > k_i$, thus the probability of the fact that two nodes are connected is decided by the node with the higher degree. Note that $P(E) \neq P(A) * P(B)$ since the two events A and B are dependent on each other.

7.3.2 Ordering Nodes to Visualize Networks

Now we can generate network visualizations by ordering nodes based on their relation scores. Given the relationship function we defined above, for a node n_i, we create a list of nodes l_i ordered by their relation to n_i from high to low. (Note that we can limit candidate nodes to those which have $R > 0$, i.e., they are connected to or share at least one neighbour with n_i.) We define the k^{th} value in this list to be l_{ik}. Here, our approach takes one input parameter s. However, as we will show in Section 5.2, s does not strongly affect the output. In practice, we usually generate several visualizations with s ranging from 2 to 8 and let the user make a choice based on their observations. For a node n_i, we define its community score C_s to be the s^{th} value in its node list l_i, i.e., $C_s(n_i) = l_{is}$, and $C_s(n_i) = 0$ if there are less than s nodes in the list. Then we define the reachability of node j with respect to i as

$$reach_s(i,j) = \begin{cases} R(i,j) & if\ C_s(n_i) > R(i,j) \\ C_s(n_i) & otherwise \end{cases}$$

Intuitively, the parameter s represents the expected number of nodes that one node is similar with in order to be a member of any community. C_s is the lowest relation score between node i and its similar neighbours in one community. Then the reachability score from node i to j ($reach_s(i,j)$) is the relation score between node i and j if j is not among the top s nodes of l_i and is the community score of i otherwise. Thus, $reach_s(i,j)$ measures the community relationship between i and j. It is their direct distance score if i and j are far away from each other, and equals to the community radius of i if j is close enough. Therefore, a decreasing order of the reachability scores (RS) indicates a node list for i, starting from i's most related neighbours to the least ones.

We present our algorithm to generate node lists ordered by their reachability scores in Algorithm 5. More specifically, our algorithm creates an ordering of net-

Algorithm 5 The ONDOCS Algorithm: Network Visualization

Input: A social network G with n nodes and m edges, a start node n_{start} and possible s values s_0, s_1, s_2....

Output: A list of nodes L with their Reachability Scores RS for each s.

1. Sort a node list l_i for each node n_i, ordered by their relation score to n_i, from high to low.

2. For each s :

Initialize a max-heap h, insert n_{start} in h with $RS = 0$.

Select the s^{th} largest element in l_i for each node n_i as its community score $C_s(n_i)$.

While (there is still nodes in heap h) :

Pop the node α in h with largest value ϵ.

Store α in L_s with $RS_\alpha = \epsilon$.

For all nodes x in l_α:

If $x \notin h$, insert x into h with $reach_s(\alpha, x)$.

If $x \in h$, update its value if $reach_s(\alpha, x)$ is larger.

Update max-heap h.

3. Return list L_s with RS values for each s value.

work nodes, additionally storing a reachability score $RS(i)$ for each node i. It starts at a given node n_{start} and inserts n_{start} into a max-heap structure h, which is maintained to store the reachability of candidate nodes. At each step, the node j, which has the highest reachability score in h, is chosen to be the next node in order and the popped score is stored as $RS(j)$. All nodes that are in j's neighbourhood are then inserted into h with their reachability according to j if they are not yet in h. The value in h is updated if the node is already in h and its new score is higher. Then h is updated to maintain its max-heap property. Therefore, the top node of heap h has the highest RS value to one of the nodes that have already been included in the list L, i.e., the RS score for each node in the list represents its highest reachability from any of the prior nodes in the sequence. The algorithm stops after all nodes in the network are visited.

The computational complexity of ONDOCS is $O(n \log n)$ for dense graphs and $O(n)$ for sparse ones. The list generation and sort step takes $O(c \log cn)$ where constant c is the average number of similar nodes for each node. Note that based on our relationship function, one node can only be similar to another if they are connected or share one or more neighbours. In step 2, there are n insertions to

the heap h and updating h for each insertion takes $O(\log n)$ time for dense graphs and $O(1)$ for sparse networks. Thus, the actual running time of our algorithm on experimental networks is $O(n)$ as shown in Section 5.2.

In summary, given a network with a list of s values, Algorithm 5 produces a sequence of nodes with their reachability scores for each s value, which can be visualized as a 2-D graph by tools such as GNUPlot [86]. The visualizations show interesting community information such that nodes in the same communities are consecutive in the list with high RS scores while the RS score apparently drops between two groups of community nodes (See Figure 7.3). The goal of visual data mining is to help users acquire accurate parameters by observing this phenomenon, which is presented in the next section. (A detailed example of how to choose the parameters is given in Section 7.4.2 and Figure 7.3 after explaining the experiments.)

7.3.3 Detecting Overlapping Community Structure: Communities, Hubs and Outliers

We have generated lists of nodes given specific s values, where we found that the ordering of the corresponding RS values has interesting community properties. For example, if we start from one node i, we will first visit other nodes in i's community in sequence. This is because the reachability score from i to these nodes are higher than nodes outside i's community. Therefore, each community can be seen as a group of consecutive nodes with high RS scores. In a 2D visualization, these groups are represented as curves in a "mountain" shape or peak. A noticeable drop of subsequent RS scores after a "mountain" indicates that this community has ended, which is represented as a curve in a "valley" shape or trough. The "valley" between two "mountains" represents a set of hubs, which belong to several communities. For instance, if we start from nodes in community α, the fact that hubs have neighbours from different communities makes RS scores of hubs lower than that of those single-community nodes in α but still higher than nodes in communities other than α. Therefore, after all single-community nodes in α are visited, hubs are next to follow before nodes in other communities, which form the "valley" between "mountains."

As we have discussed in the introduction, there is no global community definition, thus communities in specific networks need to be defined by parameters given by the user. For this purpose, our visual data mining approach generates visualizations with different s values first. After the user selects the suitable one based on their observation, they need to further provide two parameters to define the communities in this network, *Community Threshold (CT)* and *Outlier Threshold (OT)*. While such parameters are usually hard to obtain for previous methods, parameter selection for our approach becomes easy since we provides a visualization of the network structure with "mountains" representing strongly related communities, and "valleys" representing hub nodes that connect to both communities. Outliers are usually found at the end of the list, since their RS scores to any other nodes in the network are low. Examples of choosing parameters for real networks are presented in Section 5.2. Note that we do not require k, the number of communities to discover, as a parameter. The number of communities is a byproduct of the mining process given the parameters OT and CT which are determined by the user after exploiting our visualization output. The visualization of the network helps the user understand the structure first and then decide about reasonable thresholds for communities and outliers, i.e not the numbers per se but has a similar effect.

Given the two parameters CT and OT, our algorithm works as the following: from the first node in the sequence as the starting community, we scan all nodes along the list. One node n_i is merged into the current community if $RS(n_i) \geq CT$. If $CT > RS(n_i) > OT$, n_i is classified as a hub. If $OT \geq RS(n_i)$, it is an outlier. Since the first node of a community in the list has a low RS score, e.g., the starting node has $RS = 0$, we refine the outlier and hub nodes by moving any node n_i into corresponding communities if we have $RS(n_{i+1}) \geq CT$. (Also see Algorithm 6) The complexity of Algorithm 6 is $\theta(n)$.

To represent that hubs can belong to k communities, for each hub node i, we use a vector of "belonging factors" $v = (f_{(i,1)},\ f_{(i,2)}\ ...\ f_{(i,k)})$ where each coefficient $f_{(i,k)}$ measures the strength of the relationship between node i and community k. For every community C_k, we can quantify the Overall Relationship between i and

Algorithm 6 The ONDOCS Algorithm: Overlapping Community Structure Detection

Input: A list L of nodes $n_0, n_1, ...$ and their RS scores, the Community Threshold CT and the Outlier Threshold OT.
Output: A list of communities $c_0, c_1, ...$, hubs $h_0, h_1, ...$ and outliers $o_0, o_1, ...$
1. Create a community c, set $k = 0$.
2. for each $n_i \in L$ **do**
  If $RS(n_i) \geq CT$, classify n_i as a community node.
   else if $CT > RS(n_i) > OT$, classify n_i as a hub.
   else classify n_i as an outlier.
  end if
  If i is not classified as a community node but $RS(n_{i+1}) \geq CT$
   classify i as a community node.
  end if
  If n_i is a community node, insert n_i into c.
   else (n_i is a hub or an outlier)
    If $|c|! = 0$, save c as a community c_k for output
          reset c for the next community, increase index k by 1
    end if
  end if
 end for
3. Return communities $c_0, c_1, ...$, hubs $h_0, h_1, ...$ and outliers $o_0, o_1, ...$

C_k as

$$OR_{(i,k)} = \begin{cases} \sum_{x \in C_k} R(i,x) & if \ \sum_{x \in C_k} R(i,x) > 0 \\ 0 & otherwise \end{cases}$$

We then normalize the vector to get the coefficients so that we have $\sum_{x=1}^{k} f_{(i,x)} = 1$. Therefore, one node can belong to many communities at the same time, weighted by the relationship value in the range $[0, 1]$ and the sum of belonging coefficients to communities is the same for all nodes in the network, except outliers.

In summary, the community mining process is aided by visual data mining in our approach. Instead of asking the user to arbitrarily provide vital parameters, we generate visualizations of the network in question so that the user is able to observe the structure and relations between communities before they give parameters. After appropriate parameters are determined, hubs and outliers are extracted together with communities. Note that another advantage of our approach is that while parameters are easy to be altered, the impact on the change of discovered communities can be clearly perceived by observing the visualization.

Datasets	Vertices	Edges	Runtime / s			
			CONGO [89]		CF [170]	ONDOCS
			h = 3	h= 2		
football [221]	180	787	8	2	1	< 1
protein_protein [170]	2640	6600	114	11	3	11
blogs [89]	3982	6803	41	8	4	12
PGP [27]	10680	24316	772	104	>20000	62
word_association [170]	7207	31784	15922	230	102	161
blogs2 [89]	30557	82301	15148	380	319	269
cond-mat [146]	27519	116181	> 20000	1486	490	544

Table 7.2: Results on Real World Networks

7.4 Experiment Results

Here we evaluate the ONDOCS approach using both synthetic and real world datasets. The performance of ONDOCS is compared with CFinder [170] and CONGO [89], which are shown to be two of the most efficient algorithms for finding overlapping communities [89]. Note that they apply different mechanism to generate overlaps (See Chapter 3.5), instead of defining similarity metrics and visual data mining. The comparison is measured by the well known F-measure score and Adjusted Rand Index (ARI) [223] (See Chapter 5.2.2). All experiments were conducted on a PC with a 3.0 GHz Xeon processor and 4GB of RAM.

7.4.1 ONDOCS Scalability

To evaluate the scalability of our algorithm, we generated ten random graphs of vertices ranging from 10,000 to 500,000 and the number of edges ranging from 20,000 to 1,000,000. The edges are randomly distributed in the network. Figure 7.2 shows the performance of our algorithm on those networks. It clearly shows that, although the running time of ONDOCS is $O(n \log n)$ in the worst case, our approach actually runs very close to linear time with respect to the number of vertices and edges.

To further evaluate the efficiency of the algorithm, we apply three algorithms on several real-world networks. Table 7.2 shows the source of each network, its statistics, and the execution times for CONGO to compute the entire dendrogram, CFinder (v1.21) to generate solutions for $3 \leq k \leq 8$ and ONDOCS to create dataset

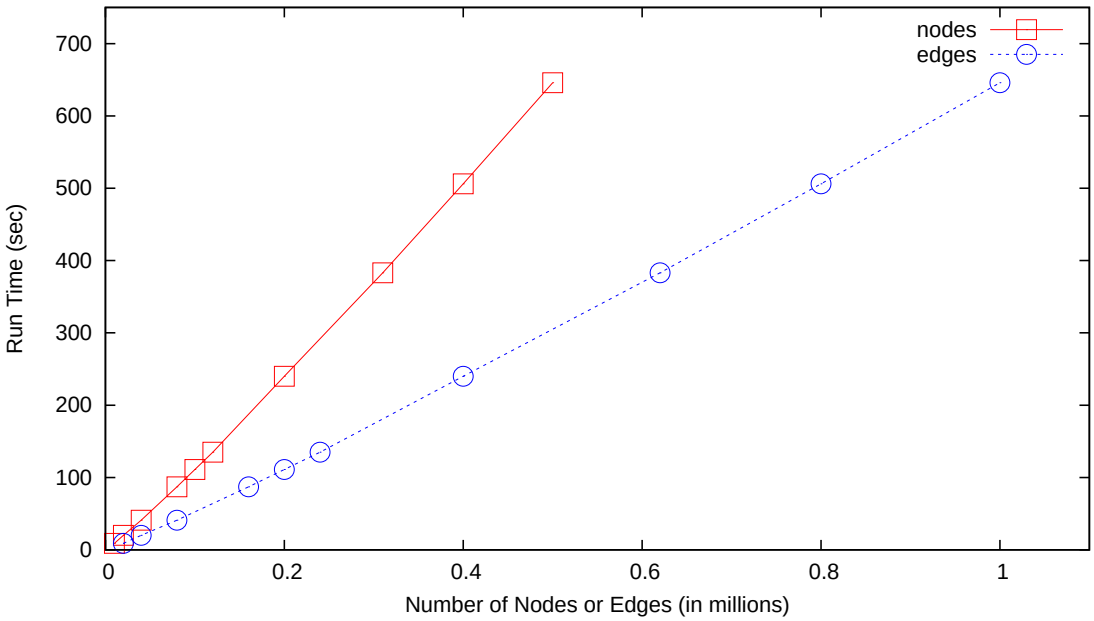
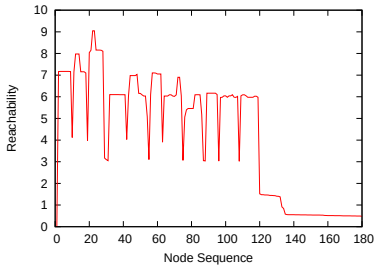


Figure 7.2: Algorithm Running Time

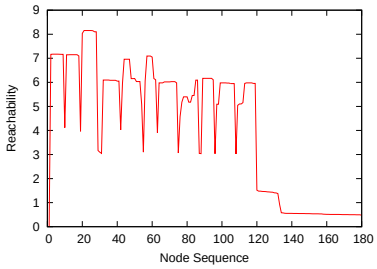
visualizations for $2 \leq s \leq 8$. From the table, we can see that ONDOCS works well overall, while CONGO's running time increases dramatically with respect to h and CF's clique detection becomes slow on some particular networks. However, we do not have ground truth to validate the accuracy of our results for these datasets, thus we turn to several real world datasets with ground truth to evaluate the accuracy of our approach.

7.4.2 ONDOCS Accuracy

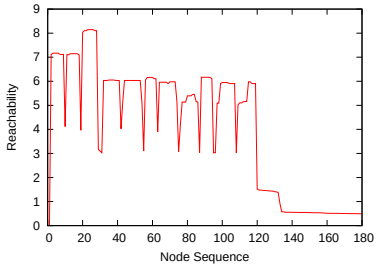
The first dataset we examine is the schedule for 787 games of 2006 National Collegiate Athletic Association (NCAA) Football Bowl Subdivision (also known as Division 1-A) [221]. In the NCAA network, there are 115 universities divided into 11 conferences. In addition, there are four independent schools at this level, namely Navy, Army, Notre Dame and Temple, as well as 61 schools from lower divisions. Each school in the division plays more often with schools in the same conference than schools outside. Independent schools do not belong to any conference and play with teams in different conferences, while lower division teams play only very few games. In our network vocabulary, this network contains 180 vertices (115 nodes



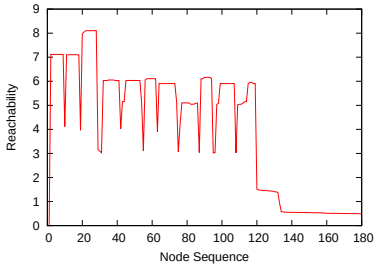
(a) S=3



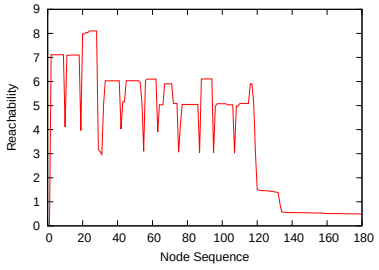
(b) S=4



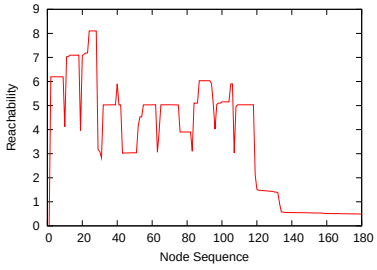
(c) S=5



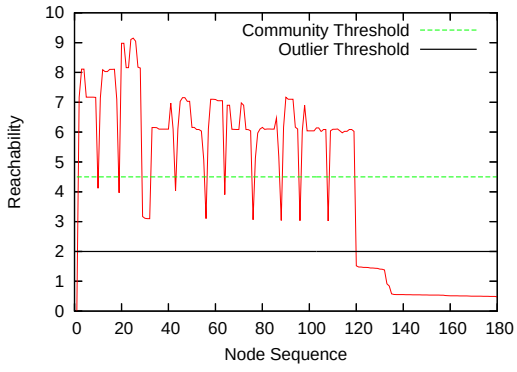
(d) S=6



(e) S=7



(f) S=8



(g) S=2

Figure 7.3: Community Visualizations of the football network with different S value

Data Setting		Algorithms		
		CONGO (h=2)	CF (k=4)	ONDOCS (s=2) (CT = 4.5, OT = 2)
115 Nodes in 11 Clusters	Cluster	11*	11	11
	Hub	92	6	0
	ARI	0.047	0.945	1.00
Plus 4 Hubs	Cluster	11*	12	11
	Hub	100	8	3
	F-measure	0.038	0.167	0.857
Plus 4 Hubs and 61 Outliers	Cluster	11*	12	11
	Hub	96	8	3
	Hub F-measure	0.04	0.167	0.857
	Outlier	0	61	61
	Outlier F-measure	0	1.00	1.00

Table 7.3: Result Comparison on the Football Dataset. (*The right cluster number is provided as a parameter for the CONGO algorithm.)

as 11 communities, 4 hubs and 61 outliers), connected by 787 edges. (The same dataset has been used in Chapter 6 to evaluate local communities.)

