Course Project Report

Community Detection In Complex Network Using Leader Knowledge

Submitted By

Abhishek Kumar (202222)
Mohd Asif Khan Khaishagi (202200)
Nitin Sharma (202230)

as part of the requirements of the course

Web and Social Computing (IT752) [Feb-Jun 2021]

in partial fulfillment of the requirements for the award of the degree of

Master of Technology in Information Technology

under the guidance of

Dr. Sowmya Kamath S, Dept of IT, NITK Surathkal

undergone at

DEPARTMENT OF INFORMATION TECHNOLOGY

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA, SURATHKAL

JUNE 2021

DEPARTMENT OF INFORMATION TECHNOLOGY

National Institute of Technology Karnataka, Surathkal

C E R T I F I C A T E

This is to certify that the Course project Work Report entitled **"Leader aware community detection in complex network"** is submitted by the group mentioned below -

Details of Project Group

Name of the Student	Register No.	Signature with Date
Abhishek Kumar	202222	Abhishek Kumar
Mohd Asif Khan Khaishagi	202200	Asif
Nitin Sharma	202230	[illegible]

this report is a record of the work carried out by them as part of the course **Web and Social Computing (IT752)** during the semester **Feb-June 2021**. It is accepted as the Course Project Report submission in the partial fulfillment of the requirements for the award of the degree of **Master of Technology in Information Technology.**

(Name and Signature of Course Instructor)
Dr. Sowmya Kamath S

DECLARATION

We hereby declare that the project report entitled **"Leader aware community detection in complex network"** submitted by us for the course **Web and Social Computing (IT752)** during the semester **Feb-June 2021**, as part of the partial course requirements for the award of the degree of Master of Technology in Information Technology at NITK Surathkal is our original work. We declare that the project has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles elsewhere.

Details of Project Group

Name of the Student	Register No.	Signature with Date
1. Abhishek Kumar	202222	Abhishek Kumar
2. Mohd Asif Khan Khaishagi	202200	Asif
3. Nitin Sharma	202230	[illegible]

Place: NITK, Surathkal
Date: 21-05-2021

Community Detection In Complex Network Using Leader Knowledge

Abhishek Kumar[1], Mohd Asif Khan Khaishagi[2], Nitin Sharma[3]

Abstract— **Complex networks are rich of different kinds of communities. These communities are often of varying sizes and understanding them is an important task as it gives us insight on different attributes of our network. Focus should also be given to identifying nodes of extra importance in these communities. These nodes are called Leaders of communities, and they have significance influence over other members of the community. One thing to notice in most of the research that has been done in the field of community detection is that they don't emphasis on selection of leaders in these communities.**

Here, in our work, we implement Auto-leader algorithm, which not only focuses on finding clusters of communities but also finds leaders among them. This algorithm focuses on finding leadership of nodes and then depending on how much adjacent nodes attract a particular node, local leader of that node is decided. Apart, from implementation of this algorithm, we also propose our changes to it and then perform comparative analysis between them.

Keywords: **Community, Leadership, Social Network, Cluster, Leader, Dependence Tree**

I. INTRODUCTION

People have different behaviors and preferences which leads to different decision makings. However, in some cases, based on available limited options, a decision is to be made which showcases preferences of individuals. A community or cluster in a social network is a grouping of people with similar tastes and preferences. A community or a cluster is a sub graph of the network in which intracluster edges are way higher than inter-cluster edges. Finding communities in networks helps us in understanding certain behavior of a group of people. Most of the algorithms out there do not find the leaders as they just focus on finding the communities in a network. But If we talk about the communities in real life then there always exists some nodes which have influence over other nodes in the community. Now, the node which has more influence over other nodes can be a potential leader of that community. So, our main task here is to find the leader based communities. For doing that we shall partition the entire network into clusters/communities such that the node which belongs to the same cluster will be the part of the same community and vice versa. Different community detection methods can be broadly classified into node centric, network centric or hierarchy centric community detection. Our approach is based on Network centric community detection.

