


RESEARCH

Open Access



# Advancing community detection using Keyword Attribute Search

Sanket Chobe<sup>1</sup> and Justin Zhan<sup>2\*</sup> 

\*Correspondence:

jzhan@uark.edu

<sup>2</sup> University of Arkansas, 227  
N. Harmon Ave., Fayetteville,  
AR, USA

Full list of author information  
is available at the end of the  
article

## 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 paper, we propose a novel algorithm 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, we 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.

**Keywords:** Keyword search, Attributed graphs, Attribute index, Personalized community detection, Generalized community detection

## Introduction

Graphs have played an important role in the big data and social network analysis in recent years [1]. 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 network, and edges represent the 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 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 [2, 3] in graph analytics. Several different techniques of community detection are already defined. Although more advance work is in progress, there are small number of 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 relationship 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* [4], *Random Walk* [5], and *Modularity Optimization* [6] focus on the probability of edge creation or connectivity between two nodes. These probabilistic models derive a community based on the actual and possible connection between different nodes, and group them together based on the connectivity of 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 each node, 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 types 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, we introduce a novel method to derive the community of nodes in an online manner based on the attribute similarity and the connectivity of nodes in a graph. This paper defines a new mechanism to extract community of nodes from attributed 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, proposed method is able to derive these 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 [2, 7–12].
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. A new measure, called as *node-weight*, is defined to derive the influential attributes among all the nodes.
4. Apart from *Node-weighted* attributes, it is necessary to find the probability of connectivity of nodes which share similar attributes. Hence, another threshold value, called as *edge-weight*, is given to filter out the attributes which have the least probability of being shared among different nodes, and find the influential nodes in terms of attribute similarity.
5. Once the influential attributes are determined, we assign a vector of keyword attributes on each node. These node attributes can be classified into different classes based on the similarity of two attributes. We use *Jaccard Similarity Index* to measure the similarity between two nodes.
6. These similar attributes between the two nodes contain the *Node-weighted* and *Edge-weighted* attributes. The average value of the weight of common attributes give the probability of edge creation between two nodes based on shared similar attributes. Thus, these nodes can be classified together based on probability of connectivity, and attribute similarity.
7. 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.
8. The experimental analysis shows that the proposed algorithm is able to derive the personalized community for a query, and generalized communities for all class 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 work is organized as follows. “[Related work](#)” section presents the related work and background. “[Methods](#)” section gives the detailed explanation of the proposed approach. “[Keyword search based algorithm](#)” section describes *Keyword Search* based algorithm for personalized as well as generalized community detection in detail. “[Algorithm analysis](#)” section analyzes the performance of the proposed approach. “[Experimental results](#)” section describes experimental analysis, and “[Concluding remarks](#)” section provides the conclusive remarks for the paper.

## Related work

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

### Community detection on attributed graphs

Since the inception of graph theory, many algorithms have been proposed for different applications of graph theory [2, 7, 9]. For instance, social network analysis started in 1930s and has become one of the most critical and revolutionary areas of research in the big data community.

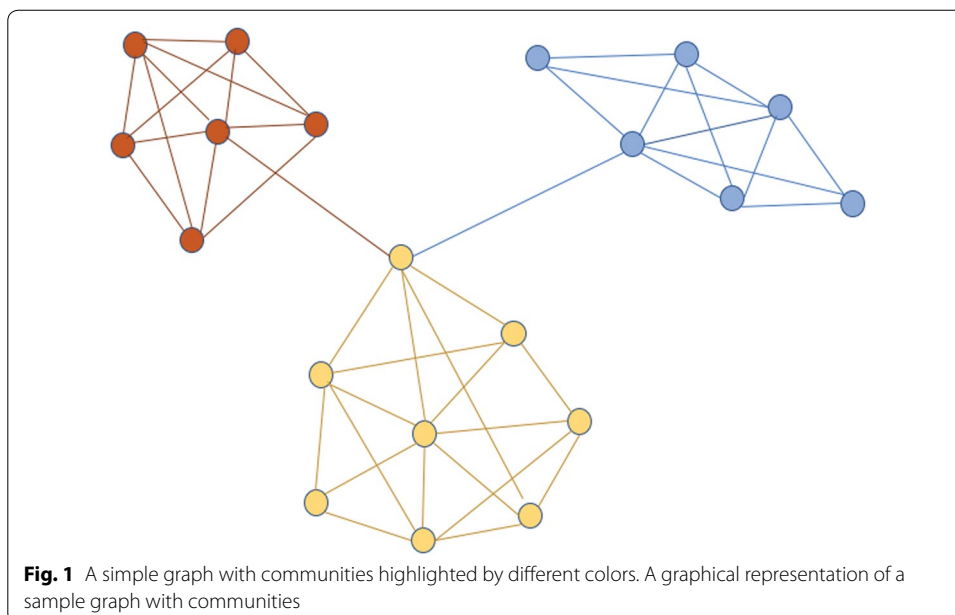
#### Community detection

A community is also known as a *cluster*, 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 vertices. In Fig. 1, a graphic representation of a sample graph with communities is shown.

A community [13] 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. The community detection techniques can be categorized as follows:

#### Structure-based community detection

This class of community detection determines the probability of possible connection based on the existing connections between nodes, and groups them together in one community. The existing methods like *Label Propagation* [4] and *Random Walk* [5] generate the community structure based on a probabilistic model. The *Label Propagation* [4] assigns label to each node and changes the label of each node based on the label of neighbors. Though *Label Propagation* [4] 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* [5] 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* [5] based methods do not consider the attributes associated with nodes, hence, the generated communities may not have attribute similarity between the nodes in a community. Girvan and Newman proposed a new measure known as the *modularity* [6, 9], which is defined as the fraction of connections within a community in the actual network minus expected fraction of connections in a random network. As per the definition of *modularity*, a maximum *modularity* gives the best partition of a network [9]. Fortunato and Barthelemy discovered that the *modularity optimization* [13] 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* [13] approach is that it does not consider the overall size of a community. To overcome this limitation, Rosvall and Bergstrom [3] used the full description length of a partition of a network to compress the network-based communication process. Li et al. [14] 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.

#### ***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 few techniques of community detection for the network graphs having node attributes [15]. Fang et al. [16] proposed the community detection on large attributed graphs by creating an index tree based on the keyword attribute information. Zhou et al. [17] 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. [18] 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 [19], the attributed graph community detection is done based on probabilistic inferences. CESNA [20] detects the overlapping communities by assuming communities generate content. He et al. proposed another method known as MISAGA, [21] for mining subgraphs in an attributed graph. MISAGA [21] 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 few techniques which achieve this objective [22].

### 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 [23–26] 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. [23] proposed the first algorithm known as the *Global* to find the  $\widehat{k\text{-core}}$  containing vertex  $q$ . Cui et al. [25] 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* [24] and *k-truss* [27] which searches 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.

### Keyword search over graphs

The keyword search over graphs [28–31] have attracted significant attention in recent years since it provides valuable information to users without the knowledge of underlying entities, schema, or access mechanism. Many advanced keyword search techniques [28–32] are developed to search information over such large complex graphs. We use the concept of keyword search over graphs to generate different communities in the graph. We 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 section defines the problem, and describes the key definitions and major aspects of the proposed approach.

In this paper, we consider some important measures described by *structure-based*, *attribute-based*, *online* community detection, and *keyword search methods* to design a novel method for more accurate community detection over large attributed graphs.

## Methods

The proposed approach for community detection using Keyword Attribute Search is explained in detail in this section. Before explaining the proposed approach, it is necessary to define the problem, provide some corresponding definitions, and lemmas to support the proposed approach.

### Data model

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  $\Lambda$  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  $\Lambda = \{attr_1, attr_2, attr_3 \dots attr_n\}$ , and the set of attribute values associated with each vertex  $v_i$  is denoted as  $attr_{ji} = \{attr_{1i}, attr_{2i}, \dots attr_{ji}\}$ .

In this paper, an undirected attributed graph  $G = (V, E, \Lambda)$  is considered. Let  $n$  and  $m$  be the size of  $V$  and  $E$ , respectively. Table 1 represents the meaning of all the symbols used in the paper.

### 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 vertices in a community is  $k$  or more [23, 25, 26, 33, 34], also known as *k-core*, which is an important condition for structure cohesiveness of  $G$ .

### Definition 1 (*k-core*)

Given an integer  $k$  ( $k \geq 0$ ), the *k-core* [23, 33, 35] 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* [16, 23, 33] 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.

### Definition 2 (*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* [36]  $Sim(v_1, v_2)$  and defined as follows:

$$Sim(v_1, v_2) = \frac{|(X_1 \cap X_2)|}{|(X_1 \cup X_2)|} \quad (1)$$

**Table 1** Symbols and meanings

Symbol	Meaning
$G(V, E)$	An undirected graph with set of vertices $V$ and set of edges $E$
$\lambda$	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 $q2$
$\theta_c$	Minimum threshold constant for similarity between $q1$ and $q2$
$\theta_w$	Minimum threshold constant for average weight of the attributes shared between $q1$ and $q2$
$\theta_d$	Maximum threshold constant for geodesic distance between $q1$ and $q2$
$W_v$	Minimum threshold constant for Node-weight on attribute type-value pairs
$W_e$	Minimum threshold constant for Edge-weight on attribute type-value pairs
$GD(q1, q2)$	Shortest path length or geodesic distance between $q1$ and $q2$



The *Jaccard Similarity Index* [36] 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.

