May 2018

Advancing Community Detection Using Keyword Attribute Search

Sanket Chobe

ADVANCING COMMUNITY DETECTION USING KEYWORD ATTRIBUTE SEARCH

by

Sanket Chobe

Bachelor of Technology (I.T.)
Government College of Engineering, Amravati, India
2011

A thesis submitted in partial fulfillment of
the requirements for the

Master of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2018

UNLV | GRADUATE COLLEGE

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

April 6, 2018

This thesis prepared by

Sanket Chobe

entitled

Advancing Community Detection Using Keyword Attribute Search

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Justin Zhan, Ph.D.
Examination Committee Chair

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Interim Dean

Hal Berghel, Ph.D.
Examination Committee Member

Wolfgang Bein, Ph.D.
Examination Committee Member

Xiangning Chen, Ph.D.
Graduate College Faculty Representative

Abstract

As social network structures evolve constantly, it is necessary to design an efficient mechanism to track the influential nodes and accurate communities in the networks. The attributed graph represents the information about properties of the nodes and relationships between different nodes, hence, this attribute information can be used for more accurate community detection. Current techniques of community detection do not consider the attribute or keyword information associated with the nodes in a graph. In this thesis, I propose a novel ideal of online community detection using a technique of keyword search over the attributed graph. First, the influential attributes are derived based on the probability of occurrence of each attribute type-value pair on all nodes and edges, respectively. Then, a compact *Keyword Attribute Signature* is created for each node based on the unique id of each influential attribute. The attributes on each node are classified into different classes, and this class information is assigned on each node to derive the strongest association among different nodes. Once the class information is assigned to all the nodes, I use a keyword search technique to derive a community of nodes belonging to the same class. The keyword search technique makes it possible to search community of nodes in an online and computationally efficient manner compared to the existing techniques. The experimental analysis shows that the proposed method derive the community of nodes in an online manner. The nodes in a community are strongly connected to each other and share common attributes. Thus, the community detection can be advanced by using keyword search method, which allows personalized and generalized communities to be retrieved in an online manner.

Acknowledgements

Firstly, I would like to express my sincere gratitude and appreciation to Dr. Justin Zhan for being a wonderful academic advisor and committee chair. Dr. Zhan provided a strong support to my research work [CZ18, WZC18] by sharing all the knowledge, resources, and opportunities. He encouraged and guided me throughout my Master's program.

Besides my advisor, I would like to thank Dr. Hal Berghel for serving on my thesis committee. It has been a great pleasure and honor working with Dr. Berghel as a Teacher Assistant(TA). He mentored and supported me in my TA work, due to which I could handle multiple responsibilities.
I would like to gratefully acknowledge Dr. Wolfgang Bein and Dr. Xiangning Chen for serving on my thesis committee and evaluating my research work.
I shall remain indebted to the Graduate College, Graduate Financial Services, and Office of International Students and Scholars forever, for their help and support throughout my Master's program.

A special thanks to Mr. Rizwan Patel, who is the Director at Caesars Entertainment Corporation located at Las Vegas, USA. He has mentored and encouraged me to explore different advanced technologies during my tenure as an intern in Caesars Entertainment, which helped in my research.
Finally, I would like to thank my family: my parents and my sister for their tremendous support to my academic pursuit. It would have been an impossible journey without their support and encouragement.

Sanket Chobe

University of Nevada, Las Vegas

May 2018

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

INTRODUCTION

Graphs have played an important role in the big data and social network analysis in recent years [GEJN02, AB02]. It is straightforward to represent and manage the information from different domains with the help of graphs. It is efficient to define the relationship between different entities and get the required knowledge from graphs. Due to changing dynamics of users over the Internet, different applications of the social network, and the tremendous rise in the volume of information, it is critical to design a method to efficiently extract the knowledge and discover hidden patterns among the group of users. The community detection is widely used to derive a group of nodes closely interacting and having a strong relationship with each other, which is helpful to get more positive results from social network analysis. For example, if the nodes in a network represent the user profiles in a social or professional network, and edges represent the association or interaction between these nodes, then a community of nodes provides the group of users who are closely interacting with each other and share some similar characteristics. The community detection can be used to derive the information about a group of people who go to the same school, who work at the same organization, or group of books by the same publication. This close interaction and strong association among the nodes in a network can also be used to predict the link formation or edge creation. Deriving the strongest community among the large network graph has become an increasingly important and critical task [SS17, RTB07] in graph analytics. Several different techniques of community detection are already defined. Although more advance work is in progress, there are limited efficient mechanisms to get the knowledge from attributed graphs. The large volume of information is represented in the form of network graph, where some key attributes are assigned to the nodes, and relationships between different nodes are represented in the form of edges. It is important to consider these attributed graphs for the community detection, which

can give us much more useful information than general network graphs. Current techniques of community detection can be categorized into three different categories. First, the *Structure-based* community detection, where the community of nodes is formed based on the connectivity between nodes. The techniques like *Label Propagation* [RAK07], *Random Walk* [AF02], and *Modularity Optimization* [ZWW$^+$09] focus on the probability of edge creation or connectivity between two nodes. These probabilistic models derive a community based on the actual connection and the possible connection between different nodes, and group them together based on the connectivity of the nodes. This type of community detection gives the nodes which have high connectivity with each other and form a cohesive structure. However, these techniques do not consider the attributes associated with each node, hence, the accurate communities may not be derived from the attributed graphs. Another class of community detection technique considers the attributes associated with the nodes, also known as the *Attribute-based* community detection. However, it is possible that the two nodes which share the same attributes may not be connected to each other, hence the community of nodes may not be structurally cohesive. There are some methods which consider the attribute similarity as well as the connectivity between nodes while deriving a community of nodes. The fundamental principle behind all these methods is to create a group of nodes which share some common features. But, we do not have any prior information about how many communities and what type of relationships are present between the nodes in a community. Finding communities in an online manner is a more efficient and accurate way of extracting knowledge on a real-time scale. Thus, I introduce a novel method [CZ18] to derive the community of nodes in an online manner based on the attribute similarity and the connectivity of nodes in a graph. This thesis defines a new mechanism [CZ18] to extract different community of nodes from attributed network graphs by using the keyword search technique. The proposed method is used to derive the communities in an online manner, which makes it possible to generate personalized communities based on the user queries. Since a keyword search approach is used to detect the communities, the proposed method is able to derive communities in a more accurate and efficient manner. The key contributions to the novel method are listed as follows:

1. A novel technique of community detection is developed based on the existing revolutionary research [GEJN02, SS17, FC12, LNMG09, NAXC08, ENG04, SCFS12].

2. The *node attributes* are used to represent the keywords in the attributed graphs to design a novel algorithm of community detection using keyword search over attributed graphs.

3. Since the multiple keyword attributes are present on every node, it is important to find the influential attributes from the set of attributes, which in turn leads to influential nodes. The probability of each attribute on all the nodes of the graph is derived and the less important attributes are filtered out if the probability of occurrence is less than a threshold value. The attributes with a greater probability of occurrence are called as *node-weighted* attributes, and considered as the influential attributes.

4. Apart from *node-weighted* attributes, it is necessary to find the probability of connectivity of nodes which share similar attributes. The probability of each attribute shared among different nodes helps to filter out the attributes which are shared the least among different nodes. Hence, another threshold value is given to filter out the attributes which have the least probability of being shared among different nodes. The attributes with a greater probability of sharing between two connected nodes are called as *edge-weighted* attributes, and considered as the influential attributes.

5. Once the influential attributes are determined, I create and assign a vector of keyword attributes on each node. These node attributes can be classified into different class of attributes based on the similarity of two attributes. I use *Jaccard Similarity Index* to measure the similarity between two nodes.

6. A class label is assigned to each node while classifying the attributes in different classes. This class information is used as a keyword on each node, and can be used for personalized as well as generalized community detection by using keyword search techniques.

7. The experimental analysis shows that the proposed algorithm is able to derive the personalized community for a query, and generalized communities for all classes of attributes. Thus, the proposed mechanism is able to derive community of nodes in an online and efficient manner based on the keyword attributes.

The rest of the thesis is organized as follows. Chapter 2 presents the related work and background. Chapter 3 gives the detailed explanation of the proposed approach. Chapter 4 describes the *Keyword Search* based algorithm for personalized as well as generalized community detection in detail. Chapter 6 analyzes the performance of the proposed approach. Chapter 7 describes the experimental analysis, and chapter 8 concludes the thesis.

Chapter 2

LITERATURE REVIEW

This chapter reviews the related work about the *community detection* and *keyword search* over the large network graphs.

2.1 Community Detection on Attributed Graphs

Since the inception of graph theory, many algorithms have been proposed for different applications of graph theory [SS17, FC12]. Significant research has been done on the different properties and applications of graphs in different domains like biology, social, and informational networks [GEJN02]. The different mathematical properties of graphs add an advantage to the use of graphs in different domains. For instance, social network analysis started in 1930's and has become one of the most critical and revolutionary areas of research in the big data community.

2.1.1 Community Detection

A community is also known as a *cluster* or *module*, and defined as a group of vertices which probably share common features, or have a strong relationship with each other formed by the strong distribution of edges between the vertices. In *Figure 2.1*, a graphic representation of a sample graph with communities is shown.

A community [For10] can also be defined as a dense subgraph, since the nodes in same communities have dense connection with each other than that of different communities. Each community represents a functional system or working unit, due to which it is necessary to find different effective techniques of community detection. Since the inception of graph-based community detection, many algorithms have been already proposed. The different community detection techniques can

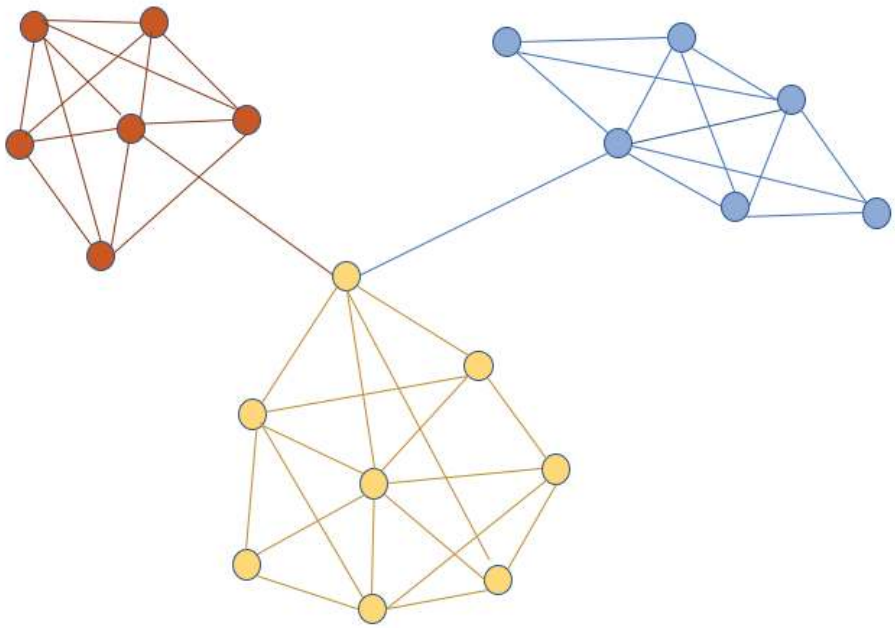


Figure 2.1: A Simple Graph with Communities Highlighted by Different Colors.

be categorized as follows:

2.1.2 Structure-based Community Detection

This class of community detection considers the connectivity between different nodes and determines the probability of possible connection between nodes, and groups them together in one community. Existing methods like the *Label Propagation* [RAK07] and *Random Walk* [AF02] generate the community structure based on a probabilistic model. The *Label Propagation* [RAK07] assigns the label to each node and changes the label of nodes based on the label of neighbors. Though the *Label Propagation* [RAK07] is computationally efficient, it does not consider the attributes associated with each node, hence, may not be an effective method to create a group of nodes based on the label of neighbors for attributed graphs. The *Random Walk* [AF02] method works on the principal of *Markov Chain Process*, where nodes are grouped together based on the probability of transition from one node to another. Since the current *Random Walk* [AF02] based methods do not consider the attributes associated with nodes, hence, the generated communities may not have attribute similarity between the nodes in same community. Girvan and Newman proposed a new measure known as the *modularity* [GEJN02, ZWW$^+$09], which is defined as the fraction of connections within a community in the actual network minus expected fraction of con-

nections in a random network. As per the definition of *modularity*, a maximum *modularity* gives the best partition of a network [GEJN02]. Fortunato and Barthelemy discovered that the *modularity optimization* [FB07] gives extra importance to the number of connections in a network, and hence this method cannot correctly classify some specific cases of networks. The main limitation of the *modularity optimization* [FB07] approach is that it does not consider the overall size of a community. To overcome this limitation, Rosvall and Bergstrom [RTB07, RB07] used the full description length of a partition of a network to compress the network-based communication process. Li et al. [LBL+16] proposed a fast and accurate measure of mining community structure by providing a kernel function to measure the leadership of each node. Once the leader nodes are determined, a discrete-time dynamical system is used to assign the community for each node dynamically. All these methods consider the actual and possible connection between different nodes, and generate the structurally cohesive community structure. However, none of the above methods considers the node attributes, and hence, may not be the accurate measure to generate community structures for an attributed graph.

2.1.3 Attributed-based Community Detection

Many different techniques of community detection have considered large complex graph without any keyword or attributes on its nodes. There are limited techniques of community detection for the network graphs having node attributes [TFGER07]. Fang et al. [FCLH16] proposed the community detection on large attributed graphs by creating an index tree based on the keyword attribute information. Zhou et al. [ZCY09] proposed a method to derive the clusters by computing the pairwise similarity between the nodes using keywords and links between the nodes. Ruan et al. [RFP13] proposed a method called as CODICIL, where new edges are created based on the content similarity, and then effective graph sampling is done to boost the efficiency of graph clustering. In another approach [XKW+12], the attributed graph community detection is done based on probabilistic inferences. CESNA [YML13] detects the overlapping communities by assuming communities generate content. He et al. proposed another method known as MISAGA, [HC18] for mining subgraphs in an attributed graph. MISAGA [HC18] defines a probabilistic measure to determine the strength of association between a pair of attribute values, then it determines the degree of association between each pair of vertices to group them together in one community. All these methods consider the degree of association between a pair of vertices based on a set of attributes on vertices. However, these methods may not consider the structure cohesiveness or the

connectivity between a pair of vertices while creating the community structure. It is necessary to design a mechanism which will consider both, structure cohesiveness and attribute similarity for a group of nodes belonging to the same community. There are limited techniques which achieve this objective [YJCZ09].

2.1.4 Online Community Detection

There are some techniques to determine the communities in an *online* manner, that is based on a query request. Few of these methods [SG10, CXWW14, LQYM15, CXW$^+$13] obtain the community for given vertex V based on the query over q. Such a personalized community detection technique requires different measures like the *minimum degree*, *k-core*, etc., which generate the structurally cohesive communities. Sozio et al. [SG10] proposed the first algorithm known as the *Global* to find the $\widehat{k\text{-}core}$ containing *vertex q*. Cui et al. [CXWW14] proposed the *Local* to enhance the efficiency of the *Global* by expanding techniques to local search space. There are many other methods like *k-clique* [CXW$^+$13] and *k-truss* [HCQ$^+$14] which search the communities in large complex networks, but all these techniques assume non-attributed graphs, and does not consider the important keyword attribute information on nodes which can be used for the generation of more accurate communities in the graph.

In this thesis, I consider some important measures described by *structure-based*, *attribute-based*, and *online* community detection, and design a novel method to generate more accurate communities for large attributed graphs.

2.1.5 Keyword Search over Graphs

The keyword search over graphs [BHN$^+$02, DYW$^+$07, KPC$^+$05, KA11] have attracted significant attention in recent years since it provides valuable information to users without the knowledge of underlying entities, schema, or access mechanism. To search information over such large complex graphs, many advanced keyword search techniques [BHN$^+$02, DYW$^+$07, KPC$^+$05, KA11, YLC$^+$17] are already discovered. I use the concept of keyword search over graphs to generate different communities in the graph. I apply the naive keyword search approach to search the keyword attributes on the nodes, and group them together based on the similarity between two nodes. The keyword search approach gives the flexibility of searching the required group of nodes in an online manner. Since the social network graphs may have multiple attributes assigned to their nodes, a mechanism is designed to derive clusters in the graph based on the attribute similarity among

different nodes. The next chapter defines the problem, and describes the key definitions and major aspects of the proposed approach.

Chapter 3

PROPOSED APPROACH

In this chapter, the proposed approach of community detection using keyword attribute search is explained in detail. Before explaining the proposed approach in detail, it is necessary to define the problem and provide some corresponding definitions and lemmas to support the proposed approach.

3.1 Problem Statement

Let a network graph is denoted as $G = (V, E, \Lambda)$, which is an undirected and attribute graph. V is the set of vertices, E is the set of undirected edges, and Λ is the set of attributes assigned to each vertex $v_i \in V$ in the graph. If two vertices v_i and v_j are connected to each other through an undirected edge, then such a graph is known as a *connected graph*. The set of attributes for all vertices $v_i \in V$ is denoted as $\Lambda = \{attr_1, attr_2, attr_3 ... attr_n\}$, and the set of attribute values associated with each vertex v_i is denoted as $attr_{ji} = \{attr_{1i}, attr_{2i}, ... attr_{ji}\}$.

In this thesis, an undirected attributed graph $G = (V, E, \Lambda)$ is considered. Let n and m be the size of V and E, respectively. *Table 3.1* represents the meaning of all the symbols used in the thesis.

3.1.1 Preliminary

A community can be defined as a subgraph of G that have the nodes densely connected to each other to form a cohesive structure, and sparsely connected to nodes in other communities. The structure cohesiveness of a graph G is defined by how different nodes are connected to each other. The *minimum degree* of all the vertices in a community is k or more [CXWW14, DGM06, LQYM15, Sei83, SG10], also known as $k - core$, which is an important condition for structure cohesiveness

Table 3.1: Symbols and Meanings

Symbol	Meaning
$G(V, E)$	An undirected graph with set of vertices V and set of edges E
λ	A set of attributes for set of vertices V
q	keyword or query to be searched on attributed graph G
$deg_G(V)$	The degree of vertex V in G
$L(X(q))$	The length of the set of attribute X for vertex q
$G[S']$	The largest connected subgraph of G such that $q \in G[S']$, and $S \subset X(v)$
$Sim(q1, q2)$	Jaccard Similarity Index to measure similarity between $q1$ and $q2$
θ_c	Maximum threshold constant for similarity between $q1$ and $q2$
W_v	Maximum threshold constant for Node-Weight on attribute type-value pairs
W_e	Maximum threshold constant for Edge-Weight on attribute type-value pairs
$GD(q1, q2)$	Shortest path length or Geodesic distance between $q1$ and $q2$

of G.

3.1.2 Definition 1 (k-core)

Given an integer k ($k \geq 0$), the $k - core$ [Sei83, SG10, 10.14] of G denoted by G_k is the largest subgraph of G such that, $\forall v \in G_k deg_{G_k}(v) \geq k$. The notion of $K - core$ [Sei83, SG10, FCLH16, Sei83] makes sure that all the nodes in a community G_k are densely connected to each other in some way, and hence makes the structure cohesive.

3.1.3 Definition 2 (Core Number)

Given a vertex $v \in V$, the *core number* [Sei83, SG10, FCLH16] of v, denoted by $core_G(v)$, is defined as the highest order of a *k-core* that contains the vertex v.

3.1.4 Definition 3 (Jaccard Similarity Index)

Given a graph $G = (V, E)$ and two vertices $v_1, v_2 \in V$ which have set of attributes X_1 and X_2 respectively, the measure of similarity between the two vertices can be given by the *Jaccard Similarity Index* [WBW+13] $Sim(v_1, v_2)$ and defined as follows:

$$Sim(v_1, v_2) = \frac{|(X_1 \cap X_2)|}{|(X_1 \cup X_2)|} \tag{3.1}$$

The *Jaccard Similarity Index* [WBW+13] measure can be useful to give some threshold constant, which can be used to classify the vertices with a similarity of attributes greater than or equal to the threshold constant into a common group of vertices.

3.1.5 Definition 4 (Shortest Path Length or Geodesic Distance)

Apart from the attribute similarity, it is necessary that the two nodes v_1 and v_2 are connected to each other at a minimum possible distance. If two nodes v_1 and v_2 are connected to each other, but far away from each other in a network than the other nodes, then such nodes may not be the part of a community. Since the communities contain nodes which are densely connected, the shortest path distance between two nodes should be minimal to make the community structure dense, or structurally cohesive. The *shortest path length* or *geodesic distance* [HKA16] can be used to calculate and specify the maximum threshold on the distance between two nodes v_1 and v_2. The *shortest path length* or *geodesic distance* [HKA16] can be given as follows:

$$GD(v_1, v_2) = min(\forall_i \forall_j d(v_i, v_j)) \tag{3.2}$$

3.2 Problem Definition

The problem is defined as follows:
For a given large complex graph G, an efficient mechanism should be designed to partition the graph into K disjoint subgraphs, where each subgraph will hold the following properties:

1. The vertices $v_i \in V$ in a subgraph should be connected to each other to form a cohesive structure, and sparsely connected to other subgraphs.

2. The vertices $v_i \in V$ which have similar attributes should be partitioned into the same group, while the vertices with different attributes should be partitioned into separate groups.

Based on the above properties, the formal problem definition for the community detection in attributed graphs is stated in the following definition:

3.2.1 Problem 1 (Attributed Community Detection)

Given a graph *G(V,E, λ)*, a positive integer constant k, a vertex $v \in V$, and a set of attributes $S \subset X(V)$, return a set of subgraphs such that following properties should be satisfied $\forall G_v \in G$:

1. *Connectivity:* $G_v \in G$ is a connected subgraph and contains $v \in V$;

2. *Structure Cohesiveness:* $\forall v \in G_v \ deg_{G_v}(v) \geq k$; all the nodes $v \in V$ in a subgraph have degree greater than or equal to the k_core value.

3. *Attribute Cohesiveness:* The number of shared attributes $L(G_v, S)$ among all the vertices v in G_v should be maximal, where $L(G_v, S) = \cap_{v \in G_v}(X(v) \cap S)$ represents the set of attributes from S shared among all the vertices v in G_v.

The above properties like *k-core and k-clique* make sure the community structure is structurally cohesive, and the property of *Jaccard Similarity Index* makes sure the community structure follows the attribute cohesiveness. Both of these requirements are critical for an accurate community detection. The focus is to design an efficient mechanism to derive communities not only in terms of interaction between the vertices in a community, but also in terms of attribute similarity, or the characteristics shared by vertices in a community.