In Network centric method of community detection, nodes of a network are divided into different disjoint sets and that is the base of our work too. We have followed "Leader aware community detection in complex networks"[1] and by applying some modifications to it we have prepared our own algorithm for finding the communities and leaders of the communities as well. In the algorithm firstly we have find local leaders of all the nodes present in the network based using leadership score of each node. Then we store leader and follower as key value pair and the we have applied transitive property and merging function. Finally, we recreate the communities with their leaders based on the updated key-value pairs which we got after applying the merging function.

II. RELATED WORK

Complex network community detection has attracted many researchers over the years. In this section, we first touch on different approaches for community detection, and then we look at some work done in this field which helped us in understanding what was needed to be done next.

Community detection methods can be broadly classified into three categoriues : Node Centric, Network centric and Hierachy centric methods. Node centric community detection is based on satisfying different properties of nodes like complete mutuality, reachability of members, node degrees and clique percolation method. Network based community detection divides the network into multiple disjoint sets. Common approaches for doing this are structural equivalence, spectral clustering and modularity optimization. Louvian algorithm is an example of this method. Hierarchy based methods focuses on topology of network and exploits hierarchical structure of network for community detections. They are two approaches in hierarchical based method, namely divisive clustering and agglomerative clustering. Girvan-Newmann algorithm is the most famous example of this type of algorithm.

Now lets look at some work done in this field. We primarily are mentioning three papers that have been extensively referred by other researchers too.

Community Detection Based on Girvan Newman Algorithm and Link Analysis of Social Media (Sathiyakumari and Vijaya, 2016) : In this Girvan-Newman algorithm (Girvan and Newman, 2002), our focus is on computing edge betweenness values. We focus on iteratively removing edges from the graph that have highest edge-betweeness value and the resulting network gives us different communities. The limitation of this was that it did not talk about the leader in the community, and we have to specifically mention that up to what number of cluster we want.

A density-based algorithm for discovering clusters in large spatial databases with noise (Ester et al., 1996): They worked on a density-based clustering algorithm called DBSCAN for finding clusters of communities, which was able to discover clusters of arbitrary shape. The limitation of this was that here also we can identify the clusters, but there is no way to find the leaders of those clusters. Moreover, choosing the appropriate parameters here, i.e. epsilon value and min points, is a difficult task and since it is more sensitive to outliers, it leads high number of clusters than ground truth value

LICOD, A Leader-driven algorithm for community detection in complex networks (Yakoubi and Kanawati, 2014): In their work, they proposed a general framework for implementing algorithms and detecting communities and their respective leaders in social networks. Limitation of the work was that they were unable to find the right number of clusters compared to ground truth and in absence of leaders, communities detection task becomes more difficult.

III. DATASET

1) **Zachary karate Club** : This is a well known karate club network which used to ran by Wayne Zachary.Here in this network each node represents the member of the karate club and edge between two members represents fight between them. This is an undirected Network.
2) **American college football network** : This is nothing but a network in which a football team of America is represented by a node of the network and a match between any two teams represented by an edge.In this graph there are 115 nodes and 613 edges.
3) **Polity books dataset** : This is nothing but a network of political books which were written around 2004 U.S presidential election which used to be sold by the online bookseller amazon.com. In this network edges are nothing but the purchase of a book by the same user.In this network there are a total 105 nodes and 441 edges.

Dataset Properties			
Datasets	#vertices	#edges	clusters
Zachary karate club	34	78	2
American college football network	115	613	12
Polity books dataset	105	441	3

We have used these three datasets of real world network for experimentation.