**Definition 3 (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* [37] 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* [37] can be given as follows:

$$GD(v_1, v_2) = \min(\forall_i \forall_j d(v_i, v_j)) \quad (2)$$

**Problem definition**

The problem is defined as follows:

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:

**Problem 1 (attributed community detection)**

Given a graph  $G(V, E, \lambda)$ , positive integer constants  $k$ ,  $d$ ,  $\theta_c$ ,  $\theta_w$ ,  $W_v$ , and  $W_e$ , a vertex  $v \in V$ , and a set of attributes  $attr_i \subset X(v_i \in V)$ , 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. *Dense structure*  $\forall v \in G_v$  following properties should be satisfied;
  - a. all the nodes  $v \in V$  in a subgraph have degree greater than or equal to the  $K\_core$  value,  $deg_{G_v}(v) \geq k$ ,  $k$  is a minimum threshold for  $k$ -core value.
  - b. if two nodes  $(v_i, v_j) \in V$  are connected, then the distance between them should be less than or equal to the maximum threshold constant  $d$  of the *shortest path length* or *geodesic distance*,  $GD(v_i, v_j) \leq d$ .
3. *Attribute sharing*  $\forall v \in G_v$  and  $\forall X \in \lambda$  following properties should be satisfied;
  - a. each node  $v \in V$  should have a set of influential attributes such that,  $P(attr_i \in v) \geq W_v$  and  $P(attr_{ij} \in v_j, attr_{ik} \in v_k) \geq W_e$ , where  $P(attr_i \in v)$  is



the probability of attribute  $attr_i$  on all vertices,  $P(attr_{ij} \in v_j, attr_{ik} \in v_k)$  is the probability of attribute  $attr_i$  shared between vertices  $v_j$  and  $v_k$ ,  $W_v$  and  $W_e$  are minimum threshold constants for probability of attributes on nodes and edges, respectively.

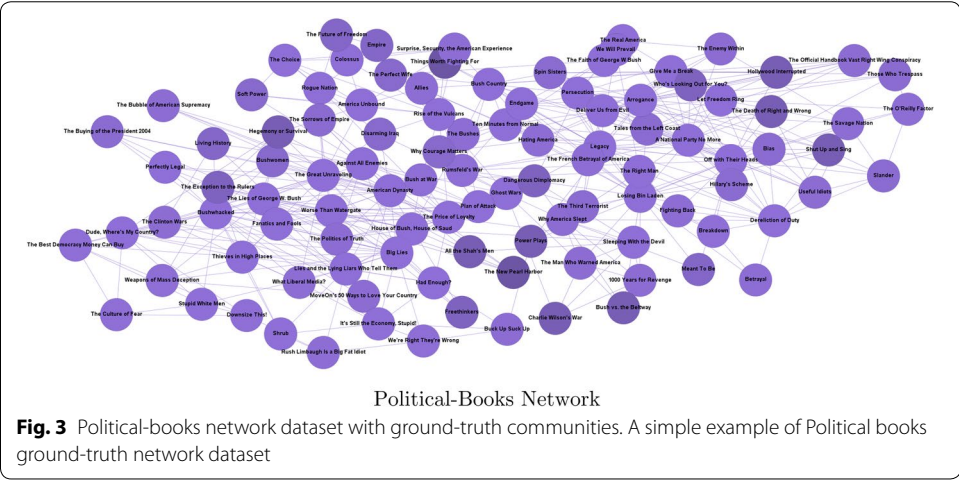
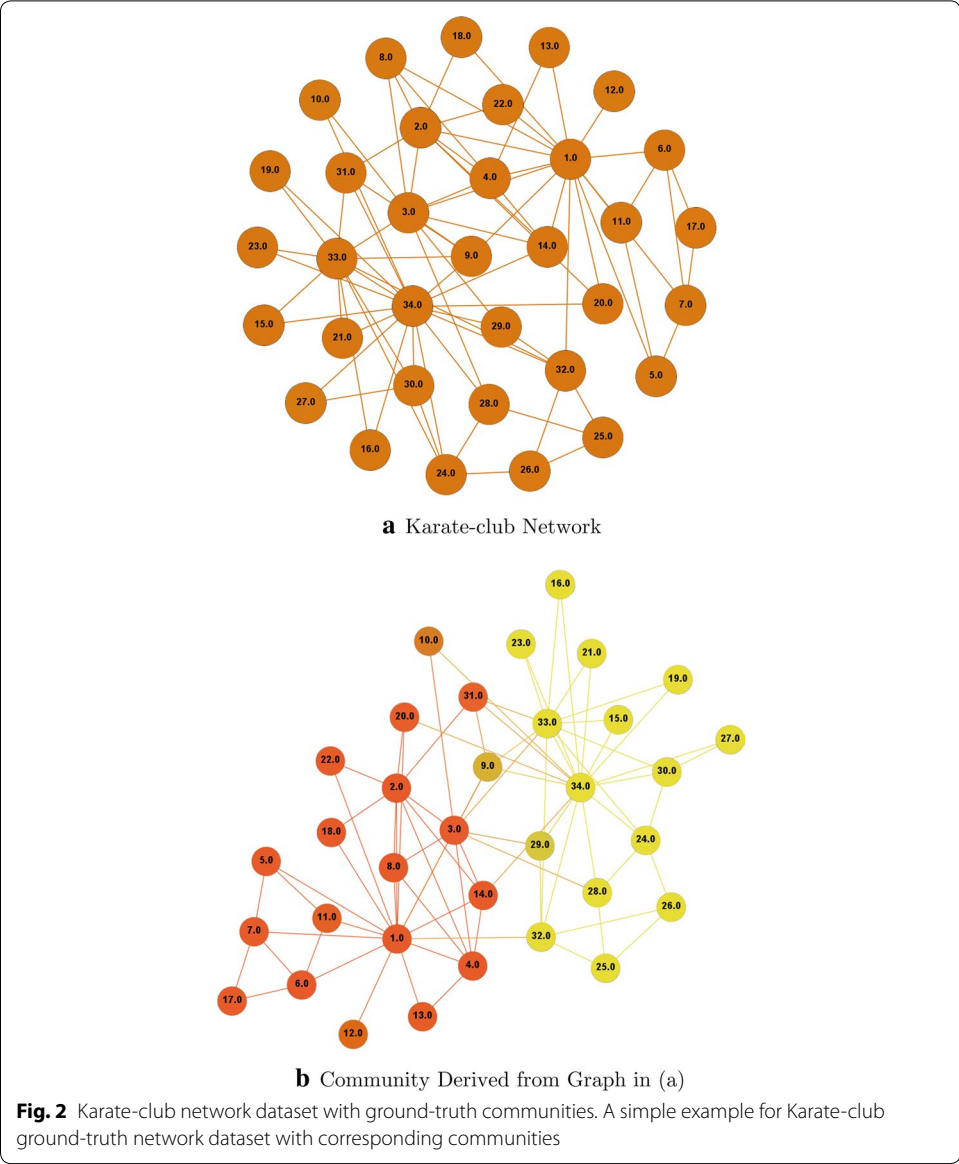
- b. if two nodes  $v_i, v_j \in V$  are connected, then the two nodes should share the maximal similar influential attributes, and the average weight of common influential attributes should be greater than or equal to *Average Edge weight*,  $W_e(v_i, v_j) \geq \theta_w$ .
- c. if two nodes  $v_i, v_j \in V$  are connected, then the two nodes should share the maximal similar attributes, and the similarity between these attributes should be greater than or equal to *Jaccard Similarity Index*,  $Sim(v_i, v_j) \geq \theta_c$ .

The above properties like *k-core* and *shortest path length* make sure the community structure is structurally cohesive, and the property of *Node-weight*, *Edge-weight*, and *Jaccard Similarity Index* makes sure the community structure follows the attribute cohesiveness. Both of these requirements are critical for an accurate community detection. Figure 2a, b shows the attributed graph with the corresponding communities based on the problem definition. Figure 3 shows another example of ground truth network dataset having certain number of communities. Our 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 between all the vertices in a community.

### Community detection using keyword search

Although the following method [16] is applicable to attributed graphs and generate communities which satisfies the structure and attribute cohesiveness, but it does not consider the attribute type-value pair for the attribute cohesiveness. Since there can be multiple values for one attribute type, it is necessary to identify the influential attribute type-value pairs, and derive the communities based on these influential attributes. Thus the proposed method is significant to identify the important attribute type-value pairs, and derive more accurate communities for these attributes. The proposed method highlights the importance of finding the influential attributes. The *Node-weighted* and *Edge-weighted* attributes are useful to find the probability of edge creation between two nodes based on the connectivity as well as attribute similarity. The existing method [16] does not find such influential attributes or probability of edge creation, hence the proposed method is useful to find accurate communities. The vertices which have strong connectivity and common *keyword attributes* can be grouped together in the same community. An algorithmic framework is designed to classify *keyword attributes* on all vertices based on the similarity between a set of attributes on two vertices, and the connectivity between two vertices. These classes 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. 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. 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.
4. The factor *Jaccard Index*, and average weight of the *Edge-weighted* attributes between a pair of vertices, is used to measure the similarity of attributes, while *k-core* and *geodesic distance* are used to measure the connectivity of the vertices sharing similar attributes.
5. While classifying each vertex in the graph based on the attributes, a new 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.
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 section, an algorithmic framework is defined and explained in detail for each major step.

### Keyword search based algorithm

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, we use two network datasets *Karate-club* and *Political Book* in Figs. 2a and 3 respectively, which have ground-truth communities associated with them. We 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 2.

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

#### 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 each node  $v \in V$  contains a set

**Table 2 Node attributes of sample graph in Fig. 2a**

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'}

**Table 3 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

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. 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:

$$P(attr_i) = \frac{O(attr_i)}{N} = \frac{\sum_{j=1}^N attr_i}{N} \quad (3)$$

For the example network shown in Fig. 2a, if we assume the maximum threshold for Node-weight  $W_v = 20.0\%$ , then the corresponding important attributes are listed in Table 3.

### 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, we 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:

$$P(attr_{ij}, attr_{ik}) = \frac{\sum_{i=1}^N \sum_{j=1}^N O(attr_{ij}, attr_{ik})}{N} \quad (4)$$

We can determine the *Edge-weight* from the above important attribute type-value pairs listed in Table 3, and the all the edges in *Karate-club* network graph sharing these attribute type-value pairs. If we assume the maximum threshold value for *Edge-weight*  $W_e = 10.0\%$ , then, corresponding influential *Edge-weight* attributes are listed in Table 4.