3.2.2 Community Detection Using Keyword Search

It is useful to assign the *keyword attributes* on vertices to gain an accurate measure of a community. The vertices which have strong relationships and common properties can be grouped together in the same community based on *keyword attributes*. An algorithmic framework can be designed to classify *keyword attributes* on all vertices based on the similarity between a set of attributes on two vertices, and these classes of attributes can be used to search the vertices strongly related to each other, and hence, eventually the community. Following major steps are involved in the *community detection* using *keyword search* method:

1. Since each node $v \in V$ contains a set of attributes $X \in \lambda$, and each attribute type $attr_{ij} \in X$ might be distributed on the different nodes with different values, it is important to find the influential attribute type-value pairs among all the node attributes. The probability of each attribute type-value pair on all vertices can be used to get influential attributes. A probability threshold value called as $Node - weight$ is defined, which derive the influential attributes. If the probability of an attribute type-value pair is greater than or equal to $Node - weight$, then that attribute type-value pair is considered as an important attribute.

2. Once all the influential attribute type-value pairs are determined, I determine the connectivity of all the attribute type-value pairs which makes sure the structure cohesiveness. The

probability of each attribute type-value pair shared between all pair of vertices can be useful to get this information. A probability threshold value called as $Edge - weight$ is defined, which derives the influential attributes in terms of the connectivity. If the probability of sharing of any attribute type-value pair between all pair of vertices is greater than or equal to $Edge - weight$, then that attribute type-value pair is considered as an important attribute.

3. All the influential *keyword attribute* information on nodes can be used to construct an *Attribute Index Structure*, where different keyword attributes are classified and grouped together in separate groups based on the attribute similarity between the nodes, and their degree structure. The factors like the *Jaccard Index* and *K_core* are used to measure the similarity of attributes between a pair of vertices and the connectivity of the vertices sharing similar attributes, respectively.

4. While classifying each vertex in the graph based on the attributes like *city, school, mutual interest or activities, events, etc.*, a different class of attributes can be identified and stored in the *Attribute Index Structure*. This attribute class information can be assigned as a label to each node of the graph with the *keyword attribute* information, to find strongly associated nodes.

5. Since I create an index structure for a different class of attributes in a keyword information, I create queries for the *personalized community detection* using attribute class information assigned to different nodes with the *keyword attributes*.

6. The *naive keyword search* algorithm is used to determine the *personalized communities* in the graph based on a query by using different *class of attributes* assigned to all vertices in the graph.

7. For a *generalized community detection*, the *naive keyword search* algorithm is used to determine all communities in the graph based on the class information and *keyword attribute* information on different nodes in the graph. All the classes can be accessed iteratively to determine different nodes which belong to the same class, and which share strong association in terms of keyword information. The group of such nodes will form a community structure in the graph.

In the next chapter, an algorithmic framework is defined and explained in detail for each major step.

Chapter 4

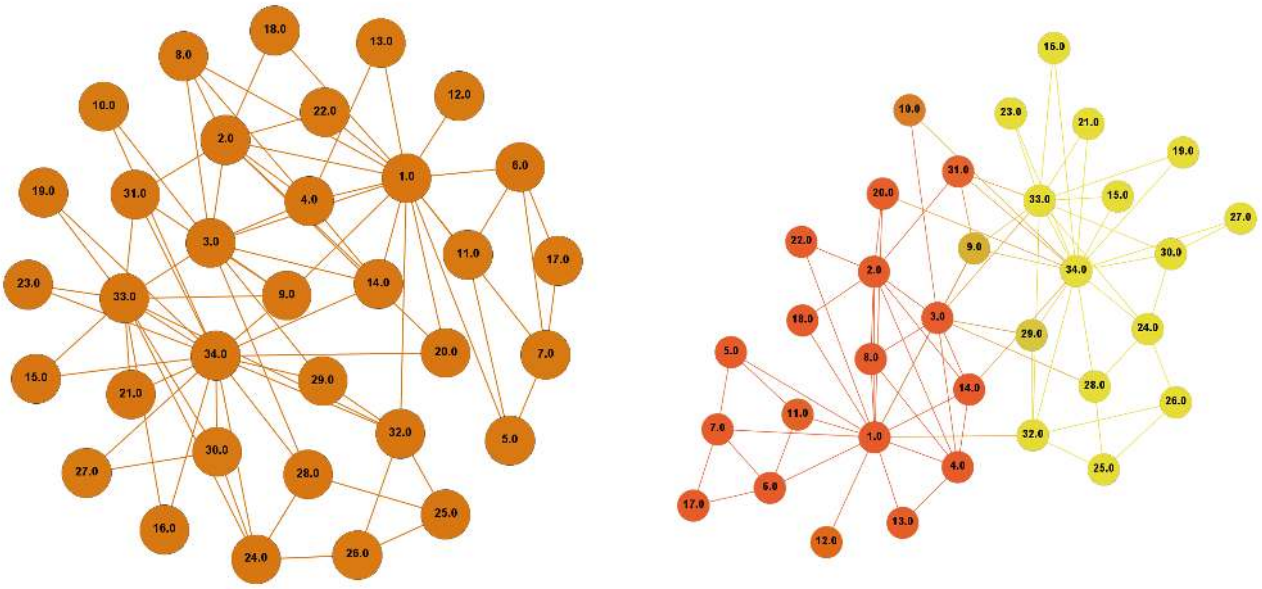
KEYWORD SEARCH-BASED ALGORITHM

The proposed approach is explained in detail in this chapter. The *Keyword Attribute Search-based* algorithm requires that every node should have some keyword or attributes associated with them, these *keyword attributes* are used to search a specific type of attribute and group them together in one class. For example, I use two network datasets *Karate-club* and *Political Book* which have ground-truth communities associated with them. The *Karate-club* network is shown in *Figure 4.1(a)* and the corresponding ground-truth communities are displayed in *Figure 4.1(b)*. The *Political Books* network is shown in *Figure 4.2(a)*. I assign random attributes on each node of both the datasets, and derive communities by using *Keyword Search-based* algorithm. The random attributes assigned to each node of the *Karate-club* network is shown in *Table 4.1*.

Table 4.1: Node Attributes of Sample Graph in Figure 4.1(a)

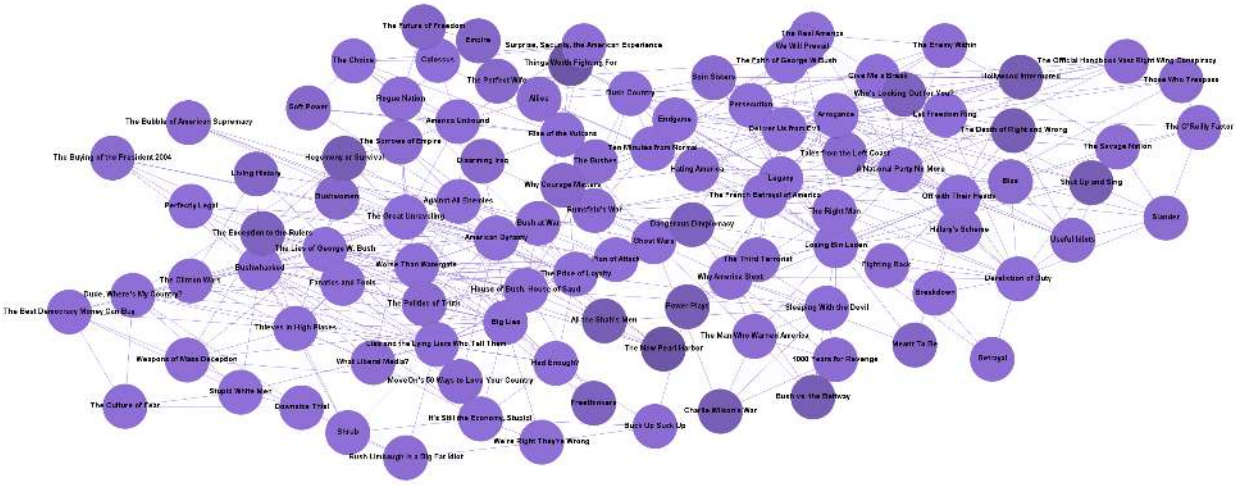
Attribute Type	**Set of Values**
School	{'UNLV','SUNY','ASU'}
Employer	{'Caesars','MGM','Amazon','Google'}
Role	{'Student','Professor','Software Engineer','Manager','Team Lead'}
Sports	{'Soccer','Baseball','Badminton','Basketball'}
Vehicle	{'Toyota','Hyundai','Mercedes','Audi','BMW','Chevrolet'}
City	{'Las Vegas','New York','Phoenix'}
Country	{'USA'}

Following major steps are involved in the *Keyword Attribute Search-based* algorithm:



(a) Karate Club Network (b) Communtiy Derrived from Graph in (a)

Figure 4.1: Karate Club Network Dataset with Ground-truth Communities



(a) Political Books Network

Figure 4.2: Political Books Network Dataset with Ground-truth Communities

4.1 Node-weight

The first step, in the process of our *community detection* problem, is to determine the influential nodes among all the nodes in a graph. Since there are multiple attributes on each node, it is necessary to find the influential attributes among a set of attributes, and filter out the less important attributes. The influential attributes can be found based on the probability of each attribute type-

value pair among all the nodes in the graph. Let $O(attr_i)$ is the number of times $attr_i$ present among all the vertices, N be the number of nodes in the graph, then the probability of $attr_i$ can be given as follows:

$$P(attr_i) = \frac{O(attr_i)}{N} = \frac{\sum_{j=1}^{N} attr_i}{N} \tag{4.1}$$

For the example network shown in *Figure 4.1(a)*, let's assume that the maximum threshold for *Node-weight* W_v =20.0%, then the corresponding important attributes are listed in *Table 4.2*.

Table 4.2: Node-weight for Karate-Club Network

Id	**Attribute Type-Value pair**	**Probability**
1	Country : USA	100.0
2	School : UNLV	50.0
3	School : SUNY	35.0
4	City : New York	50.0
5	City : Las Vegas	50.0
6	Role : Student	38.00
7	Sports : Baseball	38.00
8	Sports : Soccer	32.00
9	Employer : Caesars	30.00
10	Employer : Google	27.00
11	Employer : Microsoft	24.00
12	Employer : MGM	21.00
13	Vehicle : Hyundai	22.00
14	Vehicle : Chevrolet	21.00

4.2 Edge-weight

Once all the important attributes are determined based on the probability of each attribute type-value pair on all the nodes of a graph, I consider the probability of occurrence of each attribute type-value pair on each edge, that is, how many times each attribute type-value pair is shared between all pair of vertices. The threshold value of probability of each attribute type-value pair on each edge can be called as the *Edge-weight*. If the probability of each attribute type-value pair on each edge is greater than or equal to *Edge-weight*, then that attribute type-value pair is considered to be influential. Let $O(attr_{ij})$ and $O(attr_{ik})$ be the number of times attribute $attr_i$ present on

vertices j and k, respectively. $O(attr_{ij}, attr_{ik})$ represents the number of times $attr_i$ shared between vertex j and k, when j and k are connected to each other. The probability of an attribute type-value pair shared among all the pair of vertices can be given as follows:

$$P(attr_{ij}, attr_{ik}) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} O(attr_{ij}, attr_{ik})}{N} \tag{4.2}$$

The *Edge-weight* is determined from the above important attribute type-value pairs listed in *Table 4.2*, and all the edges in the *Karate-club* network graph which share these attribute type-value pairs. Let's assume that the maximum threshold value for *Edge-weight* W_e =10.0%, then corresponding influential *Edge-weight* attributes are listed in *Table 4.3*.