IV. METHODOLOGY

A. RELEVANT CONCEPTS

Given a target graph G with set of nodes V and set of edges E, we define :

- **Structural Neighborhood** - Structural neighborhood of a node is defined as neighbors of the node along with the node itself.

$$\Gamma(u) = \{u\} \cup \{v \in V | (u,v) \in E\} \tag{1}$$

- **Common Neighbors** - Common neighbors of two nodes is defined as the intersection of the structural neighborhood of these two nodes.

$$CN(u,v) = \Gamma(u) \cap \Gamma(v). \tag{2}$$

- **Unique Neighbors** - Unique neighbors of a node w.r.t to another adjacent node is nothing the structural neighborhood of that node minus common neighbors of that two nodes.

$$UN(u,v) = \Gamma(u) - CN(u,v), \tag{3}$$

- **Jaccard similarity** - Jaccard similarity between two adjacent nodes is nothing but the ratio of common neighbors and total unique neighbors between those two nodes.

$$sim(u,v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|} \tag{4}$$

- **Jaccard distance** - It is nothing but inversely proportional to the jaccard similarity. Means as similarity increases then distance decreases and vice versa.

$$d(u,v) = \frac{1}{sim(u,v)} \tag{5}$$

- **Leadership score** - Leader of a node is nothing but the summation of jaccard similarity of that node with all the neighbors(remember it just neighbors not structural neighborhood).

$$LS(u) = \sum_{v \in \Gamma(u), v \neq u} sim(u,v) \tag{6}$$

- **Edge Compactness** - It is defined using the formula given below.

$$\begin{aligned} EC(u,v) = sim(u,v) + \sum_{t \in CN(u,v)} sim(u,t) \times sim(t,v) \\ + \sum_{t \in UN(u|v)} sim(u,t) \times \rho(t,v), \end{aligned} \tag{7}$$

where

$$\rho(t,v) = \begin{cases} sim(t,v) & \text{sim(t,v)} > \lambda \\ sim(t,v) - \lambda & \text{otherwise} \end{cases} \tag{8}$$

- **Attractive force** - Just like force of attraction in physics, where force of attraction was inversely proportional to square of distance between the masses in the same way force of attraction of node(u) over other node (v) is inversely proportional square of jaccard distance between the two nodes. Now for removing the proportionality we shall introduce a constant which is nothing but ratio of degrees multiplied with leadership score of ‘u’:

$$f(u,v) = \frac{deg(u)}{deg(v)} * \frac{LS(u)}{d(u,v)^2} \quad (9)$$

- **Local leader** - Local leader of node (u) is a neighbor of ‘u’ which has a higher force of attraction over ‘u’.

$$l_{loc}(v) = \{u | \arg\max_{u \in N(v)} f(u,v)\} \quad (10)$$

where

$$N(v) = u | u \in \Gamma(v) \wedge LS(u) > LS(v) \wedge EC(v,u) \geq 0$$

- **Dependence tree** - A tree like structure where all the followers of a leader will point to their leader.
- **Transitive property** - Let say leader of node ‘u’ is ‘v’ and leader of ‘v’ is ‘w’ then we can say updated leader of ‘u’ will be ‘w’.
- **Candidate set** - For a node ‘u’, It is nothing but the set of potential nodes which may be the updated leader of ‘u’. It is defined as the neighbors of ‘u’ which have more similarity with ‘u’ than average value.

$$C(v) = \{u | u \in \Gamma(v), sim(v,u) \geq \overline{sim(v)}\} \quad (11)$$

where,

$$\overline{sim(v)} = \frac{1}{|\Gamma(v)|} * \sum_{u \in \Gamma(v)} sim(v,u).$$

Now, lets look at the tweaked auto-leader algorithm.

B. ALGORITHM

There are 3 major sub-parts of our algorithm for detecting communities and leader corresponding to them.

1) **Updated Auto-Leader** : Here we are iterating over all the nodes in the network and checking that which node in neighborhood of u has highest force of attraction over u i.e. find highest f(u,v) such that EC(u,v) >=0 (Edge compactness should be greater or equal to zero). Then v will be the local leader of u. Finally we will return follower information for all local leaders using key as a leader and value as a list of followers corresponding to that leader.

2) **Merging** : There might be a chance that some nodes are wrongly clustered and end up with some other leader than the actual one. So, for removing this limitation we shall apply the merging function. Here we will iterate over local leaders found in part 1 and if that local leader i.e. u is alone then we’ll go in the neighborhood of u and find which node has the highest leadership score and this node will become the updated local leader of ‘u’. Otherwise, we shall check for the same in candidate set and find the potential updated local leader for ‘u’. i.e. check which node in the candidate set has highest leadership score and that will become the updated local leader of ‘u’.