#### 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 a set of attributes on each node. Thus, the *Keyword Attribute Signature* is a compact representation of the attribute values on each node. The *unique Id* associated with each attribute type-value pair shown in Table 4 is used to create the *Keyword Attribute Signature* on each node.

#### Attribute Index Structure

In the next step, we create an *Attribute Index Structure* for the *Keyword Attribute Signature* assigned to all nodes. Given a large attributed graph  $G = (V, E)$ , the task is to determine different classes of attributes from all the *Keyword Attribute*

**Table 4** 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

*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 each node with other nodes. If the two nodes have degree greater than or equal to  $k_{core}$ , then they are considered for the classification. If the two nodes have degree greater than or equal to  $k_{core}$ , and 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  $I = \{C_1: \{X_1, V_1, count(C_1)\}, C_2: \{X_2, V_2, count(C_2)\}, \dots, C_n: \{X_n, V_n, count(C_n)\}\}$ , 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 pseudo-code given in *Algorithm 1* is used to create the *Attribute Index Structure*:

Table 5 represents the *Keyword Attribute Index* structure created for the *Karate-club* network dataset.

#### 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* is to create 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 its relationship with other vertices, this *Keyword Attribute Signature* based search leads to different nodes which share the same attribute information, which eventually helps derive 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  $\theta_c$  to depict a maximum threshold of similarity between two sets of *Keyword Attribute Signatures* 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

**Table 5** 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

---

**Algorithm 1** Keyword Attribute Index Structure
 

---

**Input:** $G = (V, E)$ ; $V = (V_1, V_2, \dots, V_n)$  : Set of vertices in the graph; $X = (X_1, X_2, \dots, X_m)$  : Set of influential attributes derived from Node-weight & Edge-weight; $X_i = \{attr_1(V_i), attr_2(V_i), \dots, attr_j(V_i)\}$  : Set of attributes associated with each vertex in the graph;**Output:**
 $I = \{C_1 : \{X_1, V_1, count(C_1)\}, C_2 : \{X_2, V_2, count(C_2)\}, \dots, C_n : \{X_m, V_n, count(C_n)\}\}$  :  
 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  $C_n$  ;

```

1: Initialize  $i, j, k, count\_node = \emptyset$ 
2: for Vertices  $v_i \in V$  do
3:   for Class  $C_j \in I$  do
4:      $k\_core \leftarrow$  get  $K\_core(G, v_i)$  value;
5:      $attr_i \leftarrow$  attribute list  $X(v_i)$  of  $V_i$ ;
6:      $attr_j \leftarrow$  attribute list  $X_j \in C_j$  of class  $C_j$ ;
7:      $Sim(attr_i, attr_j) \leftarrow |attr_i \cap attr_j| / |attr_i \cup attr_j|$ ;
8:      $Avg\_weight(attr_i, attr_j) \leftarrow |W_e(attr_i \cup attr_j)| / |attr_i \cup attr_j|$ ;
9:
10:    if degree  $deg_{v_i} \geq k\_core$  then
11:      if  $Sim(attr_i, attr_j) \geq \theta_c$  then
12:        if  $Avg\_weight(attr_i, attr_j) \geq \theta_w$  then
13:           $attr_i \cup attr_j \leftarrow X_j$  {for  $X(v_i), X_j$ }
14:           $V_j \leftarrow v_i$  classify  $v_i$  in  $C_j$ ;
15:           $count(v_i)++$ ; {Number of nodes for each class of attribute}
16:          Merge attributes of  $v_i, X(v_i) \leftarrow X(v_i) \cup C_j$  in class  $C_j$ ;
17:        end if
18:      else
19:         $C_k \leftarrow k+1$  {Create a separate class for attribute  $X(v_i)$ };
20:         $V_k \leftarrow v_i$  ;
21:         $X_k \leftarrow X(v_i)$ ;
22:         $count(C_k) \leftarrow 1$ ;
23:        Set  $I = \{C_k : \{(X_k, V_k, count(C_k))\}\}$  ;
24:        Set  $X(v_i) \leftarrow X(v_i) \cup C_k$  where  $X(v_i) \in X$ ;
25:      end if
26:    end if
27:  end for
28: end for
29: return  $I$ 

```

---

graph. We 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 we 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 sets 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  $\theta_c$ , 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  $\theta_c$ , 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*. We also 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.

**Lemma 1** Given  $G = (V, E)$  and set of attributes  $X$ , if  $X_q = \{X_{q1}, X_{q2}, \dots, X_{qn}\}$  and  $\hat{X}_q = \{\hat{X}_{q1}, \hat{X}_{q2}, \dots, \hat{X}_{qn}\}$  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, we use a proof by contradiction. Let  $X_q = \{X_{q1}, X_{q2}, \dots, X_{qn}\}$  such that  $X_q \in C_q$ , and  $\hat{X}_q = \{\hat{X}_{q1}, \hat{X}_{q2}, \dots, \hat{X}_{qn}\}$  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.  $\square$

### 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  $X_i(v_i)$  on each node  $v_i \in V$  is updated to include the class  $C_i$ . 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  $Q = \{q_1, q_2, \dots, q_i\}$ , with  $i$  keywords in the query. For a given query  $Q$  which has keywords or class of attributes, we 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 naive keyword search starts at the node  $V_i$  derived from  $I$  and finds the *Breadth First Search (BFS)* path with the node  $V_i$  as the root node. It continues to search the nodes with same class  $C_i$  of attributes as node  $v_i$  along the *BFS* path of the node  $V_i$ . The *Breadth First Search (BFS)* path makes sure that all the nodes are covered in the network, and every node is compared with other nodes to find the nodes having similar class label. 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. The pseudo-code for deriving the  $k\_core$  value for all the nodes, and the retrieval of a *Breadth First Search Tree (BFS)* for root node  $V_i \in I$  is given by *Algorithm 4* and *Algorithm 5*, respectively.

---

**Algorithm 2** Personalized Community Detection
 

---

**Input:**

$G = (V, E, X)$ ;  
 $V = (V_1, V_2, \dots, V_n)$  : Set of vertices in the graph;  
 $X = (X_1, X_2, \dots, X_m)$  : Set of attributes in the graph;  
 $X_i = \{attr_1(V_i), attr_2(V_i), \dots, attr_j(V_i)\}$  : set of attributes associated with each vertex in the graph;  
 $I = \{C_1 : \{X_1, V_1, count(C_1)\}, C_2 : \{X_2, V_2, count(C_2)\}, \dots, C_n : \{X_m, V_n, count(C_n)\}\}$  : 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  $C_n$  ;  
 $Q = \{q_1, q_2, \dots, q_n\}$  : Query having a list of keywords or class information;

**Output:**

$Comm(C_i)$ : Communities with the nodes sharing the keyword information in the query  $Q$ ;

```

1: Initialize  $i, j, node = \emptyset$ ;
2: for Query  $q_i \in Q$  do
3:   Class  $C_i \leftarrow q_i$ ;
4:   if Class  $C_i \in I$  then
5:      $node_i = V_i$ ; {Attribute Index Structure  $I = \{C_i : \{X_i, V_i, count_i(C_i)\}\}$ }
6:      $k\_core \leftarrow \text{get } K\_core(G, v_i) \text{ value}$ ;
7:     if degree  $deg_{node_i} \geq k\_core$  then
8:        $node\_count_i \leftarrow count_i(node_i)$ ;
9:        $bfs\_list(node_i) \leftarrow BFS\_Tree(G, v_i)$ ; {Get BFS list of node  $v_i$ }
10:      while  $count \leq node\_count_i$  do
11:        for Node  $v_j \in bfs\_list(node_i)$  do
12:          if degree  $deg_{v_j} \geq k\_core \wedge GD(v_i, v_j) \leq \theta d$  then
13:             $Class_j \leftarrow attr_j(v_j)$  {Attribute list  $attr_j(v_j)$  of  $v_j \in \text{Class } C_i$ }
14:            if Query  $q_i \in Class_j$  then
15:               $Comm_i(q_i) \leftarrow v_j$ ; {Mark  $v_j$  for community }
16:            end if
17:          end if
18:        end for
19:      end while
20:    end if
21:  end for
22: end for
23: return  $Comm(Q)$ ;
  
```

---

**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. 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 all the required information about different attributes  $X_i(v_i)$  on node  $v_i \in V$  and class  $C_i$  of these attributes. 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 pseudo-code given in *Algorithm 3* explains the generalized community detection process in detail:

---

**Algorithm 3** Generalized Community Detection
 

---

**Input:** $G = (V, E, X)$ ; $V = (V_1, V_2, \dots, V_n)$  : Set of vertices in the graph; $X = (X_1, X_2, \dots, X_m)$  : Set of attributes in the graph; $\text{attr}(V_i) = \{\text{attr}_1(V_i), \text{attr}_2(V_i), \dots, \text{attr}_j(V_i)\}$  : set of attributes associated with each vertex in the graph; $I = \{C_1 : \{X_1, V_1, \text{count}(C_1)\}, C_2 : \{X_2, V_2, \text{count}(C_2)\}, \dots, C_n : \{X_m, V_n, \text{count}(C_n)\}\}$  : 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:** $\text{Comm}(C_i)$ : Communities having the nodes  $v_i$  belong to  $C_i \in I$ ;

```

1: Initialize  $i, j, \text{node} = \emptyset$ ;
2: for Class  $C_i \in I$  do
3:   Class  $C_i \leftarrow$  query  $q_i$ ;
4:    $\text{node}_i = V_i$ ;  $\{I = \{C_i : \{X_i, V_i, \text{count}_i(C_i)\}\}\}$ 
5:    $k\_core \leftarrow$  get  $K\_core(G, v_i)$  value;
6:   if degree  $\text{deg}_{\text{node}_i} \geq k\_core$  then
7:      $\text{node\_count}_i \leftarrow \text{count}_i(\text{node}_i)$ ;
8:      $\text{bfs\_list}(\text{node}_i) \leftarrow \text{BFS\_Tree}(G, v_i)$ ;  $\{\text{Get BFS list of node } v_i\}$ 
9:     while  $\text{count} \leq \text{node\_count}_i$  do
10:      for node  $v_j \in \text{bfs\_list}(\text{node}_i)$  do
11:        if degree  $\text{deg}_{v_j} \geq k\_core \wedge GD(v_i, v_j) \leq \theta d$  then
12:           $\text{Class}_j \leftarrow \text{attr}_j(v_j) \{ \text{attr}_j(v_j) = C_i \}$ 
13:          if  $q_i \in \text{Class}_j$  then
14:             $\text{Comm}_i(q_i) \leftarrow v_j$ ;  $\{\text{Mark } v_j \text{ for community } \}$ 
15:          end if
16:        end if
17:      end for
18:    end while
19:  end if
20: end for
21: return  $\text{Comm}(Q)$ ;

