4. DIRECTED NETWORK BASED ON COMMUNITY DETECTION

Community detection is the graph theoretical problem of identifying meaningful subgroups within a network. Within Twitter, this corresponds to groups of users who share similar interests, or who are engaging with each other on a particular topic. Though studies have used community detection methods to analyze twitter networks, these have generally ignored the directionality of the edges. Most of the widely-used community detection methods are defined for undirected networks and are not easily adapted to the directed. Most of the community detection strategies are designed for directed networks and several applications to which community detection is highly relevant are better modeled in directed networks. The directed network-based community detection methods and their implementation on sample twitter network are described in this chapter.

4.1 INTRODUCTION

In recent years, social community research has been carried out using a large amount of data collected from online interactions and from explicit courting hyperlinks in online social community systems including Facebook, Twitter, LinkedIn, Flickr, Instant Messenger, and so on. Twitter is a new shape of media and utilized in numerous fields, consisting of corporate advertising and marketing, education, broadcasting and etc. Structural characteristics of such social networks can be explored using socio metrics to understand the structure of the network, the properties of links, the roles of entities, information flows, evolution of networks, clusters/communities in a network, nodes in a cluster, center node of the cluster/network, and nodes on the periphery. The functionality of the associated objects from network groups are observed based on interaction modules, characteristic values and are expecting unobserved connections among nodes. The nodes have many relationships among themselves in communities. Identifying community is a complex task of clustering nodes into small communities and a node may be belonging to a couple of communities straight away in a community structure [78].

Two exclusive assets of facts are used to carry out the clustering in community detection. First is nodes and its attributes and the second one is the connection between nodes. The attributes of nodes in community structure are known properties of users like network profile, author publication, publication histories help to determines similar nodes and community module to which the node belongs. The connection between nodes provides information about friendships, authors collaborate, followers, and topic interactions.

Few clustering algorithms employ node attributes but ignore the relationships among nodes. The network detection algorithms use corporations of nodes which can be densely linked but ignore the node attributes. Some community detection algorithms fail to describe the critical shape in a community. For example, attributes may also inform about which community node with few hyperlinks belonging to and it is difficult to determine from community structure alone. The community offers detail about nodes belongs to the equal community even some of the nodes have no attributes values. Node attributes can balance the network structure which ends up in accurate detection of communities. Thus, community detection becomes a challenging task when taking into account both nodes attribute and network topology.

The problem of clustering in directed networks is measured to be a more challenging task as compared to the undirected case. It is clear that ignoring edge directionality and considering the graph as undirected is not a meaningful way to cluster directed networks as it fails to confine the asymmetric relationships implied by the edges of a directed network. Therefore, the main challenge is to suggest meaningful ways to incorporate edge directionality in the clustering process. Some additional essentials that strengthen the challenging nature of the problem are:

- Graph concepts are theoretically well formulated for undirected graphs, but not enough effort has been taken to extend these concepts on directed graphs
- Extension of the solutions proposed for the undirected graph based community detection problem to community detection on a directed graph is not straightforward
- The intuition based on the intra-cluster and inter-cluster edge density cannot be easily extended to the directed case, due to the absence of link symmetry.
- The presence of directed edges implies more sophisticated types of clusters that do not exist in undirected networks and cannot be captured using only density and edge concentration characteristics

The proposed method overcomes the above hassle by identifying groups based totally on the node and its attributes with the aid of implementing Girvan-Newman set of rules. In this work, the Girvan-Newman algorithm based on Edge-Betweenness Modularity and Random walk is applied for discovering communities in networks with node attributes [79].

These two algorithms for community detection in the directed network are described in the following sections.

4.2 GIRVAN-NEWMAN EDGE BETWEENNESS ALGORITHM

Girvan and Newman is a general community finding algorithm. It performs natural divisions among the vertices without requiring specifying the number of communities. Girvan and Newman have proposed an algorithm which has three definitive features; (1) edges is gradually removed from a network (2) the edges to be removed are chosen by computing betweenness scores (3) the betweenness scores are recomputed for removal of each edge.

Girvan-Newman method of community detection is a divisive method where edge weight is the number of shortest paths passing through the edge. This value is called as edge betweenness and it is a generalization of central vertex betweenness which determines vertex influence on other vertices in the network. Vertex betweenness is the number of shortest paths passing through the vertex, therefore, edge betweenness is the number of shortest paths passing through the endpoints of the edge.