First, the ONDOCS approach generates several visualizations with different s values for the user to choose. We show all visualizations for $2 \leq s \leq 8$ in Figure 7.3. As we can see, most images are very similar to each other. The only one that shows a different structure is the visualization for $s = 8$. Recall that the parameter s represents the expected number of nodes that one node is similar with in order to be considered as a community member. When s is raised to a large value, some communities might disappear if their size is smaller than s. In this case, ONDOCS visualizations only show the structure of communities whose size is greater or equal to s. The larger the s value is, the smoother the curves are and the fewer "spikes" we have. Nevertheless, we have seven visualizations that clearly represent the network structure, where there are 11 communities, a few hubs and a set of outliers.

The parameter selection is solely based on users' visual interpretation of the visualized network. First we choose the visualization with $s = 2$, where the community structure is shown in most detail since pair relations are mostly measured as direct distance. In Figure 7.4, we note that nodes in sequence from 120 to 180 are barely related to the rest and can be considered as outliers, therefore we set $OT = 2$.

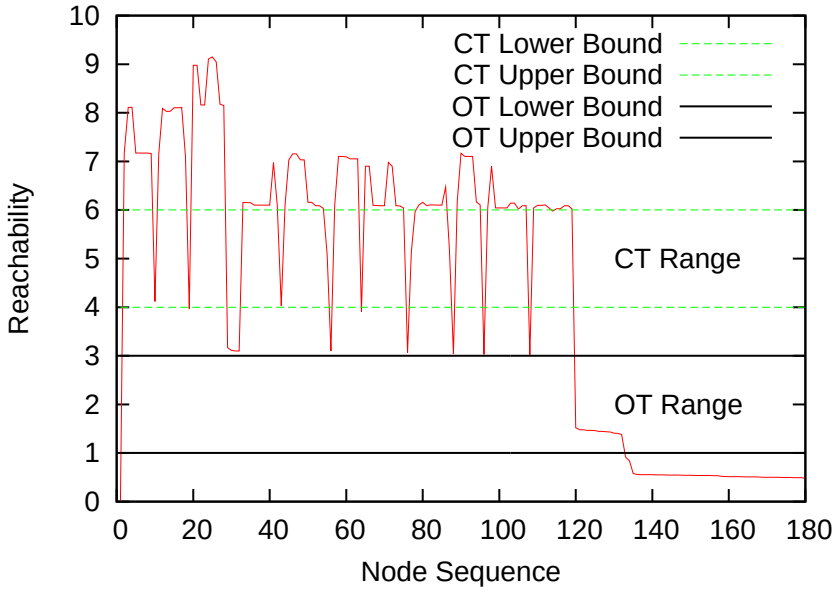


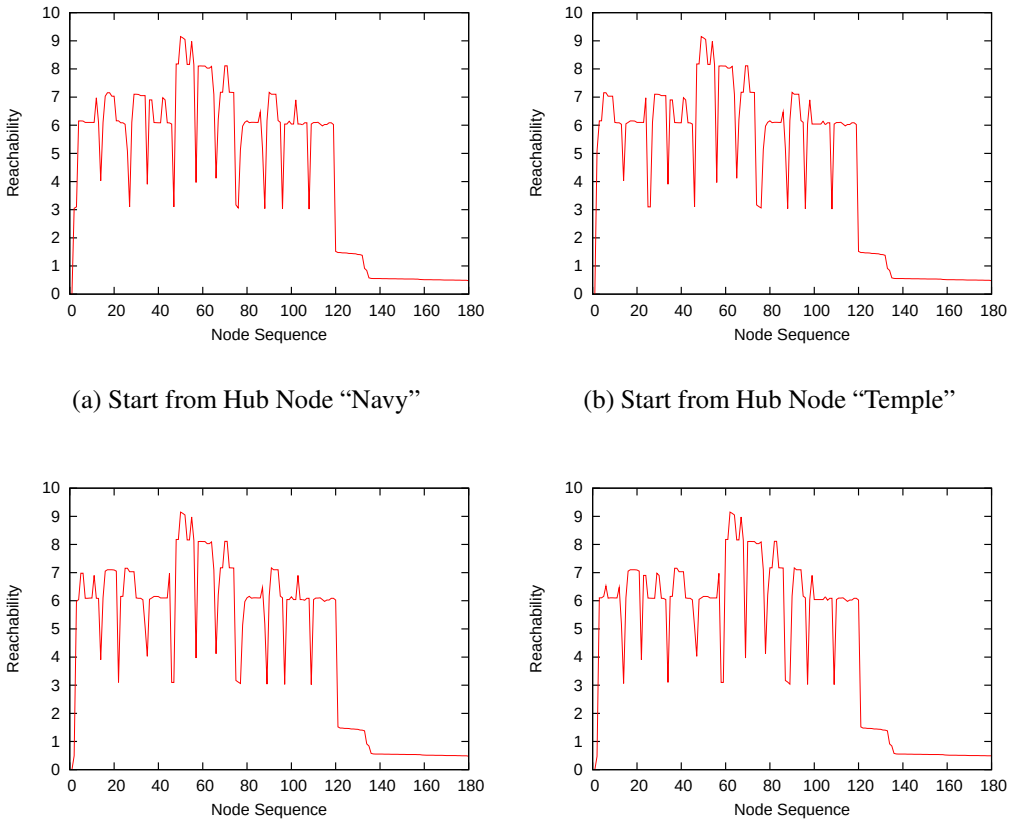
Figure 7.4: Selecting CT and OT for ONDOCS

CT	OT = 2					OT	CT = 4.5				
	Cluster	Hub	H-FM	Outlier	O-FM		Cluster	Hub	H-FM	Outlier	O-FM
4.0	9	3	0.857	61	1.0	1.0	11	16	0.30	48	0.880
4.5	11	3	0.857	61	1.0	1.5	11	4	0.75	60	0.991
5.0	11	3	0.857	61	1.0	2.0	11	3	0.857	61	1.0
5.5	11	6	0.8	61	1.0	2.5	11	3	0.857	61	1.0
6.0	12	7	0.77	61	1.0	3.0	11	3	0.857	61	1.0

Table 7.4: Comparing ONDOCS Accuracy with Different CT and OT. (H-FM means F-measure for Hubs and O-FM means F-measure for Outliers.)

Note that OT can also be set as 2.5, or any other close number. Different OT value will not give completely different results and the impact can be perceived directly from the visualization. Furthermore, we see a community usually ends with a RS score between 3 and 5, thus we set $CT = 4.5$ so that all communities are separated. The range of possible thresholds are shown in the figure. Table 7.4 shows results of varying CT and OT in the range. As can be noticed, it is quite easy for one to select parameters given the network visualization, and the results are stable enough for a large range of parameters.

To evaluate how algorithms detect overlapping community structure, we provide the data to our algorithms in three different ways. At first, we give only 115 community nodes and connections between them, then we measure the accuracy of discovered communities by the ARI score based on the ground truth, which is the conference assignment. Then we add the 4 hubs and their connections into the network. Although these hubs clearly belong to multiple communities, we do not have exact ground truth for overlapping community structure, i.e., which communities these hubs should go. However, we do have ground truth for which nodes are hubs (outliers) and which are not. Therefore, we measure the accuracy of the output hubs and outliers by the F-measure score, which is defined as the harmonic mean of precision and recall. Finally we give the complete network with communities, hubs and outliers. Table 7.3 shows the experiment results for the three algorithms. As we can see, the CONGO algorithm always detects overlaps, even for the first network where there are only community nodes. Additionally, it requires the cluster number as the input parameter, which is usually unavailable for real world networks, and it still fails to find any outliers. The CF algorithm gives its best result when $k = 4$, where it detects all outliers and finds 12 clusters, which is very close to the truth. However, CF also finds hubs when there is no overlap and the accuracy of its overlap detection is low with only a 0.167 F-measure score. Our ONDOCS algorithm works the best overall. It finds all outliers and only detects hubs when there is indeed some overlap between communities. The hub detection accuracy is not perfect, however, when we look into the data, we find out that the only missing hub team (Temple) plays half of its games (6 out of 12) with teams from the Mid-



(a) Start from Hub Node "Navy"

(b) Start from Hub Node "Temple"

(c) Start from Outlier Node "Maine"

(d) Start from Outlier Node "Furman"

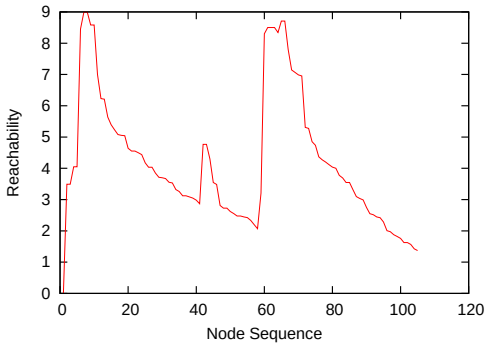
Figure 7.5: ONDOCS Visualizations with different starting nodes

American conference, which explains why it is classified into that community. Note that the result of our algorithm depends on two parameters (CT and OT), however, we believe that appropriate values are easy to find based on direct observation on network visualizations.

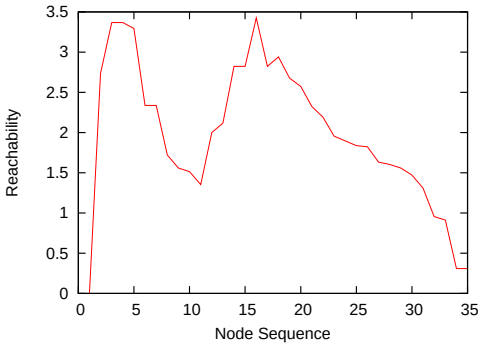
In ONDOCS, the node sequence might change if we choose different node n_{start} to start with. For previous experiments, we choose a community node to start the process. In Figure 7.5, visualizations that start from hub nodes and outlier nodes are shown. However, as we can see, a community, represented by a "mountain" curve, is found first. It is because our algorithm intends to visit the closest nodes in the sequence, which have higher RS scores, before nodes that are far away. Thus, no matter where the start node is, the closest community is found first, followed by other communities ordered by their RS values. Hubs are found as "valley" between

communities.

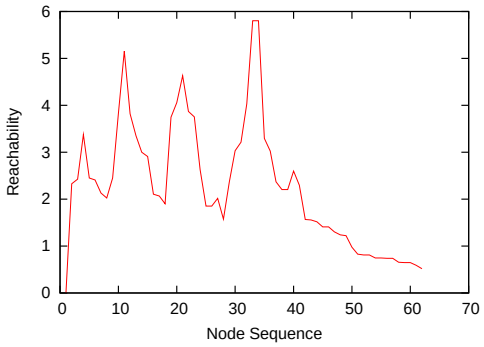
We also apply our algorithm on other real world networks, including the Political Book network [119], the Mexican Politician network [168], the Dolphin network [155] and the Les Miserables network [118]. Although we do not have exact overlapping truth for these networks, approximate community structure information is provided by previous research. In the Political Books dataset, nodes represent political books sold by Amazon.com and edges represent frequent co-purchasing of books by the same buyers, as indicated by the "customers who bought this book also bought these items" feature on Amazon. Nodes are manually labeled as "Liberal," "Neutral," or "Conservative" by Mark Newman [145]; In the Mexican Politicians dataset, edges indicate social relations between people and nodes represent politicians, who are classified based on their background as "Citizen" or "Military". The Dolphin Network gives the community structure of a group of bottle-nose dolphins. The network can be approximately divided into four main groups [155]. Finally, the Les Miserables network represents the coappearance network of characters in the novel Les Miserables. Note that for these datasets, we only have indefinite community information instead of perfect ground truth, which is the common case for overlapping community detection and evaluation. We show visualizations for these datasets generated by ONDOCS in Figure 7.6. One can see that the images correctly depict the approximate community information we have. Accurate CT and OT values should be easy to determine based on these figures. Also note that if the reachability plots are not clear for some datasets, the users may have problems selecting parameters. This could be the case when a large number of real communities exist, where the plot would present a jagged graph with many close peaks for a vague community structure. This is a limitation of the visualization and may be addressed by increasing the screen real-estate or a progressive hierarchical method, which selects parameters for each level of the community hierarchy. However, it is nevertheless reasonable to believe that other approaches with no visual data mining support, when faced with a large number of existing communities, would provide less information and do even worse in the mining process.



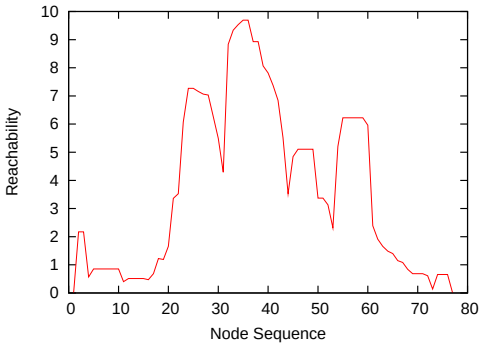
(a) Political Book Network



(b) Mexican Politician Network



(c) Dolphin Network



(d) Les Miserables Network

Figure 7.6: Community Visualizations for Various Networks by ONDOCS

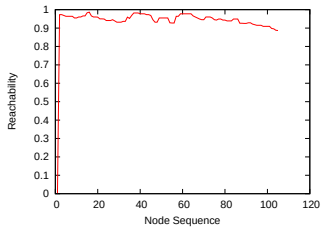
7.4.3 Comparing Metrics within ONDOCS

We have reviewed previous community mining metrics (Q and S) and proposed our relational metric R. We then evaluated them from a theoretical perspective. Here we apply these three metrics to measure the similarity between two nodes in our ONDOCS system and compare the images generated for several real world datasets respectively in order to further evaluate the effectiveness of the metrics.

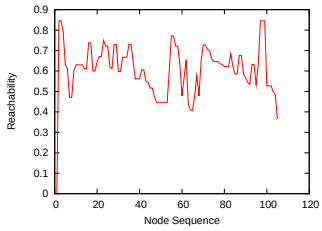
The visualizations for four different datasets based on metrics Q, S and R are shown in Figure 7.7(a) to 7.7(l) respectively (s is set to 2 for all metrics). We see that the plots using the R metric accurately depict the network structure since they match the vague community information that we have for those datasets. On the other hand, visualizations using the S metric are ambiguous and the community structure is hard to read. Also note that the R visualizations provide a much wider range for the user to observe accurate CT and OT values to detect the right number of communities than the S visualizations. Finally, visualizations based on the Q metric do not show any community structure. The reason is that Q does not consider local structure thus similarity scores of all node pairs are smaller than and close to 1 after node ordering, which makes the plots into a nearly-horizontal line.