```

---

The personalized community detection can be used to derive a particular community from a graph in an online manner. However, the generalized

---

**Algorithm 4** Find  $K\_core$  for All Vertices
 

---

**Input:** $G = (V, E, X)$ ;**Output:** $K\_core$  : List of  $K\_core$  values for all the vertices;

```

1: Compute the degree of all the vertices  $\text{deg}_{v_i}$ ;
2: Arrange the set  $V$  of vertices in the increasing order of their degrees;
3: for  $v_i \in V$  do
4:    $K\_core[v_i] = \text{deg}(v_i)$ 
5:   for  $v_j \in \text{Neighbors}(v_i)$  do
6:     if  $\text{deg}(v_j) > \text{deg}(v_i)$  then
7:        $\text{deg}(v_j) = \text{deg}(v_j) - 1$ 
8:       Arrange  $V$  in increasing order accordingly
9:     end if
10:   end for
11: end for

```

---

---

**Algorithm 5** Breadth First Search Tree (BFS\_Tree)
 

---

**Input:** $G = (V, E, X)$ ; $Q$  : Queue to add the visited vertices in the path;**Output:** $bfs\_list$  : List of vertices in the BFS path starting at vertex  $v_i \in I$ ;

```

1: Add  $v_i \in V$  to  $Q$  and mark it as visited;
2: while  $v_i \in Q$  do
3:    $w = QueueFront(Q)$ ;
4:   Remove  $Q$ ;
5:   for each visited  $v_j$  adjacent to  $w$  do
6:     mark  $v_j$  as visited;
7:     Add  $v_j$  to  $Q$   $\{Add(Q, v_j)\}$ ;
8:   end for
9: end while

```

---

community detection can be used to derive all the communities from a graph without any query in an online manner.

**Illustration**

The proposed method is explained by illustrating a simple example. We consider a small graph with corresponding node attributes and edges as displayed in Fig. 4a. If we consider the *Node-weight* and *Edge-weight* as 50% each, we get the influential attributes as  $\{1:School = UNLV\}$ ,  $\{2:City = Las Vegas\}$ ,  $\{3:Role = Student\}$ , and  $\{4:Role = Professor\}$ . We create the compact signature from the influential attributes as shown in Fig. 4a. If we consider the Jaccard Similarity Index  $JCD$  as 40%,  $k\_core$  and geodesic distance  $GD$  as 1 each, then we can apply the method to get the attribute index structure shown in Fig. 4b. Each class identified in the attribute index structure is assigned as an attribute to each node in the graph. The generalized community detection algorithm gives all the communities for the sample graph as shown in Fig. 4c.

For example, if we look at the *Karate-Club* network graph shown in Fig. 2a with different attributes on each node and the *Attribute Index Structure* shown in Table 5, we derive the community of nodes that belong to the same class of attributes as shown in Fig. 5a. Similarly, based on the previously defined measures, we derive the communities for the *Political-Books* network dataset shown in Fig. 3. All the communities are displayed in Fig. 5b.

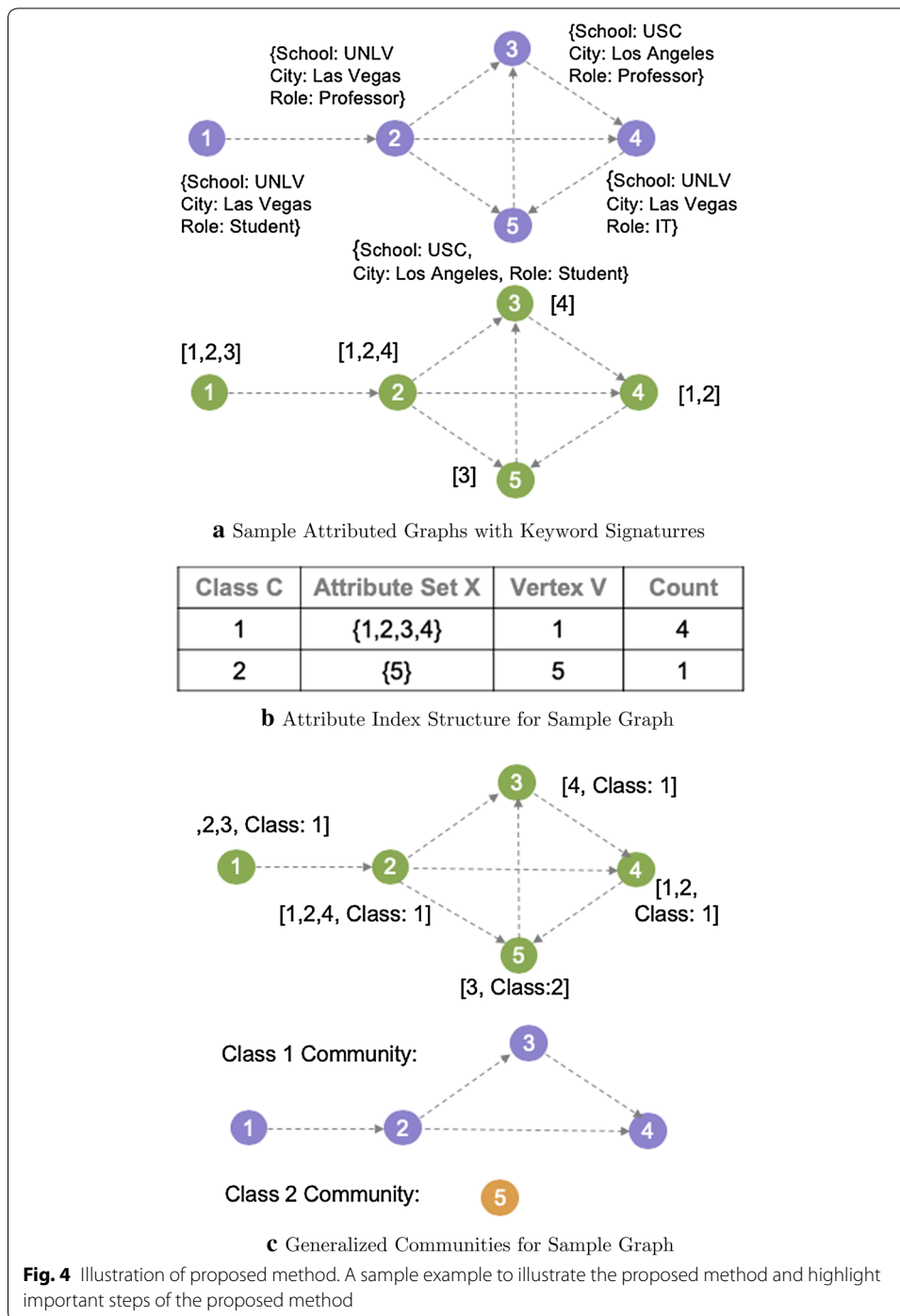
**Algorithm analysis**

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.

**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:

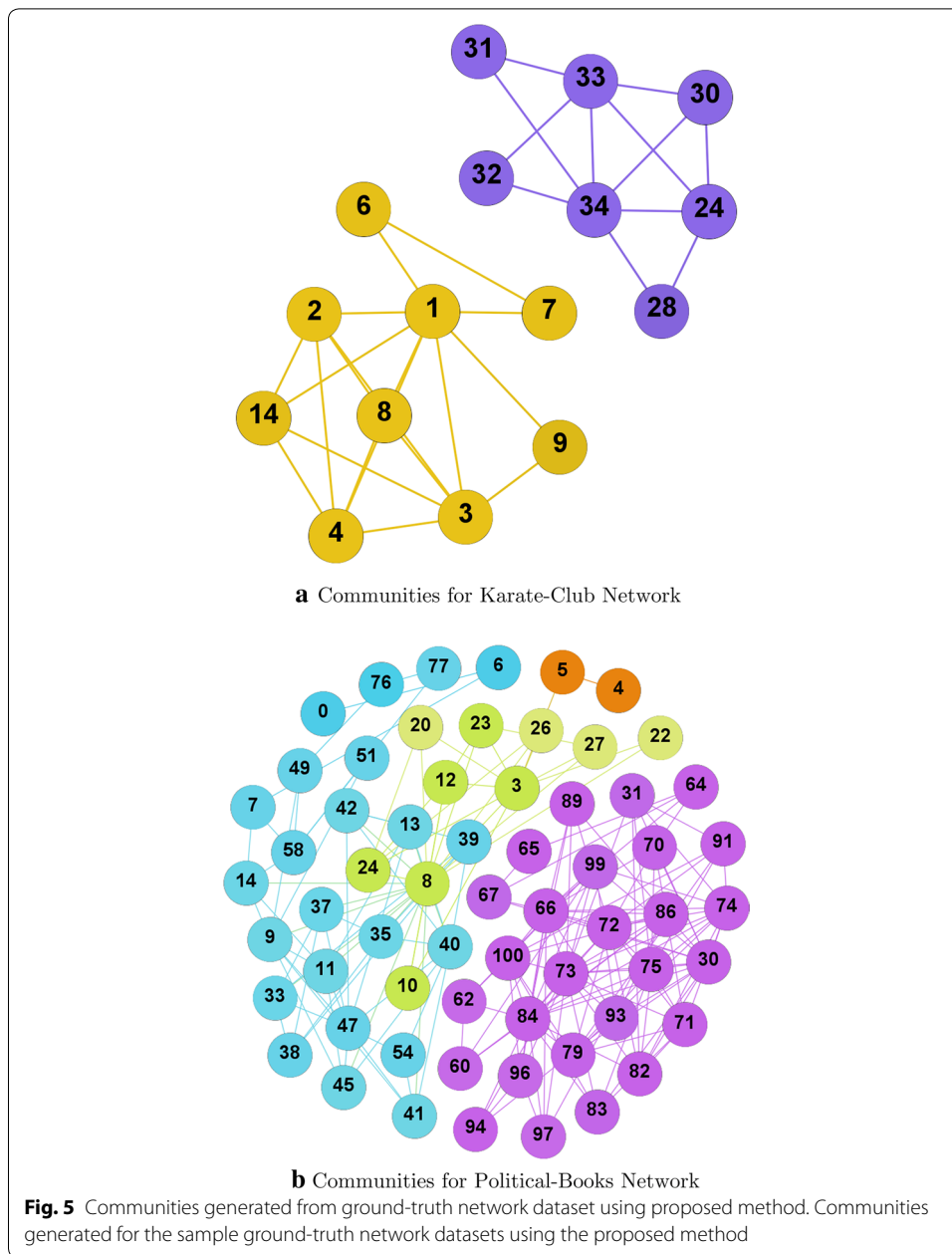
$$[H]O(attr_i(v_j)) = \sum_{i=1}^c \sum_{j=1}^n O(attr_i(v_j)) = O(c * n) = O(n) \quad (5)$$