3) **Transitive Operation** : We discussed a property ‘transitive property’ where we discussed all terminologies i.e. if ‘v’ is the leader of ‘u’ and ‘w’ is the leader of ‘v’ then finally ‘w’ will be the leader of ‘u’. Now by using this property we shall update the leader follower such that if a leader ‘x’ is follower of another leader ‘y’ then finally we will take union of their follower and they will follow ‘y’ and delete the ‘x’ finally. And finally, based on the updated leader follower list we got by autoleader we will plot the clusters.

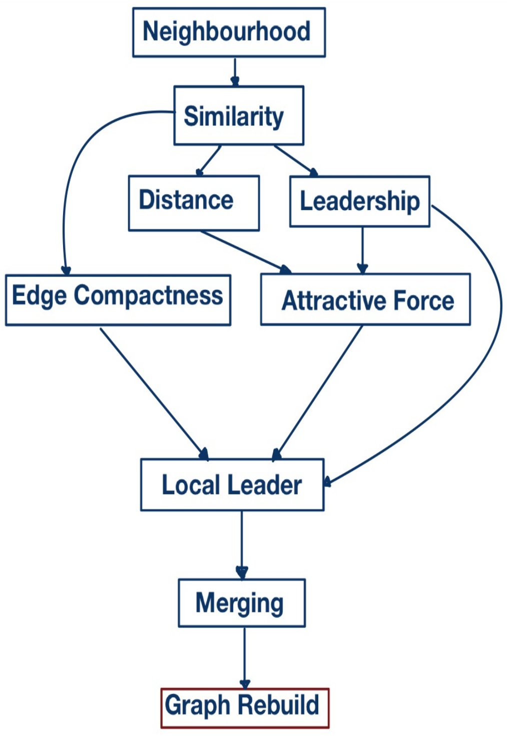


Fig.1 : Flow Diagram

C. PSEUDOCODE

Coding for our whole work was done in python with Google- collab as working environment. Firstly, code for each component described in the section above was written and then code was merged as per requirement. There were three main components in code, and pseudo-code for each one of them is mentioned below.

Updated Auto-Leader

```
INPUT :Graph G=(V,E),
          parameter λ
OUTPUT : {leader : list(followers)}

1: for each edge  e=(u,v) ∈ E do
2:    Compute the parameters
      jaccard similarity sim(u,v) and distance d(u,v)
3: end for
4: for each node v ∈ V do
5:    Compute the leadership score LS(v)
6: end for
7: leader_follower= {}
8: for each v ∈ V do
9:     force={}
10:    for each node u ∈ structural neighbor(v) do
11:        force[u] = attractive force f(u,v)
           Compute edge compactness EC(u,v)
12:    end for
13:    find argmax(force)  and corresponding 'u' = local leader
14:    if u exist in leader_follower do
15:        leader_follower[u].append(v)
16:    else do
17:        leader_follower[u]=[]
18:        leader_follower[u].push(v)
19:    end if
20: end for
21: new_leader_follower = merging(leader_follower)
22: updated_leader_follower=transitive(new_leader_follower)
23: return updated_leader_follower
```

Transitive Opearation

```
INPUT : new_leader_follower{}
           Having key=leader, value=list of followers)
OUTPUT : Updated leader_follower list
0: deleted_key=[]
1: for each k1 ∈ new_leader_follower.key() do
2:     for each k2 ∈ new_leader_follower.key() do
3:         if k1 ∈ new_leader_follower(k2) do
4:             leader_follower[k2] =
                   Union(  new_leader_follower[k1],
                           new_leader_follower[k2] )
5:             delete(k1)
6:             deleted_key.append(k1)
7:         end if
8:     end for
9: end for
10:  for each k1 ∈ deleted_key do
11:     if  k2 ∈  new_leader_follower.key() do
12:         delete(k2)
13:     end if
14: end for
15: return new_leader_follower
```