Girvan-Newman algorithm has the following steps

1. Compute edge betweenness for every edge in the graph
2. Remove the edge with the highest edge betweenness
3. Calculate edge betweenness for remaining edges
4. Repeat steps 2-4 until all edges are removed

The core of the Girvan-Newman algorithm is the calculation of edge betweenness. In order to calculate edge betweenness, it is necessary to find all the shortest paths in the graph. The algorithm starts with one source vertex $s \in V$ in a graph, calculates edge weights for paths going through that vertex, and then repeats it for every vertex in the graph and additions the weights for every edge.

There are two cases of graphs considered here to find the number shortest paths. In the simplest case, the graph is a tree as shown in Fig.4.1a and there exists only one shortest path from the source vertex to any other vertex. Starting from the leaves it allocates the value 1 to the edges that connect leaves with the rest of the tree as there is only one shortest path to s passing through that edge. It assigns edge value for an edge as a sum of values assigned to the edges directly below it. The number of the shortest paths in the tree from the source vertex to every other vertex passing through a particular edge is determined as the edge weight value. By repeating the process for every vertex and calculating the sum of weight values for every edge, the edge betweenness for every edge is calculated.

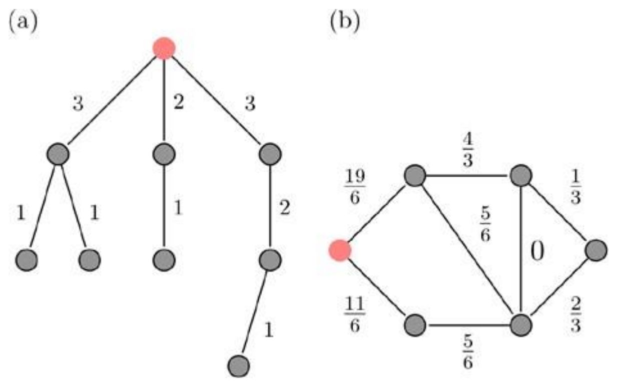

Fig. 4.1 Number of Shortest Paths in a Graph

If the graph is not a tree as in Fig.4.1b, it is possible that more than one shortest path connects source vertex with some other vertex. In that case, value k / l is assigned to the edge on the shortest path to source vertex, where *k* is the number of shortest paths to source vertex from the endpoint of the edge that is closer to source and *l* is the number of shortest paths from source vertex to another endpoint of the edge. This value is by multiplying by the number of shortest paths from a source that pass-through edges below farther vertex increased by one to find the edge betweenness. Assignment starts from the edge that has a maximum distance from source vertex.

The algorithm for calculating edge betweenness is performed in two phases. In the first phase of the algorithm, using breadth-first search, distance from source vertex is assigned for every vertex and also the number of shortest paths from source to vertices is determined. This part of the algorithm is efficiently implemented using an abstract data type queue. The second phase starts from edge incident to the vertex with maximum distance covering from the source vertex as endpoints. The numbers of shortest paths passing through edges are calculated for every edge.

Mathematically, for every vertex i $\in$V, the triple (d_i,w_i,b_i) is calculated, where d_i is the distance from the source vertex, w_i is the number of shortest paths from source vertex to vertex i, and b_i is the number of shortest paths between source vertex to any vertex in graph that passes through vertex i. Let Adj (v) is defined as the set of all vertices adjacent to v such that v$\in$V.

Algorithm – Calculation of edgebetweenness (phase 1- vertex marking)

```
1. For initial vertex s $\in$V let $d_s = 0$, =1 $w_s$, $b_i = 0$.
2. Let = inf, $d_v$, $w_v = 0$, $b_v$ =1 for all $v \neq s \in V$.
3. Create queue Q, Q $\leftarrow$ {s}. Create list L, L $\leftarrow$ {s}.
4. While Q is not empty:
(a) Dequeue i $\leftarrow$ Q.
(b) For each vertex j $\in$ Adj(i):
    i. If = inf then $d_j = d_i + 1$, $w_j = w_i$. Enqueue j $\rightarrow$ Q.  Push j $\rightarrow$ L.
    ii. If $d_j \neq$ inf and $d_j = d_i + 1$ then $w_j + = w_i$.
    iii. If $d_j \neq$ inf and $d_j < d_i + 1$, do nothing.
```

The second phase of edge betweenness calculation starts from the vertex that was last marked in the first phase and visits vertices in reverse order than they were visited in the first phase. Only one shortest path from the source passes through the last marked vertex.