Table 4.3: Edge-weight for Karate-Club Network

Id	**Attribute Type-Value pair**	**Probability**
1	Country : USA	100.0
2	School : UNLV	38.0
3	School : SUNY	36.0
4	City : New York	38.0
5	City : Las Vegas	22.0
7	Sports : Baseball	16.00
9	Employer : Caesars	10.00
11	Employer : Microsoft	15.00
13	Vehicle : Hyundai	12.00
14	Vehicle : Chevrolet	13.00

4.2.1 Keyword Attribute Signature

The influential attributes can be stored in a decreasing order of the probability value, and a unique index $k_i \rightarrow attr_i | \forall attr_i \in X$ value can be assigned to the attribute type-value pair. The *Keyword Attribute Signature* contains a vector of these unique index values for each attribute type-value pair on a node. Thus, the *Keyword Attribute Signature* is a compact representation of the attribute values on each node. The *unique index* associated with each attribute type-value pair shown in *Table 4.3* is used to create the *Keyword Attribute Signature* on each node.

4.3 Attribute Index Structure

In the next step, I create an *Attribute Index Structure* for the *Keyword Attribute Signature* assigned to all nodes. Given a largely attributed graph *G = (V, E)*, the task is to determine the different class of attributes from all the *Keyword Attribute Signatures* on all vertices. Since we do not have prior information on the type of attributes in the graph, this information can be determined from the *keyword attributes* on different nodes. The *Attribute Index Structure* is created iteratively for all nodes and will contain all the class of attributes in the large network graph. Each class of attribute is created iteratively by comparing every node with each other, if the two nodes are similar to each other based on *Jaccard Similarity Index*, then the *Keyword Attribute Signature* on two nodes can be merged into one class, and two nodes belong to the same class. The *Attribute Index Structure* would consist of the inverted list of *Keyword Attribute Signature* for each node with a corresponding class, first visited node, and the total number of vertices which belong to the same class of attributes. The *Attribute Index Structure* can be denoted as I = $\{C_1 : \{X_1, V_1, count(C_1)\}, C_2 : \{X_2, V_2, count(C_2)\}, ..., C_n : \{X_n, V_n, count(C_n)\}\}$, where C_i is the class assigned to each attribute set X_i, and V_i is the first node from which the class C_i is derived. The following pseudo-code is used to create the *Attribute Index Structure*:

Table 4.4 represents the *Keyword Attribute Index* structure created for the *Karate-Club* network dataset.

Table 4.4: Attribute Index Structure for Karate-Club graph

Class	**Node V**	**Attributes**	**count**
1	1	{1, 2, 3, 4, 5, 7, 9, 11, 13, 14}	10
2	24	{1, 3, 7, 9, 11, 13}	7

4.3.1 Keyword Attribute Class Information

Once the *Attribute Index Structure* is created, it is used to assign the *class* of attribute C_i on each vertex $v_i \in V$ along with the *Keyword Attribute Signature* information. The main idea behind the *class* creates awareness about strong relationship or common properties between the nodes. Initially, each vertex with the *Keyword Attribute Signature* is considered as a separate community while searching it's relationship with other vertices, this *Keyword Attribute Signature* based search leads to different nodes which share the same attribute information, which eventually helps derive

Algorithm 1 Keyword Attribute Index Structure

Input:
G = (V, E);
V = $(V_1, V_2 \ldots, V_n)$: Set of vertices in the graph;
X = $(X_1, X_2 \ldots, X_m)$: Set of influential attributes derived from Node-Weight and Edge-Weight;
$X_i = \{attr_1(V_i), attr_2(V_i), \ldots, attr_j(V_i)\}$: Set of attributes associated with each vertex in the graph;

Output:
I = $\{C_1 : \{X_1, V_1, count(C_1)\}$, $C_2 : \{X_2, V_2, count(C_2)\}$,.., $C_n : \{X_m, V_n, count(C_n)\}\}$: Attribute index structure where C_n is the class assigned to each attribute set $X_m(V_n)$, and V_n is the first node which belongs to C_n ;

Initialize $i, j, k, count\_node = \emptyset$
for $v_i \in V$ **do**
  for $C_j \in I$ **do**
    $k\_core \leftarrow K\_core(G, v_i)$;
    $attr_i \leftarrow X(v_i)$;
    $attr_j \leftarrow X_j \in C_j$;
    $Sim(attr_i, attr_j) \leftarrow |attr_i \cap attr_j| \div |attr_i \cup attr_j|$;
    if $deg_{v_i} \geq k\_core$ **then**
      if $Sim(attr_i, attr_j) \geq \theta$ **then**
        $attr_i \cup attr_j \leftarrow X_j for X(v_i), Xj$;
        $V_j \leftarrow v_i$ for first $v_i \in V$ and $V_j \in C_j$;
        $count(v_i) + +$; {Number of nodes for each class of attribute}
        Set $X(v_i) \leftarrow X(v_i) \cup C_j$ where $X(v_i) \in X$;
      else
        $C_k \leftarrow k + 1$ {Create a seperate class for attribute $X(v_i)$};
        $V_k \leftarrow v_i$;
        $X_k \leftarrow X(v_i)$;
        $count(C_k) \leftarrow 1$;
        Set $I = \{C_k : \{(X_k, V_k, count(C_k)\}\}$;
        Set $X(v_i) \leftarrow X(v_i) \cup C_k$ where $X(v_i) \in X$;
      end if
    end if
  end for
end for
return I

the communities in graph. However, for a large network graph, there are numerous type of attributes associated with all nodes, hence it becomes necessary to classify *Keyword Attribute Signature* at each node in different classes. There should be a parameter θ_c to depict a maximum threshold of similarity between two set of *Keyword Attribute Signature* on two vertices v_1 and v_2, respectively, based on which it will be easy to prune the search space over a large network graph. I can verify

if an attribute is present in *Attribute Index Structure I*. If an attribute $attr_{ij} \in X(j)$ is present in the inverted list of attributes $\{C_j : X_j, V_j, count(C_j)\}$, then I assign the corresponding *class* C_j of attribute $attr_{ij}$ to the node v_i. If an attribute $attr_{ij} \in X(j)$ is not present in the inverted list, then the attribute $attr_{ij} \in X_j$ is compared with the existing attributes in I. The similarity between the two set of attributes is determined based on the *Jaccard Similarity Index* $Sim(v_i, v_j)$. If the $Sim(X_i(v_i), X_j(v_j))$ is greater than or equal to the threshold value θ_c, then the two attributes are combined together $X_i(v_i) \cup X_j(v_j)$, and same class C_i is assigned to the combination of attributes. If $Sim(X_i(v_i), X_j(v_j))$ is less than the threshold value θ_c, then a new *class* C_j is assigned to the attribute $X_j(V_(j))$ along with the node V_j in the *Attribute Index Structure I*. I keep track of the count of nodes $count_i(C_i)$ which belong to same class of attributes while creating the class of attributes in the *Attribute Index Structure I*. This information is useful in the personalized as well as generalized community detection. Based on the key definitions and *Attribute Index Structure I*, following lemma is derived to prove that the *Attribute Index Structure* is useful to classify nodes in different clusters, or groups to form communities.

Theorem 4.1. *Given* $G = (V, E)$ *and set of attributes* X*, if* $X_q = \{X_{q1}, X_{q2}, \ldots, X_{qn}\}$ *and* $\hat{X}_q = \{\hat{X_{q1}}, \hat{X_{q2}}, \ldots, \hat{X_{qn}}\}$ *are the set of attributes on the two vertices* q *and* $\hat{q}$ *respectively, and* $(q, \hat{q}) \in G_{X_q}$ *i.e.* q *and* $\hat{q}$ *belong to the same subgraph* G_{X_q}*. If* $L(X_q \cap \hat{X}_q) \geq K_q$*, then* $X_q \cup \hat{X}_q \in C_q$*, where* $C_q \in I$*.*

Proof. To prove this lemma, I use a proof by contradiction. Let $X_q = \{X_{q1}, X_{q2}, \ldots, X_{qn}\}$ such that $X_q \in C_q$, and $\hat{X}_q = \{\hat{X_{q1}}, \hat{X_{q2}}, \ldots, \hat{X_{qn}}\}$ such that $\hat{X}_q \in \hat{C}_q$. Assuming $\hat{X}_q \notin C_q$, then it means that X_q and $\hat{X}_q$ does not have any attribute in common, which proves C_q is not equal to $\hat{C}_q$ and $L(X_q \cap \hat{X}_q) = \phi$. Thus, for every query or keyword vertex $X_q \in C_q$ and $q \in G_q$ i.e. q belongs to the subgraph G_q, and $\hat{X}_q \in \hat{C}_q$ and $\hat{q} \in \hat{G}_q$ i.e. $\hat{q}$ belongs to the subgraph $\hat{G}_q$. This contradicts the given assumption that $(q, \hat{q}) \in G_{X_q}$ and also fails to satisfy the *attribute cohesiveness*. This proves the given lemma. ■

4.4 Personalized Community Detection

As discussed in the previous steps, the *Attribute Index Structure I* contains the class C_i of attributes , first node, and the number of nodes which belongs to class C_i, also, keyword attribute $X_i(v_i)$ is updated to include the class C_i for each node $v_i \in V$. This information is crucial for the personalized community detection. The *personalized community detection* can be defined as, a group of nodes

having same class C_{q_i} on each node for a given query $Q = \{q_1, q_2, \ldots, q_i\}$, with i keywords in the query. For a given query Q which has keywords or class of attributes, I access the class information C_i, the number of nodes which belong to the same class $count_i(C_i)$, and the first vertex V_i which belongs to class C_i from I. Once this information is fetched from the *Attribute Index Structure* I, the naive keyword search algorithm is used to search nodes which belong to class C_i. This keyword search starts at the node V_i derived from I and continues to search the nodes with same class C_i of attributes as node v_i. The keyword search has the upper bound of threshold length $count_i(C_i)$, and continues the keyword search until the $count_i(C_i)$ number of nodes are not matched with the given class C_i. Following pseudo-code is used for the *Keyword Search* required in personalized community detection.

