Leader-aware Community Detection in Social Networks

Heli Sun[a], Hongxia Du[a], Jianbin Huang[b,*], Zhongbin Sun[a], Liang He[a], Xiaolin Jia[a], Zhongmeng Zhao[a]

[a]*Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China*
[b]*School of Software, Xidian University, Xi'an 710071, China*

Abstract

Community structures are very common in social networks. Detecting these communities is important for understanding the hidden features of networks. Besides, each community usually has one leader, which presents its significant influence over the whole community. However, most existing methods just focus on the problem of graph clustering, ignoring the role of community leaders. To solve this problem, in this paper, we propose a novel leader-aware community detection algorithm, which can find community structures as well as leaders of each community. This algorithm measures the leadership of each node and let each one adhere to its local leader, forming dependence trees. Once all dependence trees are definitely settled, the community structures emerge because one tree actually is a cluster. Additionally, each root node of the tree is exactly the leader of corresponding community. This method can quickly determine the belonging of each node. Experimental results on real-world and benchmark networks demonstrate the effectiveness and the efficiency of our algorithm compared with other state-of-the-art approaches.

Keywords: Complex networks, Community detection, Leader-aware, Dependence tree

1. Introduction

Community structures are very common in social networks and information networks. A community or a cluster is a subgraph containing nodes which are more densely linked to each other than to the rest of the graph

*Email: jbhuang@xidian.edu.cn.

[1]. Detecting communities contributes to revealing structural properties of networks. Many approaches are proposed to identify communities based on different objective functions, such as normalized cut [2] and modularity [3]. However, all those algorithms can just discover community structures while totally ignoring the existence of cluster leaders.

In fact, each community usually has a representative leader node, which has the greatest influence over the whole community. For example, in reality, a class has a monitor and a group of horses also have one bellwether. Linyuan Lv [4] et al. indicate that identifying influential leaders is essential in online communities. These leaders play an important role in social networks because they have cohesive connections with members of the communities. Mining community leaders is also meaningful in the field of information dissemination. We can inform each leader of some urgent news so that everyone in the corresponding community can know about it as soon as possible. As Froome stated [5], leadership is essential among organizations, in external policy support, and in facilitating links between the two.

Recently, some researchers recognize the importance of leader nodes during the process of community detection. The leader-follower model is utilized to identify community structures since they hold that a community is a set of follower nodes around a leader [6, 7]. But these existing methods simply evaluate the node leadership based on the node degree. This is not very reasonable because the degree just considers the number of connections but ignoring the tightness of each link. In real world, community structures in social networks usually are formed by all members spontaneously. Considering there is one group of people that need to be divided into several partitions, everyone in fact can only make decision according to its neighbors. One person may choose to follow another person who has the largest influence on him. Everyone can identify its local leader in its neighborhood. Thus the cluster is formed once all the leader-follower chains are determined and the cluster leader emerges naturally. According to this motivation, our method is put forward.

In this paper, we propose a novel algorithm which can detect community structures automatically and identify community leaders. A measure of node leadership is introduced to evaluate the importance of nodes in the local perspective. A leader in a cluster should have cohesive enough connections as many as possible. The attractive force from one node to another is computed. Based on the leadership and attractive force, each node attaches to the local leader in its neighborhood, forming a directed dependence link. After scan-

ning all the nodes, a forest which contains at least one tree emerges. Merging is also necessary to improve the clustering quality since there maybe exist some tiny branches. Finally, the whole available links are settled. Each tree represents a community and the root node is the leader of the corresponding community. From the dependence tree, we can clearly recognize the hidden rule of how one community is formed up.

The main contributions are summarized as follows:

1) We study the problem of community detection in social networks and propose a novel method, which can detect community structures automatically as well as identifying community leaders.

2) Some new metrics are introduced to evaluate the node leadership and identify local leaders of each node. We take full advantage of the neighborhood information, which is easy to be parallelized for large-scale networks.

3) Based on the leadership, each node points to its local leader in a cascade way, forming original community structures spontaneously. Dependence trees are built and the community leaders are obtained during the process of forming communities.

4) We compare the performance of our algorithm with several state-of-the-art community detection approaches on real-world and synthetic datasets. The results show that our method is scalable and achieves comparable performance with a low time complexity.

The rest of the paper is organized as follows. In Section 2, we review some related work in community detection. Section 3 presents our algorithm Autoleader in detail. Next we evaluate our method against several approaches and present the experimental results in Section 4. Finally, we make a conclusion in Section 5.

2. Related Work

The problem of community detection in social networks has attracted interests of many researchers. Until now, plenty of approaches are available, such as modularity-based methods [1], spectral methods [8] and density-based methods [9]. A comprehensive and thorough report has been summarized by Fortunato [10].

Modularity-based methods. Modularity-based algorithms are very popular for solving community detection problem. The concept of modularity is first proposed by Newman and Girvan [1], which becomes the most used and best known quality function. Modularity optimization is an NP-complete problem, so many algorithms aim to obtain the approximation of the modularity maximum of the graph partition. FastModularity [11], BGLL [12] and simulated annealing [13] algorithms achieve excellent performance. Yang et.al [14] propose a novel nonlinear reconstruction method by adopting deep neural networks based on modularity optimization model. However, modularity optimization has a resolution limit that may prevent it from detecting clusters which are comparatively small with respect to the graph as a whole [10].

Density-based methods. Density-based methods obtain widely attention in recent years. Xu et al. propose a structural clustering algorithm SCAN [9] which is based on DBSCAN [15]. The limit of SCAN is that it need user-defined parameters, which influences the results of graph clustering to a certain extent. Some methods [16, 17] are proposed to improve the performance of SCAN. Recently, Arif Mahmood et.al [18] utilize geodesic space density gradients to decide the community membership of each node and achieve excellent performance.

Leader-follower methods. TopLeader [6] is a typical methods. It assumes that a community is a set of follower nodes assembling close to a leader. It first identifies promising leaders in a given network then iteratively assembles followers to their closest leaders to form communities. The major limit is that it requires the number of communities k as a predefined parameter. There are also some approaches [7, 19, 20, 21] based on the leader-follower model. But the clustering results are not very well and they also face the problem of parameter dependence.

Other methods. There are other novel approaches for community detection problem. LPA [22] is an algorithm which utilizes label propagation over the whole network. It can detect community structures in nearly linear time but the results are not stable because of its random update order. Infomap [23] aims to find the optimal divisions based on minimum description length principle, which obtains the best results among all the methods. Shao et.al propose a novel method based on distance dynamics and achieve good performance [24]. There are also some powerful methods [8, 25, 26] on spectral clustering. The main problem of this kind of methods is that the definition of community is too restrictive.

Table 1: Definitions of main notations.