Algorithm – Calculation of edge betweenness (phase 2)

```
While L is not empty:
(a) Pop i $\leftarrow$ L.
(b) For each vertex j $\in$ Adj (i):
    i. If $d_i < d_j$ then = $+\sum_j b_i \ 1 \ \sigma_{ij}$.
    ii. If $d_i > d_j$ then $\sigma_{ij} = w_j/w_i$. $b_i \cdot$
```

Both phases of the algorithm are performed for all source vertices s and edge betweenness for every edge is calculated as a sum of the edge betweenness calculated in every step. The computational complexity of this part of the algorithm is O *(mn*), where m is the number of edges and n is the number of vertices. After each edge betweenness calculation, the edge with highest edge betweenness is removed and the algorithm is repeated until there is no remaining edge. The complexity of the Girvan-Newman algorithm is, therefore, O (m^2n) [80].

4.3 RANDOM WALKS ALGORITHM

Community detection in a large complex network can be carried out by capturing the network structure using a random walk in the network. The intuition behind the random walk is that the network tends to be trapped inside a denser region or community for a longer period of time. This idea is used for inclusion of nodes in the community. A random walk is a mathematical concept formalizing a procedure consisting of a sequence of random steps. In case of graphs, given a node that corresponds to a starting point, a random walk is defined as the sequence of nodes formed by a repeating process starting from the initial node and randomly moving to neighborhood nodes. At each step, the random walker is situated on a node of the graph and jumps to a new node selected randomly and uniformly among its neighbors. Random walk in a graph is the process of visiting a neighboring node randomly from the source node and continuing the process of visiting throughout the graph. Random walk process is similar to Markov chain, where the set of states corresponds to vertices in the visited path.

Mathematically, let $G_U = (V, E)$ be a directed graph and v_0 be the starting node of the random walk. At the t^{th} step, the random walk is situated at node i. At (t + 1) step, the random walk is moving from node i to node j with transition probability $1/ d_i$ where d_i is the degree of node i. The probability of visiting all nodes from all other nodes in the network through k length random walk is represented by transition matrix T^k. Each tuple in the transition matrix corresponds to probabilities of visiting all other nodes from node i in k walk length. These probabilities are based on structural information in the network. From the structure of the network, the following inferences are drawn:

- If two nodes i and j, are in the same community, the probability of visiting node j from i would be higher as compared to visiting a node outside the community. If the probability is high, it does not mean that they belong to the same community.
- The probability $T^k_{i,j}$ depends on the degree of j because the walker tends to visit towards vertices, where the degree is high.
- Two vertices belonging to the same community tend to see all other vertices in the same way and $T^k_{i,m} \approx T^k_{j,m}$, $\forall$ i, j $\in$ same community and m $\in$ [1, n]

Transition matrix obtained by a random walk in the graph is considered for detection of communities. Transition matrix describes the probability of visiting each node from every other node in k number of steps. $T^k_{i,j}$ corresponds to the probability of visiting node j from i in k number of steps. T^1, T^2, $T^{3,}$ and T^k are the transition matrices for random walk corresponding to 1, 2, 3 and k walk length, respectively. Transition probability from vertex i to vertex j in one length random walk is defined by the following equation:

$$T^1_{ij} = \frac{A_{ij}}{d_i} \qquad (4.1)$$

where A_{ij} is the adjacency matrix of the network and d_i is the degree of vertex i.

A node belonging to the same community will have similar behavior as compared to nodes outside the communities. Any two nodes inside a community look the same way as other nodes in the network. The similarity between two vertices is identified from the transition matrix $T^k_{i,j}$ based on the walk length k. The probability of reaching one node from another would be different for different walk lengths. The similarity between i and j for k walk length can be computed by the Euclidean distance between row vectors corresponding to nodes i and j, in matrix T^k

$$Sim(i,j) = \sqrt{\sum_{l=i}^{n} \frac{(Tki,l - Tkj,l)2}{dl}} \qquad (4.2)$$

The similarity between nodes calculated based on the random walk in the network is the core of community detection. The time complexity of walktrap algorithm is found to be O (mn^2) in the worst case, where m is the number of edges and n is the number of nodes in the network [81].