4.5 Generalized Community Detection

The generalized community detection can derive all communities from a network graph based on the *Keyword Attribute Signature* associated with each node in the graph. Since each node $v_i \in V$ has a keyword attribute $X_i(v_i)$ associated with it, and this attribute belongs to some class $C_i \in I$, this information can be used to determine the similarity between two nodes and group them together in one community. The personalized community detection is based on the keyword search query Q, while generalized community detection may not need keyword search query Q. The *Attribute Index Structure* I has required information about different attributes $X_i(v_i)$ on node $v_i \in V$ and class C_i of these attributes. It also contains the information of first node V_i and the number of nodes $count_i(C_i)$ that belongs to class C_i. All this information can be used for the generalized community detection. The process starts by scanning the *Attribute Index Structure* I for each class C_i of the attribute, then perform the keyword search at node V_i associated with C_i in I. The *Keyword Search* process explained in the personalized community detection leads to a community of nodes for each class $C_i \in I$. Same process is performed recursively for each class C_i of the attribute, corresponding node V_i, and the attributes $X_i(V_i)$ associated with the node V_i. The following pseudo-code explains the generalized community detection process in detail:

The personalized community detection can be used to derive a particular community from a graph in an online manner. However, the generalized community detection can be used to derive all the communities from a graph without any query in an online manner. For example, if we look at the *Karate-Club* network graph shown in *Figure 4.1(a)* with different attributes on each node, and the *Attribute Index Structure* shown in *Table 4.4*, I derive the community of nodes that belong

Algorithm 2 Personalized Community Detection

Input:
$G = (V, E, X)$;
$V = (V_1, V_2 \ldots, V_n)$: Set of vertices in the graph;
$X = (X_1, X_2 \ldots, X_m)$: Set of attributes in the graph;
$X_i = \{attr_1(V_i), attr_2(V_i), \ldots, attr_j(V_i)\}$: set of attributes associated with each vertex in the graph;
$I = \{C_1 : \{X_1, V_1, count(C_1)\}$, $C_2 : \{X_2, V_2, count(c_2)\}$,.., $C_n : \{X_m, V_n, count(C_n)\}\}$: Attribute index structure where C_n is the class assigned to each attribute set $X_m(V_n)$, and V_n is the first node which belongs to C_n ;
$Q = \{q_1, q_2, \ldots, q_n\}$: Query which has a list of keywords or class information;

Output:
$Comm(C_i)$: Community of the nodes which share the keyword information in the query Q;

Initialize $i, j, node = \emptyset$;
for $q_i \in Q$ **do**
  $C_i \leftarrow q_i$;
  if $C_i \in I$ **then**
    $node_i = V_i$; $\{I = \{C_i : \{X_i, V_i, count_i(C_i))\}\}\}$
    $k\_core \leftarrow K\_Core(G, v_i)$
    if $deg_{node_i} \geq k\_core$ **then**
      $node\_count_i \leftarrow count_i(node_i)$
      $bfs\_list(node_i) \leftarrow BFS\_Tree(G, v_i)$
      while $count \leq node\_count_i$ **do**
        for $v_j \in bfs\_list(node_i)$ **do**
          if $deg_{v_j} \geq k\_core$ **then**
            $Class_j \leftarrow attr_j(v_j)$ $\{attr_j(v_j) = C_i\}$
            if $q_i \in Class_j$ **then**
              $Comm_i(q_i) \leftarrow v_j$; {Mark v_j for community }
            end if
          end if
        end for
      end while
    end if
  end if
end for
return $Comm(Q)$;
$BFS\_Tree(G, v_j)$
Initialize $j, k = \emptyset$;
for each $v_k \in in\_neighbor(v_j)$ **do**
  Pick up the $v_k \in BFS\_TREE$
end for

to the same class of attributes as shown in *Figure 4.3(a)*. Similarly, based on the previously defined measures, I derive the communities for the *Political-Books* network dataset shown in *Figure 4.2*.

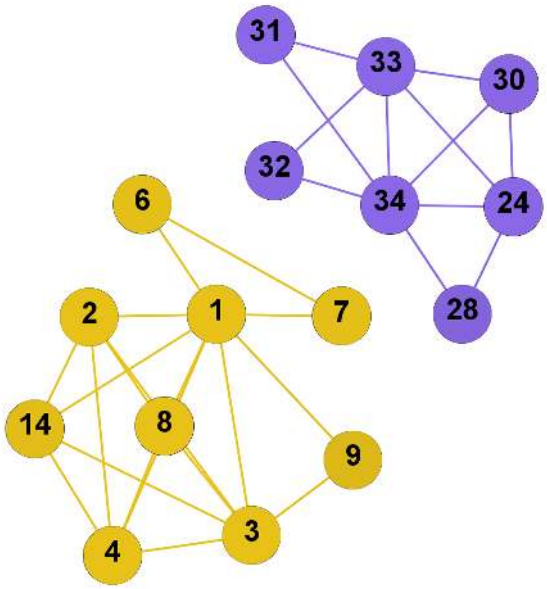
Algorithm 3 Generalized Community Detection

Input:
$G = (V, E, X)$;
$V = (V_1, V_2 \ldots, V_n)$: Set of vertices in the graph;
$X = (X_1, X_2 \ldots, X_m)$: Set of attributes in the graph;
attr(V_i) = $\{attr_1(V_i), attr_2(V_i), \ldots, attr_j(V_i)\}$: set of attributes associated with each vertex in the graph;
I = $\{C_1 : \{X_1, V_1, count(C_1)\}$, $C_2 : \{X_2, V_2, count(C_2)\}$,.., $C_n : \{X_m, V_n, count(C_n)\}\}$: Attribute index structure where C_n is the class assigned to each attribute set $X_m(V_n)$, and V_n is the first node which belongs to C_n ;

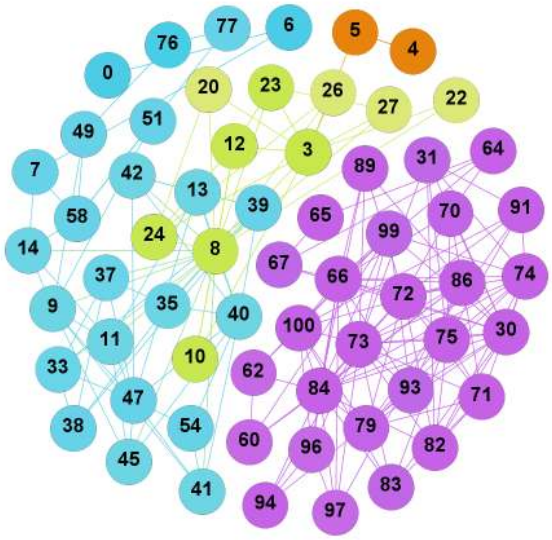
Output:
$Comm(C_i)$: Communities which have the nodes v_i belong to $C_i \in I$;

Initialize $i, j, node = \emptyset$;
for $C_i \in I$ **do**
  $C_i = q_i$;
  $node_i = V_i$; $\{I = \{C_i : \{X_i, V_i, count_i(C_i))\}\}\}$
  $k\_core \leftarrow K\_core(G, v_i)$
  if $deg_{node_i} \geq k\_core$ **then**
    $node\_count_i \leftarrow count_i(node_i)$
    $bfs\_list(node_i) \leftarrow BFS\_Tree(G, v_i)$
    while $count \leq node_count_i$ **do**
      for $v_j \in bfs\_list(node_i)$ **do**
        if $deg_{v_j} \geq k\_core$ **then**
          $Class_j \leftarrow attr_j(v_j)$ $\{attr_j(v_j) = C_i\}$
          if $q_i \in Class_j$ **then**
            $Comm_i(q_i) \leftarrow v_j$; {Mark v_j for community }
          end if
        end if
      end for
    end while
  end if
end for
return $Comm(Q)$;
$BFS\_Tree(G, v_j)$
Initialize $k = \emptyset$;
for each $v_k \in in\_neighbor(v_j)$ **do**
  Pick up the $v_k \in BFS\_TREE$
end for

All the communities are displayed in *Figure 4.3(b).*



(a) Communities for Karate Club Network



(b) Communities for Political Books Network

Figure 4.3: Communities Generated from Network Dataset Using the Proposed Method

Chapter 5

ALGORITHM ANALYSIS

The complexity analysis of the proposed method can be divided into following major parts based on the major steps involved in the community detection process.

5.1 Node-weight and Edge-weight

The $Node-weight$ and $Edge-weight$ are calculated by calculating the probability of each attribute on each node v_i and edge e_{ij}, respectively. Let's assume that there are n nodes and m edges in a network graph, and each node contains an average of c attributes, then the time required to calculate $Node-weight$ and $Edge-weight$ can be given as follows:

$$O(attr_i(v_j)) = \sum_{i=1}^{c}\sum_{j=1}^{n} O(attr_i(v_j)) = O(c*n) = O(n) \tag{5.1}$$

$$O(attr_i(e_{jk})) = \sum_{i=1}^{c}\sum_{j=1}^{m}\sum_{k=1}^{m} O(attr_i(e_{jk})) = O(c*m) = O(m) \tag{5.2}$$

5.2 Maximizing Degree of Nodes

Identifying the nodes, which have a maximum degree, satisfy the requirement of the *structure cohesiveness*. The *k-core* measure is used to identify such nodes with a maximum possible degree. The *k-core* of a graph G can be identified within the time complexity of $\mathcal{O}(m)$, where m is the number of lines. Since all the n nodes of graph G are traversed to identify the *k-core* value, the complexity of the process to maximize the degree of nodes can be given as follows:

$$O(K\_core) = min(deg_{K\_core}(G, V)) = O(n+m)) \tag{5.3}$$

$$O(K\_core) = O(n) = O(|V| + |E|) \tag{5.4}$$

5.3 Attribute Index Structure Creation

As explained in the previous section, the *Attribute Index Structure* is created to classify *Keyword Attribute Signature* on each node into different classes. These classes are used to generate the communities based on the similarity of a set of attributes on two nodes of the graph. Since I consider the nodes which have degree greater than or equal to the threshold value of *k-core*, nodes which have degree less than the *k-core* are rejected and will not be part of any community. This criterion prune the search space over a large network graph. There is another threshold known as *Jaccard Similarity Index* θ_c, which verify the similarity between two *Keyword Attribute Signatures*. If the similarity between two sets is greater than or equal to θ_c, then such sets are combined into one class C_i, otherwise, the two sets are classified into two different class of attributes C_i and C_j, respectively. If I consider the worst case scenario where all the nodes are densely connected to each other, hence, they have maximal degrees associated with them, then all the n nodes of the graph G are considered for the *Attribute Index Structure* creation. Also, if I assume that there are c attributes on each node of the graph G, then the time required for the comparison of the two *Keyword Attribute Signatures* would be some constant value. Hence, the total time required for the creation of *Attribute Index Structure* can be given by the following equation:

$$O(I) = \sum_{i=1}^{n} \sum_{j=1}^{n} O(n * c) = O(n) = O(|V|) \tag{5.5}$$

5.4 Keyword Search for Community Detection

Every class of attributes in the *Attribute Index Structure* $C_i \in I$ is associated with a vertex V_i, and the number of nodes $count_i(C_i)$ which belong to the class C_i. This information is used for the personalized as well as generalized community detection by using the *Keyword Search* method. Since the generalized community detection uses each class of attributes $C_i \in I$, the *Keyword Search* method is executed for every class and generate communities. The *Keyword Search* starts at the first vertex V_i associated with class C_i, then it searches iteratively for keyword C_i on every node in *Breadth First Search (BFS) tree* oriented at root V_i ($v_i \in BFS\_TREE(V_i)$). During the keyword search, the class C_i is compared with the class information present in attributes of node $attr_i(v_i)$, if the attribute information matches with the class C_i, then those nodes are grouped together in the same community marked with class C_i. It is also necessary to consider another important

requirement for cohesive community structure, i.e. *shortest path length* between two nodes. The *shortest path length* between two nodes should be less than or equal to a maximum threshold path length. Thus, the time required for the generalized community detection with keyword search would consist of the total time required for finding *Node-weight*, *Edge-weight*, and *k-core*, creating *Attribute Index Structure*, retrieving the *BFS tree* for vertex $v_i \in C_i$, and determining the nodes having *shortest path length* less than or equal to a maximum threshold path length. The time required for the *BFS tree* creation is $O(|V| + |E|)$, but the nodes in the graph have maximum degree forming a dense structure, hence, it is safe to assume that the time required for the *BFS tree* creation is dominated by the number of edges, that is $O(|E|)$, or $O(m)$. Now, the generalized community detection searches for all the classes $C_i \in I$, hence, the *BFS tree* is retrieved for each vertex v_i associated with $C_i \in I$. In the worst case, each vertex v_i belongs to separate class C_i, hence, the *BFS tree* requires the $O(n * |E|)$ or $O(|V| * |E|)$ time. The time required to calculate the shortest path length for n or $|V|$ vertices is $O(n^2)$ or $O(|V|^2)$. Thus, the total time complexity for the generalized community detection can be given as follows:

$$O(Comm_i(I)) = max(O(|V|), O(|E|), O(|V| + |E|), O(|V| * |E|), O(|V|^2)) \tag{5.6}$$