7.5 Conclusions

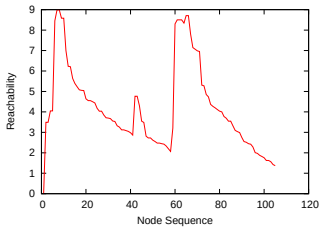
In this chapter, we first propose a general list of requirements for a good similarity metric to detect overlapping communities. We analyze existing metrics based on those criteria and then propose a new similarity metric R which satisfies all of those requirements. A visual data mining approach for overlapping community detection in networks is then proposed based on metric R. The method first generates lists of nodes, ordered by their reachability scores. Network visualizations are then provided to help the user determine important parameters. Finally, overlapping communities, i.e., communities, hubs and outliers, are extracted based on these parameters. Experiment results show that our approach not only scales well for large networks, but also achieves a high accuracy for real world networks. Unlike previous approaches, our method only detects overlap when overlap exists. Moreover,



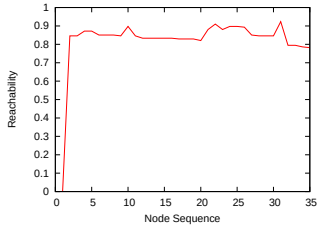
(a) Political Book by Q



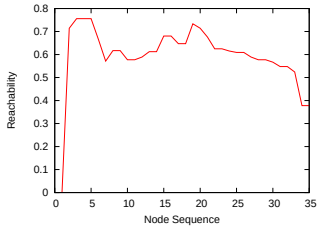
(b) Political Book by S



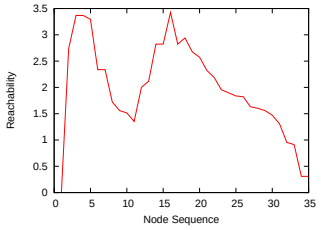
(c) Political Book by R



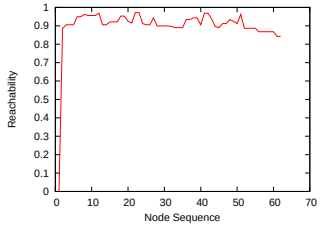
(d) Mexican Politics by Q



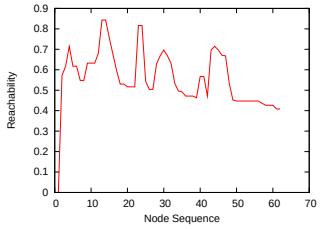
(e) Mexican Politics by S



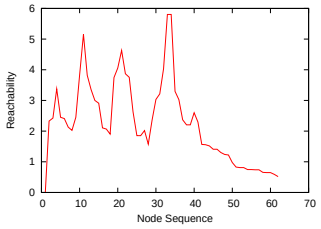
(f) Mexican Politics by R



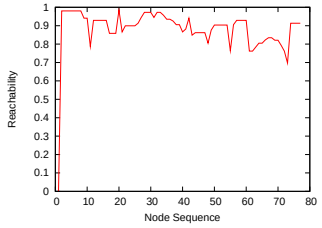
(g) Dolphin Network by Q



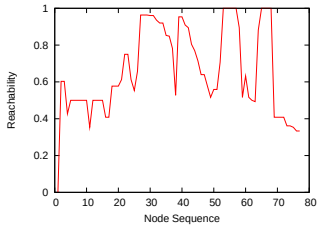
(h) Dolphin Network by S



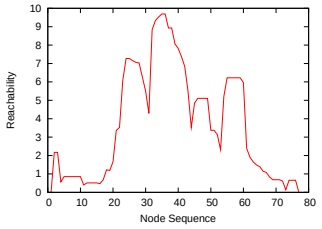
(i) Dolphin Network by R



(j) Les Miserables by Q



(k) Les Miserables by S



(l) Les Miserables by R

Figure 7.7: Comparing Metric Q, S and R with ONDOCS Visualizations

appropriate parameters are easy to obtain by means of visual data mining. The effectiveness of R over previous metrics are also confirmed by comparing ONDOCS visualizations.

Chapter 8

A Community Mining Application: Clustering Web Search Results Based on Word Sense Communities

After presenting works on community mining with different assumptions, in this chapter we propose a solution for a real problem, which is web search result clustering, based on community mining metrics and algorithms [42]. Effectively organizing web search results into clusters is important for any search engine, since it facilitates user's quick navigation through relevant documents. Traditional clustering techniques generate page clusters even when the original search result contains no ambiguity. However, in that situation the organization effort is unnecessary and will only distract the user. In this chapter, we reformulate the document clustering problem as a query sense community detection problem. Given a query and a list of documents returned by a certain web search engine, our unsupervised approach detects query sense communities in the extracted keyword network. The documents are then assigned to several refined query sense communities to form clusters. We use the modularity score of the discovered keyword community structure to measure page clustering necessity. Experimental results verify our method's effectiveness impact on real search engine data and show its advantages over traditional clustering methods. (This work has been filed as patent [225] in December 2008 and is currently under review).

8.1 Related Work

Organizing search engine results. In general, there are two ways to organize the information returned by search engines. The first, and the most popular approach, is to rank results by perceived relevance. This is applied by most popular search engines, including Google, Yahoo!, MSN Search, etc. However, this method is highly inefficient since there are usually thousands of retrieved pages for a typical query, and most users just view the few top results, possibly missing relevant information. Additionally, the criteria used for ranking might not fit the user's needs. Finally, a majority of the queries tend to be short, thus making them non-specific or imprecise [181]. A query might have multiple meanings, thus the inherent ambiguity in interpreting a word or phrase in the absence of its context means that a large percentage of the returned results can be irrelevant to the user [121].

Another way to organize documents is by *clustering*: query result pages are organized into groups based on their similarity between each other. The idea of clustering search results has already been applied in industry and commercial web services such as Vivisimo [204], Kartoo [114] and Koru [140]. Hearst et al. [100, 175] proposed the Scatter/Gather algorithm to cluster top documents returned from an information retrieval system. Zamir et al. [228, 229] proposed to cluster query result pages based on the snippets or contents of returned documents using the Suffix Tree Clustering (STC) algorithm. Chim et al. [48] combined the STC model with the TF-IDF weight to improve its accuracy. In [121], the authors proposed a monothetic clustering algorithm to assign documents to clusters based on a single feature, which is used as cluster labels. In [56] search engine results are grouped in a tree of word meanings using WordNet. Zeng et al. [230] proposed a supervised learning approach to extract relevant phrases from the query result snippets, which are used to group search results. Wang et al. [209] proposed an approach to learn from search logs for a more user-oriented partitioning of the search results. Similarly, Freyne and Smyth [77] recently proposed a Collaborative Web Search (CWS) approach to record and reuse search histories to learn a preference model for a user group.

There are other methods for document clustering [194] and document categorization [111] but do not apply to the search engine result groupings as they either require training or are not efficient enough for speedy realtime categorization. To summarize, all of the above methods are able to find document clusters but the effectiveness in practical search engine result partitioning is questionable. In addition, the performance of supervised classifiers is limited by the training data, which is hard to achieve for the dynamic web. Moreover, none of the proposed methods measures clustering necessity.

Word Sense Disambiguation. Selecting the most appropriate sense for an ambiguous word in a sentence, i.e., Word Sense Disambiguation (WSD), is a central problem in Natural Language Processing (NLP). The solution of this problem impacts many other tasks, including reference resolution, coherence and inference. WSD methods can be generally classified into three types: methods that make use of information provided by dictionaries [174, 137, 127], e.g. Wordnet [69]; methods that use information gathered from training on a corpus (supervised training methods) [159, 50]; and methods that use information gathered from the raw data (unsupervised methods) [222, 24]. Similar to our approach, Pantel et al. proposed an unsupervised centroid-based clustering method to automatically discover word senses [172]. As Chen et al. suggested in [41], text classification methods are helpful for organizing search results, since keywords and phrases can also be treated as features of web pages. While previous WSD works focus on distinguishing word senses, we further create representative keyword communities related to discovered senses of a query in order to categorize search result pages into clusters.

8.2 Preliminaries

8.2.1 Query Sense Community

The *Contextual Hypothesis for Sense* states that the context in which a word appears can be used to determine its sense [184]. For example, a web page discussing Jaguar as a *car* is likely to talk about other types of cars, car companies, etc. Whereas, a page on Jaguar the *cat*, is likely to contain information about animals. Naturally,

the words or phrases that frequently appear in the document together with Jaguar the car, e.g., *engine, Ford* and *vehicle*, are very unlikely to also appear frequently in web pages about Jaguar the cat, and vice versa. Moreover, there are extremely few pages that discuss both senses of Jaguar in detail at the same time. Therefore, the senses of the word Jaguar and other words or phrases that frequently appear together with one of the senses can be used to cluster search result pages for this particular query "jaguar", which is the intuition of our approach.

We define a query sense community as *a group of words or phrases that co-occur together frequently in a set of search result pages for a particular query*. There are a variety of definitions for "keyword co-occurrence", e.g., two words appear in the same document, within a given word range distance d or in the same paragraph, etc. We define two words to "co-occur" if they are located in the same sentence. Note that similar to many WSD algorithms in NLP area, e.g. [161], we only treat nouns as keywords in our approach and ignore other lexical categories such as verbs, adjectives, adverbs and pronouns to avoid noises. Words in these categories may appear frequently in various contexts, thus they can be strongly related to several senses of the same query in the same time. Also note that our query sense does not equal to but includes the traditional understanding of word sense. For example, as shown in Table 8.3 of Section 8.4.4, the query "tiger woods" has three query sense communities, which are related to the video game named by the player, his golf career and his new-born child. All these communities are about the very same "Tiger Woods" but are discussed in different page set s, thus we treat them as separate "senses" of the query. However, our method merges communities that share the same word sense and are from the same pages, as we explain in Section 8.3.4.

Now assume that we have generated a keyword graph G, where each node represents a unique keyword/phrase and the edge between two nodes represents their relations, the edge weight w is the frequency of the two corresponding keywords co-occurring in one sentence in the documents (details are given in Section 8.3.1), how can we find word sense communities on this graph? To solve this problem, we extend Newman's Modularity metric [155] to its weighted version and then use a

hierarchical algorithm on the keyword graph to detect query sense communities.

8.2.2 Extending Modularity Q

In this dissertation, we have proposed several metrics, which are designed for community mining in different situations. The Max-Min Modularity in Chapter 5 gives the best performance if complete network information and domain knowledge are available. The local metric L in Chapter 6 only detects a local community for given starting nodes. The visual data mining approach in Chapter 7 is designed especially for overlaps, with visual aids to determine the parameters. Unfortunately, all of these new metrics are inappropriate for the word sense community discovery problem. For this problem, in particular, we have access to the complete network, thus it does not belong to the local community discovery problem. Moreover, we do not have domain knowledge and it is very hard to generate such knowledge because the web context is too dynamic to model, even for the best linguists, thus the Max-Min modularity is not appropriate. Finally, word sense communities may have overlaps, but global thresholds for hubs and outliers do not exist. The option of providing visualizations for every particular query is unrealistic for a search engine, and would also increase the complexity of the user interface. Therefore, in order to find query sense communities from the weighted keyword network, we use a different metric by extending the modularity to its weighted version. We believe that it is the best available metric for this particular problem, after analyzing all other possibilities. The modularity extension (the original definition can be found in Chapter 3) is as follows:

Given an undirected network $G = (V, E)$, $|V| = n, |E| = m$, let A_{xy} be an element of the adjacency matrix of G.

$$A_{xy} = \begin{cases} w & \textit{if vertices x and y are connected} \\ 0 & otherwise \end{cases}$$

where w is the edge weight. Also, $P_{xy} = \frac{w_x w_y}{2W}$ where w_x is the total weight of all edges connect to x and W is the total weight of the network. $Q_{weighted}$ equals to:

$$Q_{weighted} = \frac{1}{2W} \sum_{xy} [A_{xy} - P_{xy}] \phi(C_x, C_y) \tag{8.1}$$

Assume node x belongs to community C_x, the ϕ function $\phi(C_x, C_y)$ is 1 if C_x and C_y are the same community and 0 otherwise.

8.3 Our Approach

The basic idea of our approach is to cluster search results based on the query sense communities discovered by community mining algorithms using the modularity metric. Given an input query, the general procedure of our approach (shown in Figure 8.1) is listed as follows.

- Phase I: Extract keywords from crawled documents. Keywords may be any sequence of characters that might be searched, and include words and phrases in any language.
- Phase II: Build a keyword graph based on the corresponding keyword lists.
- Phase III: Find query sense communities using a clustering or community mining algorithm on the word graph.
- Phase IV: (optional) Refine detected communities by merging similar ones and deleting outliers. (An outlier here is a community that is too small).
- Phase V: Assign pages to sense communities to generate and label clusters.

8.3.1 Phase I: Keyword Extraction

The first step to discover query sense communities from web pages is to build the keyword network, weighted by the frequency of the co-occurrence between keywords in sentences. However, keyword extraction is usually too slow for realtime search, thus it should be done offline and the keyword lists stored and indexed together with the crawled web pages by the search engine during the crawling process whenever possible. For practicality, we proceed on-line as follows: given a query q, we send q to the Google search engine and retrieve the top k returned web pages. We parse the content of text pages (such as pages ending with *.html, .php*, etc.) and ignore multimedia pages since we are not able to extract keywords from them.

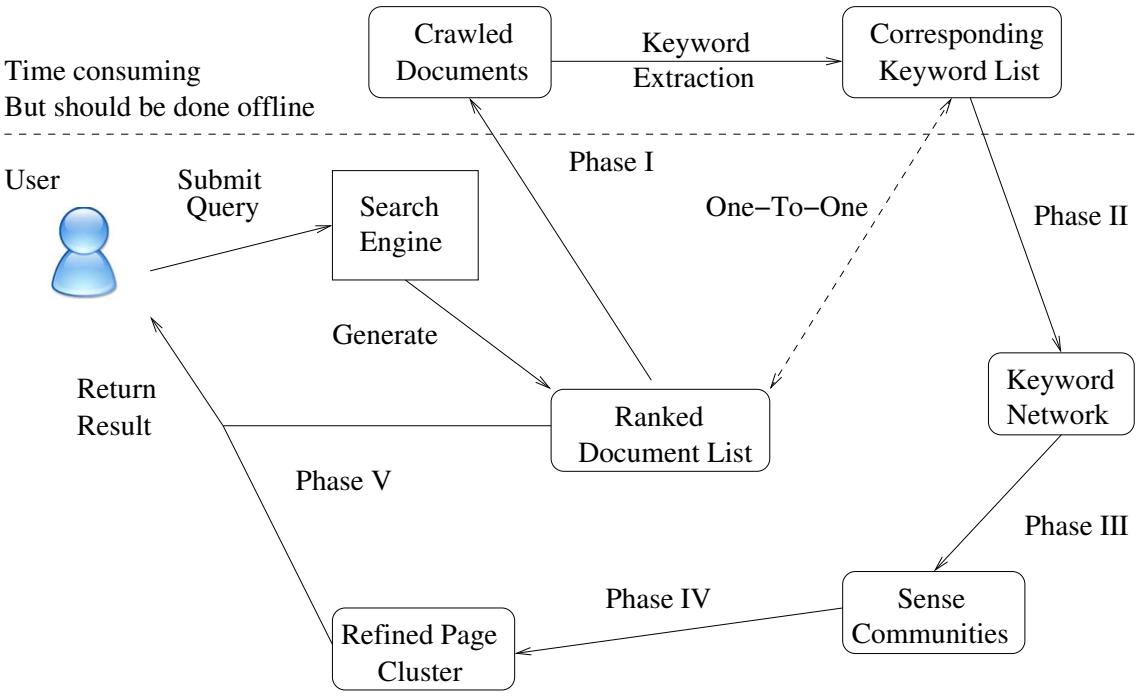


Figure 8.1: Web Page Clustering based on Query Sense Communities

Irrelevant information such as HTML tags and javascript code is stripped and only text is recovered.

We use Minipar [129], a broad-coverage English parser, to parse the clean text. Minipar is able to transform a complete sentence into a dependency tree and classify words and phrases into lexical categories. All retrieved keywords are then stemmed using the Porter Stemming Algorithm [190] and stopwords are removed. Again, note that although keyword extraction here is slow since Minipar parsing is time-consuming (500 words/second on a normal PC [172]), it should be done offline during web page crawling and indexing by the search engine, therefore the running time of our approach during query time is not affected by this step.

8.3.2 Phase II: Generate Keyword Graph

After text parsing and stemming, each page is represented as a list of pairs of keywords, confirmed to be in the same sentence. Assume a search engine now receives a query q from a user, it generates a ranked document list consisting of k result pages and also provides k keyword lists corresponding to these documents, as shown in

Figure 8.1.

The first online phase of our approach is to build a keyword graph from those k keyword relation lists, where each node represents a keyword and edges between nodes represent relations between corresponding keywords while the edge weight w is the co-occurrence frequency of the two keywords. However, it is neither accurate nor practical to include all listed keywords in the graph since words that only appear a few times in the documents are usually unrelated to the particular sense or the context. Moreover, a larger graph would make our approach slower and less effective for realtime search. In order to find important keywords in these documents to build the keyword network, we measure the importance of keywords by the Document Frequency (DF) score, which is calculated by dividing the number of documents containing the keywords by the number of all documents. We then select those keywords that have DF score higher than a given threshold t_{df} (discussed in Section 8.4.3) to be the nodes of the keyword network. (Note that stopwords are removed before this.) Two nodes are connected if we find the pair of corresponding keywords in any list and the weight of the edge is the pair frequency over all lists. The words in query q are removed since they certainly belong to all sense communities. We use the DF score instead of TF-IDF to filter keywords because we want to build a keyword network that represents a global property of this document set. Unfortunately, the TF-IDF score only indicates the importance of a keyword for a particular document. Therefore, the DF score is used to select these generally important keywords. Whether they have strong community structure or not is revealed by the community mining algorithm and the modularity score.

8.3.3 Phase III: Finding Query Sense Communities

Any modularity-based clustering algorithm could be applied here to find query sense communities. Here we adapt a hierarchical clustering algorithm to find the clusters by greedily optimizing the modularity score [52]. It starts with every node being a community of its own, then at each step, it merges a pair of communities that increase the overall modularity the most and stops when there is no such pair. Since a high modularity score represents strong community structure, the intuition

of this algorithm is to greedily optimize the overall modularity. Details of the algorithms are as follows: Given a weighted keyword network, three data structures are maintained.

- A sparse matrix containing ΔQ_{ij} for each pair i, j of communities with at least one edge between them. Each row of the matrix is stored as a balanced binary tree and as a max heap. The ΔQ matrix is initialized as follows.

$$\Delta Q_{ij} = \begin{cases} \frac{1}{2W} - \frac{w_x w_y}{(2W)^2} & \textit{if i and j are connected} \\ 0 & otherwise \end{cases}$$

- A max-heap H containing the largest element of each row of the matrix along with the labels i, j of the corresponding communities.

- A vector array with element $a_i = \frac{w_i}{2W}$

The algorithm then greedily merges pairs of communities that give the highest modularity gain as follows:

- Pop the max-heap with the largest element of each row of the matrix ΔQ.