Notation	Definition
G	Given graph
V	Set of nodes in G
E	Set of edges in G
$deg(u)$	The degree of node u
$\Gamma(u)$	The structural neighborhood of node u
$UN(u\|v)$	The unique neighbors of node u compared to v
$CN(u,v)$	Common neighbors of node u and v
$sim(u,v)$	Structural similarity between node u and v
$d(u,v)$	Jaccard distance of node u and v
$LS(u)$	The leadership of node u
$EC(u,v)$	The edge compactness of pair (u,v)
$f(u \leftarrow v)$	The attractive force exerted on node v by u

In this paper, we provide a novel algorithm to investigate the community structure in a chained mode, which is not only able to detect communities, but also identifies leaders of each cluster.

3. Leader-aware Community Detection Algorithm

In this section, we firstly introduce some necessary concepts and definitions that will be used later. The problem that we need solve is to detect non-overlapping community structures and their leaders in networks. For convenience, we exploit unweighted and undirected graphs as the research object.

3.1. Preliminaries

Given a target graph $G(V, E)$, where V is the set of nodes and E is the set of edges. Table 1 lists some main notations and their definitions.

Definition 1 (Structural Neighborhood). *Given an undirected graph $G = (V, E)$ and a node $u \in V$, the structural neighborhood of node u is the node set $\Gamma(u)$ containing itself and its adjacent nodes which have a connection with node u.*

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

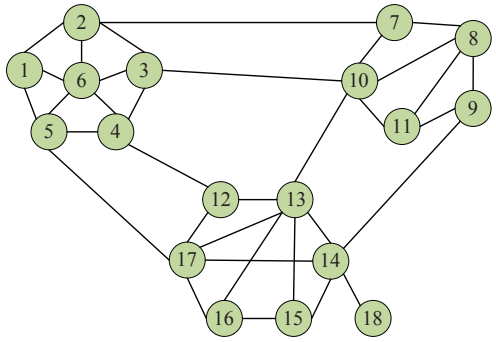


Figure 1: A toy network which contains three communities.

Definition 2 (Unique Neighbors). *For an edge $(u, v) \in E$ of a graph $G = (V, E)$, the unique neighbors of node u are the nodes that only link to u while having no connection with node v, which can be denoted as:*

$$UN(u|v) = \Gamma(u) - CN(u, v), \tag{2}$$

where

$$CN(u, v) = \Gamma(u) \cap \Gamma(v). \tag{3}$$

$CN(u, v)$ is the common neighbors of the two nodes u and v.

As shown in Fig. 1, the unique neighbors $UN(12|13) = \{4\}$, while $UN(13|12) = \{10, 14, 15, 16\}$.

Definition 3 (Jaccard Similarity). *Given an undirected graph $G = (V, E)$, the Jaccard similarity of node u and v is:*

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

Based on Definition 3, the Jaccard distance between two adjacent nodes can be calculated.

Definition 4 (Jaccard Distance). *The Jaccard distance of two nodes u and v in a graph $G = (V, E)$ is defined as:*

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

Naturally, the higher the similarity, the shorter the distance.

3.2. Leader-aware Community Detection Algorithm

Our leader-aware community detection algorithm Autoleader is introduced in this section. The basic idea is that each node tends to attach to its local leader, building up a forest which contains at least one dependence tree. Then a merging step is taken to merge some tiny branches. Finally, each tree represents a community and the root of tree is the leader node of the corresponding cluster. The first thing that we need to do is to identify the local leaders.

3.2.1. Identifying Local Leaders

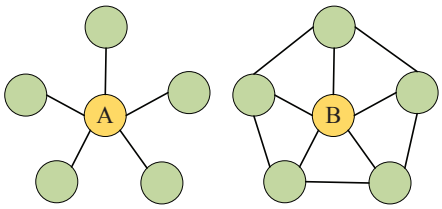
The first phase aims to identify local leader of each node from its neighborhood. In order to detect community structure as well as the community leaders, a novel measurement is proposed to quantify the leadership of nodes in social networks.

Definition 5 (Leadership). *Given a node $u \in V$ in an undirected graph $G = (V, E)$, the leadership of node u is defined as:*

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

According to Definition 5, the level of the leadership of nodes depends on the number of its neighbors as well as the extent of intimacy between itself and the neighbors. This is also reasonable in real-world scenarios. Someone who has many close connections with others is more likely to be a leader. Just considering the degree of node can not reflect its leadership. For example, in Fig. 2, the degree of nodes A and B is the same, but node B has the larger leadership since the connections between itself and its neighbors are denser, meaning that node B has greater influence.

Leadership is used to measure the influence of node in a local perspective. It decides whether one node can be a local leader of its neighbor to a great extent. For any pair of nodes that are connected, the leader-follower relationship can be settled if their leaderships are known. For an edge $(u, v) \in E$ in graph G, if $LS(u) > LS(v)$, node v can not be the leader of node u. Since $LS(u) > LS(v)$, we can know from Definition 5 that the whole connections between node u and its neighbors are denser than that of node v, which means that the influence of node u is greater than v in their local region. Thus node v can not be the leader of node u.



(a) Leader A (b) Leader B

Figure 2: Two subgraphs where their central nodes have the same degree but different leaderships.

The rule implies that having larger leadership is one of the prerequisites of being a leader with respect to another linked node. For one node u, in order to identify its local leader among its structural neighborhood $\Gamma(u)$, the strength of the link should also be taken into account. There may exist several candidate nodes whose leadership is larger than $LS(u)$. Considering the structure of the network is complex, an edge (u, v) may be a bridge which locates between two communities, i.e. node u is maybe in the border of a cluster, while node v belongs to another cluster. In this case, node v can not be the leader of node u even though $LS(v) > LS(u)$. So it is important to make sure that the node and its potential local leader are in the same cluster. The concept of edge compactness is proposed to deal with this problem.

Definition 6 (Edge Compactness). *The edge compactness of an edge* $(u, v) \in E$ *in graph* $G = (V, E)$ *measures the closeness between the two endpoints. There are three patterns that need to be considered.*

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

where

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

The first term is easily understandable since it measures the similarity of the two nodes. The second term emphasizes the importance of common neighbors of two endpoints, which strengthen their connection. The situation is more complex in the third term. If the similarity between one unique neighbor and another endpoint is smaller than the threshold, it leads to

opposite effect. Otherwise, it is the same as the second case. The parameter λ ranges from 0 to 1, which determines the direction of the interaction force caused by these unique neighbors: closing or moving far away [24]. We will discuss the parameter further in the section of experiments.