Modularity

Modularity is used to measure the quality of network partitioning of a network. Modularity (Q) quantifies the community strength by comparing the fraction of edges within the community with such fraction when random connections between the nodes are made. The justification is that a community should have more links between themselves than a random gathering of people. The Q value close to 0 means that the fraction of edges inside communities is no better than the random case and the value of 1 means that a network community structure has the highest possible strength. Mathematically, modularity (Q) is defined as:

$$Q = \sum_{c_i \in C} \left[\frac{\left|E_{C_i}^{in}\right|}{|E|} - \left(\frac{2|E_{C_i}^{in}| + \left|E_{C_i}^{out}\right|}{2|E|} \right) \right] \qquad (4.3)$$

where C is the set of all the communities, c_i is a specific community in C, $|E_{C_i}^{in}|$ is the number of edges between nodes within community c_i , $\left|E_{C_i}^{out}\right|$ is the number of edges from the nodes in community c_i to the nodes outside c_i , and |E| is the total number of edges in the network.

Modularity can also be expressed in the following form:

$$Q = \frac{1}{2|E|} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2|E|} \right] \delta_{ci,cj} \qquad (4.4)$$

where k_i is the degree of node i, A_{ij} is an element of the adjacency matrix, δc_i, c_j is the Kronecker delta symbol, and c_i is the label of the community to which node i is assigned.

Since larger Q means a stronger community structure based on modularity optimization. The modularity measure defined above is suitable only for directed and unweighted networks. This definition can be naturally extended to apply to directed networks as well as to weighted networks. Weighted and directed networks contain more information than undirected and unweighted ones and are therefore often viewed as more valuable but also as more difficult to analyze than their simpler counterparts.

Alternatively, the modularity Q proposed by Newman and Girvan [68] as a degree of the selected division of a network is defined as follows:

Q = (range of edges inside communities) − (predicted a wide variety of such edges)

The modularity Q measures the fraction of the edges within the community that join vertices of the same type, i.e., inside-community edges, minus the expected value of the same quantity in a community with the equal network department however with random connections among the vertices If the variety of inside community edges is not any higher than random, Q = zero. A price of Q this is near 1, which is the maximum, indicates strong community shape. Q usually falls inside the range from 0.3 to 0.7 and excessive values are rare [82].

4.4 DIRECTED NETWORK COMMUNITY DETECTION MODEL

Directed network-based community detection model is built using edge betweenness and random walks. The model includes three components, input, process, and output. The input component uses twitter network data presented in chapter 3. The graph representation of the twitter data is used as input. The second component includes a community detection process wherein two different algorithms described in sections 4.2 and 4.3 are pursued. In the first case graph partitioning method is used and the number of shortest paths is determined to find the edge betweenness. In the second case, the similarity measure is used to group the nodes through random walks. The effectiveness of the algorithms is evaluated using the modularity score. The third component is the output logic which analyses the detected communities based on their membership distribution and various properties. The modularity score is evaluated based on the communities detected by these two algorithms. The inferences are drawn based on the experiment results. The architecture of the model is shown in Fig.4.2.