- Select the largest ΔQ_{ij}, merge the two communities, update ΔQ (described blow), the heap H and a_j ($a'_j = a_i + a_j$), increment Q by ΔQ_{ij}.

- Repeat until there is no $\Delta Q_{ij} > 0$

Merging community i and j by updating ΔQ as follows.

$$\Delta Q'_{jk} = \begin{cases} \Delta Q_{ik} + \Delta Q_{jk} & \textit{if community k is connected to both i and j} \\ \Delta Q_{ik} - 2a_j a_k & \textit{k is connected to i but not to j} \\ \Delta Q_{jk} - 2a_i a_k & \textit{k is connected to j but not to i} \end{cases}$$

The complexity of this algorithm is $O(mD \log n)$, where m is the edge number, n is the node number and D is the depth of the cluster dendrogram [52]. While other online phases run in linear time, the efficiency of our approach depends on the community detection algorithm. In a web search, the number of returned documents can be huge. However, since we only select keywords with a high DF score, the size of the keyword graph is stable in the thousands in our experiments, thus the running

time of the algorithm is a matter of few seconds on a single PC, which is fast enough for a search engine service. Moreover, the modularity score as an outcome of the algorithm can be used to measure the strength of the sense community structure, which indicates whether document clustering is necessary or not, as can be seen in Section 8.4.4.

8.3.4 Phase IV: Community Refinement

Using a modularity-based clustering method, we have discovered query sense communities from the weighted keyword network. However, before they can be used to categorize documents, we need to refine the structure for the following two situations:

- Delete noise communities, which are formed by keywords that always co-occur no matter what the page is about, e.g., we observe keywords *trademark, privacy* and *policy* always form a strong three-words-community.
- Merge communities that focus on different aspects of the same query sense and appear in the same page set, therefore it is not necessary to separate them. e.g., we observe two communities for the sense of *eclipse* as *astronomical event*, one focus on how to observe the event, the other introduces the phenomenon from a scientific prospective, and they are from the same page set.

Fortunately, simple heuristics can be applied to solve the problem. For noise communities, we observe that they are usually small in size. Therefore, we remove all communities that have fewer nodes than 5% of the total keywords (note that this threshold is stable enough. Varying it from 5% to 10% does not affect the result). For merging communities, recall that we assume that there is only one primary query sense for a given query in one web page, thus if two communities share the same query sense, they are likely to be covered by the same pages. Therefore, we calculate the overall TF-IDF score (described in Section 8.3.5) of pages for these two communities, and compare the two sets of pages whose scores exceed a threshold t_{merge} (discussed in Section 8.4.3). If the size of overlapping pages is

more than half of one of the page set, we merge the two communities in question. These heuristics work well as shown by our experiments in Section 5.2.

8.3.5 Phase V: Assign Documents to Labeled Communities

Our final step is to assign pages to communities and label them. In order to assign a page p to its most related query sense community, we calculate the overall TF-IDF score of p for all communities and assign p to the one that has the highest score. If more than one candidate community has the highest score, we categorize p as *miscellaneous*. The overall TF-IDF score of p for community c is defined as the sum of TF-IDF scores of all keywords, which belong to community c, for p.

We use the dependency-based word similarity data[1] [130] to label the clusters. For a keyword w in community c, we use the number of words in c that are similar to w as an overall ranking for w. We select keywords that have high overall ranking as c's label. We also tried other methods to label the clusters. For example, we have labeled the clusters by the most frequent words or words with the largest neighbourhood. However, their accuracy is not promising judging by the manual labels. More accurate document cluster labeling methods may apply here; however, cluster labeling itself is a research topic per se and thus is beyond the scope of this application.

8.4 Experiment Results

We conducted several experiments to validate the effectiveness and accuracy of the proposed approach. We use the ARI metric, which is introduced in Chapter 5.2.2, to evaluate the accuracy of our approach.

8.4.1 Data Collection and Labeling

We constructed our datasets using the Google search engine and partitioned the results manually to create the ground truth for evaluation purpose. At first, we submitted ambiguous queries to Google and parsed result pages in the top returned

[1]The dependency-based word similarity data can be downloaded at http://www.cs.ualberta.ca/∼lindek/downloads.htm

results. Pages that do not contain any keyword pairs were removed. To evaluate our idea of community mining for page clustering, we intentionally merged result pages from several related queries to create datasets that have strong query sense communities, e.g., the *amazon* dataset is merged with results from queries "amazon river", "amazon warrior" and "amazon company", and so is the *java* (island, coffee or programming language) and *eclipse* (Mitsubishi car model, obscuring of a celestial body, or programming development platform) datasets. To evaluate the approach on real queries, we queried the word "jaguar" and "salsa" only to create *jaguar* (animal, car or mac system) and *salsa* (dance/music or recipe) datasets. Table 8.1 shows the labeled datasets that are used for accuracy experiments.

In order to build ground truth to evaluate our results, we asked four graduate students to manually classify all pages into pre-defined clusters using a vote system, i.e., a page is classified to the cluster which most people agree on. If votes do not agree, we have a fifth person make the final decision to break the tie[2]. Pages can also be labeled as *miscellaneous* if the person does not think they belong to any of the pre-defined clusters. Note that while the experiment datasets are categorized by human, our approach absolutely does not rely on such manual labeling. The exercise involving human labeling is solely to gather ground truth for empirical experimentation.

For practicality, we kept the Google page sets small to be manually labeled. However, to illustrate the approach on a larger set, we used a subset of the standard text data set Reuters-21578[3] and selected three document categories with about the same size, to simulate a query with three senses. (see Table 8.1). Totalling about 950 documents, each is treated as a parsed page for a query.

8.4.2 Accuracy Evaluation

We have applied our method on the six datasets and show the results in Table 8.2 (the miscellaneous cluster for all datasets are omitted in the table). We compare our method with an effective variation of K-Means [113], which is a common algorithm

[2]The labeled dataset can be downloaded at http://www.cs.ualberta.ca/∼jiyang/WWW/.
[3]http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

Dataset	Manual Labels	Page Set Size
amazon	river, warrior, company	114
java	software, island, coffee	119
eclipse	car, astronomy, java	125
jaguar	car, animal, mac	101
salsa	dance, sauce	85
Reuters*	Trade, Crude, Money-fx	946

Table 8.1: Experimental Datasets.

of document clustering [189]. Note that while other document clustering algorithms [21, 229, 228] usually require parameters as metric thresholds, which are extremely hard to set generally for various search queries, we do not need any parameters for the clustering.

For the K-Means algorithm, every extracted keyword is treated as a feature, thus one document is represented as a vector of keyword TF-IDF scores. The distance between two documents is defined as the squared Euclidean distance between two vectors. Note that we use the final community number discovered by our approach as k to feed the k-means algorithm. While our approach detects k automatically based on senses of query words, other unsupervised algorithms often rely on such critical parameter, thus our approach is more appropriate for real time search result page clustering, where such information is unavailable.

From the table, we see that for datasets (*amazon, java, eclipse*) that are merged by three different senses of the same word, our approach achieves high accuracy, which validates our assumption that a query sense community relates to its corresponding document cluster. For real datasets (*jaguar, salsa, reuter*) with noise, our method can no longer achieve ARI score higher than 0.8. However, our method still works measurably better than the K-means algorithm and detects the right number of clusters. Note again that the running time of our clustering approach depends on the size of the *keyword network* while other clustering methods (etc. K-means) rely on the size of the document set, thus our approach is able to handle large document sets very fast since the number of extracted keywords is always stable regardless of the number of returned pages.

DataSet	Manual Label	Extracted Label	ARI score		Q score
			Our Method	K-means	
Amazon	River	lake, river, water	0.888	0.693	0.367
	Warrior	girl, battle, woman			
	Company	computer, consumer			
Java	Coffee	coffee, fruit, tea	0.889	0.728	0.403
	Island	island, mountain, city			
	Software	software, interface			
Eclipse	Car	engine, car, video	0.931	0.765	0.428
	Astronomy	sun, picture, moon			
	Java	software, interface			
Jaguar	Animal	animal, wildlife, forest	0.785	0.114	0.471
	Car	car, vehicle, sedan			
	Mac	database, software			
Salsa	Dance	music, dance, teacher	0.642	0.605	0.405
	Sauce	garlic, tomato, sauce			
Reuter	Trade	budget, tax, tariff	0.618	0.504	0.222
	Crude	oil, crude, supply			
	Money-fx	currency, market, dollar			

Table 8.2: Sense community-based clusters for six datasets (miscellaneous clusters are omitted).

8.4.3 Parameter Setting

Parameter t_{df} is used to select keywords based on their DF score. The lower it is, the more noise keyword we get in the network. On the other hand, our approach might not be able to detect small-scale communities if it is set too high since the keywords in those communities are already filtered out. In Figure 8.2, we vary t_{df} from 0.03 to 0.1 with step 0.01 for several datasets. The parameter value for the optimal performance depends on the keyword community structure of the dataset in question, however, we see that best results are mostly obtained in range $0.03 \leq t_{df} \leq 0.05$.

We further study the threshold t_{merge} for community refinement. In experiment datasets, we observed merging in the *java* dataset, where the *software* sense splits into *programming* and *company development*, and the *salsa* dataset, where the *dance* sense splits into *teaching salsa* and *introducing salsa*. We vary t_{merge} from 0.05 to 0.5 with step 0.05. For the *java* dataset, communities are mistakenly merged when $t_{merge} \leq 0.1$, and correct merging stops when $t_{merge} \geq 0.4$. Corresponding

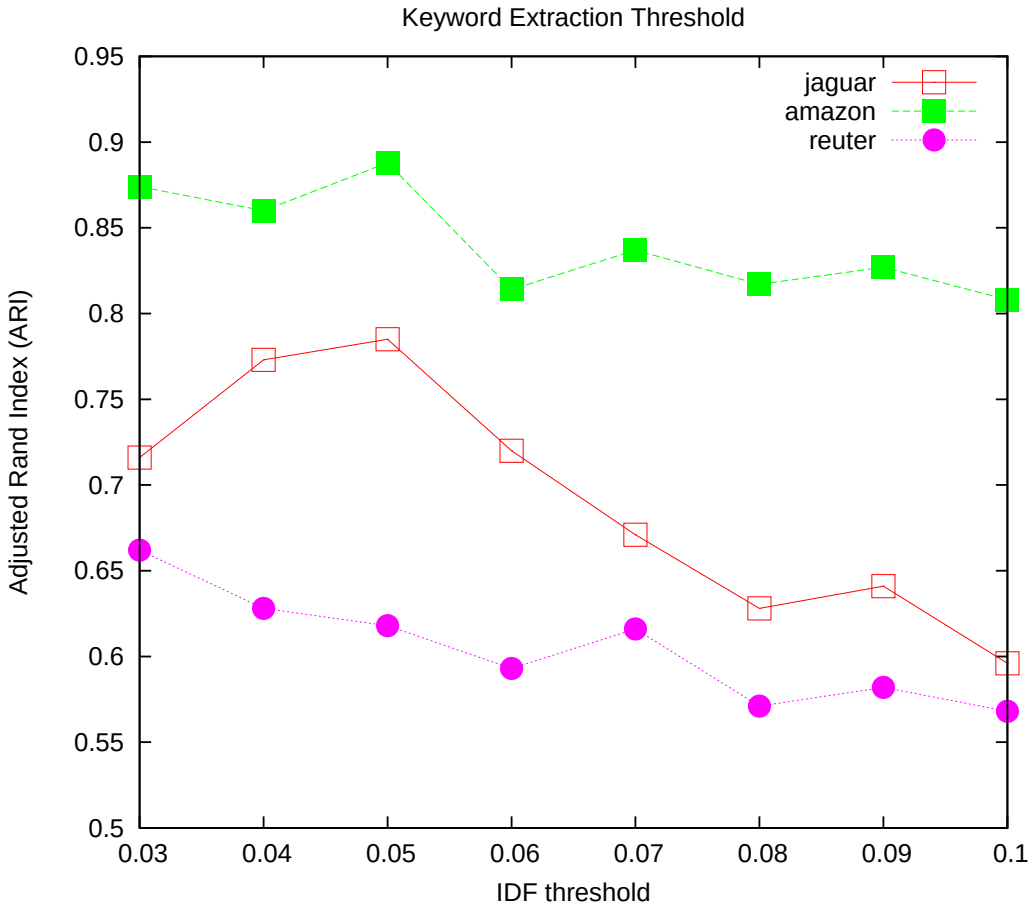


Figure 8.2: The impact of threshold t_{df}

values for the *salsa* dataset are 0.05 and 0.3. Therefore, the merging accuracy is not very sensitive to the parameter t_{merge}; a good range can be $0.15 \leq t_{merge} \leq 0.25$. In all of our experiments, we set $t_{df} = 0.05$ and $t_{merge} = 0.2$.

8.4.4 Using Q to Measure Need For Clustering

Obviously, a list of result pages does not need clustering if it only contains one primary sense of the query. The stronger its sense community structure, the more confusing the result could be. Fortunately, our method allows the strength of the community structure to be measured by the Q score, which is an outcome of the hierarchical clustering algorithm. The Q score is close to 0 if there is only one sense. The more separated the discovered communities are, the higher the Q score is and the better the search result documents are to be clustered. Typically, in Social

Network Analysis, $Q \geq 0.3$ indicates strong community structure [155]. However, in our experiments, we observe that 0.1 might be a good threshold to decide whether to cluster or not for the page clustering task.

In Table 8.3, we show automatic-labeled page clusters and corresponding Q scores of the detected sense community structure of various queries. For ambiguous queries that have strong sense communities, we further manually refine and submit the query to our system until the returned Q score is close to 0, which indicates no more clustering is necessary. As we can see, the results are very promising. The query *columbia* has many communities including educational institution, river, state, court, film company, etc. but the refined query *british columbia* has only one sense. There are two communities, vehicle and planet, for query *saturn*, and only one for query *saturn car*. *Matrix* provides three communities, which are computer tools, the movie and math, and no further clustering is required for *matrix movie*. Table 8.3 also shows other similar examples, such as *blizzard* (weather and company), *latex* (file format and allergy), *trailblazer* (scientific researcher, book series, electronic technology and car model) and *mouse* (laboratory experiment, computer device, animal and Mickey Mouse). Surprisingly, we have to refine our query twice to achieve a query string with no ambiguity for query *tiger* (animal, business, sports and software) and query *casablanca* (movie and city). The reason is that our refined query *tiger woods* (game, career and child) and *casablanca city* (capital and hotel) still show some community structure, although with a lower Q score.

To summarize, our approach generates a high Q score for queries that have several meanings in the web search context. The score decreases when we further refine the query to eliminate some of the meanings and finally reaches or becomes very close to 0, which indicates no community structure in the returned documents, i.e., there is only one main sense for this query. Therefore, the Q score should be a strong candidate for the necessity metric of page clustering.

8.5 Conclusions

This chapter proposed an approach for search result organization based on query sense communities in web pages. Our unsupervised method not only bypasses the problem of handling large result page sets by extracting and analyzing frequent keywords and phrases, but also it is able to achieve high clustering accuracy for real queries. We also show that the modularity score Q can be used to measure whether a page clustering is required. Experimental results confirm the accuracy and effectiveness of the proposed approach. Possible future work would be extending the approach to build a document hierarchy based on merged topics and discovered senses and providing a more accurate cluster labeling.

Query	Extracted Cluster Label	Q Score	Refined Query	Q Score
columbia	student, professor, institute park, town, area order, court, request river, water, lake state, district, county market, film, product season, mph, game	0.369	british columbia	0
saturn	vehicle, technology, model heat, water, hydrogen	0.259	saturn car	0.012
matrix	system, tool, database character, film, movie order, equation, rule	0.330	matrix movie	0.033
blizzard	snow, wind, weather software, product, computer	0.234	blizzard game	0.006
latex	file, format, user patient, treatment, hospital	0.401	latex allergy	0
trailblazer	student, professor, director boy, kid, book network, phone, technology car, vehicle, engine	0.324	trailblazer chevrolet	0.076
mouse	model, study, cell interface, keyboard, device animal, rat, cat film, art, character	0.316	mouse keyboard	0
tiger	animal, wildlife, habitat business, market, industry game, team, player software, user, version	0.356	tiger woods	0.209
tiger woods	game, mode, player tournament, career, record daughter, kid, child	0.209	tiger woods daughter	0.033
casablanca	movie, film, theater city, service, hotel	0.270	casablanca city	0.145
casablanca city	capital, town, region restaurant, hotel, park	0.145	casablanca city hotel	0.004

Table 8.3: Modularity score for different queries.

Chapter 9

Entity Ranking for Social Networks

The significant increase in open access digital information has created incredible opportunities for modern social network analysis and data mining research, especially in exploiting significant computational resources to determine complex relationships within entities. In this chapter, we consider the problem of analyzing heterogeneous social networks and explaining relationships between entities in order to rank entities based on a notion of relevance. For this purpose, we propose a link analysis-based method from the random walk class, which can be deployed to discover interesting relationships amongst entities of relational social networks that would otherwise be hard to expose [226]. We demonstrate our ideas on the DBLP database, where we exploit structural variations on relationships between authors, conferences and topics. We also present a navigational interface to view the ranking.