Since as per the assumption, the given graph is undirected, attributed, and dense graph, hence the number of edges $|E|$ dominate the number of nodes $|V|$, which results in the time complexity to be bounded with the number of edges or degree of the nodes. Hence, the resultant time complexity for the generalized community detection can be given as follows:

$$O(Comm_i(I)) = O(|V| * |E|)) \tag{5.7}$$

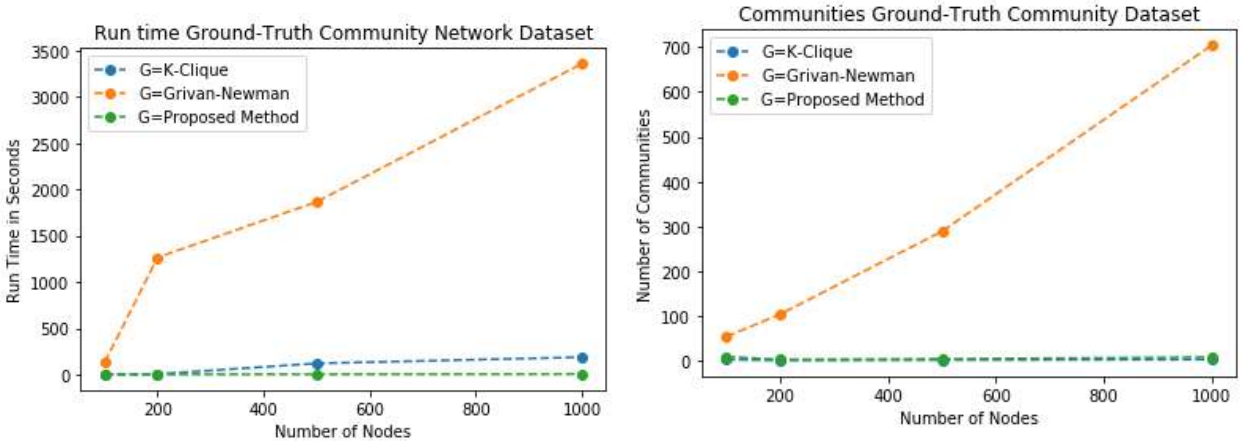
Chapter 6

EXPERIMENTAL RESULTS

Different experiments are performed to verify the accuracy of the proposed method. I consider only the undirected and attributed graph for all the experiments. All the experiments are executed on *64 GB* main memory in *Intel Core i5 @ 3.70GHz* on an Windows 10 operating system. *Python 2.7* is used to implement the algorithms with *networkx* package for graph related operations.

6.1 Experimental Setup

I divide my experiments into three parts. The first part of the experiment compares the proposed approach with the existing methods. I use the network datasets like *Karate-Club, American Football, Political-Books, Dolphin-network, email-EU-core, DBLP*, and *Amazon* [LK14] with the ground-truth communities for the comparison experiment. The second part contains the experiments on smaller datasets, where a number of nodes in the graph are less than or equal to 1000, while the third part contains experiments on the large datasets where the number of nodes in the graph is greater than 10000. Since the community structure would contain the dense subgraphs, I include the variation in a number of edges by creating synthetic graphs having $2000, 5000$, and 10000 nodes respectively, and the probability of edge creation $0.50, 0.40$, and 0.30 respectively. I distribute a number of attributes randomly on each node of all the above graphs where no attribute values are assigned to any node, so that the threshold value for attribute classification varies, and the resultant community structure can be verified. I use real datasets like *YouTube video crawl [CDL08], Twitter User Profiles [KLPM10], Skytrax Airline Reviews, Terrorist Data [GG17], Caesars Entertainment anonymous dataset, and Facebook*. I randomly assign attributes to the datasets having only edge lists, and randomly create edges with a certain probability of edge creation for



(a) Run Time for Ground-truth Community Datasets (b) Number of Communities for Ground-truth Community Datasets

Figure 6.1: Run Time vs Number of Communities for Ground-truth Community Datasets

the datasets having node list only. The *networkx* package of *Python 2.7* is used for all the graph related operation in the experiments.

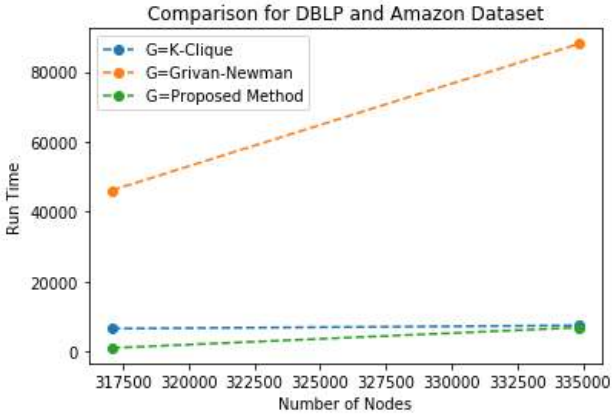
6.1.1 Experimental Result Analysis

Table 6.1: Comparison with Existing Methods

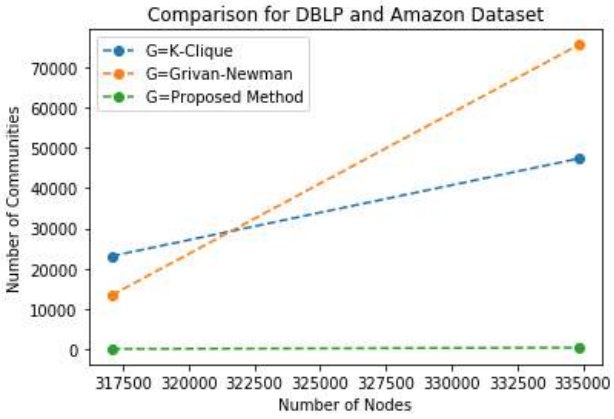
Legend: T1: Time for K-Clique; C1: #Communities for K-Clique; T2: Time for GN (Girvan-Newman); C2: #Communities for GN; T3: Time for Proposed Method; C1: #Communities for Proposed Method;

Graph	**T1**	**C1**	**T2**	**C2**	**T3**	**C3**
Karate	1	3	1	33	1	2
Football	1	4	1	114	1	5
Political-Books	1	4	1	104	1	4
Dolphins	1	3	1	61	1	3
email-EU-Core	187	3	3300	772	50	8
DBLP	6480	47307	46080	13477	936	128
Amazon	7380	23134	88080	75499	6791	468

Table 6.1 shows the details about the runtime and a number of communities in each comparison experiment. *Figure 6.1* and *Figure 6.2* displays the result for comparison of required runtime and the number of communities generated for existing methods, and the proposed method respectively. *Tables 6.2 and 6.3* show the statistics for the small as well as large graph datasets, respectively. The table shows statistics about the number of nodes, the number of edges, the probability of

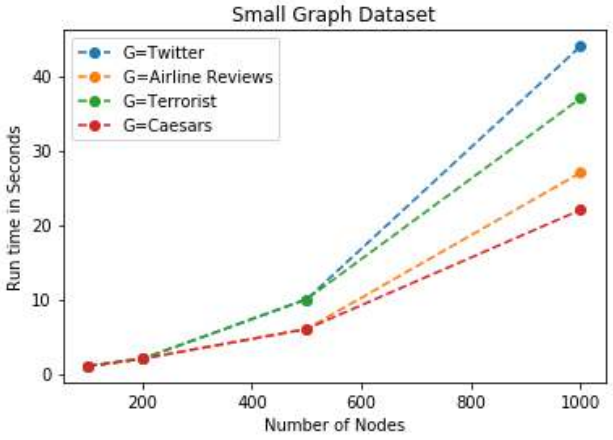


(a) Run Time for DBLP and Amazon Datasets

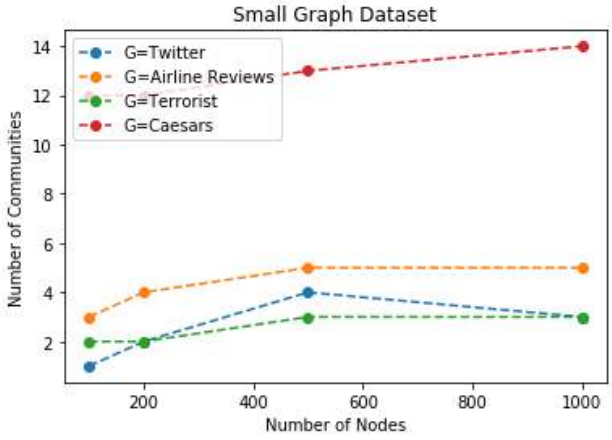


(b) Number of Communities for DBLP and Amazon Datasets

Figure 6.2: Run Time vs Number of Communities for Amazon and DBLP Datasets



(a) Run Time for Small Graph Datasets



(b) Number of Communities for Small Graph Datasets

Figure 6.3: Run Time vs Number of Communities for Small Graph Datasets

edge creation, and the number of attributes on each node. *Tables 6.4 and 6.5* show the statistics about the experimental results on the small as well as large graph datasets, respectively. The tables show detailed information of each experiment, where the number of nodes and edges are mentioned along with the threshold value for the similarity of attributes between two nodes, a threshold value for the shortest path length, the number of communities, and the time required for the proposed method to generate these communities. All this detailed information shows the authenticity of the proposed method to generate more accurate communities. All the experimental results are depicted in *Figure 6.3* and *Figure 6.4*, respectively.

Table 6.2: Small Graph Datasets

Graph	Nodes V	Edges E	Prob. of edge	Attributes on each node
Twitter User Profile	100	3426	P=0.70	26
Twitter User Profile	200	11793	P=0.70	26
Twitter User Profile	500	74625	P=0.60	26
Twitter User Profile	1313	517068	P=0.60	26
Skytrax Airline Reviews	100	3514	P=0.70	7
Skytrax Airline Reviews	200	13882	P=0.70	7
Skytrax Airline Reviews	500	74907	P=0.60	7
Skytrax Airline Reviews	1005	300202	P=0.60	7
Terrorist Data	100	3457	P=0.70	7
Terrorist Data	200	13969	P=0.70	7
Terrorist Data	500	74753	P=0.60	7
Terrorist Data	1000	299420	P=0.60	7
Caesars Entertainment	100	3439	P=0.70	6
Caesars Entertainment	200	13929	P=0.70	6
Caesars Entertainment	500	74835	P=0.60	6
Caesars Entertainment	1000	299953	P=0.60	6

6.1.2 A Case Study

I create a small case study based on a real time graph dataset provided by *Caesars Entertainment Corporation, Las Vegas, USA* . This dataset contains the anonymous real time attribute data collected from the *Caesars Entertainment Corporation WiFi* data, a test graph is created to represent the information about patrons visiting different properties of *Caesars Entertainment*, time, and places of their visit at a particular property. I create a graph $G = (V, E)$ with $|V| = 200$ nodes and $|E| = 12000$ edges with 60% probability of edge creation in the graph at different nodes, this makes the graph structurally cohesive. I assign different attributes $X_i(v_i)$ like, *Patron_Id, Path_From_Property, Path_To_Property, Time_From_Property, Time_To_Property*, and *Place_Visited* on all the vertices $v_i \in V$. Since each attribute may have different values on each node, the probability threshold value of *Node-weight =10.0* and *Edge-Weight =1.0* is set. *Table 6.6* and *Table 6.7* represent all the attribute type-value pairs having probability greater than or equal to *Node-weight* and *Edge-weight*, respectively. *Table 6.8* represents the *Attribute Index Structure* for the graph dataset. *Figures 6.5, and 6.6(a), 6.6(b), 6.6(c), 6.6(d), 6.6(e), and 6.6(f)* show the