$$HO(attr_i(e_{jk})) = \sum_{i=1}^c \sum_{j=1}^m \sum_{k=1}^m O(attr_i(e_{jk})) = O(c * m) = O(m) \quad (6)$$

### 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:

$$O(K\_core) = \min(deg_{K\_core}(G, V)) = O(n + m) \quad (7)$$

$$O(K\_core) = O(n) = O(|V| + |E|) \quad (8)$$

### 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 we 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 criteria prune the search space over a large network graph. There is another threshold known as *Jaccard Similarity Index*  $\theta_c$ , which verify the similarity between two *Keyword Attribute Signatures*. If the similarity between two sets is greater than or equal to  $\theta_c$ , 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 we 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 we 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:

$$O(I) = \sum_{i=1}^n \sum_{j=1}^n O(n * c) = O(n) = O(|V|) \quad (9)$$

### 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  $V_i$  ( $v_i \in BFS\_TREE(V_i)$ ). 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:

$$O(Comm_i(I)) = \max(O(|V|), O(|E|), O(|V| + |E|), O(|V| * |E|), O(|V|^2)) \quad (10)$$

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:

$$O(Comm_i(I)) = O(|V| * |E|) \quad (11)$$

### Experimental results

Different experiments are performed to verify the accuracy of the proposed method. We 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 @ 3.70 GHz* on an Windows 10 operating system. *Python 2.7* is used to implement the algorithms with *networkx* package for graph related operations.

#### Experimental setup

We divide our experiments into three parts. The first part of the experiment compares the proposed approach with the existing methods. We use the network datasets like *Karate-Club*, *American Football*, *Political-Books*, *Dolphin-network*, *email-EU-core*, *DBLP*, and *Amazon* [38] 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 10,000. Since the community structure would contain the dense subgraphs, we include the variation in a number of edges by creating synthetic graphs having 2000, 5000, and 10,000 nodes respectively, and the probability of edge creation 0.50, 0.40, and 0.30 respectively. We 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. We use real datasets like *YouTube video crawl* [39], *Twitter User Profiles* [40], *Skytrax Airline Reviews*, *Terrorist Data* [41], *Caesars Entertainment anonymous dataset*, and *Facebook* for the second and third part of the experiment. We randomly assign attributes to the datasets having only edge lists, and randomly create edges with a certain probability of edge creation for the datasets having node list only. The *networkx* package of *Python 2.7* is used for all the graph related operation in the experiments.

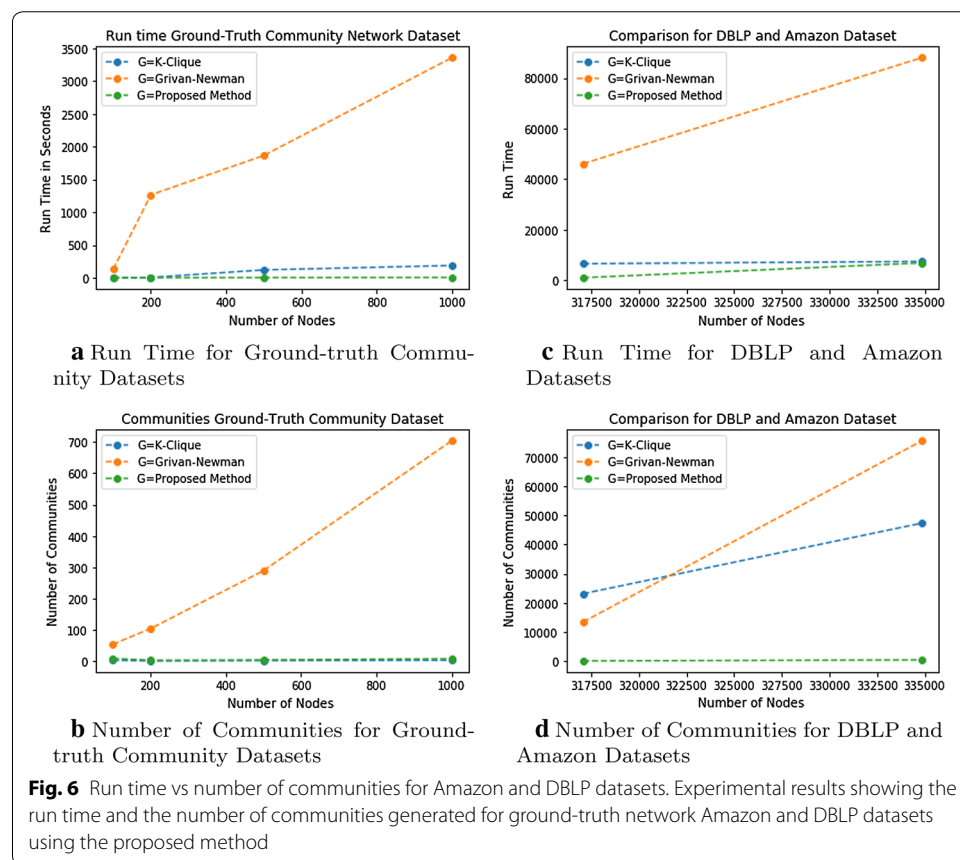
#### Discussion

Table 6 shows the details about the runtime and a number of communities in each comparison experiment. Figure 6 displays the result for comparison of runtime and

**Table 6** Comparison with existing methods

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	47,307	46,080	13,477	936	128
Amazon	7380	23,134	88,080	75,499	6791	468

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, C3 #Communities for proposed method



the number of communities generated for existing methods and the proposed method, respectively. Tables 7 and 8 show the statistics for the small as well as large graph datasets, respectively. The tables show statistics about the number of nodes, the number of edges, the probability of edge creation, and the number of attributes on each node. Tables 9 and 10 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

**Table 7 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	11,793	$P = 0.70$	26
Twitter User Profile	500	74,625	$P = 0.60$	26
Twitter User Profile	1313	517,068	$P = 0.60$	26
Skytrax Airline Reviews	100	3514	$P = 0.70$	7
Skytrax Airline Reviews	200	13,882	$P = 0.70$	7
Skytrax Airline Reviews	500	74,907	$P = 0.60$	7
Skytrax Airline Reviews	1005	300,202	$P = 0.60$	7
Terrorist Data	100	3457	$P = 0.70$	7
Terrorist Data	200	13,969	$P = 0.70$	7
Terrorist Data	500	74,753	$P = 0.60$	7
Terrorist Data	1000	299,420	$P = 0.60$	7
Caesars Entertainment	100	3439	$P = 0.70$	6
Caesars Entertainment	200	13,929	$P = 0.70$	6
Caesars Entertainment	500	74,835	$P = 0.60$	6
Caesars Entertainment	1000	299,953	$P = 0.60$	6

**Table 8 Large graph datasets**

Graph	Nodes V	Edges E	Prob. of edge	Attributes on each node
Facebook	2000	1,000,025	$P = 0.50$	7
Facebook	5000	4,997,567	$P = 0.40$	7
Facebook	10,000	14,998,579	$P = 0.30$	7
Youtube video crawl	2000	998,910	$P = 0.50$	9
Youtube video crawl	5000	4,997,518	$P = 0.40$	9
Youtube video crawl	10,000	14,996,737	$P = 0.30$	9
Terrorist data	2000	1,000,470	$P = 0.50$	7
Terrorist data	5000	4,998,900	$P = 0.40$	7
Terrorist data	10,000	15,004,225	$P = 0.30$	7
Caesars entertainment	2000	998,937	$P = 0.50$	6
Caesars entertainment	5000	4,887,265	$P = 0.40$	6
Caesars entertainment	10,000	14,997,657	$P = 0.30$	6

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 Figs. 7 and 8, respectively.