9.1 The Network Model

As mentioned in Chapter 4.5, we can consider the web as a network over a binary relation between pages. It can be well represented by a bipartite graph, where two partitions contain the page set and edges between partitions represent the hyperlinks that connect corresponding pages. We now generalize this idea to heterogeneous social networks.

Given a social network $D = (X, Y, R_{X \leftrightarrow Y})$, where entity set $X = \{x_i | 1 \leq i \leq n\}$ and relation set $Y = \{y_j | 1 \leq j \leq m\}$ (there are n entities in X and m relations

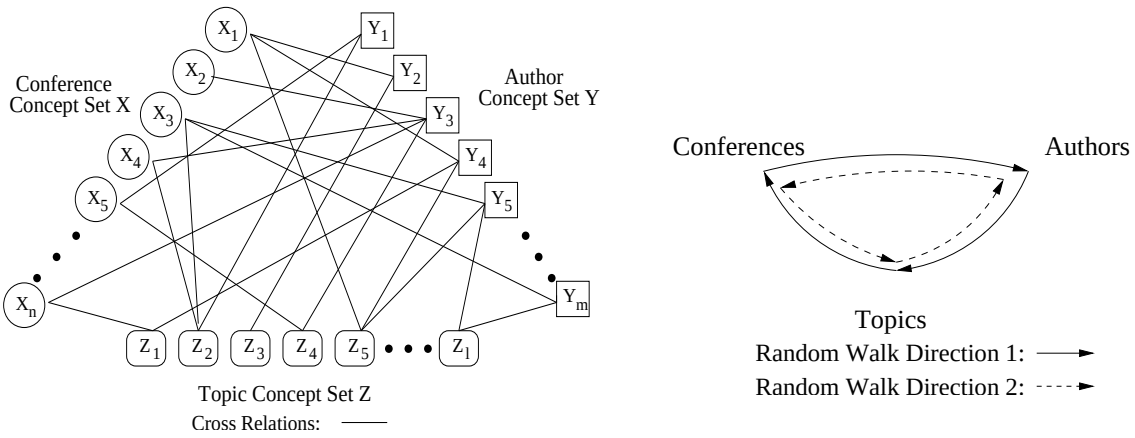
in Y), $R_{X \leftrightarrow Y}$ represents the connections between X and Y: $r(x_i, y_j) \in R_{X \leftrightarrow Y}$ if x_i has relation y_j, e.g., if x_i represents an author and y_j represents a conference, $r(x_i, y_j)$ could mean that the author x_i has published a paper in conference y_j. We refer to $R_{X \leftrightarrow Y}$ as the *cross relation* between X and Y.

9.1.1 Bipartite Network Model

As we have shown in Chapter 4.5, a social network $D = (X, Y, R_{X \leftrightarrow Y})$ can easily be transformed to a undirected bipartite graph $G = (X, Y, E)$, where X and Y are node sets ($X = \{x_i \mid 1 \leq i \leq n\}$, $Y = \{y_j \mid 1 \leq j \leq m\}$) and E is an edge set ($E = \{e(x_i, y_j) \mid r(x_i, y_j) \in R_{X \leftrightarrow Y}\}$), i.e., each partite represents an entity/relation set and edges represent connections between entities and relations. We assume the edges are undirected, but they can also be directed for some social networks, such as the web. The weight of edge $e(x_i, y_j)$ is $w(x_i, y_j)$, which can be 1 for an unweighted graph, or non-negative value for a weighted graph, where the weight is the sum of the weights of relation y_j edges that connect entity x_i in the original network. One important property of the structure of the bipartite model is that nodes in X that are similar to each other usually connect to similar nodes in Y. Therefore, we could achieve the ranking by generating the adjacency matrix of graph G and applying a random walk algorithm on that matrix starting from a given node.

9.1.2 K-partite Network Model

We have presented the bipartite network model for a social network $D = (X, Y, R_{X \leftrightarrow Y})$ which has only one type of cross relation, namely $R_{X \leftrightarrow Y}$. However, real world networks might have multiple cross relations between several types of entities and relations. For example, there are seeds, peers and file types (multimedia, text, etc.) for P2P system; conferences, authors and research topics for research publication data; customers, movies, genre and actors for a movie purchase database. We may have more correlated entities in other fields, however, in this dissertation, we experiment on graphs that have cross relations between three types of entities, network models for more can be achieved by extending the graph model and applying the proposed method.



(A) Tripartite Network Model (B) Random walk on Tripartite Graph

Figure 9.1: Tripartite Network Model for Multiple Cross Relations

We now consider a social network $D' = (X, Y, Z, R_{X \leftrightarrow Y \leftrightarrow Z \leftrightarrow X})$, and we have $R_{X \leftrightarrow Y}$, $R_{Y \leftrightarrow Z}$ and $R_{X \leftrightarrow Z}$ as cross relations between these three different types of entities. We naturally use an undirected tripartite graph $G' = (X, Y, Z, E')$ to model the data: two entity nodes are connected if they are related. Figure 9.1 (A) shows the tripartite network model for the social network.

Applying random walk algorithms on a k-partite graph can be interesting, since the direction of the random walk need also to be taken into consideration. For example, assume we have a tripartite network model representing relations between conferences, authors and topics (Figure 9.1 (B)), there are two possible ways to apply a random walk: assuming we start from authors, walk from author to conference to topic and then back to author, or walk from author to topic to conference and then back to author. Random walk algorithms with both directions rank authors based on the frequency of sharing the same topics and conferences, but there are slight differences due to the walking sequence. Nevertheless, starting from an entity in a particular partition, the random walker need to walk through all related partitions and then return to the beginning partition. By that we no longer need to arbitrarily set the authority transfer probability [103], however the sequence of involved partitions are required for the random walker to jump among different types of entities. We investigate this problem in more details in Chapter 9.3.3.

9.2 Proposed Method

In this section, we first define the relevance score and describe the random walk approach in Section 9.2.1, then present the ranking algorithm for entity sets with multiple cross relations in Section 9.2.2.

9.2.1 Relevance Score based on Random Walk

For the problem of relevance ranking in the network model, we believe that two objects are similar to each other if they are related to similar objects. Therefore, we consider that an entity is most related to itself and can be assigned a score of 1. We denote the relevance score between entities α and β by $rs(\alpha, \beta)$. We know that $rs(\alpha, \beta) \in [0, 1]$ and $rs(\alpha, \beta) = 1$ iff $\alpha = \beta$. Now the problem of internal and external ranking in the network model can be described as follows:

Given a node α in partition X of the network model, we want to compute a relevance score for all nodes $\beta \in X$ (internal ranking) and all nodes γ in other partitions (external ranking). For each partition, the result is a one-column vector containing all relevance scores of the entities with respect to α.

The basic intuition behind our approach is to apply random walks with restart (RWR) from the given node α, and count the visitation frequency that the walk does on each node in the k-partite network model, i.e., the relevance score of node β is defined as the probability of visiting β via a random walk which starts from α and goes back to α with a probability c. In more detail, RWR works as follows: assume we have a random walker that starts from node α. For each step, the walker chooses randomly among the available edges from the current node it stays. After each iteration, it goes back to node α with probability c. The final steady-state probability that the random walker reach node β is the relevance score of β with respect to α: $rs(\alpha, \beta)$. We choose the random walk approach to compute the relevance score because it gives node β high ranking if β and α are connected by many nodes; this is because the random walker has more paths to reach β from α. The purpose of the periodic restart of the random walk is to raise the chance that close related nodes are visited more often than other nodes.

9.2.2 Algorithm for Multiple Cross Relations

For a social network with multiple cross relations, we first model the related entities as an undirected k-partite graph. Without loss of generality, we present our algorithm on a tripartite network model $G' = (X, Y, Z, E')$, it can be easily extended to more complex models. Assume we have n nodes in X, m nodes in Y and l nodes in Z, we can represent all relations using three corresponding matrices: $U_{n\times m}$, $V_{m\times l}$ and $W_{n\times l}$.

In matrix U, $U(\alpha, \beta)$ denotes that there is an edge from node α to node β in G'. If we want to initiate a random walk starting from a node in X represented by row α to Y, the probability of taking the edge (α, β) is proportional to the edge weight over the weight of all outgoing edges from α in X to Y. Therefore, we normalize them such that every column sum up to 1: $Q(U) = col\_norm(U)$, $Q(U^T) = col\_norm(U^T)$. We then construct the adjacency matrices of G' after normalization:

$$J_{XY} = \begin{pmatrix} 0 & Q(U) \\ Q(U^T) & 0 \end{pmatrix}$$

$$J_{XZ} = \begin{pmatrix} 0 & Q(W) \\ Q(W^T) & 0 \end{pmatrix}$$

$$J_{YZ} = \begin{pmatrix} 0 & Q(V) \\ Q(V^T) & 0 \end{pmatrix}$$

We now transform the given node α into a $(n + m + l) \times 1$ vector $\vec{u}_\alpha$ initialized with all 0 except the value for start node α, which is set to 1. Then we need to achieve a $(n + m + l) \times 1$ steady-state vector $\vec{u}_\alpha$, which contains relevance scores over all nodes in the graph model with respect to α. The steady-state vector can be computed based on the following lemma using the RWR approach. Note that, in the lemma, vector $\vec{v}_\alpha$ is initialized in the same way as $\vec{u}_\alpha$ and is representing the probability of restarting random walk in each iteration.

Lemma 1 *Let c be the probability of restarting random walk from node α and P is the adjacency matrix. Then the steady-state vector $\vec{u}_\alpha$ satisfies the following equation:*

$$\vec{u}_\alpha = (1 - c)P\vec{u}_\alpha + c\vec{v}_\alpha$$

Proof: The adjacency matrix P is a Markov matrix, where every entry of P is nonnegative and every column of P adds to 1. We know that multiplying a nonnegative $\vec{u}$ by a Markov matrix produces a nonnegative $\vec{u}'$, whose components also add to 1. We have the pattern for Markov chains: if P is a positive Markov matrix, then its eigenvector is the steady state vector. More details for the proof can be found in Chapter 8 in Strang's book [191].

Algorithm 7 applies the above lemma repeatedly until $\vec{u}_\alpha$ converges. For all experiments, the restarting probability c is set to 0.15 and converge threshold ϵ is set to 0.1, which give the best performance for RWR according to experiment results in [193]. In step 3, the k-partite structure of the graph model is used to save the computation of applying Lemma 1. The result vector $\vec{u}_{\alpha(1:n)}$, $\vec{u}_{\alpha(n+1:n+m)}$ and $\vec{u}_{\alpha(n+m+1:n+m+l)}$ respectively represent vectors of first n, m and last l elements of $\vec{u}_\alpha$, and contain the relevance score for X, Y and Z nodes.

In algorithm 7, the random walk starts from X to Y, then to Z and finally returns to X. The algorithm applies the random walk repeatedly until the vector converges. As mentioned in Section 9.1.2, there are several possible directions of random walks in a k-partite network model, which could lead to different ranking results. Algorithm 7 describes the direction of $X \rightarrow Y \rightarrow Z$. The computation sequence and involving matrices in step 3 of the algorithm would be different if the random walk direction is changed.

9.3 Experiments

We tested our ranking approach on the DBLP database, the data structure of which is shown in Figure 9.2. As a social network, DBLP originally has only two entity types: conference and author. Papers are relations that connect these two kinds of entities. In order to test our approach on k-partite network models, we extracted a third type of entities from DBLP data, which is the research topic. Details are explained below.

In the DBLP database, we safely discarded all journal publications since they are only a small fraction compared to the whole database. We also observe that

Algorithm 7 Random Walk Algorithm for Multiple Cross Relations

Input: starting node α, tripartite graph model G', restarting probability c, converge threshold ϵ.

Output: relevance score vector $\vec{x}$, $\vec{y}$ and $\vec{z}$ for X, Y, Z nodes.

1. Compute the adjacency matrices J_{XY}, J_{YZ} and J_{XZ} of the graph model G'.

2. Initialize $\vec{u}_\alpha = \vec{v}_\alpha = 0$, set element for α to 1: $\vec{u}_\alpha(\alpha) = \vec{v}_\alpha(\alpha) = 1$.

3. While ($\Delta\vec{u}_\alpha > \epsilon$)

$\vec{u}_{\alpha(n+1:n+m)} = (Q(U^T) * \vec{u}_{\alpha(1:n)})$

$\vec{u}_{\alpha(n+m+1:n+m+l)} = (Q(V^T) * \vec{u}_{\alpha(n+1:n+m)})$

$\vec{u}_{\alpha(1:n)} = (Q(W) * \vec{u}_{\alpha(n+m+1:n+m+l)})$

$\vec{u}_\alpha = (1-c)\vec{u}_\alpha + c\vec{v}_\alpha$

4. Set vector $\vec{x} = \vec{u}_{\alpha(1:n)}$, $\vec{y} = \vec{u}_{\alpha(n+1:n+m)}$, $\vec{z} = \vec{u}_{\alpha(n+m+1:n+m+l)}$.

6. Return $\vec{x}$, $\vec{y}$, $\vec{z}$.

authors in different research areas publish in a certain group of conferences and seldom publish across multiple areas, i.e., the undirected author-conference graph can be partitioned into several nearly non-overlapping subgraphs, where few edges between authors and conferences connect across partitions. Therefore, it is unnecessary to run the algorithm on the whole graph, instead, we applied a graph partitioning algorithm first and then performed random walks only on the partition containing the given node. There are several algorithms available for graph partitioning. Note that the proposed random walk approach is independent of the selected partitioning algorithm. In our work, we used the METIS algorithm [115] to partition the large graph into 10 subgraphs of about the same size. We examined all our experiments on the biggest partition with 1,170 conferences and 35,926 authors. This partition includes most conferences in the area of Database and Data Mining, e.g., KDD, SIGMOD and VLDB.

Since DBLP data provide paper titles as the only content-related information, we obtained as many paper abstracts as possible from Citeseer [1], then extracted topics based on keyword frequency from both titles and abstracts. At first, we manually selected a list of stop words to remove meaningless but frequent words, e.g., "approach", "using". Then we counted the frequency of every co-located pairs

[1] http://citeseer.ist.psu.edu/

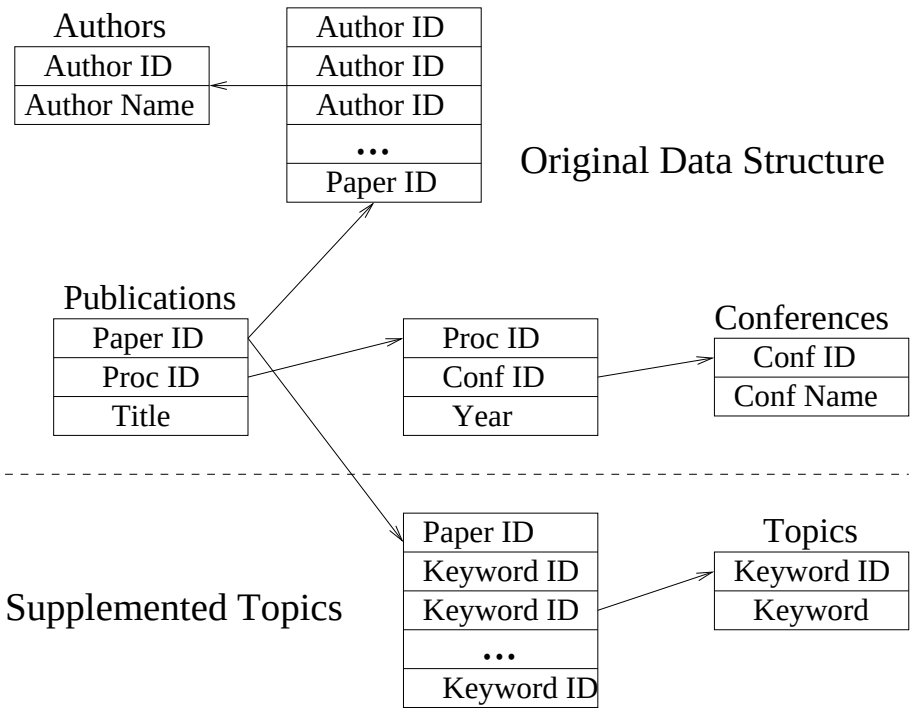


Figure 9.2: Our Data Structure extracted from DBLP Database

of words (stemmed bi-grams) and selected the top k most frequent bi-grams as our topic entities (k=1000 in our experiments). We chose to represent topics by bi-grams because most of the research topics can be well described by two words, e.g., artificial intelligence, software engineering and machine learning. Moreover, we added several tri-grams, e.g. Support Vector Machine, World Wide Web, if we observed both bi-grams from them (e.g. Support Vector and Vector Machine) to be frequent.

9.3.1 Ranking for Conference Entities

After we extracted conference-author-topic data from the DBLP database to build the network model, we checked whether entities with high relevance scores are truly related to the given node. In the following, we select related entities with top 10 relevance scores for given conferences and verify that the result makes sense in the academic context.