For nodes u and v, if $EC(u,v) \geq 0$, we can know from Definition 6 that $sim(u,v)+\sum_{t\in CN(u,v)} sim(u,t)*sim(t,v)+\sum_{t\in UN(u|v)} sim(u,t)*\rho(t,v) \geq 0$, in which $\rho(t,v) < 0$ under the condition of $sim(t,v) < \lambda$. Then $sim(u,v) + \sum_{t\in CN(u,v)} sim(u,t) * sim(t,v) + \sum_{t\in UN(u|v),sim(t,v)\geq\lambda} sim(u,t) * sim(t,v) \geq \sum_{t\in UN(u|v),sim(t,v)<\lambda} sim(u,t) * (\lambda - sim(t,v))$. The left term of the inequality denotes the cohesive force between node u and v, while the right term represents the repulsive force. The cohesive force is larger than repulsive force, thus node u and v are in the same cluster. So we can see that judging the edge compactness is essential when determining the cluster belongings. The requirements of leadership and edge compactness can filter reasonable potential leader nodes. From these candidates, we make further efforts to identify leader of each node.

Based on the leadership, the attractive force between two connected nodes can be counted out, which is derived from the physical conceptions. In physics, the force between two objects is related to their mass and the distance. Similarly, the attractive force in social networks is proportional to the leadership and in inverse proportional to the square of distance. Besides, considering that the degree also plays an important role in social organization, we take the degree into account.

Definition 7 (Attractive Force). *For a node v and its adjacent node u in a graph $G = (V, E)$, the attractive force exerted on node v caused by node u is:*

$$f(u \leftarrow v) = \frac{deg(u)}{deg(v)} \cdot \frac{LS(u)}{d(u,v)^2}. \tag{9}$$

Note that attractive force is directed, thus $f(u \leftarrow v)$ may be not equal to $f(v \leftarrow u)$. Based on the definitions above, the local leader of each node can be determined now.

Definition 8 (Local Leader). *For a node v in graph $G = (V, E)$, the local leader that the node v chooses to follow is defined as:*

$$locleader(v) = \{u \in V | \underset{u\in\Gamma(v)\wedge LS(u)>LS(v)\wedge EC(v,u)\geq 0}{\arg\max} f(u \leftarrow v)\}. \tag{10}$$

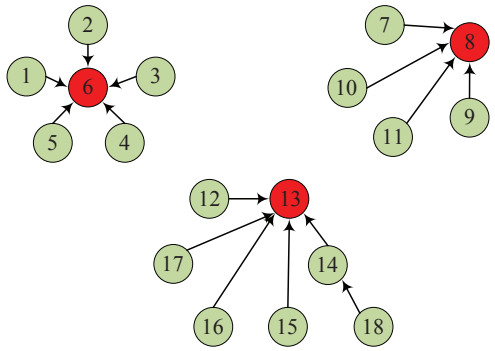


Figure 3: Three dependence trees after each node attaching to its local leader. Nodes 6, 8 and 13 are the root nodes, which are also the leaders of communities.

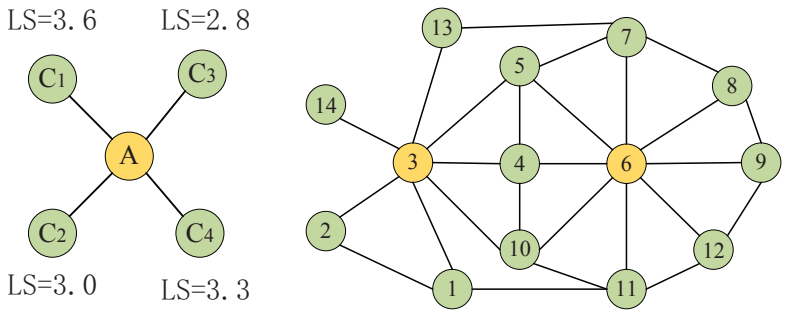
For one node v, in order to identify its local leader from v's neighborhood $\Gamma(v)$, we need to consider three aspects. Firstly, we can know that its local leader u should satisfy $LS(u) > LS(v)$. Secondly, the leader-follower relationship should be in the same cluster according to the discussion above, thus we can get $EC(u, v) \geq 0$. Finally, qualified leader candidates are selected after above two steps. We choose the one with the largest attractive force on node v as v's local leader. For example, in Fig. 1, the local leader of node 4 is node 6, because node 6 has the largest leadership value in node 4's neighborhood and $EC(4, 6) > 0$.

3.2.2. Building Dependence Trees

Since the local leader of each node can be settled according to Definition 8, the initial clustering structures emerge after scanning all nodes of the network. Each node has a pointer pointing to its local leader in a cascade way, the same is true for the local leader. Then we can get the original dependence trees.

Definition 9 (Dependence Tree). *A dependence tree in a graph $G = (V, E)$ is composed of nodes and edges, where each node in the tree just has one exit link pointing to its local leader except for the root of the tree.*

As shown in Fig. 3, there are three dependence trees obtained from Fig. 1 after scanning all nodes. For example, node 18 points to its local leader 14, while node 14 attaches to its local leader node 13. Node 13 cannot find its local leader, thus it becomes an initial leader node.



(a) Isolate node (b) Separated leaders

Figure 4: Two conditions of merging.

We can come to a conclusion that if $locleader(u) = v$ and $locleader(v) = w$, then node u, v and w are in the same cluster. According to Definition 8, we can know that node u and v are in the same cluster if $locleader(u) = v$, because $EC(u, v) \geq 0$. Similarly, node v and w are in the same cluster. Since the network is partitioned into disjoint communities, node u, v and w are surely in the same cluster.

The discussion implies that all nodes in a dependence tree are in the same cluster. When the process on all nodes ends up, the preliminary clustering results emerge. We can see that each tree represents a cluster. In Fig. 1, red nodes 6, 8 and 13 are root nodes, which are also the leaders of corresponding communities. Furthermore, the structure of dependence tree clearly reveals the process of community formation. However, in reality, there are usually many noise edges in the network, thus a merging phase is necessary. The pseudo code of our algorithm Autoleader is described in Algorithm 1.

3.2.3. Merging Branches

After building all the dependence trees, the initial community structure is uncovered. However, there may exist some tiny branches that are incorrectly separated out since we just utilize the local information to determine the belongings of each node. Thus, merging is necessary to achieve more accurate results especially when the network structure is very complicated.