Merging Function

```
INPUT:  leader_follower(Given by transitive function),
          Graph G(V,E)
OUTPUT: New leader_follower_list
1: for each leader l ∈ leader_follower.key() do
2:     if l.number of followers == 0 then
3:        l.leader= arg max_{m ∈ structural neighbor(l)} LS(m)
4:        leader_follower.remove(l)
5:        leader_follower[l.leader] = l
6:    else
7:       Compute candidate set C(l)
8:       dictionary dic ={}
9:       for each node u ∈  C(l) do
10:          if localleader(u) not in dic then
11:            dic[localleader(u)] = sim(u,l)
12:          else
13:             dic[localleader(u)] += sim(u,l)
14:          end if
15:     end for
16:     sort the dictionary dic by value in descending order
17:     newLeader = the first key of dic
18:     if newLeader != l then
19:        for each node u ∈ leader_follower[l] do
20:            update the leader of followers of l as well.
21:        end for
22:     end if
23: end for
```

V. OBSERVATION AND RESULT

We considered 3 different real world social network datasets and noted their ground truth values for performance evaluation using performance metrics : nmi, ari, purity of clusters. Details of how our algorithms performed against Auto leader algorithm and other state-of-the-art algorithms like Scan end Girvan-Newmann is represented in tables given below.

Here are the observations from the results obtained :

a) It can be observed that in case of Zachary dataset, both Autoleader and our implementation provide better results than SCAN and Girvan Newmann.
b) In case of Football and Polbook dataset, our implementation, if not better, is at least as good as any of the other algorithm.

ZACHARY DATABASE				
Metrics	#cluster	N MI	ARI	Purity
Auto-Leader	2	0.837	0.882	0.971
OUR	2	0.8371	0.882	0.9705
SCAN	4	0.3836	0.1901	0.8235
Girvan-Newmann	4	0.6516	0.6446	0.9705

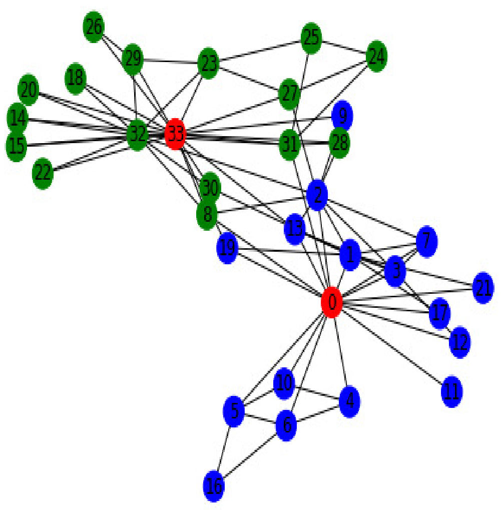


Fig.2 : Zachary Clusters

POLITICAL BOOKS				
Metrics	#cluster	N MI	ARI	Purity
Auto-Leader	4	0.553	0.664	0.810
OUR	4	0.5629	0.6805	0.8476
SCAN	5	0.4238	0.5314	0.8095
Girvan-Newmann	3	0.5754	0.6795	0.8476

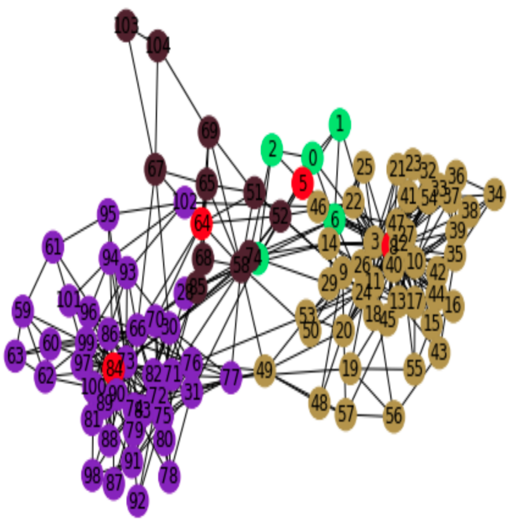


Fig.3: Polity book clusters

American College Football Network				
Metrics	#cluster	N MI	ARI	Purity
Auto-Leader	12	0.902	0.814	0.835
OUR	12	0.9268	0.8893	0.9304
SCAN	11	0.9204	0.8595	0.9043
Girvan-Newmann	12	0.9213	0.8846	0.9304