Table 9.1 and Table 9.2 shows the top 10 relevance ranked conferences for a given conference α in the bipartite (without topics) and tripartite (with topics) net-

α	*KDD*	*ICDM*	*SIGMOD*	*VLDB*
1	VLDB	KDD	VLDB	SIGMOD
2	ICDE	ICDE	ICDE	ICDE
3	NIPS	VLDB	PODS	DEXA
4	ICML	PAKDD	DEXA	PODS
5	SIGMOD	SIGMOD	EDBT	CIKM
6	ICDM	ICML	CIKM	EDBT
7	IJCAI	SDM	KDD	KDD
8	AAAI	IJCAI	DASFAA	BDA
9	PKDD	PKDD	SSDBM	ER
10	CIKM	NIPS	ER	DASFAA

Table 9.1: Top 10 Related Conferences for Conference using bipartite model: Conference → Author → Conference

α	*KDD*	*ICDM*	*SIGMOD*	*VLDB*
1	VLDB	VLDB	VLDB	ICDE
2	ICDE	ICDE	ICDE	SIGMOD
3	SIGMOD	KDD	DEXA	DEXA
4	NIPS	NIPS	CIKM	SAC
5	DEXA	SIGMOD	DASFAA	DASFAA
6	PAKDD	PAKDD	SAC	CIKM
7	IJCAI	DEXA	ER	ER
8	SAC	IJCAI	IJCAI	IJCAI
9	ICML	SAC	SIGIR	SIGIR
10	SIGIR	ICML	KDD	KDD

Table 9.2: Top 10 Related Conferences for Conference using tripartite model: Conference → Topic → Author→ Conference

α	*SIGMOD*	*PODS*	*SAC*	*VLDB*
1	Database System	Database System (1^{st})	Genetic Algorithm (15^{th})	Database System (1^{st})
2	Relational Database	Query Language (13^{th})	Neural Network (6^{th})	Relational Database (2^{nd})
3	Management System	Concurrency Control (17^{th})	Web Service (5^{th})	Information System (4^{th})
4	Information System	Relational Database (2^{nd})	Information Retrieval (12^{th})	Management System (3^{rd})
5	Web Service	Deductive Database ($> 20^{th}$)	Information System (4^{th})	Data Modeling (10^{th})
6	Neural Network	Query Optimization (11^{th})	Data mining (9^{th})	Data Management (7^{th})

Table 9.3: Related Topics for Conference using bipartite model: Conference → Topic → Conference ((x), x is the rank of the topic with respect to the SIGMOD conference

work model. Note we do not make any claim on which list is better, since comparing ranking results itself is still an open problem. The list based on a conference-author network implies that the given conference α shares researchers with these conferences who have high relevance scores. One can observe that our ranking puts database conferences as the most relevant conferences for SIGMOD and VLDB. For KDD and ICDM, however, it appears that database conferences are the most relevant. This is due to historical reasons since prominent data mining authors used to and still publish in VLDB, SIGMOD and ICDE venues. The ranking changed in the tripartite graph list since the network model now includes topics, which increase relevance scores of conferences that share the same research topics with the given conference. For example, after the topic set is taken into account, PODS is no longer top 10 related to SIGMOD. Instead, we see SAC appear in the list due to the similarity of covered research topics. As we can see in Table 9.3, the most related topics to SIGMOD and VLDB (1^{st} for SIGMOD in Table 9.1) are almost the same, while SAC (6^{th}) and SIGMOD share several topics as well. On the other hand, PODS (ranked as 13^{th}), which is surprisingly not in the top 10 related conferences for SIGMOD based on a topic-included network model, does not have many topics in common with SIGMOD (except "relational database" and "database system"). As mentioned before, topics in our experiments were extracted from paper titles and abstracts when available. A better topic extraction means could lead to even better rankings.

Table 9.4 shows the top 10 related authors for SIGMOD, ICDM and KDD. All of these authors are researchers in the database and/or data mining area, and have a large number of publications in DBLP with attached topics related to the topics of the conferences in question. Notice that our ranking tends to favor objects center to the social networks. In other words, if an author is highly connected to other authors related to a conference α, and is central in the social network of topics and other conferences related to the conference α, our approach would tend to rank this author high vis-à-vis this conference, which justifies why Jiawei Han for instance is ranked the highest for SIGMOD.

α	*SIGMOD*	*ICDM*	*KDD*
1	Jiawei Han	Jiawei Han	Philip S. Yu
2	Hans-Peter Kriegel	Philip S. Yu	Hans-Peter Kriegel
3	Michael Stonebraker	Jian Pei	Nick Cercone
4	Elisa Bertino	Masaru Kitsuregawa	Eamonn J. Keogh
5	David J. DeWitt	Wei Wang	Shusaku Tsumoto
6	Philip S. Yu	Eamonn J. Keogh	Wei Wang
7	Georges Gardarin	Hans-Peter Kriegel	Rakesh Agrawal
8	Elke A. Rundensteiner	Hongjun Lu	Masaru Kitsuregawa
9	Michael J. Carey	Ming-Syan Chen	Christos Faloutsos
10	Hongjun Lu	Osmar R. Zaïane	Jian Pei

Table 9.4: Top 10 Related Authors for Conference using tripartite model: Direction Conference $\rightarrow$ Topic $\rightarrow$ Author $\rightarrow$ Conference

9.3.2 Ranking for Author Entities

We ranked various entities (conferences, topics, authors) with respect to a given conference in previous experiments. However, the given entity to start the random walk is not limited to one particular set, it can be any entity across the k-partite network model. For example, we can also rank related topics, conferences and authors for a given author or topic. Among these possibilities, ranking author for author could provide promising potential collaboration recommendations for researchers, i.e., high relevance score between authors implies that their research topics and fields, which are represented by conferences, are very similar.

We applied our random walk algorithm on the conference-author-topic network model. We removed all the researchers that have already co-authored a paper with the given author from the ranked list. Therefore, there are no direct collaboration connections between the given author and the listed authors. In Table 9.5, we list the top 5 authors recommended as possible future collaborators for Philip S. Yu. To validate our results, we show the paths between them, i.e. degree of separation ($A \rightarrow B$ means A and B are co-authors) and their most frequent topic and conference in common. Obviously, all recommended collaborators are strongly connected, only one or two steps away via co-authorship, to Philip S. Yu, and the listed topics and conferences are apparently all relevant to his research interests.

Path: Philip S. Yu → ...	Future Collaborator	*Top Topic*	*Top Conf.*
→Jiawei Han→Martin Ester→	Hans-Peter Kriegel	Similarity Search	ICDE
→Jun Yang→	Hector Garcia-Molina	Database System	ICDE
→Jiawei Han→Beng Chin Ooi→	Elisa Bertino	Data Mining	ICDE
→Yun-Wu Huang→	Elke A. Rundensteiner	Data Stream	CIKM
→Balakrishna R. Iyer→	Divyakant Agrawal	Data Stream	VLDB

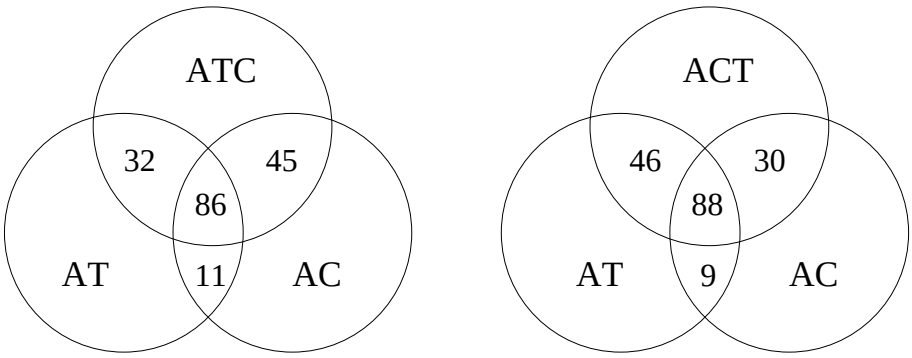
Table 9.5: Top 5 Related Author for Philip S. Yu with most recommended Topic and Conference to collaborate ($A \rightarrow B$ means A and B are co-authors)

9.3.3 Random Walk on Tripartite Graph

As mentioned in Section 9.1.2, there are two possible directions of random walks in a tripartite graph, (and C_k^2 options for a k-partite graph), which could lead to different ranking results. To choose the most appropriate direction for a given network model is an interesting and unsolved problem. We experimented on the conference-author-topic database from DBLP and present our result in the following.

Assume we start from an author node to rank related authors for that author, the two possible random walk directions of the conference-author-topic database are: *author* → *conference* → *topic* → *author* and *author* → *topic* → *conference* → *author* (Figure 9.1 (B)). Each entity set affects the final ranking result based on its cross relations, e.g., relation set conference↔author affects the relevance score result since authors that published in the same conference as the given author would have high ranking. Usually those relation sets are not equal in significance, for example, one may think relations between topics and authors are more important and should be emphasized more in the random walk than other relations. We need to investigate influence changes of different directions to find the appropriate sequence of entity sets that the random walker should follow, based on the importance of these entity sets.

Given author "Jiawei Han", we run the random walk algorithm on the bipartite network model between authors and conferences (AC), on the bipartite network model between authors and topics (AT), on the tripartite graph following the direction *author* → *conference* → *topic* → *author* (ACT), and on the tripartite graph following the direction *author* → *topic* → *conference* → *author* (ATC). We selected authors with top k (k=200) relevance score as a test list for the given author.



Total Recommendation: AT=AC=ATC=ACT=200

Figure 9.3: Random Walks on Tripartite Model

We show the result in Figure 9.3, where numbers in the overlapping area of the circles represent the number of authors that these ranking lists share. For both directions, most of the authors in the list (81.5% of ATC, 82% of ACT) can be found in either AT or AC, which confirms the initial observation that each entity set has influence on the result of the multiple cross relation model. However, their influences change for different directions. AT contributes 32 authors to ATC and that number increase to 46 for ACT. On the other hand, AC's contribution drops from 45 for ATC to 30 for ACT. The experiment shows that, for author "Jiawei Han", the influence of entity sets on the ranking is increasing following the sequence of the random walk, i.e., topics are more emphasized in ACT and conferences are more important in ATC, as the last partition of the random walk. In order to investigate whether this phenomenon is general, we run the same experiment on more random authors, whose results all show similar characteristics. Therefore, we believe that the random walk direction in a tripartite graph for the multiple cross relation model can affect the relevance ranking result. The later the entity set is visited by the random walk, the more influence it has on result scores and the more important it should be. Similar behaviour is expected for k-partite graphs.

9.3.4 Discussion

Ranking entities in a social network, where objects are cross-linked with each other via multi-type links, is important in discovering the rich semantic information these

links may contain and in identifying the paramount relationships among objects. Due to limitations of the DBLP database, it is hard to extract accurate topics for conference papers since the only available content-related information are the title of the paper and incomplete abstracts from Citeseer. In experiments where we used only single keywords as topics, we observed that the majority of the entities of ACT or ATC (see Section 9.3.3) can be found in AT, and the change of direction did not affect much AT's contribution. In other words, inaccurate topic-author/topic-conference relations have an overwhelming influence on the random walk in the tripartite graph model and the result ranking score. Using bi-grams greatly improves the quality of relations and ranking results. Better performance can be expected if the topics are extracted or provided more accurately. However, classifying topics is itself a huge research issue and is out of the scope of this dissertation.

9.4 DBConnect

In the above, we use DBLP data to generate bipartite (author-conference) and tripartite (author-conference-topics) graph models, where topics are frequent n-grams extracted from paper titles and abstracts. Moreover, we present an iterative random walk algorithm on these models to compute the relevance score between entities to understand the relations between communities. We now present DBconnect [226], which is an interactive interface for navigating the DBLP community structure, as well as recommendations and explanations for these recommendations.

After the author-conference-topic data extraction from the DBLP database, we generate lists of people with high relevance scores with respect to different given researchers. *DBconnect*, which is a navigational system to investigate the community connections and relationships, is built to explore the result lists of our random walk approach on the academic social network. Figure 9.4 shows a screenshot of the author interface of our *DBconnect* system. There are eight lists displayed for a given author in the current version. Clicking on any of the hyper-linked names will generate a page with respect to that selected entity. We explain details of each of the lists below.

Figure 9.4: DBconnect Interface Screenshot for an author

- *Academic Information*

  Academic statistics for the given author are shown in this list, which contain three components: conference contribution, earliest publication year and average publication per year are extracted from DBLP; the H-index [214] is calculated based on information retrieved from Google Scholar[2]; approximate citation numbers are retrieved from Citeseer[3]. The query terms for Google Scholar and Citeseer are automatically generated based on the author names. Users can submit an alternative query which gives a more accurate result from the search engines. We also provide a visualization of the H-index.

[2]http://scholar.google.com/
[3]http://citeseer.ist.psu.edu/

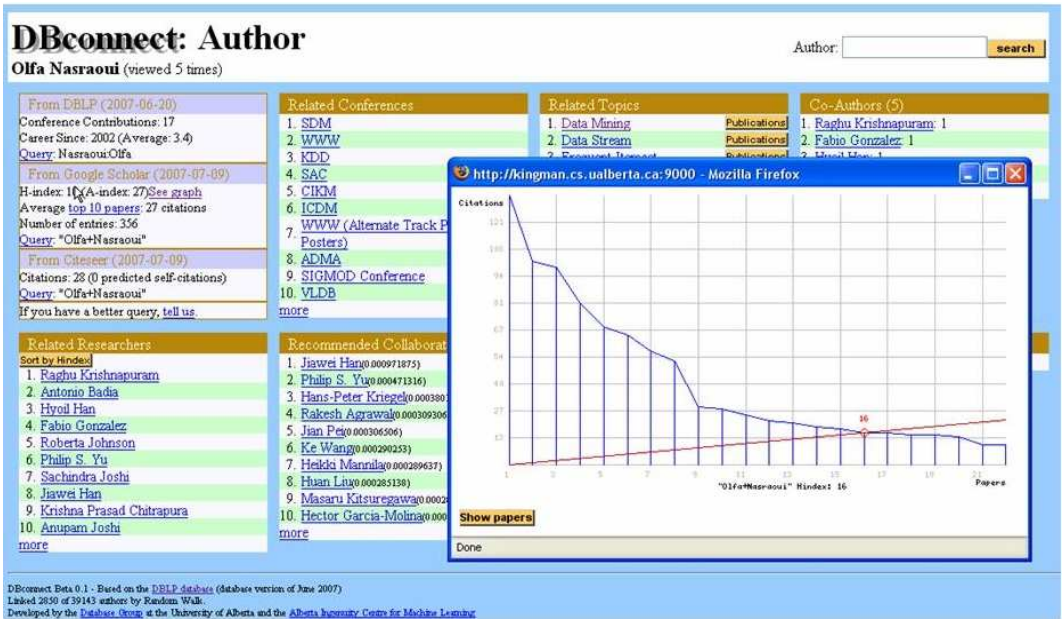


Figure 9.5: DBconnect Interface Screenshot for H-Index Visualization

One can click the "See graph" link beside the H-index numbers. Figure 9.5 shows an example of H-index visualization.

- *Related Conferences*

  This list is generated by the random walk, which starts from the given author, on an author-conference-topic model and is ordered by their relevance score, in descending order. These are not necessarily the conferences where the given researcher published but the conferences related to the topics and authors that are also related to the reference researcher. Clicking on the conference name leads to a new page with topics and authors related to the chosen conference.

- *Related Topics*

  This list is ordered by the relevance scores from a random walk on the tripartite model. Clicking on the button "Publications" after each topic provides the papers that the given author has published on that topic, i.e. the papers of the given author that contains the N-gram keywords in their titles or abstracts. Similarly, these are not necessarily the topics that the given author has worked on, but the topics most related to his topics, attended conferences

Figure 9.6: DBconnect Interface Screenshot for conference ICDM

and colleagues.

- *Co-authors*

  The co-author list reports the number of publications that different researchers co-authored with the given person.

- *Related Researchers*

  This list is based on the bipartite graph model with only conference and author entities. The result implies that the given author is related to the same conferences and via the same co-authors as these listed researchers. In most cases, most related researchers to the given author are co-authors and co-authors of co-authors.

- *Recommended Collaborators*

  This list is based on the tripartite graph author-conference-topic. Since co-

Figure 9.7: DBconnect Interface Screenshot for topic Data Mining

authors are treated as "observed collaborators", their names are not shown here. The result implies that the given author shares similar topics and conference experiences with these listed researchers, hence the recommendation. The relevance score calculated by our random walk is displayed following the names. Clicking on the "why" button brings the detailed information of the relationship between the two authors. For example, in Figure 9.4, relations between Philip Yu and Osmar Zaïane are described by the topics and conferences they share, and the degree of separation in the co-authorship chain ($A \rightarrow B$ means A and B are co-authors). Here, the "Share Topics" table lists the topics that these two authors both have publications on and the "Related Topics" table shows the topics that appear in the Related Topics lists of both authors. Similarly, the "Shared Conferences" table displays the conferences that the two authors have attended and the "Related Conferences" table

shows the conferences that can be found in the Related Conferences lists of both authors.

- *Recommended To*

  The recommendation is not symmetric, i.e., author A may be recommended as a possible future collaborator to author B but not vice versa. This phenomenon is due to the unbalanced influence of different authors in the social network. For example, Jiawei Han has a significant influence with his 196 conference publications, 84 co-authors and H-index 63. He has been recommended as collaborator for 6201 authors, but apparently only a few of them is recommended as collaborators to him. The Recommended To list shows the authors that have the given author in their recommendation list, ordered by the relevance score.

- *Symmetric Recommendations*

  This list shows the authors that have been recommended to the given author and have the given author on their recommendation list.