Since each node has chosen its local leader, we just need to consider these leader nodes of trees. There are two conditions. If the leader does not have any followers (i.e. it is an isolate node), let the leader attach to the one with the maximum leadership in its neighbors. For example, in Fig. 4(a), node A has no followers, so we focus on its neighbors and select

Algorithm 1 Autoleader algorithm

Input: Network $G = (V, E)$, parameter λ
Output: Set of communities C and leaders

for each edge $e = (u, v) \in E$ **do**
  $G.edge[u, v].link = 0$;
  compute the Jaccard similarity $sim(u, v)$ and distance $d(u, v)$;
end for
for each node $v \in V$ **do**
  compute the leadership $LS(v)$;
end for
leaderList=[];
for each $v \in V$ **do**
    for each node $u \in \Gamma(v)$ **do**
      compute the attractive force $f(u \leftarrow v)$ and the edge compactness $EC(u, v)$;
    end for
  compute the local leader $locleader(v)$;
  if $locleader(v)$ exists **then**
    $v.leader = locleader(v)$, $G.edge[v, locleader(v)].link = 1$;
  else
    $leaderList.add(v)$;
  end if
end for
$merging(G, leaderList)$;
for each edge $e = (u, v) \in E$ **do**
  if $G.edge[u, v].link = 0$ **then**
    remove the edge e from G;
  end if
end for
find the communities (components) of G;
return C and $leaderList$

the one with the maximum leadership like C_1. Here we just consider non-overlapping communities, so simply choosing one according to the leadership is reasonable.

For the second case, the solution is a little complicated. In fact, this kind of situation is more common. If there is a community formed around a leader, while a local dense subgraph is remote from the leader, the small subgraph tends to be regarded as a separate cluster. For instance, Fig. 4(b) plots a toy network. After the former steps, we can obtain two initial clusters $\{1, 2, 3, 13, 14\}$ and $\{4, 5, 6, 7, 8, 9, 10, 11, 12\}$, where node 3 and 6 are leaders of responding communities. However, in reality, the two small clusters should be merged into one community.

Since some nodes can not find their local leaders based on leadership and attractive force, an alternative method which is similar to voting is exploited by collecting the information of their neighborhood. Considering there are many noise in the network, we should only focus on the nodes that meet the condition of similarity from the neighbors. The definition of candidate set is defined as follows to identify these qualified nodes.

Definition 10 (Candidate Set). *The candidate set of a node v is the subset of its neighbors that satisfies the similarity constriction:*

$$\mathcal{C}(v) = \{u | u \in \Gamma(v), sim(v, u) >= \overline{sim(v)}\}, \tag{11}$$

where

$$\overline{sim(v)} = \frac{1}{|\Gamma(v)|} \sum_{u \in \Gamma(v)} sim(v, u). \tag{12}$$

Based on the candidate set, we need to select from the candidates to determine whether an initial leader node should be updated. For one leader node v, by scanning the local leaders of its candidate set $\mathcal{C}(v)$, we can count the weights of unique local leaders of these candidates. The weight of a unique leader l equals the sum of similarity of the node v and the candidate whose local leader is exactly l. Then we can sort them in descending order and select the leader p with the maximum weight. Finally, node v can randomly select one node u from the candidate set as its updated local leader so long as the local leader of u is p. The formal definition is presented in the following.

Definition 11 (Updated Local Leader). *For one leader node v, its updated local leader is defined as:*

$$UpdLeader(v) = \{u \in \mathcal{C}(v) | \arg\max \sum_{m \in \mathcal{C}(v) \wedge locleader(m) = locleader(u)} sim(v, m)\}. \tag{13}$$

For the case in Fig. 4(b), we only need to consider leader nodes 3 and 6 in the merging phase. For leader 3, its candidate set is $\{1, 2, 4, 5, 10\}$ because of the similarity limitation. Candidate nodes 4, 5 and 10 are all followers of local leader 6, thus the weight of 6 equals to the sum of similarities of $sim(3, 4)$, $sim(3, 5)$ and $sim(3, 10)$. Without doubt, local leader 6 is the one with the maximum weight based on the candidate set of node 3. Thus node 3 need to update its own local leader. It can randomly select one from nodes $\{4, 5, 10\}$ to follow, resulting in the cohesion of small branches. For leader 6, we can find that itself is the one with maximum weight, meaning that there is no need to change the leader role of node 6. The procedure in detail of merging is illustrated in Algorithm 2.

3.3. Property Analysis

Our algorithm exploits the conception of local leader, which is used to build dependence trees step by step. There are some particular properties that help us understand the algorithm better.

Property 1. ***Order-independency****. The result of our algorithm on a graph G is unique and irrelevant to the order of visit of nodes in G.*

For each node u in graph G, its local leader $locleader(u)$ is unique. So the final community structure is uniquely determined by the dependence trees. The order of nodes for visiting has no impact on the results. The order-independency property enables Autoleader algorithm to be parallelizable because only local structural information is needed to identify local leaders. And it can be easily combined in a MapReduce framework.

Property 2. ***Fast inquiry****. Given a graph $G(V, E)$ and two nodes u and v, our algorithm Autoleader can quickly confirm whether these two nodes are in the same cluster without uncovering all community structure.*

Algorithm 2 Merging function

Input: Network $G(V, E)$, initial set of leader nodes L
Output: Updated leader nodes L

for each leader $l \in L$ **do**
  if $l.numberOfFollowers = 0$ **then**
    $l.leader = \arg\max_{m \in \Gamma(l)} LS(m)$, $L.remove(l)$;
  else
    compute candidate set $\mathcal{C}(l)$;
    dictionary $dic = \{\}$;
    for each node $u \in \mathcal{C}(l)$ **do**
      if $locleader(u)$ not in dic **then**
        $dic[locleader(u)] = sim(u, l)$;
      else
        $dic[locleader(u)] += sim(u, l)$;
      end if
    end for
    sort the dictionary by the value in descending order;
    $newLeader$ = the first key of dic;
    if $newLeader != l$ **then**
      for each node $u \in \Gamma(l)$ **do**
        if $locleader(u) = newLeader$ **then**
          $l.leader = u$, $G.edge[l, u].link = 1$, $L.remove(l)$;
          break;
        end if
      end for
    end if
  end if
end for
return $\mathcal{C}$

What we need to do is just starting from the two nodes u and v respectively, follow their local leader pointers until reaching the roots. If their roots are the same one, there's no doubt that the two nodes are in the same cluster. If not, nodes u and v cannot be the members of a group at the same time. By utilizing this method, we can quickly identify the community feature of different people in social networks without dividing the whole graph.