### Case study

We create a small case study based on a real time graph dataset provided by *Caesars Entertainment Corporation* located at *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

**Table 9 Experimental results on small graphs**

G	V	E	Sim (%)	L	C	T
Twitter user profile	100	3426	70	3	2	1
Twitter user profile	200	11,860	70	3	2	3
Twitter user profile	500	74,625	70	3	4	10
Twitter user profile	1313	517,072	70	5	3	25
Skytrax airline reviews	100	3514	50	3	3	1
Skytrax airline reviews	200	13,882	50	3	3	2
Skytrax airline reviews	500	74,907	50	3	4	6
Skytrax airline reviews	1005	300,202	50	3	5	28
Terrorist data	100	3457	70	3	2	1
Terrorist data	200	13,969	70	3	2	2
Terrorist lata	500	74,753	70	3	3	10
Terrorist lata	1000	299,420	70	3	3	37
Caesars entertainment	100	3439	70	3	10	1
Caesars entertainment	200	13,929	70	3	12	2
Caesars entertainment	500	74,835	70	3	12	6
Caesars entertainment	1000	299,953	70	3	13	22

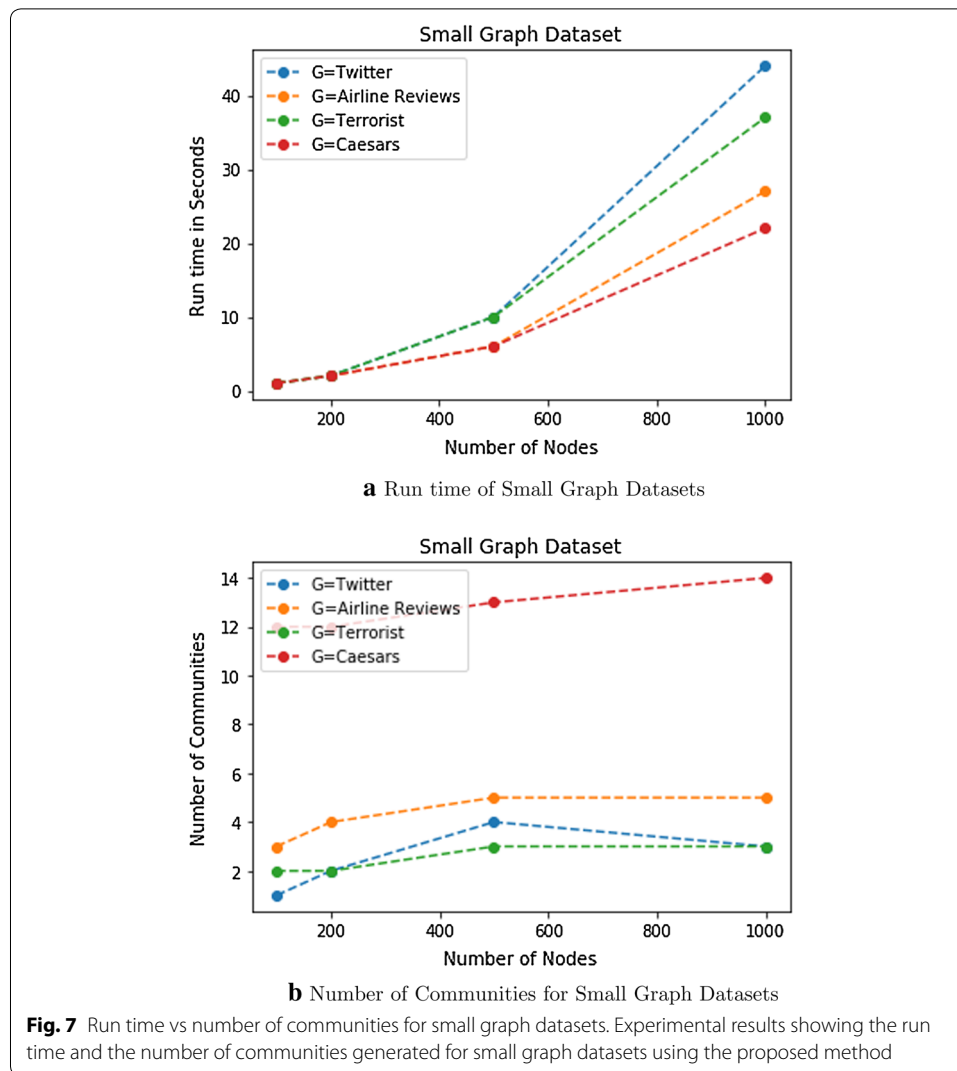
G Graph, V #Nodes in the graph, E edges in the graph, Sim #attribute similarity, L shortest path length, C #number of communities, T #time in seconds

**Table 10 Experimental results on large graphs**

G	V	E	Sim (%)	L	C	T
Facebook	2000	1,000,025	70	3	60	100
Facebook	5000	4,997,567	70	3	37	2577
Facebook	10,000	9,996,858	70	3	51	1701
Youtube video crawl	2000	998,910	70	3	5	62
Youtube video crawl	5000	4,997,518	70	3	5	817
Youtube video crawl	10,000	14,996,737	70	3	6	6785
Terrorist data	2000	1,000,470	70	3	9	100
Terrorist data	5000	4,998,900	70	5	10	2149
Terrorist data	10,000	15,004,225	70	5	10	25,000
Caesars entertainment	2000	998,937	70	3	7	109
Caesars entertainment	5000	4,997,265	70	3	10	1321
Caesars entertainment	10,000	14,997,657	70	3	10	7383

G graph, V #nodes in the graph, E edges in the graph, Sim #attribute similarity, L shortest path length, C #number of communities, T #time in seconds

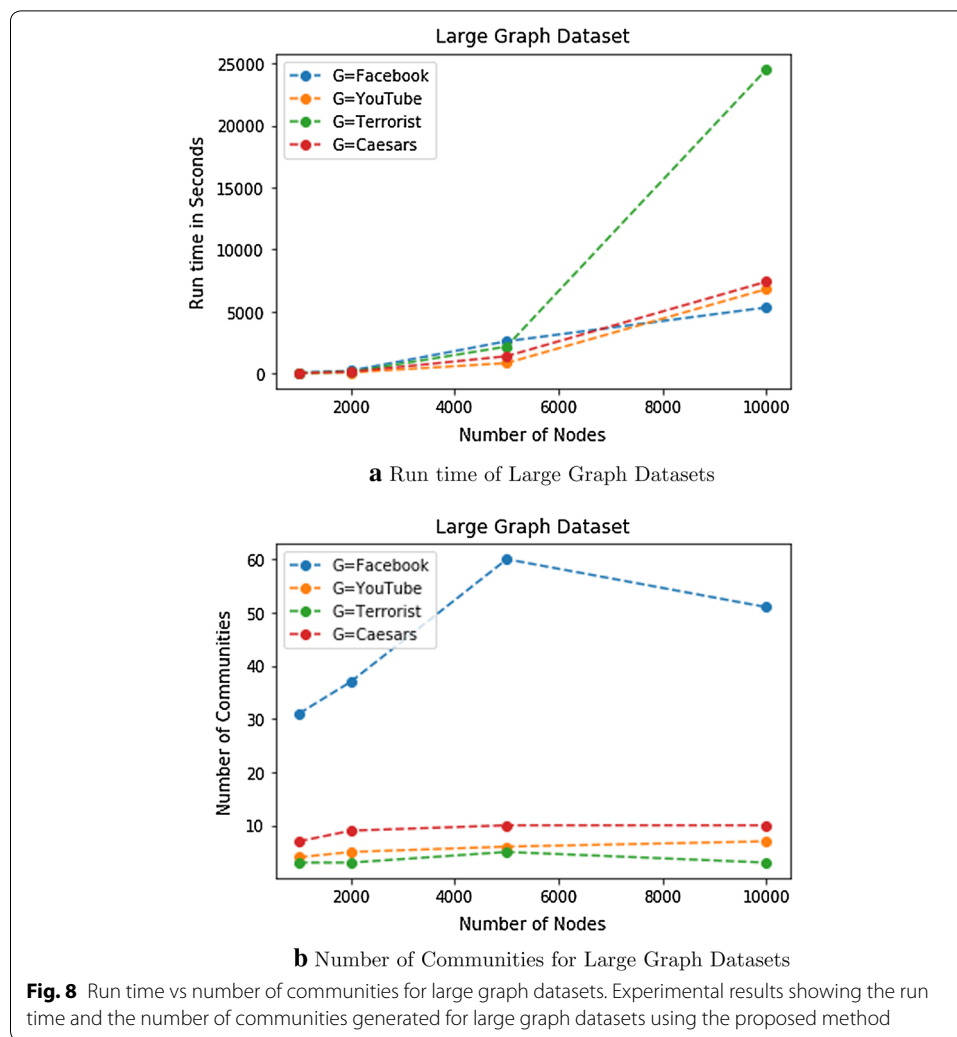
particular property. We create a graph  $G = (V, E)$  with  $|V| = 200$  nodes and  $|E| = 12000$  edges with 60% probability of edge creation in the graph at different nodes, this makes the graph structurally cohesive. We assign different attributes  $X_i(v_i)$  like, *Patron\_Id*, *Path\_From\_Property*, *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. Tables 11 and 12 represent all the attribute type-value pairs having probability greater than or equal to *Node-weight* and *Edge-weight*, respectively. Table 13 represents the *Attribute Index Structure* for graph dataset. Figs. 9, 10a–c, 11a–c show the original graph and communities generated through the proposed method, respectively.



Now, for the personalized community detection, we 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.