Note that while there is some overlap between the list of related researchers and the list of recommended collaborators, there is a fundamental difference and the difference by no means implies that collaboration with the missing related researchers is discouraged. They are simply two different communities in the network even though they overlap. The list of related researchers is obtained based on a conference-author bipartite graph. The list of recommended collaborators could be perceived as a more distant community and thus as an interesting discovery since it is obtained with relations from topics and we use a RWR on a tripartite graph authors/conferences/topics. The explanation on the why collaborators are recommended (i.e. common conferences and topics, and degree of separation) establishes more trust in the recommendation. A systematic validation of these lists is difficult but the cases we manually substantiated were satisfactory and convincing.

Clicking on any conference name shows a conference page. Figure 9.6 illustrates an example when the entity "ICDM" is selected. Conferences have their own related

conferences, authors and topics. Note that the topics here mean the most frequent topics used within titles and abstracts of papers published in the given conference.

Clicking on the topics leads to a new page with conferences, authors and topics related to the chosen topic. Note again that this relationship to topics comes from paper titles and abstracts. Figure 9.7 shows an example when the topic "Data Mining" is selected.

9.5 Conclusions

A wide range of social networks can be described as related entities sets and can be modeled as k-partite graphs, such as P2P networks, author-conference relationships, customer-movie rental records, etc. In this chapter, a model and algorithm for multiple cross relations are presented and the consequences of different random walk directions are investigated. We validate results of the algorithms on existing publication databases. Our experimental results confirm the accuracy and effectiveness of the proposed methods. We also present DBconnect, which can help explore the relational structure and discover implicit knowledge within the DBLP data collection.

Chapter 10

Conclusion

10.1 Conclusions

In this Ph.D study, we have tackled several social network community mining problems from different aspects. We summarize our solutions as follows.

1. A solution to the problem stated in Chapter 4.1 has been proposed: as shown in Chapter 5, we propose the Max-Min Modularity metric to include domain knowledge, which is provided by domain experts, as a new guiding criteria in the community detection process without increasing algorithm complexity. Therefore, distinct properties of different networks could be used to help community mining.

2. We propose a new metric in Chapter 6 for the problem stated in Chapter 4.2. We analyze the drawbacks of previous approaches and correct them by using the connection density to evaluate the local community structure so that outliers can be detected. We propose a greedy algorithm to discover local communities based on our metric.

3. For the problem discussed in Chapter 4.3, we propose a visual data mining approach (ONDOCS) in Chapter 7 to help finding overlapping communities. A new metric is defined based on proposed metric requirements. Network visualizations are then generated by ordering nodes based on their relation scores. Thus appropriate parameters for overlapping community identifica-

tion are easy for the user to decide after observing generated network visualizations.

4. In Chapter 8, we solve a real problem for web search engines, which is defined in Chapter 4.4, by community mining approaches. We propose an algorithm to identify query sense communities on a network of keywords that are extracted from search result pages. These pages are then classified into different sense communities based on the keywords they contain. We also propose to use the modularity metric to assess whether page clustering is necessary for a particular query.

5. Finally, we propose a solution in Chapter 9 for the problem of ranking entities in heterogeneous social networks, which is defined in Chapter 4.5. For social networks with multiple relations and entity types, we first transform the network into a k-partite graph and then apply a random walk algorithm on the network model to generate rankings for a given starting node.

10.2 Summary of Contributions

This Ph.D dissertation makes the following contributions:

1. We propose a method to include domain knowledge as guiding criteria in the community detection process by either rewarding or penalizing the metric that evaluates the discovered structure, without increasing algorithm complexity. Our new measure and algorithm improve the accuracy for community detection over previous algorithms when applied to real world networks for which the community structures are already known, and also give promising results when applied to randomly generated networks for which we only have approximate information about communities. This shows the robustness of the algorithm against noise.

2. In contrast to existing local community mining approaches, our method proposed in Chapter 6 requires no parameters and our metric L is robust against

outliers. The proposed algorithm not only discovers local communities without an arbitrary threshold, but also determines whether a local community exists or not for certain nodes.

3. For overlapping community identification, our visual data mining approach could assist the user in finding appropriate parameters to describe the communities they are looking for. The new metric R can effectively quantify the relations between entities. The method is scalable and is able to discover communities, hubs, and outliers in social networks.

4. Our method to identify query sense communities on a network of extracted keywords gives good results. The unsupervised approach, which categorizes web pages into meaningful categories with information of the query and the result page content only, does not require additional information about the query in question and is feasible for real time page clustering for search engine results. The use of a community mining metric of the discovered query sense community structure to assess whether page clustering is required for search results is new. While previous methods overlook this issue, experimental results show that our method is useful.

5. While previous ranking methods focus on homogeneous networks, our entity ranking approach is useful for heterogeneous networks where there exist multiple types of entities and relations. We also provide a convenient interface for users to navigate the ranking and relations.

10.3 Future Research

Community mining is increasingly attracting people's attention as an area of study. Nevertheless it faces many challenges, including accurate data extraction, entity ranking, relation selection, community evaluation and discovery. Each has received significant attention in the literature, and is known to be rather challenging itself. In this dissertation, we have proposed solutions in many aspects of community mining problems, however, there is still much left to do. Possible future work can

be summarized as follows:

- Domain knowledge is represented as node pairs in our approach, i.e, our approach takes a pair of nodes as "related nodes" or "unrelated nodes" to help the mining process. However, knowledge in some networks might not be represented in this form. For example, a group of genes can belong to the same category in a biological network. It would be redundant to transform this type of knowledge into a set of node pairs. A better way of knowledge representation may be necessary.

- Web pages need to be classified into different sense communities once these communities are discovered. In Chapter 8 we use a straightforward method, which classifies a page to the sense community that has the largest sum of frequencies of keywords that belong to both the page and the community. To achieve a more accurate result, other factors, such as the topic of the page or the importance of the keyword, may be necessary to take into consideration.

- Again in Chapter 8, our sense communities are labeled using NLP rankings based on a different database. These generated labels show the content of the communities, to some extent, but a better cluster labeling method is desired to practically improve the usability of our approach.

- In this dissertation, relations in a multiple-relation social network are considered to be equal. For example, when we transform the network into a bipartite graph in Chapter 4, the weight of every connection is 1 regardless of which relation this connection is. However, in the real world, different relations in the same social network have different significance levels and thus should be ranked. How to improve the performance of community mining approaches by taking advantage of relation ranking is an interesting topic.

- In Chapter 9, we justify our ranking results in a bibliographical database by common sense. However, we do not evaluate the results systematically. Such metric for ranking evaluation is very hard to define, if not impossible. We

are not aware of any feasible approaches to evaluate our rankings. Further investigation is desired.

- We focus on social relations in this dissertation. However, attributes can also be important for community mining. How to extend proposed methods to incorporate attribute analysis would be a very interesting topic.

- We assume the social network to be "static". It would be interesting to investigate the possibility of extending the metrics and algorithms proposed in this dissertation to discover and rank communities in a dynamic social network.

Bibliography

[1] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek. Robust, complete, and efficient correlation clustering. In *Proc. 7th SIAM International Conference on Data Mining (SDM'07)*, 2007.

[2] L. A. Adamic and N. Glance. The political blogosphere and the 2004 u.s. election: divided they blog. In *LinkKDD '05: Proceedings of the 3rd international workshop on Link discovery*, pages 36–43, 2005.

[3] B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, and S. S. Parag. Banks: Browsing and keyword searching in relational databases. In *VLDB*, pages 1083–1086, 2002.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, pages 487–499, 1994.

[5] AICML. http://www.machinelearningcentre.ca/.

[6] Reka Albert, Hawoong Jeong, and Albert-Laszlo Barabasi. Diameter of the world-wide web. *Nature*, 401:130–131, 1999.

[7] L. A. N. Amaral, A. Scala, M. Barthelemy, and H. E. Stanley. Classes of small-world networks. *Proc. Natl. Acad. Sci.*, USA97:11149–11152, 2000.

[8] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *SIGMOD*, pages 49–60, 1999.

[9] M. Ankerst, C. Elsen, M. Ester, and H.-P. Kriegel. Visual classification: an interactive approach to decision tree construction. In *KDD*, pages 392–396, 1999.

[10] M. Ankerst, M. Ester, and H.-P. Kriegel. Towards an effective cooperation of the user and the computer for classification. In *KDD*, pages 179–188, 2000.

[11] M. Ankerst and D. A. Keim. Visual data mining, Tutorial at SIAM Int. Conf on Data Mining 2003.

[12] S. O. Aral, J. P. Hughes, B. Stoner, W. Whittington, H. H. Handsfield, R. M. Anderson, and K. K. Holmes. Sexual mixing patterns in the spread of gonococcal and chlamydial infections. *American Journal of Public Health*, 89:825–833, 1999.

[13] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *KDD*, pages 44–54, 2006.

[14] J. P. Bagrow. Evaluating local community methods in networks. *J.STAT.MECH.*, page P05001, 2008.

[15] J. P. Bagrow and E. M. Bollt. Local method for detecting communities. *Physical Review E*, 72(4), 2005.

[16] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *Machine Learning*, pages 238–247, 2002.

[17] J. A. Barnes. Class and committee in a norwegian island parish. *Human Relations*, 7:39–58, 1954.

[18] S. Basu, A. Banerjee, and R. J. Mooney. Semi-supervised clustering by seeding. In *ICML*, pages 27–34, 2002.

[19] J. Baumes, M. K. Goldberg, and M. Magdon-Ismail. Efficient identification of overlapping communities. In *ISI*, pages 27–36, 2005.

[20] D. Bean and E. Riloff. Unsupervised learning of contextual role knowledge for coreference resolution. In *Proc. of HLT/NAACL*, pages 297–304, 2004.

[21] F. Beil, M. Ester, and X. Xu. Frequent term-based text clustering. In *KDD*, pages 436–442, 2002.

[22] T. Y. Berger-Wolf and J. Saia. A framework for analysis of dynamic social networks. In *KDD*, pages 523–528, 2006.

[23] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of SIGIR-98*, pages 104–111, Melbourne, AU, 1998.

[24] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *DMKD '04: Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 11–18, 2004.

[25] DBLP Bibliography. http://www.informatik.uni-trier.de/ ley/db/.

[26] A. Bjorck. Numerical methods for least squares problems, SIAM 1996.

[27] M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas. Models of social networks based on social distance attachment. *Phys. Rev. E*, 70(5):056122, 2004.

[28] P. Bonacich. Power and centrality: A family of measures. *Am. J. Social*, 92(5):1170–1182, 1987.

[29] P. Bonacich and P. Lloyd. Eigenvector-like measures of centrality for asymmetric relations. *Social Networks*, 23(3):191–201, July 2001.

[30] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.

[31] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW*, pages 107–117, Brisbane, Australia, 1998.

[32] R. S. Burt. Positions in networks. *Social Forces*, 55:93–122, 1976.

[33] D. Cai, Z. Shao, X. He, X. Yan, and J. Han. Community mining from multi-relational networks. In *PKDD*, pages 445–452, 2005.

[34] D. Cai, Z. Shao, X. He, X. Yan, and J. Han. Mining hidden community in heterogeneous social networks. In *LinkKDD '05: Proceedings of the 3rd international workshop on Link discovery*, pages 58–65, 2005.

[35] D. Cartwright and F. Harary. Structural balance: a generalisation of heider's theory. *Psychological Review*, 63:277–293, 1956.

[36] S. C. Cazella and L. O. C. Alvares. An architecture based on multi-agent system and data mining for recommending research papers and researchers. In *Proc. of the 18th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pages 67–72, 2006.

[37] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *WWW*, 1998.

[38] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD*, pages 307–318, 1998.

[39] P.K. Chan, M. D. F. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. In *Proceedings of the 30th International Conference on Design Automation*, pages 749–754, 1993.

[40] M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. In *FOCS*, page 524, 2003.

[41] H. Chen and S. Dumais. Bringing order to the web: automatically categorizing search results. In *CHI '00*, pages 145–152, 2000.

[42] J. Chen, O. R. Zaïane, and R. Goebel. An unsupervised approach to cluster web search results based on word sense communities. In *Proceedings of IEEE/WIC/ACM International Conference*, pages 725–729, 2008.

[43] J. Chen, O. R. Zaïane, and R. Goebel. Detecting communities in social networks using max-min modularity. In *Proceedings of the SIAM Data Mining Conference*, pages 105–112, 2009.

[44] J. Chen, O. R. Zaïane, and R. Goebel. Local community identification in social networks. In *Proceedings of the International Conference on Advances in Social Network Analysis and Mining(ASONAM)*, pages 237–242, 2009.

[45] J. Chen, O. R. Zaïane, and R. Goebel. A visual data mining approach to find overlapping communities in networks. In *Proceedings of the International Conference on Advances in Social Network Analysis and Mining(ASONAM)*, pages 338–343, 2009.

[46] J. Chen, O. R. Zaïane, and R. Goebel. Detecting communities in large networks by iterative local expansion. In *CASoN*, pages 105–112, 2009.

[47] C. K. Cheng and Y.-C. Wei. An improved two-way partitioning algorithm with stable performance. *IEEE. Trans. on Computed Aided Design*, 10:1502–1511, 1991.

[48] H. Chim and X. Deng. A new suffix tree similarity measure for document clustering. In *WWW*, 2007.

[49] J. Cho, H. Garcia-Molina, and L.Page. Efficient crawling through url ordering. In *WWW*, 1998.

[50] M. Chodorow, C. Leacock, and G. A. Miller. A topical/local classifier for word sense identification. *Computers and the Humanities*, 34(1-2):115–120, 2000.

[51] A. Clauset. Finding local community structure in networks. *Physical Review E*, 72:026132, 2005.

[52] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, 2004.

[53] D. Cohn and H. Chang. Learning to probabilistically identify authoritative documents. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 167–174, 2000.

[54] D. Cohn and T. Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In *NIPS*, 2001.

[55] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*, 118(1-2):69–113, 2000.

[56] H. Cui and O. R. Zaiane. Hierarchical structural approach to improving the browsability of web search engine results. In *DEXA Workshop on Digital Libraries*, pages 956–960, 2001.

[57] L. Danon, J. Duch, A. Diaz-Guilera, and A. Arenas. Comparing community structure identification. *J. Stat. Mech*, page P09008, 2005.

[58] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic coclustering. In *KDD*, pages 89–98, 2003.

[59] C. Ding, X. He, P. Husbands, H. Zha, and H. D. Simon. Pagerank, hits and a unified framework for link analysis. In *SIGIR '02*, pages 353–354, 2002.

[60] C. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 107–114, 2001.

[61] A. Doan, R. Ramakrishnan, F. Chen, P. DeRose, Y. Lee, R. McCann, M. Sayyadian, and W. Shen. Community information management. *IEEE Data Engineering Bulletin, Special Issue on Probabilistic Databases*, 29(1), 2006.

[62] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, pages 85–96, 2005.

[63] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Phys. Rev. E*, 72:027104, 2005.

[64] R. O. Duda, P. E. Hart, and D. G. Stork. Pattern classification, Wiley-Interscience, 2nd edition, 2000.

[65] U. Elsner. Graph partitioning - a survey. In *Technical report 97-27, Technische University Chemnitz*, 1997.

[66] A. Esuli and F. Sebastiani. Pageranking wordnet synsets: An application to opinion-related properties. In *Proceedings of the 35th Meeting of the Association for Computational Linguistics*, pages 424–431, 2007.

[67] T. Falkowski, J. Bartelheimer, and M. Spiliopoulou. Mining and visualizing the evolution of subgroups in social networks. In *Web Intelligence*, pages 52–58, 2006.

[68] A. M. Fard and M. Ester. Collaborative mining in multiple social networks data for criminal group discovery. In *CSE (4)*, pages 582–587, 2009.

[69] C. Fellbaum. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, 1998.

[70] M. Fiedler. Algebraic connectivity of graphs. *Czech. Math. J.*, 23:298–305, 1973.

[71] P. O. Fjallstrom. Algorithms for graph partitioning: A survey. *Linkoping Electronic Articles in Computer and Information Science*, 3(10), 1998.

[72] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *KDD*, pages 150–160, 2000.

[73] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *PROC.NATL.ACAD.SCI.USA*, 104:36, 2007.

[74] L. Freeman. Visualizing social networks. *Journal of Social Structure*, 1(1), 2000.

[75] L. C. Freeman. A set of measures of centrality based upon betweenness. *Sociometry*, 40:35–41, 1977.

[76] L. C. Freeman. Centrality in social networks - conceptual clarification. *Social Networks*, 1:215–239, 1978.

[77] J. Freyne and B. Smyth. Cooperating search communities. In *Proc. of 4th International Conference on Adaptive Hypermedia and Adaptive WebBased Systems (AH'2006). Lecture Notes in Computer Science*, pages 101–111. Springer Verlag, 2006.

[78] N. E. Friedkin. Theoretical foundations for centrality measures. *Am. J. Social*, 96:1478–1504, 1991.

[79] G. P. Garnett, J. P. Hughes, R. M. Anderson, B. P. Stoner, S. O. Aral, W. L. Whittington, H. H. Handsfield, and K. K. Holmes. Sexual mixing patterns of patients attending sexually transmitted diseases clinics. *Sexually Transmitted Diseases*, 23:248–257, 1996.

[80] F. Geerts, H. Mannila, and E. Terzi. Relational link-based ranking. In *VLDB*, pages 552–563, 2004.

[81] L. Getoor and C. P. Diehl. Link mining: a survey. *SIGKDD Exploration Newsletter*, 7(2):3–12, 2005.