3.4. Time Complexity

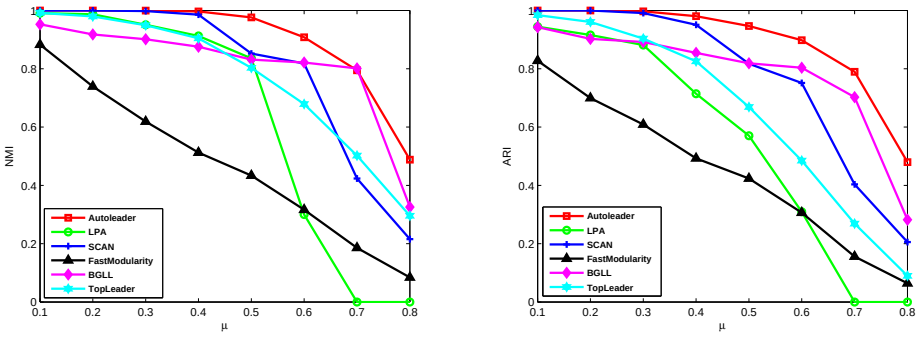
Let us suppose that there is a network with n nodes and m edges. Firstly, the similarity of any two linked nodes is required, and the time complexity is $\mathcal{O}(m)$ (Algorithm 1 (Line $1-4$)). We also need to compute the leadership of each node (see Line $5-7$), whose time computation is $\mathcal{O}(n)$. During the process of building up dependence trees, each node has to choose its local leader, thus the time complexity is $\mathcal{O}(n)$.

Assume that the number of initial leader nodes is n_l, dealing with merging phase needs $\mathcal{O}(k * n_l)$, where k is approximately the average number of candidate set. Besides, the number of leaders n_l is much smaller than the number of nodes n. In total, the worst time complexity of our algorithm is $\mathcal{O}(n+m)$.

4. Experiments

In this section, we evaluate the performance of our algorithm on both synthetic and real-world datasets compared with several representative community detection methods: TopLeader [6], LPA [22], SCAN [9], FastModularity [11] and Louvain [12]. Since TopLeader needs the number of output communities as the input, we set the parameter as the number of communities our algorithm output for fair comparison. For Louvain, default parameters are used to do experiments. For others, the best results are chosen for comparison. For Autoleader, the parameter λ is set as 0.5. All experiments are performed on a workstation with 3.3 GHz CPU and 32.0 GB RAM.

Evaluation Metrics. To quantitatively compare the results of different community detection methods, we adopt several measures that typically employed for clustering evaluations. When the ground truth of the network is available, the performance is measured by Normalized Mutual Information (NMI) [27], Adjusted Rand Index (ARI) [28] and cluster purity. If the community structure of networks is unknown, the popular measure modularity [3] is applied.



(a) NMI (b) ARI

Figure 5: NMI and ARI performance on the LFR benchmark networks.

4.1. Synthetic Networks

In this section, we generate some synthetic datasets to evaluate the performance of our algorithm. The LFR benchmark networks [29] are applied for fair comparison. We can tune the mixing parameter μ to control the difficulty of community separation.

We generate several synthetic networks containing 10000 nodes. For each type of graphs, the average degree and community size are fixed, while the mixing parameter μ ranges from 0.1 to 0.8. The NMI value and ARI performance of different algorithms is shown in Fig. 5(a) and Fig. 5(b) respectively.

From the figure, we can observe that SCAN and Autoleader achieve excellent clustering results when the mixing parameter is not greater than 0.4. With the increase of parameter μ, their performance begins to decrease. FastModularity performs worst in most cases. This is because FastModularity is based on modularity optimization and tends to find large communities, which leads to the problem of resolution limit. TopLeader performs better than FastModularity because it need predefined number of communities as the input. For networks with groundtruth, it can obtain good results as shown in Fig.5. But in fact, the number of communities is usually unknown for real-world networks, which affects the performance of TopLeader to a large extent.

Besides, LPA cannot find community structure when the network contains many noises. So the NMI and ARI value of LPA are equal to 0 when the parameter μ is greater than 0.7. For BGLL, its performance decreases slowly with the increase of mixing parameter value. But when the network becomes

Table 2: Real-world datasets and their statistics

Dataset	$\|V\|$	$\|E\|$	$\#Clusters$	$Avadegree$	$Coefficient$
Zachary	34	78	2	4.588	0.571
Football	115	613	12	10.661	0.403
Polbooks	105	441	3	8.400	0.488
Collaboration	9877	25998	–	5.260	0.312
Friendship	58228	214078	–	7.353	0.172
Road	1088092	1541898	–	2.834	0.047
DBLP	28702	66832	–	4.914	0.641

very difficult to partition, the performance of BGLL decreases dramatically. As for the SCAN method, it also suffers from the problem of noise edges. In addition, the two user-defined parameters of SCAN are inconvenient to tune in order to get better results. Our algorithm Autoleader performs better than SCAN, FastModalrity and BGLL.

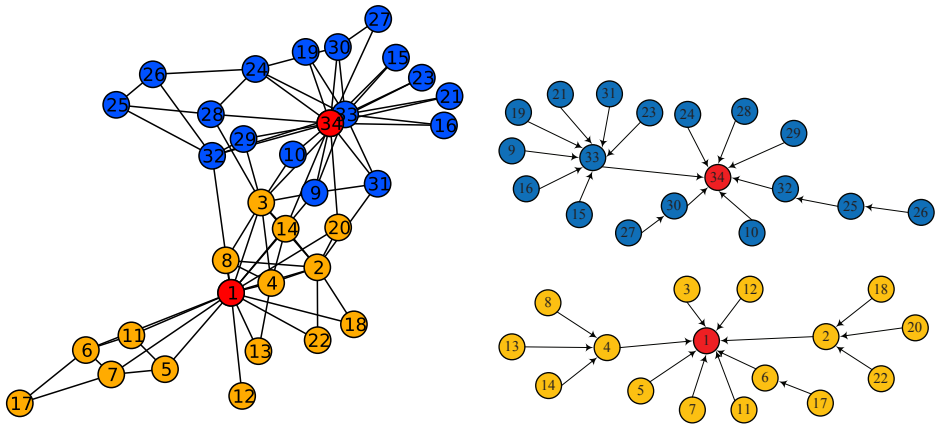
4.2. Real World Networks

We choose three frequently used real-world networks that contain ground truth for experiments. The measures like NMI, ARI and Purity are used to evaluate the accuracy of these methods. The statistics of these networks are summarized in Table 2, containing the number of nodes, the number of edges, the number of clusters, the average degree and the clustering coefficient. The computational formula of the clustering coefficient is as follows:

$$Coefficient = \frac{3 \times number\ of\ triangles}{number\ of\ connected\ triples\ of\ vertices}. \tag{14}$$

Zachary's karate club: This is a famous network describing the friend relationship among members of a karate club, which is derived by Zachary's observation [30]. The network can be divided into two partitions because the disagreement between the administrator and the instructor.

Fig. 6(a) illustrates the structure of karate network and the clustering result obtained by Autoleader. We can see that each color represents a cluster. Two groups are successfully found except one member is partitioned into wrong community (node '9'). In fact, it is difficult to judge its identification because node 9 is located between two partitions. The red nodes 1 and 34 are leaders of the corresponding communities, which is in accordance with



(a) Community structures detected (b) Corresponding dependence trees

Figure 6: The performance of Autoleader on the karate network.