Table 6.3: Large Graph Datasets

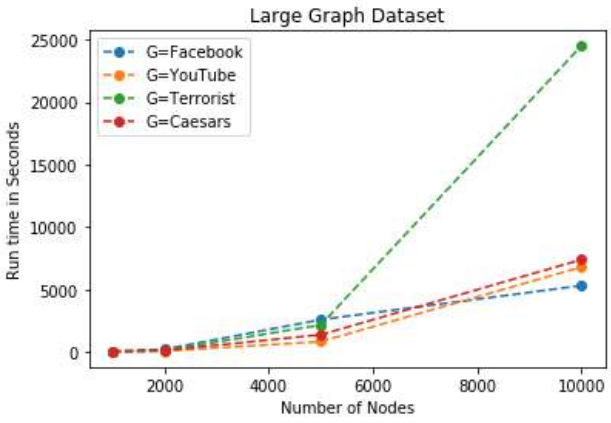
Graph	**Nodes V**	**Edges E**	**Prob. of edge**	**Attributes on each node**
Facebook	2000	1000025	P=0.50	7
Facebook	5000	4997567	P=0.40	7
Facebook	10000	14998579	P=0.30	7
Youtube Video Crawl	2000	998910	P=0.50	9
Youtube Video Crawl	5000	4997518	P=0.40	9
Youtube Video Crawl	10000	14996737	P=0.30	9
Terrorist Data	2000	1000470	P=0.50	7
Terrorist Data	5000	4998900	P=0.40	7
Terrorist Data	10000	15004225	P=0.30	7
Caesars Entertainment	2000	998937	P=0.50	6
Caesars Entertainment	5000	4887265	P=0.40	6
Caesars Entertainment	10000	14997657	P=0.30	6

original graph and the communities generated through the proposed method, respectively. Now, for the personalized community detection, I can create the queries q like, *find a community of nodes where people traveled from property A to property B*. Such a query q represents the class of *keyword attributes* $Path\_From\_Property = A, Path\_To\_Property = B$, and the *personalized community detection* algorithm finds all the nodes $v_i \in I$, where $C_i(v_i) = q$. This creates a community of nodes $Comm_i(C_i)$, where all the nodes have path from *Property A* to *B* . However, the generalized community detection finds the communities $Comm_i(C_i)$ for all $C_i \in I$, which contains all the nodes sharing the *keyword attribute* information, resulting in more accurate community detection as desired.

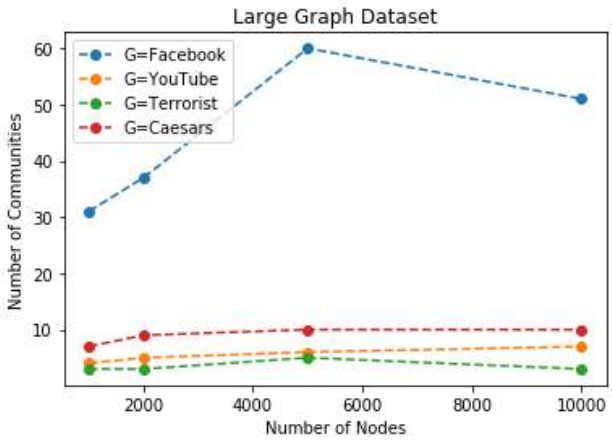
Table 6.4: Experimental Results on Small Graphs

Legend: V: Set of Vertices; E: Set of Edges; JCD: Jaccard Attribute Similarity Index; GD: Shortest Path Length(Geodesic Distance)

Graph	V	E	JCD	GD	Communities	Time
Twitter User Profile	100	3426	70%	3	2	1
Twitter User Profile	200	11860	70%	3	2	3
Twitter User Profile	500	74625	70%	3	4	10
Twitter User Profile	1313	517072	70%	5	3	25
Skytrax Airline Reviews	100	3514	50%	3	3	1
Skytrax Airline Reviews	200	13882	50%	3	3	2
Skytrax Airline Reviews	500	74907	50%	3	4	6
Skytrax Airline Reviews	1005	300202	50%	3	5	28
Terrorist Data	100	3457	70%	3	2	1
Terrorist Data	200	13969	70%	3	2	2
Terrorist Data	500	74753	70%	3	3	10
Terrorist Data	1000	299420	70%	3	3	37
Caesars Entertainment	100	3439	70%	3	10	1
Caesars Entertainment	200	13929	70%	3	12	2
Caesars Entertainment	500	74835	70%	3	12	6
Caesars Entertainment	1000	299953	70%	3	13	22



(a) Run Time for Large Graph Datasets



(b) Number of Communities for Large Graph Datasets

Figure 6.4: Run Time vs Number of Communities for Large Graph Datasets

Table 6.5: Experimental Results on Large Graphs

Legend: V: Set of Vertices; E: Set of Edges; JCD: Jaccard Attribute Similarity Index; GD: Shortest Path Length(Geodesic Distance)

Graph	**V**	**E**	**JCD**	**GD**	**Communities**	**Time**
Facebook	2000	1000025	70%	3	60	100
Facebook	5000	4997567	70%	3	37	2577
Facebook	10000	9996858	70%	3	51	1701
Youtube Video Crawl	2000	998910	70%	3	5	62
Youtube Video Crawl	5000	4997518	70%	3	5	817
Youtube Video Crawl	10000	14996737	70%	3	6	6785
Terrorist Data	2000	1000470	70%	3	9	100
Terrorist Data	5000	4998900	70%	5	10	2149
Terrorist Data	10000	15004225	70%	5	10	25000
Caesars Entertainment	2000	998937	70%	3	7	109
Caesars Entertainment	5000	4997265	70%	3	10	1321
Caesars Entertainment	10000	14997657	70%	3	10	7383

Table 6.6: Node-weight for Caesars-WiFi Dataset

Id	**Attribute Type-Value pair**	**Probability**
1	Property_Region : Gaming	21.0
2	To_Property :Cromwell	20.50
3	From_Property : Cromwell	20.50
4	From_Property : Harrahs_LV	18.0
5	From_Property : Paris	17.0
6	To_Property : Caesars Palace	16.00
7	To_Property : Paris	15.00
8	From_Property : Caesars Palace	14.50
9	To_Property : Harrahs_LV	14.00
10	From_Property : Ballys	14.00
11	Property_Region : Cromwell_Valet	11.00
12	To_Property : Flamingo	10.00

Table 6.7: Edge-weight for Caesars-WiFi Dataset

Id	**Attribute Type-Value pair**	**Probability**
1	Property_Region : Gaming	4.5
2	To_Property :Cromwell	4.0
3	From_Property : Cromwell	4.0
4	From_Property : Harrahs_LV	3.16
5	From_Property : Paris	3.0
6	To_Property : Caesars Palace	3.00
7	To_Property : Paris	2.26
8	From_Property : Caesars Palace	2.19
9	To_Property : Harrahs_LV	2.00
10	From_Property : Ballys	2.00
11	Property_Region : Cromwell_Valet	1.09

Table 6.8: Attribute Index Structure for Caesars-WiFi Dataset

Class	**Node V**	**Attributes**	**count**
1	0	{2, 4, 5, 8, 10}	48
2	2	{0, 1, 6, 9, 10}	35
3	9	{0, 1, 2, 5}	15
4	11	{0, 6, 7, 9}	16
5	14	{0, 3, 4, 5, 6}	29
6	21	{0, 1, 6, 7, 10}	6
7	31	{2, 4, 6, 8, 10}	4
8	35	{0, 1, 2, 3}	9
9	47	{0, 2, 3, 6, 8}	12
10	50	{0, 5, 7, 8, 9}	13
11	61	{1, 2, 4, 10}	5
12	188	{0, 1, 10, 3, 4}	2

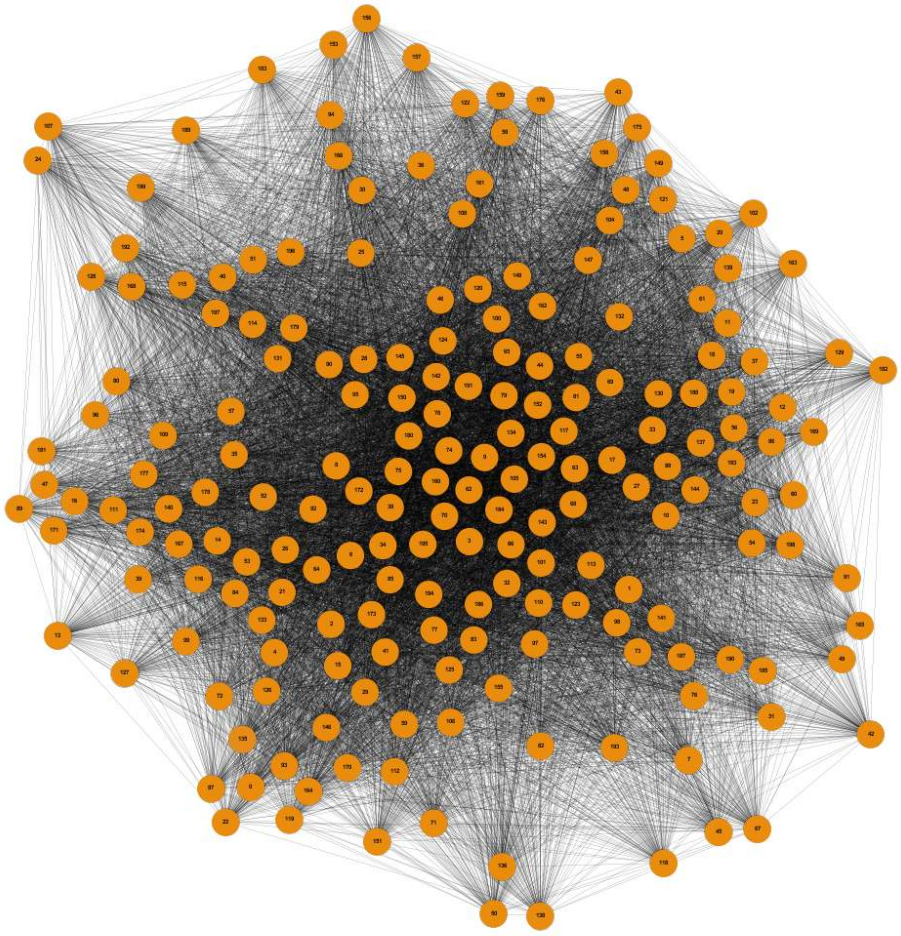
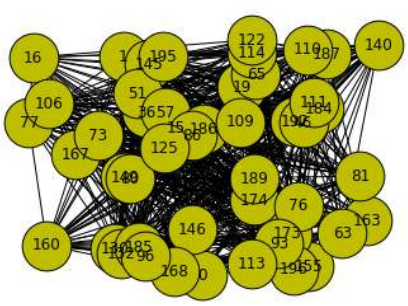
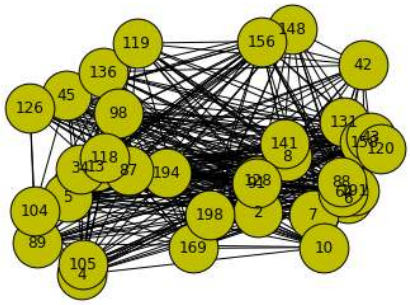