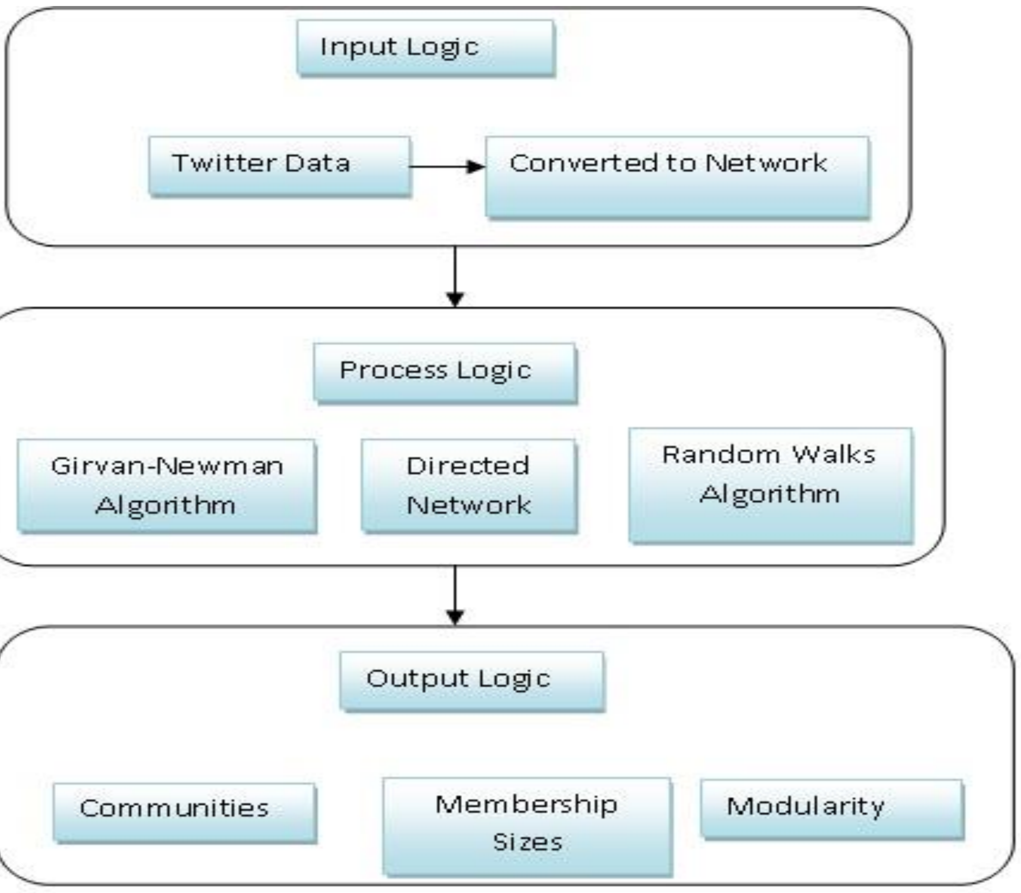

Fig. 4.2 Community Detection Framework

4.5 EXPERIMENTS AND RESULTS

Girvan-Newman Algorithm and Random walk algorithms are employed for community detection based on edge-betweenness. The real-time twitter network data described in chapter 3 is used for the experiment of identifying the communities. The experiment is carried out in R environment. Thirty-nine different communities are extracted from this network based on edge-betweenness modularity measure and demonstrated in different colors in Fig.4.3. These 39 communities are clustered based on followers, friends, and both followers and friends in the network. The distribution of nodes in various communities is shown in Fig.4.4. The modularity score for this sample network is obtained as 0.91 by edge betweenness algorithm.

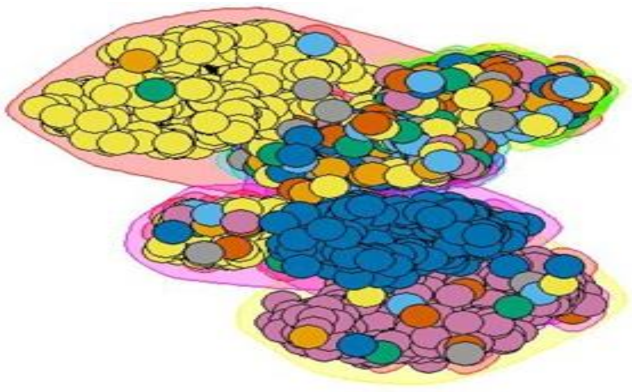
Fig. 4.3 Communities Detected using Edge-Betweenness Algorithm