Table 3: The performance of different algorithms on real-world networks with ground truth.

	Zarachy			Football			Polbooks		
	NMI	ARI	Purity	NMI	ARI	Purity	NMI	ARI	Purity
Autoleader	**0.837**	**0.882**	**0.971**	0.902	0.814	0.835	0.553	0.664	0.810
TopLeader	**0.837**	**0.882**	**0.971**	0.786	0.669	0.783	**0.556**	**0.665**	**0.829**
LPA	0.222	0.080	0.823	0.567	0.252	0.808	0.523	0.610	0.757
SCAN	0.440	0.249	0.471	**0.917**	**0.856**	**0.913**	0.536	0.587	0.800
FastModularity	0.565	0.568	0.706	0.762	0.536	0.904	0.531	0.638	0.829
BGLL	0.490	0.392	0.618	0.890	0.807	0.870	0.512	0.558	0.724

the real-world scenario since 1 and 34 are the administer and the instructor, respectively. The performance of other algorithms for comparison is shown in Table 3. From the results in NMI, ARI and Purity, it is observed that Autoleader outperforms other methods in all respects. Since TopLeader needs user-defined number of communities as the input, it obtains the same result as our method.Other approaches do not perform very well, since many nodes are wrongly grouped even by BGLL.

Fig. 6(b) shows the final dependence trees in karate network. We can clearly see the clustering process. Nodes 1 and 34 are roots of the corresponding trees apparently because they all have many followers around their neighborhood. From the structure, it is also known that node 33 also plays an important role since it has many followers. Combining the network structure in Fig. 6(a), we can find that nodes 33 and 34 both have significant influence inside their group.

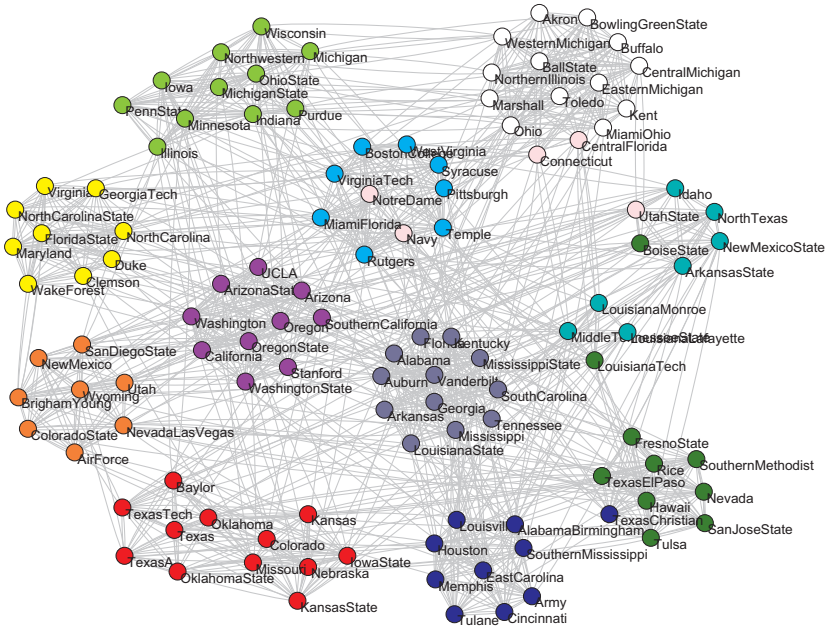


Figure 7: The performance of Autoleader on the football network.

American college football: The network is derived from the American football games, which contains 115 nodes and 613 edges. Nodes represent the teams and edges represent games between the two teams that connect. The real community structure of the network is already known [31]. The teams usually are divided into twelve conferences, containing eleven conferences and five independent teams.

The clustering result detected by our algorithm Autoleader is presented in Fig. 7. All 12 conferences in reality are represented by different colors while the communities detected are regarded as disjoint groups. Five independent conferences are denoted by pink circles. We can see that most conferences are correctly identified. Some teams are wrongly ploted because they are divided into groups with which they are most closely linked. For example, five independent teams are grouped into different conferences according to their association degree. The drawback of our clustering result is that the Big Twelve conference is divided into two parts where each part contains six teams (see red circles). This condition may be caused by the situation that the inner connection of each part is more denser than that between two parts. From Table 3, we can observe that LPA obtains the worst results. Compared

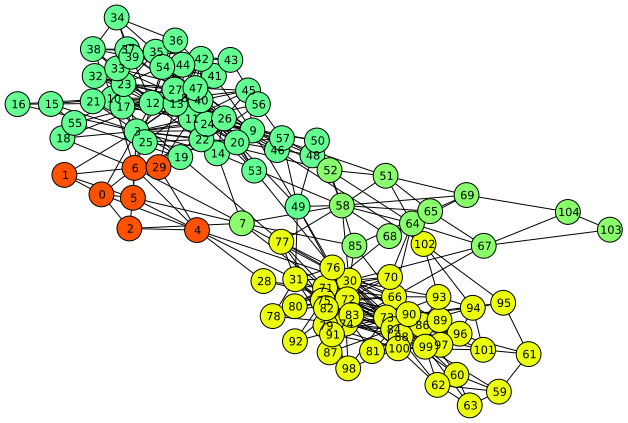


Figure 8: The performance of Autoleader on the network of political books.

with FastModularity and BGLL, our algorithm Autoleader behaves well in the aspect of NMI.

Books about US politics: This is a network of books about US politics, where nodes represent books sold by the online bookseller Amazon.com and edges represent frequent copurchasing of books by the same buyers. Each node has a label to indicate its political inclination: "liberal", "neutral" or "conservative".

Fig. 8 shows the clustering results of Autoleader on the network of political books. Autoleader partitions these books into four categories. The two communities that represent the liberal and conservative books are well identified. But the neutral ones are difficult to distinguish. According to the performance in Table 3, other methods produce relatively bad clustering results on this network except for TopLeader. This is because TopLeader gets the right number of communities as its input parameter, which is also the greatest defect of TopLeader.

We also choose three large real-world networks without class information for comparison. Modularity is used to evaluate the performance of these algorithms. Note that FastModularity is unable to deal with networks that contain more than one components, so we ignore its results. The clustering results of all these comparing methods are shown in Table 4. For TopLeader, we use the number of communities of Autoleader as the input of TopLeader.

Hepth collaboration network: This is a collaboration network of Arxiv high energy physics theory. Autoleader can find 1173 communities and the modularity equals 0.683. In terms of the modularity measure, BGLL achieves

Table 4: The performance of different algorithms on real-world networks without ground truth.