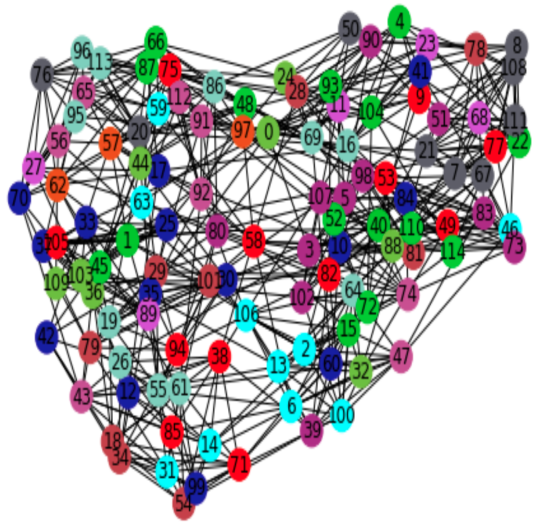


Fig.4: American college football clusters

Community structure that our algorithm found for all three datasets is mentioned above in fig. 2, fig. 3 and fig 4.

We can observe from above tables that either we are getting similar or better results than state-of-the-art algorithms and autoleader algorithm. After analyzing results we found that the reason behind this is that when we have not applied the **transitive operation**, we were not getting the number of clusters close to the ground truth value which leads to difference in the predicted cluster index of a node and the actual cluster index of that node which will result in lower value of NMI, ARI, Purity. But when we applied the **transitive operation** then we got the number of clusters either equal to the ground truth value or very close to the ground truth value, This, after extracting cluster number for each node, results in increase in NMI, ARI and cluster purity as now our nodes gets closely grouped into their original clusters.

VI. CONCLUSION AND FUTURE WORK

Here we explored some state-of-the-art algorithms and compared their performance with an altered version of AutoLeader algorithm. We used details of auto leader algorithm to calculate neighbor information and local leaders using their concept of edge compactness and attractive force. During merging phase of AutoLeader, we changed it ton handle case when a leader is present in follower list of another leader which was the main reason for the changes reflected in metrics. AutoLeader algorithm has already shown very promising results when applied on real world networks and always seems to out-perform SCAN and Girvan-Newmann algorithm. What we also proved that our implementation of algorithm after tweaking gives results at least as good AutoLeader algorithm.
Future work may focus on testing performance of our implementation on bigger dataset. What we can also

work on is to do a comparative analysis based on other algorithmic metrics like extra space required and time taken for execution of our algorithm.