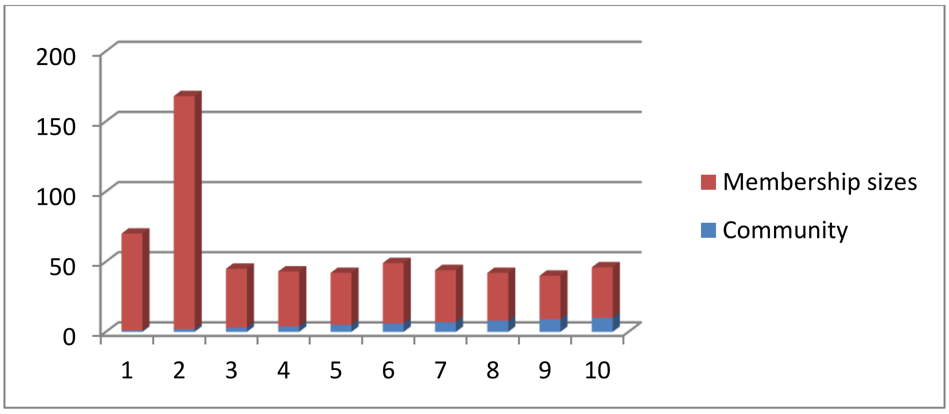

Fig. 4.4 Edge-Betweenness Community Size

The membership size of Community 1 is 69; community 2 has the highest size with 166 memberships. Communities 3 and 4 have the membership sizes 42 and 39 respectively. Communities 5 and 7 have the same membership size 37 and so on. Five communities of the network are recognized as dense and 34 communities are recognized as sparse. A sample of ten communities detected by the Girvan-Newman algorithm and their respective membership sizes are shown in Table X.

Table X Communities Identified by Edge betweenness and Respective Sizes

Commu nity	Member ship sizes	Commu nity	Member ship sizes	Commu nity	Member ship sizes	Commu nity	Member ship sizes
1	69	11	22	21	37	31	27
2	166	12	36	22	21	32	14
3	42	13	23	23	14	33	11
4	39	14	36	24	19	34	15
5	37	15	218	25	10	135	21
6	43	16	23	268	20	36	171
7	37	17	27	27	21	147	18
8	34	18	23	284	24	38	70
9	26	19	25	29	19	39	25
10	31	20	235	30	19		

In the case of random walks, eight different communities are extracted from the same twitter network based on modularity measure and these communities are demonstrated in different colors in Fig.4.5. These 8 communities are clustered based on followers, friends, and both followers and friends in the network. The distribution of nodes in various communities is shown in Fig.4.6. The modularity score yielded by random walks algorithm is 0.7.

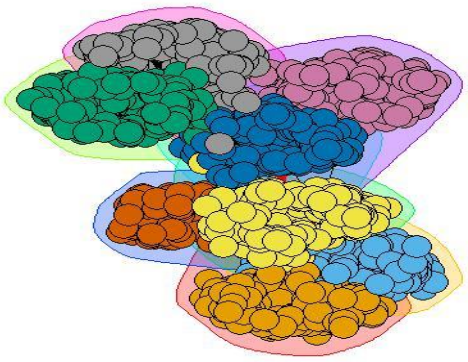

Fig.4.5 Communities Detected using Random Walk Algorithm

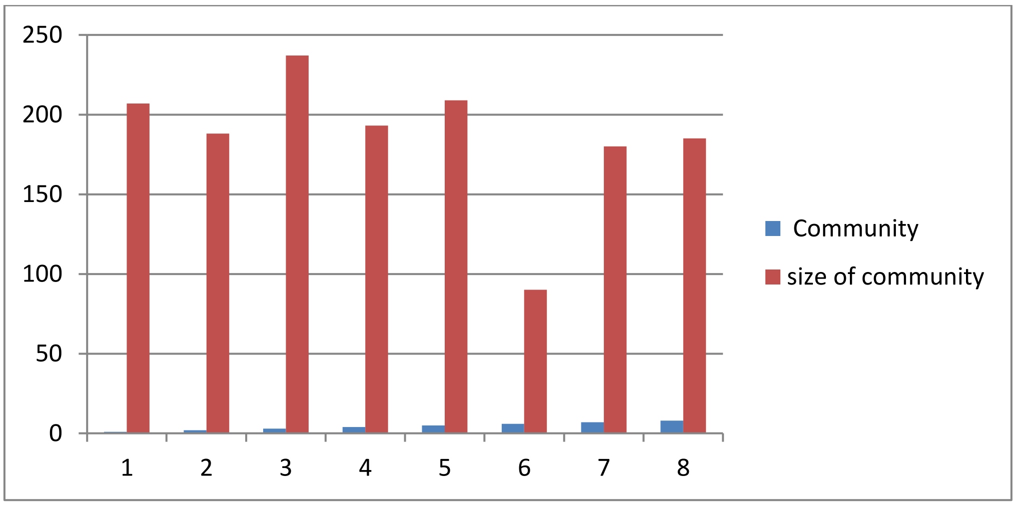

Fig.4.6 Random Walk Community Size

The membership size of Community 1 is 207, community 2 has 188 and community 3 has the highest size with 237 memberships. Communities 4 and 5 have the membership sizes 193 and 209 respectively. Communities 1, 3 and 5 have the high membership size of other communities. Five communities of the network are identified as dense and 3 communities are identified as sparse. Eight communities detected by random walks algorithm and their membership sizes shown in Table XI.