### Concluding remarks

A community structure derived from an attributed graph exhibits the structure and keyword attribute cohesiveness. The proposed *keyword search* based method derives communities with the structure and keyword attribute cohesiveness by constructing an *Attribute Index Structure*. The *Attribute Index Structure* correctly represents different



**Table 11 Node-weight for Caesars-WiFi dataset**

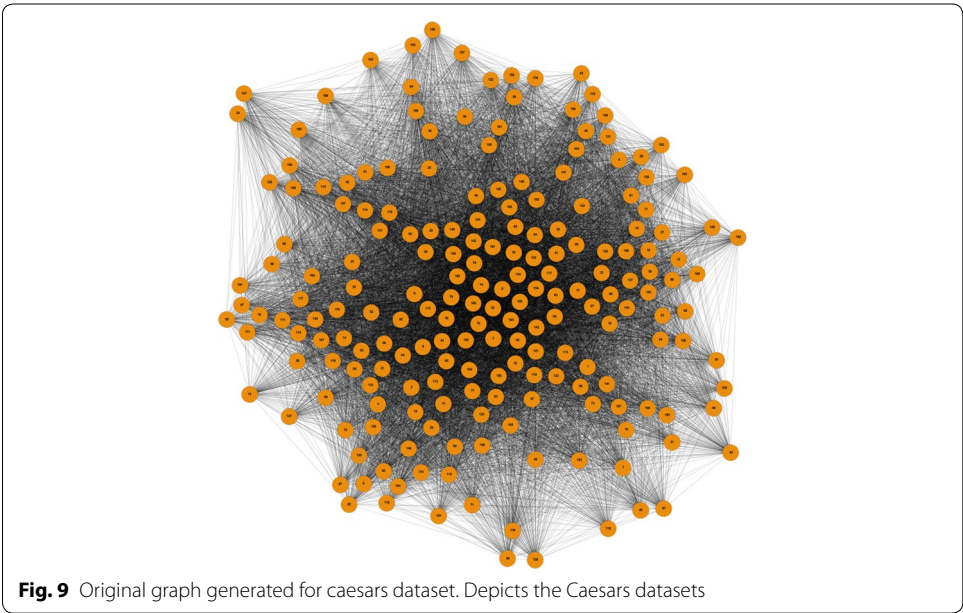
Id	Attribute type-value pair	Probability
1	Property_Region: Gaming	21.0
2	To_Property: Cromwell	20.50
3	From_Property: Cromwell	20.50
4	From_Property: Harrahs_LV	18.0
5	From_Property: Paris	17.0
6	To_Property: Caesars Palace	16.00
7	To_Property: Paris	15.00
8	From_Property: Caesars Palace	14.50
9	To_Property: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

**Table 12** Edge-weight for Caesars-WiFi dataset

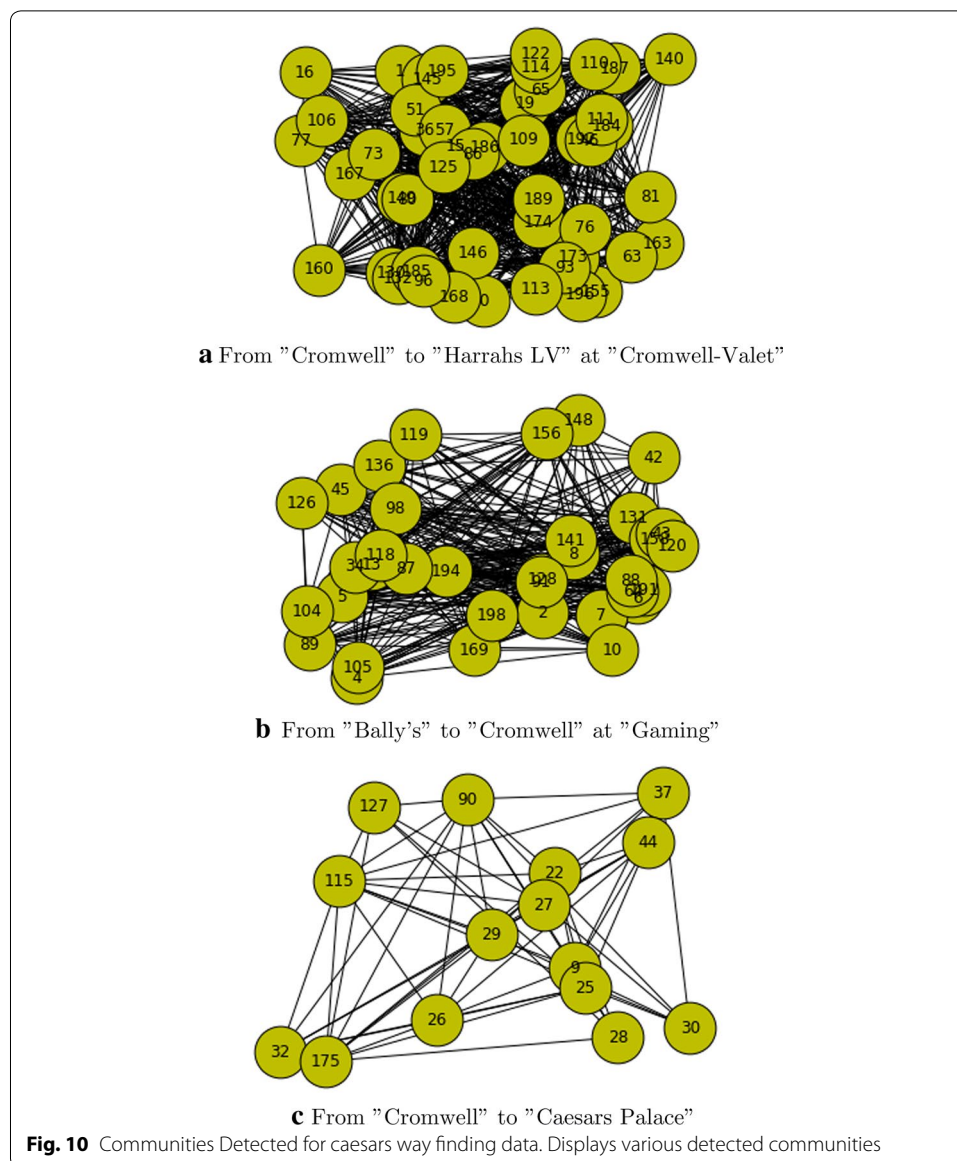
Id	Attribute type-value pair	Probability
1	Property_Region: Gaming	4.5
2	To_Property: Cromwell	4.0
3	From_Property: Cromwell	4.0
4	From_Property: Harrahs_LV	3.16
5	From_Property: Paris	3.0
6	To_Property: Caesars Palace	3.00
7	To_Property: Paris	2.26
8	From_Property: Caesars Palace	2.19
9	To_Property: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

**Table 13** Attribute Index Structure for Caesars-WiFi Dataset

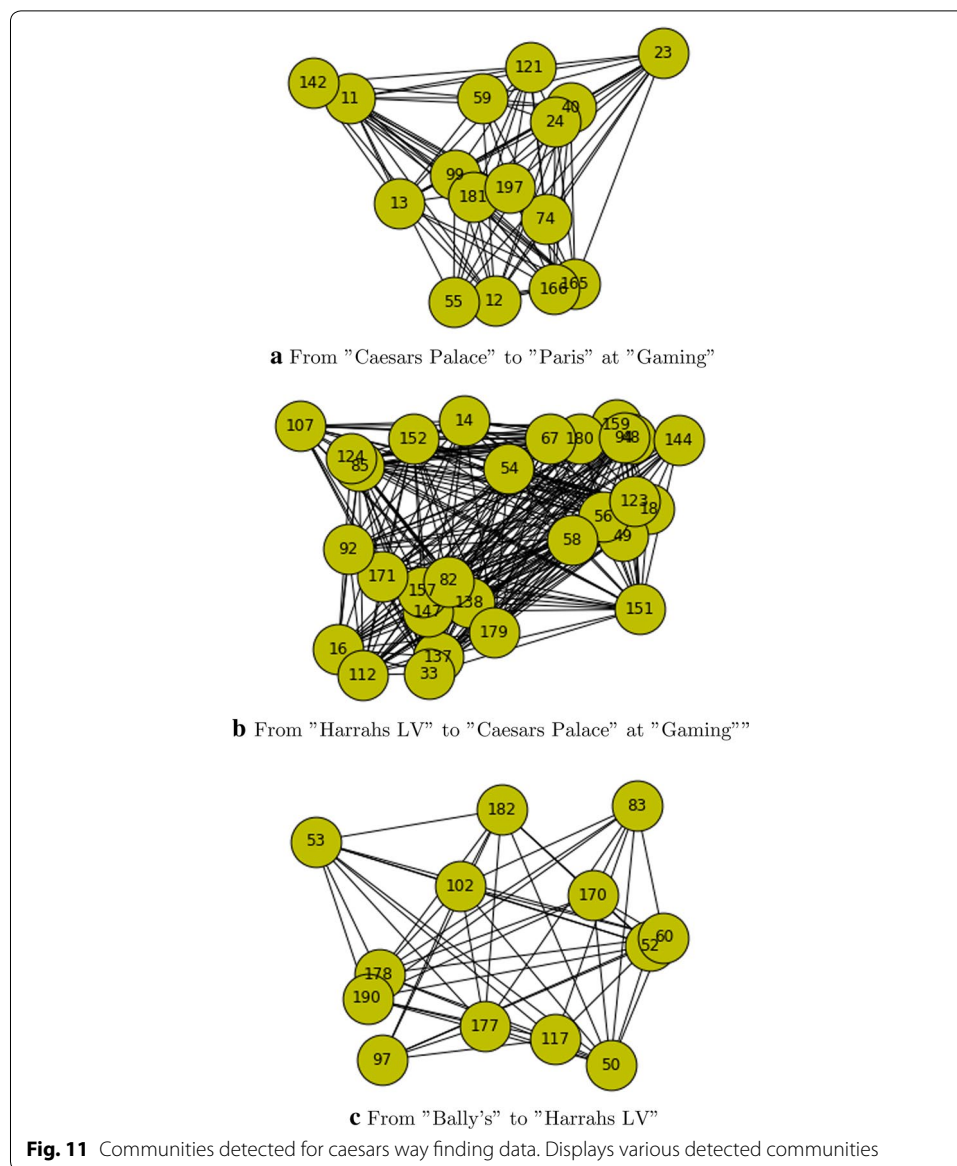
Class	Node V	Attributes	Count
1	0	{2, 4, 5, 8, 10}	48
2	2	{0, 1, 6, 9, 10}	35
3	9	{0, 1, 2, 5}	15
4	11	{0, 6, 7, 9}	16
5	14	{0, 3, 4, 5, 6}	29
6	21	{0, 1, 6, 7, 10}	6
7	31	{2, 4, 6, 8, 10}	4
8	35	{0, 1, 2, 3}	9
9	47	{0, 2, 3, 6, 8}	12
10	50	{0, 5, 7, 8, 9}	13
10	61	{1, 2, 4, 10}	5
10	188	{0, 1, 10, 3,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







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* methods, 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, we 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. We 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.