REFERENCES

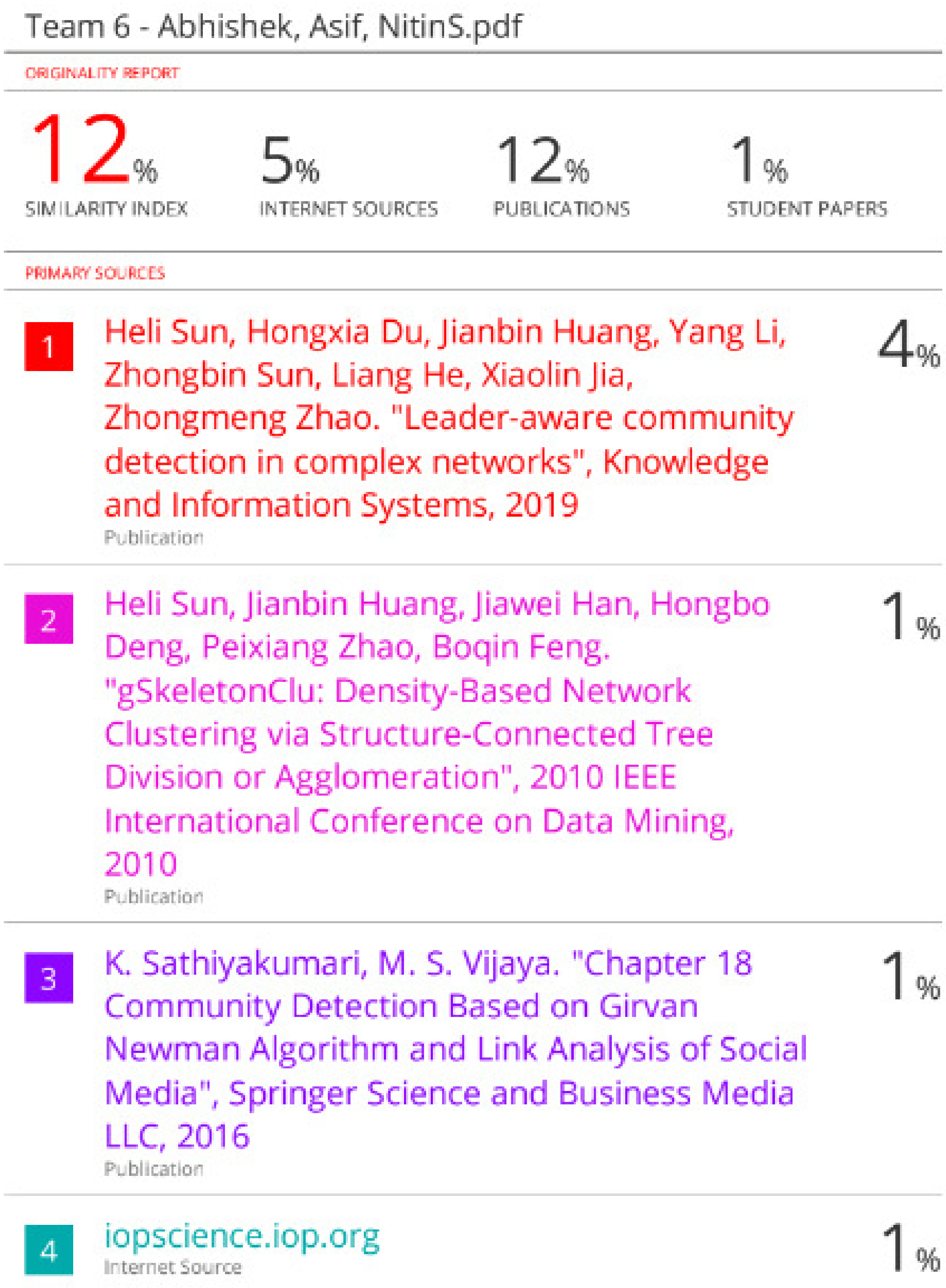
Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press.

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

Sathiyakumari, K. and Vijaya, M. S. (2016). Community detection based on girvan newman algorithm and link analysis of social media. In Subramanian, S., Nadarajan, R., Rao, S., and Sheen, S., editors, *Digital Connectivity – Social Impact*, pages 223–234, Singapore. Springer Singapore.

Yakoubi, Z. and Kanawati, R. (2014). Licod: A leader-driven algorithm for community detection in complex networks. *Vietnam Journal of Computer Science*, 1(4):241–256.

APPENDIX



Team 6 - Abhishek, Asif, NitinS.pdf

ORIGINALITY REPORT

12%	5%	12%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Heli Sun, Hongxia Du, Jianbin Huang, Yang Li, Zhongbin Sun, Liang He, Xiaolin Jia, Zhongmeng Zhao. "Leader-aware community detection in complex networks", Knowledge and Information Systems, 2019 Publication	4%
2	Heli Sun, Jianbin Huang, Jiawei Han, Hongbo Deng, Peixiang Zhao, Boqin Feng. "gSkeletonClu: Density-Based Network Clustering via Structure-Connected Tree Division or Agglomeration", 2010 IEEE International Conference on Data Mining, 2010 Publication	1%
3	K. Sathiyakumari, M. S. Vijaya. "Chapter 18 Community Detection Based on Girvan Newman Algorithm and Link Analysis of Social Media", Springer Science and Business Media LLC, 2016 Publication	1%
4	iopscience.iop.org Internet Source	1%