	Collaboration		Friendship		Road	
	#cluster	mod.	#cluster	mod.	#cluster	mod.
Autoleader	1173	0.683	2794	0.613	150931	0.869
TopLeader	1173	0.506	2794	0.430	150931	–
LPA	1264	0.656	3590	0.589	142603	0.745
SCAN	1546	0.508	805	0.295	110085	0.618
BGLL	475	0.768	746	0.684	492	0.989

better performance than Autoleader. This is because the two algorithms tend to detect few communities. In fact, the collaboration network contains 429 connected components in all, where each connected component can be partitioned into many clusters. However, we can observe that BGLL just finds 461 and 475 clusters. The number of communities identified is merely a little larger than the number of connected components, which results in the lack of many meaningful communities. Besides, SCAN and LPA obtain worse performance compared with Autoleader. For TopLeader, it gets the worst results among these methods.

Brightkite friendship network: The network is a Brightkite location based online social network contains $58,228$ nodes and $214,078$ edges. There are 547 connected components in the friendship network. From Table 4, we can see that BGLL achieves the highest value in modularity but the fewest communities. Autoleader finds 2794 clusters and outperforms the other methods except BGLL in terms of modularity. Since BGLL is an approach based on modularity optimization, the high value in modularity is reasonable. However, SCAN obtains the worst result, which maybe caused by the parameter and the structure of the network. TopLeader obtains better performance than SCAN, but worse than other approaches.

Pennsylvania road network: This is a network of road structure consisting of $1,088,092$ nodes and $1,541,898$ edges, where nodes represent intersections and edges represent roads connecting the intersections. TopLeader costs more than 10 hours but still cannot solve the problem, so here we neglect its results in road network. BGLL just gets 492 communities, while the road network has 206 connected components. Furthermore, it obtains without doubt the best result in the measure of modularity. However, considering the scale of the network, the number of communities that BGLL

Table 5: Five communities detected by Autoleader on DBLP network.

Community 1	Community 2	Community 3	Community 4	Community 5
Philip S. Yu	Jiawei Han	C. Lee Giles	Nick Craswell	Tao Li
Kevin Drummey	Kyuhyung Lee	Carl Lagoze	Hengzhi Wu	Heng Huang
James Z. Teng	Xifeng Yan	William Browuer	Brett Matson	Charles Perng
John Turek	Hong Cheng	Vivek Bhatnagar	Paul Thomas	Steven Loscalzo
Haixun Wang	Jing Gao	Kushal Dave	Steve Walker	Chris Ding
Kirsten Hildrum	Hector Gonzalez	Xiang Gao	Julie A. McCann	Charles Perng
...	...	...	...	...

finds is too small. Autoleader identifies 150931 communities and achieves better performance than the other methods like SCAN and LPA.

4.3. Case Study

In order to present the leader-aware property of our algorithm Autoleader, we extract a network from DBLP dataset. The network consists of $28,702$ nodes and $66,832$ edges, where authors are distributed mainly in four research fields (DB, IR, DM and ML).

Our algorithm Autoleader identifies 3631 communities in this network. Here we just select five representative communities from different research fields and list six members of each cluster in Table 5. Leaders of each community are presented in the second row. From Table 5, we can see that Philip S. Yu and Jiawei Han are both experts in the field of data mining, which is in accordance with practical situations. Besides, Community 3 consists of scholars in the field of database. Nick Craswell is the leader of one community in the field of information retrieval. And Tao Li leads a community that focuses on machine learning.

Fig. 9 plots the dependence tree of Community 2 whose leader is Jiawei Han. From the figure, we can clearly see that Jiawei Han is located in the central position. There are many scholars gathering around him. What's more, from the hierarchical structure, Xifeng Yan and Hong Cheng can also be regarded as influential experts in this field because they are local leaders in small-scaled scope.

4.4. Input Parameter

A parameter λ is introduced in our algorithm Autoleader. In order to analyze the influence of the parameter, we conduct experiments on the synthetic networks with the parameter ranges from 0 to 1. Fig. 10 shows the number of communities with different λ value. To compare the influence of

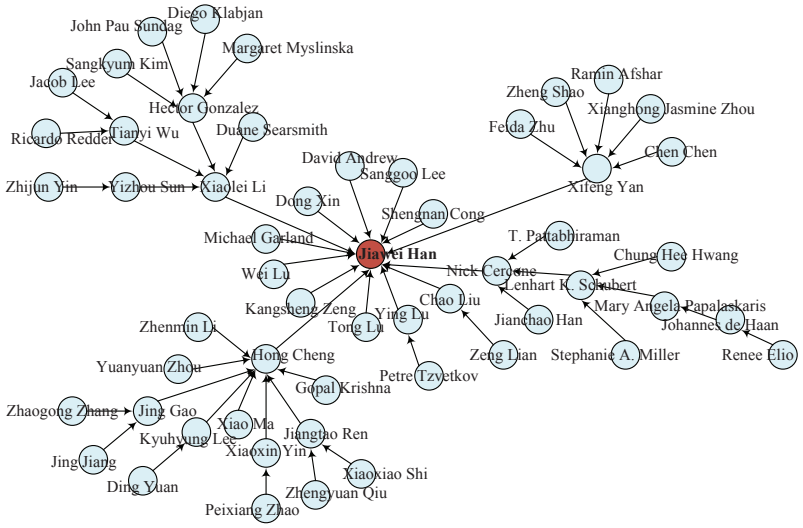


Figure 9: The dependence tree of the Community 2 in Table 5, where the leader is Jiawei Han.

mixing parameter μ, we adopt two kinds of synthetic networks of the same size but with different $\mu = 0.1$ and $\mu = 0.4$, respectively. From the plot, we can see that Autoleader identifies more communities with the increasing of parameter λ value. When $\lambda = 0$, there are only very few communities detected, illustrating that judging the strength of each dependence link is especially essential.

Besides, from the trend of networks with different mixing parameters, we find that the number of communities keeps invariable no matter what the parameter λ value is when $\mu = 0.1$. It means that the parameter λ does not have any influence on the performance when the community structure is easy to separate. But when $\mu = 0.4$, the number of communities increases slowly with the increasing of λ value. On the range $0.1 - 0.5$ of λ, the results of Autoleader can be seen stable. So based on the analysis, Autoleader is not sensitive to the value of parameter λ. A λ value between 0.1 to 0.5 is sufficient to achieve a good clustering result.

4.5. Runtime

To assess the scalability of Autoleader, we generate several benchmark networks with different node sizes ranging from $1,000$ to $500,000$ while the average degree k is 20. Fig. 11 shows the running time of different algorithms

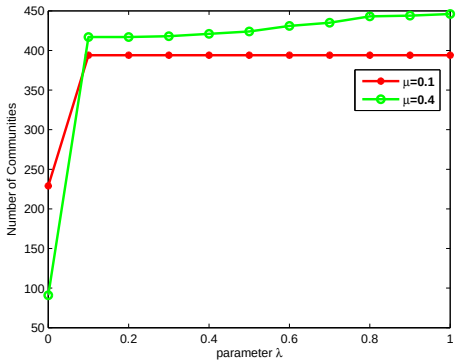


Figure 10: The sensitivity of parameter λ during the process of community detection.