#### Abbreviations

GD: geodesic distance; BFS: Breadth First Search.

#### Acknowledgements

We sincerely and gratefully acknowledge following organizations for their help and support.

*Caesars Entertainment Corporation, Las Vegas, USA*, for providing the attributed graph dataset. This graph dataset was created for the test purpose from the *Anonymous Pervasive WiFi* data, and contains the anonymous information of different patrons visiting different properties of *Caesars Entertainment*, time of visit, place of visit at the property, and path taken from one property to another.

*The United States Department of Defense* (W911NF-17-1-0088, W911NF-16-1-0416), *AEOP/REAP* programs support, the *National Science Foundation* (1625677, 1710716), and the *United Healthcare Foundation* (UHF 1592) for their support and finance for this project.

**Authors' contributions**

All authors have contributed to the research and manuscript with the order they appear. The last author being the project principal investigator. All authors discussed the final results as well as improved the final manuscript. Both authors read and approved the final manuscript.

**Funding**

The authors were funded through the UNLV Big Data Hub. This study was supported by U.S. Department of Defense (Grant numbers: W911NF1810437; W911NF1810246).

**Data availability statement**

Since we used the dataset from Caesars Entertainment for the experimental analysis. This dataset contains the anonymous information about the Caesars' customers visiting different properties at different time and hours of the day. However, we will not be able to disclose the dataset. We used some open source datasets for experimental analysis available [38]. All the implementation related source codes are available at : <https://github.com/sanketchobe/Advance-Community-Detection>.

**Consent for publication**

Not applicable.

**Competing interests**

The authors declare that they have no competing interests.

**Author details**

<sup>1</sup> University of Nevada, Las Vegas, Las Vegas, USA. <sup>2</sup> University of Arkansas, 227 N. Harmon Ave., Fayetteville, AR, USA.

Received: 27 April 2019 Accepted: 16 August 2019

Published online: 07 September 2019

**References**

- Albert R, Barabási AL. Statistical mechanics of complex networks. *Rev Mod Phys*. 2002;74:47–97.
- Sun PG, Sun X. Complete graph model for community detection. *Phys A Stat Mech Appl*. 2017;471:88–97.
- Rosvall M, Bergstrom CT. An information-theoretic framework for resolving community structure in complex networks. *Proc Natl Acad Sci*. 2007;104:7327–31.
- Raghavan UN, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks. *Phys Rev E*. 2007;76:36106.
- Aldous D, Fill J. Reversible Markov chains and random walks on graphs; 2002. <http://www.stat.berkeley.edu/users/aldous/RWG/book.html>.
- Zhang XS, Wang RS, Wang Y, Wang J, Qiu Y, Wang L, et al. Modularity optimization in community detection of complex networks. *Epl*. 2009;87:38002.
- Fortunato S, Castellano C. Community structure in graphs. In: Meyers RA, editor. *Computational complexity theory*. New York: Springer; 2012. p. 490–512.
- Sachan M, Contractor D, Faruque TA, Subramaniam LV. Using content and interactions for discovering communities in social networks. In: *Proceedings of the 21st international conference on world wide web—WWW '12*. New York: ACM; 2012. p. 331. <http://dl.acm.org/citation.cfm?doid=2187836.2187882>.
- Girvan M, Newman MEJ. Community structure in social and biological networks. *Proc Natl Acad Sci USA*. 2002;99:7821–6.
- Nallapati RM, Ahmed A, Xing EP, Cohen WW. Joint latent topic models for text and citations. In: *Proceeding 14th ACM SIGKDD international conference knowledge discovery data Min—KDD 08*. New York: ACM; 2008. p. 542.
- Liu Y, Niculescu-Mizil A, Gryc W. Topic-link LDA. In: *Proceedings of the 26th annual international conference on machine learning*. New York: ACM; 2009. p. 1–8.
- Newman M, Girvan M. Finding and evaluating community structure in networks. *Phys Rev E*. 2004;69:26113.
- Fortunato S, Barthélemy M. Resolution limit in community detection. *Proc Natl Acad Sci*. 2006;104:36–41.
- Li HJ, Bu Z, Li A, Liu Z, Shi Y. Fast and accurate mining the community structure: integrating center locating and membership optimization. *IEEE Trans Knowl Data Eng*. 2016;28:2349–62.
- Tong H, Faloutsos C, Gallagher B, Eliassi-Rad T. Fast best-effort pattern matching in large attributed graphs. In: *Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery data mining—KDD '07*. New York: ACM; 2007. p. 737.
- Fang Y, Cheng R, Luo S, Hu J. Effective community search for large attributed graphs. In: *Proceedings of the VLDB endow. VLDB Endowment*; 2016. p. 1233–44.
- Zhou Y, Cheng H, Yu JX. Graph clustering based on structural/attribute similarities. In: *Proceedings of the VLDB endow. VLDB Endowment*; 2014. p. 718–29.
- Ruan Y, Fuhry D, Parthasarathy S. Efficient community detection in large networks using content and links. In: *Proceedings of the 22nd international conference on world Wide Web—WWW '13*. New York: ACM; 2016. p. 1089–98.
- Xu Z, Ke Y, Wang Y, Cheng H, Cheng J. A model-based approach to attributed graph clustering. In: *Proceedings of the 2012 international conference on management of data—SIGMOD '12*. New York: ACM; 2012. p. 505.
- Yang J, McAuley J, Leskovec J. Community detection in networks with node attributes. In: *Proceedings of the IEEE international conference data mining. New York: ICDM*; 2013. p. 1151–6.
- He T, Chan KCC. MISAGA: an algorithm for mining interesting subgraphs in attributed graphs. *IEEE Trans Cybern*. 2018;48:1369–82.

22. Yang T, Jin R, Chi Y, Zhu S. Combining link and content for community detection. In: Alhajj R, Rokne J, editors. Encyclopedia of social network analysis and mining. New York: Springer; 2017.
23. Sozio M, Gionis A. The community-search problem and how to plan a successful cocktail party. In: Proceedings 16th ACM SIGKDD international conference on knowledge discovery data mining—KDD '10. New York: ACM; 2010. p. 939.
24. Xiao Y, Lu Y, Cui W, Wang W, Wang H. Online search of overlapping communities. In: Proceedings of the 2013 international conference on management data—SIGMOD '13. New York: ACM; 2013. p. 277.
25. Cui W, Xiao Y, Wang H, Wang W. Local search of communities in large graphs. In: Proceedings of the ACM SIGMOD international conference on management data—SIGMOD '14. New York: ACM; 2014. p. 991–1002.
26. Li R-H, Qin L, Yu JX, Mao R. Influential community search in large networks. In: Proceedings of the VLDB endow. VLDB Endowment; 2015. p. 509–20. <http://www.vldb.org/pvldb/vol8/p509-li.pdf>.
27. Yu JX, Huang X, Qin L, Cheng H, Tian W. Querying k-truss community in large and dynamic graphs. In: Proceedings of the ACM SIGMOD international conference on management data—SIGMOD '14. New York: ACM; 2014. p. 1311–22.
28. Bhalotia G, Hulgeri A, Nakhe C, Chakrabarti S, Sudarshan S. Keyword searching and browsing in databases using BANKS. In: Proceedings of the international conference on data engineering; 2002. p. 431–40.
29. Kacholia V, Pandit S, Chakrabarti S, Sudarshan S, Desai R, Karambelkar H. Bidirectional expansion for keyword search on graph databases. Vldb. VLDB Endowment; 2005. p. 505–16. <http://dl.acm.org/citation.cfm?id=1083592.1083652>
30. Ding B, Yu JX, Wang S, Qin L, Zhang X, Lin X. Finding top-k min-cost connected trees in databases. In: Proceedings of the international conference on data engineering; 2007. p. 836–45.
31. Kargar M, An A. Keyword search in graphs. In: Proceedings of the VLDB endow. VLDB Endowment; 2014. p. 681–92.
32. Lian X, Chen L, Sun Y, Wang G, Yu JX, Yuan Y. keyword search over distributed graphs with compressed signature. IEEE Trans Knowl Data Eng. 2017;29:1212–25.
33. Seidman SB. Network structure and minimum degree. Soc Netw. 1983;5:269–87.
34. Dorogovtsev SN, Goltsev AV, Mendes JFF. K-core organization of complex networks. Phys Rev Lett. 2005;96:40601.
35. Li RH, Yu JX, Mao R. Efficient core maintenance in large dynamic graphs. IEEE Trans Knowl Data Eng. 2014;26:2453–65.
36. Wu L, Bai T, Zhe W, Wang L, Hu Y, Ji J. A new community detection algorithm based on distance centrality. In: Proceedings of the 2013, 10th international conference on fuzzy system knowledge discovery FSKD 2013; 2013. p. 898–902.
37. Hutair MB, Aghbari Z AI, Kamel I. Social community detection based on node distance and interest. In: Proceedings of the 3rd IEEE/ACM international conference big data computing, application technologies—BDCAT '16; 2016. p. 274–89.
38. Leskovec J, Krevl A. SNAP datasets: stanford large network dataset collection; 2014. <https://snap.stanford.edu/data/wiki-Vote.html>.
39. Cheng X, Dale C, Liu J. Statistics and social network of YouTube videos. In: IEEE International workshop on Quality of Service IWQoS; 2008. p. 229–38.
40. Kwak H, Lee C, Park H, Moon S. What is Twitter, a social network or a news media? In: WWW '10 proceedings of the 19th international conference world wide web. New York: ACM; 2010. p. 591.
41. Gutfraund A, Genkin M. A graph database framework for covert network analysis: an application to the Islamic State network in Europe. Soc Netw. 2017;51:178–88.

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Submit your manuscript to a SpringerOpen<sup>®</sup> journal and benefit from:**

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

---

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)