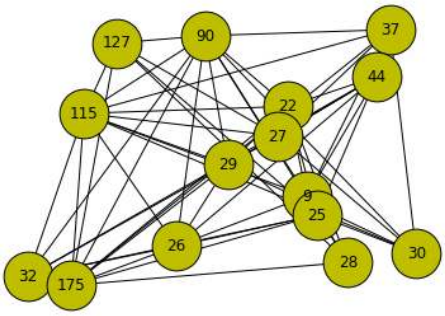
Figure 6.5: Original Graph Generated for Caesars Dataset



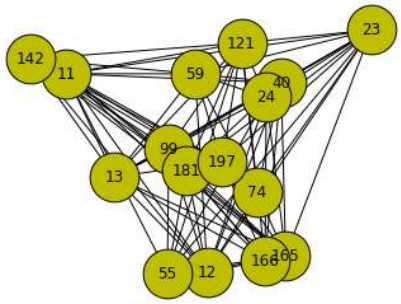
(a) From "Cromwell" to "Harrahs LV" at "Cromwell-Valet"



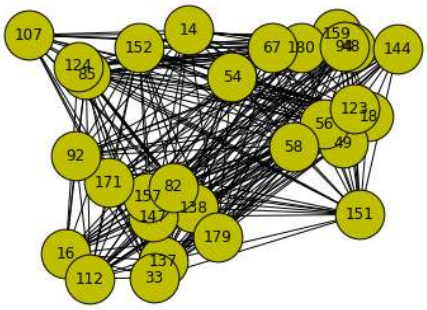
(b) From "Bally's" to "Cromwell" at "Gaming"



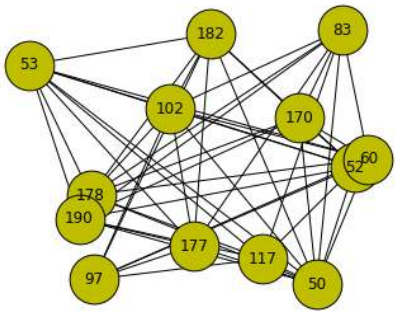
(c) From "Cromwell" to "Caesars Palace"



(d) From "Caesars Palace" to "Paris" at "Gaming"



(e) From "Harrahs LV" to "Caesars Palace" at "Gaming""



(f) From "Bally's" to "Harrahs LV"

Figure 6.6: Communities Detected for Caesars Way Finding Data

Chapter 7

CONCLUSION AND FUTURE WORK

A community structure derived from an attributed graph exhibits the structure and keyword attribute cohesiveness. The proposed *keyword search* based method derives communities with structure and keyword attribute cohesiveness by constructing an *Attribute Index Structure*. The *Attribute Index Structure* correctly represents different classes of attributes and helps to derive the community of nodes which share a finite number of *keyword attributes*, along with the cohesive structure. The proposed method is able to provide the *personalized* and *generalized community detection* method, which provides the flexibility to determine community of nodes in an online manner. Hence, the proposed method provides a more accurate measure of community detection in terms of the cohesive structure as well as keyword attribute similarity, compared to the existing algorithms. In addition, the proposed method provides a mechanism to generate communities in an online manner, which is more useful to determine real-time community of nodes. For future work, I intend to design a probabilistic model to predict the community of a node based on its connectivity with different nodes and attribute similarity. A probabilistic model can predict the class of a node, which can be further used for the *Advanced Keyword Search* techniques. I will also examine other metrics of keyword search over distributed graphs so that, the current work can be extended to a distributed environment and more efficient techniques of *keyword search* can be incorporated for the purpose of community detection.

Bibliography

[10.14] Efficient core maintenance in large dynamic graphs. *IEEE Transactions on Knowledge Data Engineering*, 26(10):2453–2465, 2014.

[AB02] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002.

[AF02] David Aldous and James Allen Fill. Reversible markov chains and random walks on graphs, 2002. Unfinished monograph, recompiled 2014, available at `http://www.stat.berkeley.edu/$\sim$aldous/RWG/book.html`.

[BHN+02] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using banks. In *Proceedings 18th International Conference on Data Engineering*, pages 431–440, 2002.

[CDL08] X. Cheng, C. Dale, and J. Liu. Statistics and social network of youtube videos. In *2008 16th Interntional Workshop on Quality of Service*, pages 229–238, June 2008.

[CXW+13] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yiqi Lu, and Wei Wang. Online search of overlapping communities. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 277–288, New York, NY, USA, 2013. ACM.

[CXWW14] Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. Local search of communities in large graphs. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 991–1002, New York, NY, USA, 2014. ACM.

[CZ18] Sanket Chobe and Justin Zhan. Advancing community detection using keyword attribute search. 2018. Manuscript submitted for publication to IEEE Access.

[DGM06] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes. *k*. *Phys. Rev. Lett.*, 96:040601, Feb 2006.

[DYW+07] B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin. Finding top-k min-cost connected trees in databases. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 836–845, April 2007.

[ENG04] Mark E.J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. 69:026113, 03 2004.

[FB07] Santo Fortunato and Marc Barthlemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[FC12] Santo Fortunato and Claudio Castellano. *Community Structure in Graphs*, pages 490–512. Springer New York, New York, NY, 2012.

[FCLH16] Yixiang Fang, Reynold Cheng, Siqiang Luo, and Jiafeng Hu. Effective community search for large attributed graphs. *Proc. VLDB Endow.*, 9(12):1233–1244, August 2016.

[For10] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75 – 174, 2010.

[GEJN02] Michelle Girvan and Mark E. J. Newman. Community structure in social and biological networks. proc. natl acad. sci. usa 99, 7821-7826. 99:7821–6, 07 2002.

[GG17] Alexander Gutfraind and Michael Genkin. A graph database framework for covert network analysis: An application to the islamic state network in europe. *Social Networks*, 51(Supplement C):178 – 188, 2017. Crime and Networks.

[HC18] T. He and K. C. C. Chan. Misaga: An algorithm for mining interesting subgraphs in attributed graphs. *IEEE Transactions on Cybernetics*, PP(99):1–14, 2018.

[HCQ+14] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. Querying k-truss community in large and dynamic graphs. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 1311–1322, New York, NY, USA, 2014. ACM.

[HKA16] M. B. Hutair, I. Kamel, and Z. Al Agbari. Social community detection based on node distance and interest. In *2016 IEEE/ACM 3rd International Conference on Big Data Computing Applications and Technologies (BDCAT)*, pages 274–279, Dec 2016.

[KA11] Mehdi Kargar and Aijun An. Keyword search in graphs: Finding r-cliques. *Proc. VLDB Endow.*, 4(10):681–692, July 2011.

[KLPM10] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 591–600, New York, NY, USA, 2010. ACM.

[KPC+05] Varun Kacholia, Shashank Pandit, Soumen Chakrabarti, S. Sudarshan, Rushi Desai, and Hrishikesh Karambelkar. Bidirectional expansion for keyword search on graph databases. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pages 505–516. VLDB Endowment, 2005.

[LBL+16] H. J. Li, Z. Bu, A. Li, Z. Liu, and Y. Shi. Fast and accurate mining the community structure: Integrating center locating and membership optimization. *IEEE Transactions on Knowledge and Data Engineering*, 28(9):2349–2362, Sept 2016.

[LK14] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[LNMG09] Yan Liu, Alexandru Niculescu-Mizil, and Wojciech Gryc. Topic-link lda: Joint models of topic and author community. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 665–672, New York, NY, USA, 2009. ACM.

[LQYM15] Rong-Hua Li, Lu Qin, Jeffrey Xu Yu, and Rui Mao. Influential community search in large networks. *Proc. VLDB Endow.*, 8(5):509–520, January 2015.

[NAXC08] Ramesh M. Nallapati, Amr Ahmed, Eric P. Xing, and William W. Cohen. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 542–550, New York, NY, USA, 2008. ACM.

[RAK07] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. , 76(3):036106, sep 2007.

[RB07] Martin Rosvall and Carl T. Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *Proceedings of the National Academy of Sciences*, 104(18):7327–7331, 2007.

[RFP13] Yiye Ruan, David Fuhry, and Srinivasan Parthasarathy. Efficient community detection in large networks using content and links. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1089–1098, New York, NY, USA, 2013. ACM.

[RTB07] Martin Rosvall and Carl T Bergstrom. An information-theoretic framework for resolving community structure in complex networks. 104:7327–31, 06 2007.

[SCFS12] Mrinmaya Sachan, Danish Contractor, Tanveer A. Faruquie, and L. Venkata Subramaniam. Using content and interactions for discovering communities in social networks. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 331–340, New York, NY, USA, 2012. ACM.

[Sei83] Stephen B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269 – 287, 1983.

[SG10] Mauro Sozio and Aristides Gionis. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 939–948, New York, NY, USA, 2010. ACM.

[SS17] Peng Gang Sun and Xiya Sun. Complete graph model for community detection. *Physica A: Statistical Mechanics and its Applications*, 471(Supplement C):88 – 97, 2017.

[TFGER07] Hanghang Tong, Christos Faloutsos, Brian Gallagher, and Tina Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 737–746, New York, NY, USA, 2007. ACM.

[WBW+13] Longju Wu, Tian Bai, Z. Wang, Limei Wang, Yu Hu, and Jinchao Ji. A new community detection algorithm based on distance centrality. In *2013 10th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 898–902, July 2013.

[WZC18] Jimmy Ming-tai Wu, Justin Zhan, and Sanket Chobe. Mining association rules for low frequency itemsets. 2018. Manuscript submitted for publication to PLOS ONE.

[XKW+12] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. A model-based approach to attributed graph clustering. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 505–516, New York, NY, USA, 2012. ACM.

[YJCZ09] Tianbao Yang, Rong Jin, Yun Chi, and Shenghuo Zhu. Combining link and content for community detection: A discriminative approach. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 927–936, New York, NY, USA, 2009. ACM.

[YLC+17] Y. Yuan, X. Lian, L. Chen, J. X. Yu, G. Wang, and Y. Sun. Keyword search over distributed graphs with compressed signature. *IEEE Transactions on Knowledge and Data Engineering*, 29(6):1212–1225, June 2017.

[YML13] J. Yang, J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *2013 IEEE 13th International Conference on Data Mining*, pages 1151–1156, Dec 2013.

[ZCY09] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph clustering based on structural/attribute similarities. *Proc. VLDB Endow.*, 2(1):718–729, August 2009.

[ZWW+09] X. S. Zhang, R. S. Wang, Y. Wang, J. Wang, Y. Qiu, L. Wang, and L. Chen. Modularity optimization in community detection of complex networks. *EPL (Europhysics Letters)*, 87(3):38002, 2009.

Curriculum Vitae

Graduate College

University of Nevada, Las Vegas

Sanket Chobe

Contact:

Phone: 725-266-1351

Email: chobe@unlv.nevada.edu

Degrees:

Bachelor of Technology in Information Technology 2011

Government College of Engineering, Amravati, India

Thesis Title: Advancing Community Detection Using Keyword Search

Thesis Examination Committee:

Chairperson, Dr. Justin Zhan, Ph.D.

Committee Member, Dr. Hal Berghel, Ph.D.

Committee Member, Dr. Wolfgang Bein, Ph.D.

Graduate Faculty Representative, Dr. Xiangnining Chen, Ph.D.