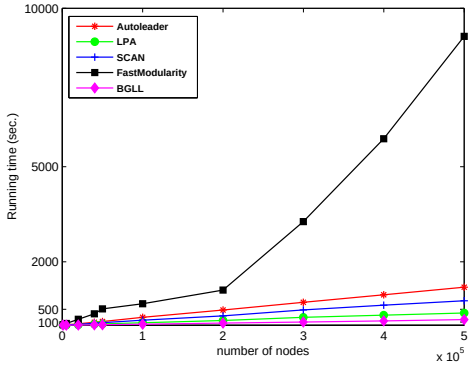


Figure 11: The running time of different algorithms.

on the benchmark networks. Since different algorithms are implemented in different programming language, we should focus on their increasing tendencies. The time complexity of TopLeader is very high so we do not draw its curve. It is observed that our algorithm Autoleader is faster than FastModularity because its time complexity is linear with the number of nodes. However, Autoleader is slower than BGLL. Although SCAN seems faster than Autoleader, their time complexity is identical. Besides, Autoleader obtains better performance in the quality of results.

5. Conclusion

In this paper, we introduce a new algorithm Autoleader, to automatically detect community structure as well as cluster leaders in social networks. A novel metric to evaluate the attractive force between two linked nodes is designed. Based on that, each node searches for its local leader and building up dependence trees. Experimental results on synthetic and real-world networks demonstrate that Autoleader can detect communities with high quality compared to state-of-the-art methods. Besides, community leaders can be identified at the same time. In the future, we plan to study the parallelization of our algorithm on large-scale networks.

Acknowledgments

The work was supported in part by the National Science Foundation of China grants 61602354, 61672417 and 61472299, the Fundamental Research Funds for the Central Universities of China. Any opinions, findings and conclusions expressed here are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] M. E. Newman, M. Girvan, Finding and evaluating community structure in networks, Physical review E 69 (2) (2004) 026113.

[2] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on pattern analysis and machine intelligence 22 (8) (2000) 888–905.

[3] M. E. Newman, Modularity and community structure in networks, Proceedings of the national academy of sciences 103 (23) (2006) 8577–8582.

[4] L. Lü, Y.-C. Zhang, C. H. Yeung, T. Zhou, Leaders in social networks, the delicious case, PloS one 6 (6) (2011) e21202.

[5] C. Froome, N. Keys, D. C. Thomsen, T. F. Smith, Opinion leaders and complex sustainability issues, Management of Environmental Quality: An International Journal 21 (2) (2010) 187–197.

[6] R. R. Khorasgani, J. Chen, O. R. Zaiane, Top leaders community detection approach in information networks, in: 4th SNA-KDD workshop on social network mining and analysis, Citeseer, 2010.

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

[8] S. White, P. Smyth, A spectral clustering approach to finding communities in graphs, in: Proceedings of the 2005 SIAM international conference on data mining, SIAM, 2005, pp. 274–285.

[9] X. Xu, N. Yuruk, Z. Feng, T. A. Schweiger, Scan: a structural clustering algorithm for networks, in: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2007, pp. 824–833.

[10] S. Fortunato, Community detection in graphs, Physics reports 486 (3) (2010) 75–174.

[11] A. Clauset, M. E. Newman, C. Moore, Finding community structure in very large networks, Physical review E 70 (6) (2004) 066111.

[12] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of statistical mechanics: theory and experiment 2008 (10) (2008) P10008.

[13] R. Guimera, L. A. N. Amaral, Functional cartography of complex metabolic networks, Nature 433 (7028) (2005) 895–900.

[14] L. Yang, X. Cao, D. He, C. Wang, X. Wang, W. Zhang, Modularity based community detection with deep learning, in: Proceedings of the 25th International Joint Conference on Artificial Intelligence, 2016, pp. 2252–2258.

[15] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise., in: Kdd, Vol. 96, 1996, pp. 226–231.

[16] H. Sun, J. Huang, J. Han, H. Deng, P. Zhao, B. Feng, gskeletonclu: Density-based network clustering via structure-connected tree division

or agglomeration, in: Data Mining (ICDM), 2010 IEEE 10th International Conference on, IEEE, 2010, pp. 481–490.

[17] H. Shiokawa, Y. Fujiwara, M. Onizuka, Scan++: efficient algorithm for finding clusters, hubs and outliers on large-scale graphs, Proceedings of the VLDB Endowment 8 (11) (2015) 1178–1189.

[18] A. Mahmood, M. Small, S. A. Al-Maadeed, N. Rajpoot, Using geodesic space density gradients for network community detection, IEEE Transactions on Knowledge and Data Engineering 29 (4) (2017) 921–935.

[19] W. Gao, W. Luo, C. Bu, Adapting the topleaders algorithm for dynamic social networks, The Journal of Supercomputing (2017) 1–23.

[20] N. A. Helal, R. M. Ismail, N. L. Badr, M. G. Mostafa, Leader-based community detection algorithm for social networks, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery.

[21] D. Shah, T. Zaman, Community detection in networks: The leader-follower algorithm, arXiv preprint arXiv:1011.0774.

[22] U. N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Physical review E 76 (3) (2007) 036106.

[23] M. Rosvall, C. T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proceedings of the National Academy of Sciences 105 (4) (2008) 1118–1123.

[24] J. Shao, Z. Han, Q. Yang, T. Zhou, Community detection based on distance dynamics, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 1075–1084.

[25] A. Capocci, V. D. Servedio, G. Caldarelli, F. Colaiori, Detecting communities in large networks, Physica A: Statistical Mechanics and its Applications 352 (2) (2005) 669–676.

[26] I. Simonsen, Diffusion and networks: A powerful combination!, Physica A: Statistical Mechanics and its Applications 357 (2) (2005) 317–330.

[27] A. Strehl, J. Ghosh, Cluster ensembles—a knowledge reuse framework for combining multiple partitions, Journal of machine learning research 3 (Dec) (2002) 583–617.

[28] W. M. Rand, Objective criteria for the evaluation of clustering methods, Journal of the American Statistical association 66 (336) (1971) 846–850.

[29] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Physical review E 78 (4) (2008) 046110.

[30] W. W. Zachary, An information flow model for conflict and fission in small groups, Journal of anthropological research 33 (4) (1977) 452–473.

[31] M. Girvan, M. E. Newman, Community structure in social and biological networks, Proceedings of the national academy of sciences 99 (12) (2002) 7821–7826.