[82] David Gibson, Jon M. Kleinberg, and Prabhakar Raghavan. Inferring web communities from link topology. In *Hypertext*, pages 225–234, 1998.

[83] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. In *ICDE*, pages 341–352, 2005.

[84] I. Giotis and V. Guruswami. Correlation clustering with a fixed number of clusters. In *SODA*, pages 1167–1176, 2006.

[85] Michelle Girvan and M. E. J. Newman. Community structure in social and biological networks. In *Proceedings of the National Academy of Science USA, 99:8271-8276*, 2002.

[86] Gnuplot. http://www.gnuplot.info/.

[87] M. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78:1360–1380, 1973.

[88] S. Gregory. An algorithm to find overlapping community structure in networks. In *PKDD*, pages 91–102, 2007.

[89] S. Gregory. A fast algorithm to find overlapping communities in networks. In *PKDD*, pages 408–423, 2008.

[90] R. Guimera and L. A. N. Amaral. Functional cartography of complex metabolic networks. *Nature*, 433:895–900, 2005.

[91] R. Guimera, M. Sales-pardo, and L. A. N. Amaral. Modularity from fluctuations in random graphs and complex networks. *Phys. Rev. E*, 70:025101, 2004.

[92] S. Gupta, R. M. Anderson, and R. M. May. Networks of sexual contacts: Implications for the pattern of spread of hiv. *AIDS*, 3:807–817, 1989.

[93] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE. Trans. on Computed Aided Design*, 11:1074–1085, 1992.

[94] J. Han and N. Cercone. Ruleviz: a model for visualizing knowledge discovery process. In *KDD*, pages 244–253, 2000.

[95] F. Harary. Graph theory, MA: Addison–Wesley, 1969.

[96] F. Harary, R. Z. Norman, and D. Cartwright. Structural models, New York: Wiley, 1965.

[97] T. Hastie, R. Tibshirani, and J. H. Friedman. The elements of statistical learning, Springer-Verlag, 2001.

[98] T. H. Haveliwala. Topic-sensitive pagerank. In *WWW*, pages 517–526, 2002.

[99] J. He, M. Li, H. Zhang, H. Tong, and C. Zhang. Manifold-ranking based image retrieval. In *MULTIMEDIA: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 9–16, 2004.

[100] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *SIGIR'96*, pages 76–84, 1996.

[101] F. Heider. Attitudes and cognitive orientation. *Journal of Psychology*, 21:107–112, 1946.

[102] P. Holme, M. Huss, and H. Jeong. Subnetwork hierarchies of biochemical pathways. *Bioinformatics*, 19:532–538, 2003.

[103] H. Hwang, V. Hristidis, and Y. Papakonstantinou. Objectrank: Authority-based keyword search in databases. In *VLDB*, pages 564–575, 2004.

[104] R. Ichise, H. Takeda, and T. Muraki. Research community mining with topic identification. In *IV '06: Proceedings of the conference on Information Visualization*, pages 276–281, 2006.

[105] R. Ichise, H. Takeda, and K. Ueyama. Community mining tool using bibliography data. In *IV '05: Proceedings of the Ninth International Conference on Information Visualisation (IV'05)*, pages 953–958, 2005.

[106] IMDB. http://www.imdb.com/.

[107] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *PKDD*, pages 13–23, 2000.

[108] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *KDD*, 2002.

[109] G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, pages 2271–279, 2003.

[110] D. Jensen. Statistical challenges to inductive inference in linked data, 1999.

[111] X. Ji and W. Xu. Document clustering with prior knowledge. In *SIGIR*, pages 405–412, 2006.

[112] I.T. Jolliffe. Principal component analysis, Springer Series in Statistics, 2nd edition(2002).

[113] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):881–892, 2002.

[114] Kartoo. http://www.kartoo.com/.

[115] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distriuted Computing*, 48(1):96–129, 1998.

[116] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49:291–307, 1970.

[117] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[118] D. E. Knuth. The stanford graphbase: A platform for combinatorial computing, Addison-Wesley, Reading, MA (1993).

[119] Valdis Krebs. http://www.orgnet.com/.

[120] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the web for emerging cyber-communities. *Computer Networks*, 31(11-16):1481–1493, 1999.

[121] K. Kummamuru, R. Lotlikar, S. Roy, K. Singal, and R. Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *WWW*, pages 658–665, 2004.

[122] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *ICDM*, pages 313–320, 2001.

[123] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.

[124] K. Lewin. Principles of topological psychology, McGraw-Hill, 1936.

[125] X. Li, B. Liu, and P. S. Yu. Discovering overlapping communities of named entities. In *PKDD*, 2006.

[126] X. Li, B. Liu, and P. S. Yu. Mining community structure of named entities from web pages and blogs. In *AAAI-CAAW*, 2006.

[127] X. Li, S. Szpakowicz, and S. Matwin. A wordnet-based algorithm for word sense disambiguation. In *In Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1368–1374, 1995.

[128] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM*, pages 556–559, 2003.

[129] D. Lin. Principar: an efficient, broad-coverage, principle-based parser. In *Proceedings of the 15th conference on Computational linguistics*, pages 482–488, 1994.

[130] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, pages 768–774, 1998.

[131] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502, 2005.

[132] F. Lorrian and H. C. White. Structural equivalence of individuals in social networks. *Journal of Mathematical Sociology*, 1:49–80, 1971.

[133] F. Luo, J. Z. Wang, and E. Promislow. Exploring local community structures in large networks. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 233–239, 2006.

[134] M. Marchiori and V. Latora. Harmony in the small-world. *Physica A*, 285:539–546, 2000.

[135] F. Martino and A. Spoto. Social network analysis: A brief theoretical review and further perspectives in the study of information technology. *PsychNology Journal*, 4(1):53–86, 2006.

[136] E. Mayo. The social problems of an industrial civilization, London; routledge and Kegan Paul, 1945.

[137] R. Mihalcea and D. I. Moldovan. A method for word sense disambiguation of unrestricted text. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 152–158, 1999.

[138] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. R. Mullers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 41–48, 1999.

[139] S. Milgram. The small world problem. *Psychology Today*, 1(1):60–67, 1967.

[140] D. N. Milne, I. H. Witten, and D. M. Nichols. A knowledge-based search engine powered by wikipedia. In *Conference on information and knowledge management (CIKM)*, pages 445–454, 2007.

[141] J. Moreno. Who shall survive, New York: Beacon Press 1934.

[142] F. Moser, R. Colak, A. Rafiey, and M. Ester. Mining cohesive patterns from graphs with feature vectors. In *SDM*, pages 593–604, 2009.

[143] Mario A. Nascimento, Jörg Sander, and Jeffrey Pound. Analysis of sigmod's co-authorship graph. *SIGMOD Record*, 32(2):57–58, 2003.

[144] T. Nepusz, A. Petróczi, L. Negyessy, and F. Bazso. Fuzzy communities and the concept of bridgeness in complex networks. *Physical Review E*, 77, 016107, 2008.

[145] M. Newman. http://www-personal.umich.edu/∼mejn/netdata/.

[146] M. E. J. Newman. The structure of scientific collaboration networks. In *PNAS USA, 98:404-409*, 2001.

[147] M. E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67, 2003.

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

[149] M. E. J. Newman. Coauthorship networks and patterns of scientific collaboration. *PROC.NATL.ACAD.SCI.USA*, 101, 2004.

[150] M. E. J. Newman. Detecting community structure in networks. *Eur. Phys. J.B*, 38:321–330, 2004.

[151] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69, 2004.

[152] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74, 2006.

[153] M. E. J. Newman. Modularity and community structure in networks. *PROC.NATL.ACAD.SCI.USA*, 103, 2006.

[154] M. E. J. Newman. Mathematics of networks, in the new palgrave encyclopedia of economics, 2nd edition, Palgrave Macmillan, Basingstoke, in Press.

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

[156] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[157] A. Y. Ng, A. X. Zheng, and M. I. Jordan. Link analysis, eigenvectors and stability. In *IJCAI*, pages 903–910, 2001.

[158] A. Y. Ng, A. X. Zheng, and M. I. Jordan. Stable algorithms for link analysis. In *Proc. 24th Annual Intl. ACM SIGIR Conference*, 2001.

[159] H. T. Ng. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 40–47, 1996.

[160] V. Ng. Semantic class induction and coreference resolution. In *ACL*, 2007.

[161] I. Nica, A. Montoyo, S. Vázquez, and M. Antònia Martí. An unsupervised WSD algorithm for a NLP system. In *NLDB*, pages 288–298, 2004.

[162] V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri. Extending modularity definition for directed graphs with overlapping communities, Eprint arXiv:0801.1647 at arxiv.org. 2008.

[163] K. Nowicki and T. A. B. Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1087, 2001.

[164] H.-J. Oh, S. H. Myaeng, and M.-H. Lee. A practical hypertext catergorization method using links and incrementally available class information. In *SIGIR*, pages 264–271, 2000.

[165] J. O'Madadhain, J. Hutchins, and P. Smyth. Prediction and ranking algorithms for event-based network data. *SIGKDD Explor. Newsl.*, 7(2):23–30, 2005.

[166] J. O'Madadhain and P. Smyth. Eventrank: a framework for ranking time-varying networks. In *LinkKDD '05: Proceedings of the 3rd international workshop on Link discovery*, pages 9–16, 2005.

[167] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In *Technical report, Stanford University Database Group*, 1998.

[168] Pajek. http://vlado.fmf.uni-lj.si/pub/networks/pajek/.

[169] G. Palla, A.-L. Barabasi, and T. Vicsek. Quantifying social group evolution. *Nature*, 446:664–667, 2005.

[170] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, 2005.

[171] J. Pan, H. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. In *KDD*, pages 653–658, 2004.

[172] P. Pantel and D. Lin. Discovering word senses from text. In *KDD*, pages 613–619, 2002.

[173] P. Pattison. Algebraic model for social networks, Cambridge University Press, 1993.

[174] S. Patwardhan, S. Banerjee, and T. Pedersen. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 241–257, 2003.

[175] P. Pirolli, P. Schank, M. Hearst, and C. Diehl. Scatter/gather browsing communicates the topic structure of a very large text collection. In *CHI*, pages 213–220, 1996.

[176] S. P. Ponzetto and M. Strube. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proc. of HLT/NAACL*, pages 192–199, 2006.

[177] A. Pothen, H. Simon, and K. P. Liou. Partitioning sparse matrices with eigenvectorsof graphs. *SIAM J. Matrix Anal. Appl.*, 11:430–452, 1990.

[178] D. Rafiei and A. O. Mendelzon. What is this page known for? computing web page reputations. In *WWW*, pages 823–835, 2000.

[179] M. Richardson and P. Domingos. The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank. In *NIPS*, 2002.

[180] J. Ruan and W. Zhang. Identifying network communities with a high resolution. *Physical Review E*, 77:016104, 2008.

[181] M. Sanderson. Word sense disambiguation and information retrieval. In *SIGIR '94*, pages 49–57, 1994.

[182] P. Sarkar and A. W. Moore. Dynamic social network analysis using latent space models. *SIGKDD Explor. Newsl.*, 7(2):31–40, 2005.

[183] V. Satuluri and S. Parthasarathy. Scalable graph clustering using stochastic flows: applications to community discovery. In *KDD*, pages 737–746, 2009.

[184] H. Schütze. Automatic word sense discrimination. *Comput. Linguist.*, 24(1):97–123, 1998.

[185] J. Scott. Social network analysis: A handbook, Sage, London 2nd edition(2000).

[186] Jerry Scripps, Pang-Ning Tan, and Abdol-Hossein Esfahanian. Exploration of link structure and community-based node roles in network. In *ICDM*, 2007.

[187] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE. Trans. on Pattern Analysis and Machine Intelligence*, 2000.

[188] A. F. Smeaton, G. Keogh, C. Gurrin, K. McDonald, and T. Sodring. Analysis of papers from twenty-five years of sigir conferences: What have we been doing for the last quarter of a century. *SIGIR Forum*, 36(2):39–43, 2002.

[189] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *Proceedings of Workshop on Text Mining, KDD'00*, pages 109–110, 2000.

[190] Porter Stemming. http://tartarus.org/ martin/PorterStemmer/.

[191] Gilbert Strang. Introduction to linear algebra, Wellesley-Cambridge Press, 3 Edition, 1998.

[192] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *KDD*, pages 687–696, 2007.

[193] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *ICDM*, pages 418–425, 2005.

[194] M. Surdeanu, J. Turmo, and A. Ageno. A hybrid unsupervised approach for document clustering. In *KDD*, pages 685–690, 2005.

[195] C. Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In *SODA*, pages 526–527, 2004.

[196] The ArnetMiner System. http://www.arnetminer.org/.

[197] The DBLife System. http://dblife.cs.wisc.edu/.

[198] The Microsoft Libra System. http://libra.msra.cn/.

[199] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. In *KDD*, pages 717–726, 2007.

[200] B. Taskar, M. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *in Neural Information Processing Systems*, 2003.

[201] S. T. Teoh and K.-L. Ma. Paintingclass: interactive construction, visualization and exploration of decision trees. In *KDD*, pages 667–672, 2003.

[202] H. Tong, C. Faloutsos, and J. Pan. Fast random walk with restart and its applications. In *ICDM*, pages 613–622, 2006.

[203] J. R. Tyler, D. M. Wilkinson, and B. A. Huberman. Email as spectroscopy: automated discovery of community structure within organizations. *Communities and technologies*, pages 81–96, 2003.

[204] Vivisimo. http://www.vivisimo.com/.

[205] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *ICML*, pages 1103–1110, 2000.

[206] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *ICML*, pages 577–584, 2001.

[207] X. Wang, N. Mohanty, and A. McCallum. Group and topic discovery from relations and text. In *LinkKDD '05: Proceedings of the 3rd international workshop on Link discovery*, pages 28–35, 2005.

[208] X. Wang, N. Mohanty, and A. McCallum. Group and topic discovery from relations and their attributes. In *NIPS*, pages 1449–1456, 2006.

[209] X. Wang and C. Zhai. Learn from web search logs to organize search results. In *SIGIR'07*, pages 87–94, 2007.

[210] S. Wasserman and K. Faust. Social network analysis: Methods and applications, Cambridge University Press, 1994.

[211] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[212] F. Wei, C. Wang, L. Ma, and A. Zhou. Detecting overlapping community structures in networks with global partition and local expansion. In *APWeb*, pages 43–55, 2008.

[213] Y.-C. Wei and C. K. Cheng. Toward efficient hierarchical designs by ratio cut partitioning. In *Proceedings of the IEEE International Conference on Computer Aided Design*, pages 298–301, 1989.

[214] Michael C. Wendl. H-index: however ranked, citations need context. *Nature*, 449(403), 2007.

[215] H. C. White, S. A. Boorman, and R. L. Breiger. Social structure from multiple networks. *American Journal of Sociology*, 81:730–780, 1976.

[216] H. C. White, S. A. Boorman, and R. L. Breiger. Social structure from multiple networks: I. blockmodels of roles and positions. *Am. J. Social*, 81:730–779, 1976.

[217] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *KDD*, pages 266–275, 2003.

[218] S. White and P. Smyth. A spectral clustering approach to finding communities in graphs. In *Proceedings of the 5th SIAM International Conference on Data Mining*, 2005.

[219] R. J. Williams and N. D. Martinez. Simple rules yield complex food webs. *Nature*, 404:180–183, 2000.

[220] A. P. Wolfe and D. Jensen. Playing multiple roles: Discovering overlapping roles in social networks. In *ICML-04 Workshop on Statistical Relational Learning and its Connections to Other Fields*, 2004.

[221] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. Scan: a structural clustering algorithm for networks. In *KDD*, pages 824–833, 2007.

[222] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

[223] K. Y. Yip and M. K. Ng. Harp: A practical projected clustering algorithm. *IEEE TKDE*, 16(11):1387–1397, 2004.

[224] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

[225] O. R. Zaïane, J. Chen, and R. Goebel. Method of producing web search result. patent filed 072-56.

[226] O. R. Zaïane, J. Chen, and R. Goebel. Dbconnect: Mining research community on dblp data. In *Joint 9th WEBKDD and 1st SNA-KDD Workshop '07 (WebKDD/SNA-KDD'07)*, 2007.

[227] O. R. Zaïane, A. Foss, C.-H. Lee, and W. Wang. On data clustering analysis: Scalability, constraints, and validation. In *PAKDD*, pages 28–39, 2002.

[228] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54, 1998.

[229] O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to Web search results. *Computer Networks*, 31(11–16):1361–1374, 1999.

[230] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *SIGIR '04*, pages 210–217, 2004.

[231] H. Zhang, C. L. Giles, H. C. Foley, and J. Yen. Probabilistic community discovery using hierarchical latent gaussian mixture model. In *AAAI*, pages 663–668, 2007.

[232] H. Zhang, B. Qiu, C. L. Giles, H. C. Foley, and J. Yen. An lda-based community structure discovery approach for large-scale social networks. In *Proceedings of the IEEE Conference on Intelligence and Security Informatics*, pages 200–207, 2007.

[233] S. Zhang, R. Wang, and X. Zhang. Identification of overlapping community structure in complex networks using fuzzy c-means clustering. *Physica A*, 374:483–490, 2007.

[234] W. J. Zhou, J. R. Wen, W. Y. Ma, and H. J. Zhang. A concentric-circle model for community mining in graph structures, Technical Report MSR-TR-2002-123, http://research.microsoft.com/research/pubs/,2002.