Table XI Communities Identified by Random Walks and Respective Sizes

Community	Membership sizes
1	207
2	188
3	237
4	193
5	209
6	220
7	180
8	285

Comparison of Edge-Betweenness and Random walks

The comparative analysis of these two directed network-based community detection algorithms is made in terms of a number of communities, membership distribution, and modularity score. The modularity score obtained through Edge-Betweenness algorithm is 0.91, which proves that the sports person's twitter network is highly dense. Also, the Girvan-Newman algorithm has detected 39 different communities from the network and found that out of 39 communities, 5 communities are dense. The modularity score found through Random walk algorithm is 0.7, which also confirms that the sports person's friends and followers network is dense. The random walk algorithm has found 8 communities from the sports person's network which are all very dense. The comparative results are given in Table XII and illustrated in Fig.4.7

Table XII Community Analysis of Edge-Betweenness and Random Walks

Algorithm	Modularity Score	Number of Communities	Number of Dense Communities	Number of Sparse Communities
Edge-Betweenness	0.91	39	4	35
Random Walks	0.7	8	5	3

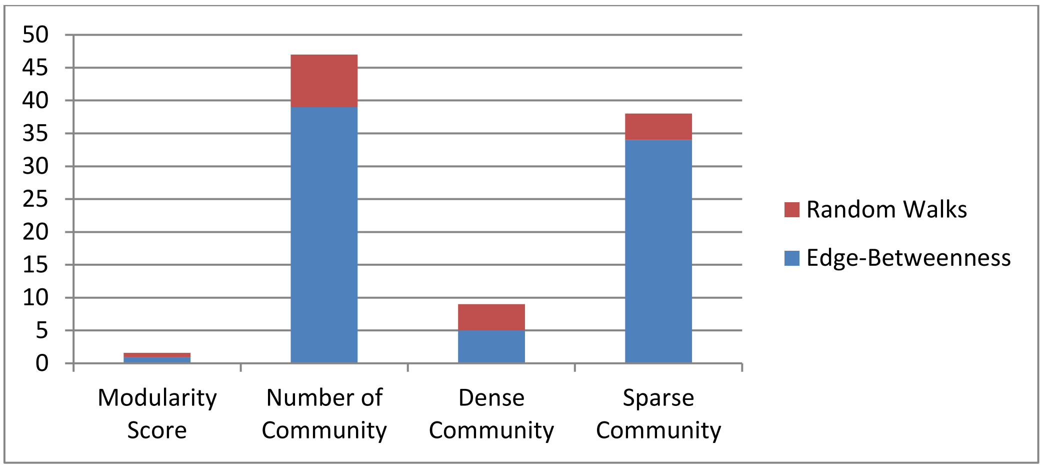

Fig. 4.7 Comparitive Results of Edge-Betweenness and Random Walks

Findings

The aim of this experiment is network detection in graphs which discovers the subgroups by using the statistics encoded inside the graph topology. Girvan–Newman algorithm has discovered a number of sparse communities than random walks algorithm and has eliminated them during clustering. Random walks algorithm finds less number of communities with high communication between the nodes. The modularity value 0.91 of the tested network by Edge-Betweenness confirms that the network is dense and the algorithm is efficient in finding different communities with sparsity. The modularity value 0.71 of the checked network by Random walk proves that the network is dense and the algorithm is efficient in finding different communities with density.

SUMMARY

The application of the Girvan-Newman algorithm based on Edge-Betweenness and random walk for detecting communities from networks with node attributes has been demonstrated in detail in this chapter. The methodology of the approach and the experiments carried out using the real-time twitter directed network of a sports person have been illustrated with tables and charts. This approach achieved the goal of detecting principal communities from the directed network data with basic network properties using graph partitioning technique. Further exploration of community detection based on subgraph analysis is elucidated in the next chapter.