LICOD: Leaders Identification for Community Detection in Complex Networks

Rushed Kanawati
LIPN - CNRS UMR 7030
University of Paris Nord
99 Av. J.B. Clément
93430 Villetaneuse, FRANCE
Email: rushed.kanawati@lipn.univ-paris13.fr

Abstract—In this work we propose a new efficient algorithm for communities construction based on the idea that a community is animated by a set of leaders that are followed by a set of nodes. A node can follow different leaders animating different communities. The algorithm is structured into two mains steps: identifying nodes in the network playing the role of leaders, then assigning other nodes to some leaders. We provide a general framework for implementing such an approach. First experimental results obtained by applying the algorithm on different real networks show the effectiveness of the proposed approach.

I. INTRODUCTION

Almost all real-world complex networks exhibit a mesoscopic level of organization, called communities [1]. A community is a connected sub-graph whose nodes are much linked with one each other that with nodes out-side the sub-graph. Nodes in a community, are generally supposed to share common properties or play similar roles within the network. This suggests that we can gain much insight into complex networked systems by discovering and examining their underlying communities.

More importantly, since the community-level structure is exhibited by almost all studied real-world complex networks, an efficient algorithm for detecting communities would be useful to implement a pre-treatment step for a number of general complex operations such as computation distribution, huge graph visualization and large-scale graph compression [2].

A huge number of algorithms have been proposed for detecting communities in complex networks. A quick review of main approaches for community detection and evaluation is made in section II. We propose a new community detection algorithm called LICOD \(^1\) inspired from human community formation process. We define a community as a set of nodes led by a set of core leaders followed by other community members. A single node can belong to different communities at once. The proposed algorithm is structured into two mains steps: 1) Identifying nodes in the network that are playing the role of communities leaders, 2) then assigning other nodes to leaders in order to construct the communities.

\(^1\)for Leader Identification for COmmunity Detection algorithm

Our algorithm computes automatically the number of communities to identify. Moreover it identifies overlapping communities rather than disjoint ones as most of existing algorithms do. The algorithm is detailed in section III. First experimental results are given and commented in Section IV. Finally, we conclude in Section V.

II. COMMUNITY DETECTION AND EVALUATION APPROACHES

A. Community detection approaches

A wide diversity of approaches has been proposed in the scientific literature. We propose to classify existing approaches according the following criteria:

1) Inter-community relationship nature: Computed communities can be of two types: disjoint communities or overlapping ones. The majority of existing approaches computes disjoint communities where each node can belong to one community at most [3], [1], [4]. However, in many real application fields, a node might belong to different communities at once. Examples of approaches for detecting overlapping communities are given in [5], [6], [7].

2) Partition-quality guidance: We distinguish between algorithms optimizing some partition quality criteria in order to construct communities and those that are not guided by such a quality parameter. Different partition quality metrics has been proposed in the scientific literature. The Newman modularity is the most widely used one [1]. Examples of modularity guided algorithms are [1], [3], [8], [9]. Examples of modularity independent approaches are [10], [4], [11].

3) Graph structure type: Most approaches propose algorithms for community identification in unipartite graphs [3], [1]. However, in many situations handled complex networks can have bipartite graphs (affiliation networks), tripartite graphs (such as the case of folksonomies) or in general case, a multi-partite graphs. Examples of work coping with community identification in multi-partite graphs are given in [12], [9], [8].

4) Computation approach: A quick review of the state of the art allows to distinguish among the following community computation schemes:
a) Agglomerative approaches: This is a bottom-up approach where an algorithm starts by considering each single node as a community, then iterates by merging some communities guided by some quality criteria. Work presented in [3] is an example of this approach.

b) Separative approaches: This a top-down approach, where an algorithm starts by considering the whole network as a community. It iterates to select ties to remove in order to split the network into communities. Different criteria can be applied for tie selection. The Newman-Girvan algorithm is the most known representative of this class of approaches [1].

c) Constructive approaches: These are relatively new approaches according to which some heuristics are applied in order to build communities directly without following a hierarchical approach as both agglomerative and separative approaches do. One example is the work presented in [10] where authors propose an algorithm that weights links in a complex network based on some node similarity metrics allowing then to identify communities by removing links whose weights are under a given threshold. More related to our approach, algorithms proposed in [11], [4] where communities are built around special nodes called also leaders. However in both algorithms each community is centered around one leader only and a node can follow one community at once. In addition the first algorithm requires the user to provide the number k of communities to identify.

d) Optimization approach: This consists of applying classical optimization approaches that are guided by some community decomposition quality criteria. Examples are work applying genetic algorithms for community detection [13], [14], [15].

III. THE LICOD APPROACH

A. Preliminaries

We start by introducing some basic notations and functions that will be used later in this work. Let $G = (V, E)$ be a graph. For sake of simplicity we consider here simple connected unweighted and undirected graphs. Let $n_G = |V|$ (resp. $m_G = |E|$) be the number of nodes (resp. edges) in $G$. Let $A$ be the binary square adjacency matrix describing $G$. We denote by $\Gamma_G(n)$ the set of direct neighbors of node $n$ in $G$. The number of direct neighbours of a node $n$ is the degree of $n$: $d_G(n) = |\Gamma_G(n)|$.

Nodes (as well edges in a very similar way) of a network can be ranked according to the importance of the role they play applying some basic topological metrics. In this work, we make use of the two following basic centrality metrics:

- **The degree centrality** $dc(v)$ is defined as the fraction of nodes that are directly connected to node $v$. This is given by:

  $$dc(v) = \frac{d_G(v)}{n_G - 1} \quad (1)$$

- **Betweenness centrality** $BC(v)$ is the sum of the fraction of all-pairs shortest paths that pass through $v$. Formally, we have:

  $$BC(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)} \quad (2)$$

  where $\sigma(s,t)$ is the number of shortest paths linking $s$ to $t$, and $\sigma(s,t|v)$ is the number of paths passing through node $v$ other than $s$ and $t$.

B. LICOD: Informal presentation

The basic idea underlaying the proposed algorithm is that a community is composed of two types of nodes: Leaders and Followers. Roughly speaking, leaders form a subset of nodes (eventually one node) whose removal form the network implies community collapse. Algorithm 1 sketches the general outlines of the proposed approach. The algorithm functions as follows. First it searches for nodes in the network that are likely to be leaders in a community. Different node ranking metrics can be used in order to estimate the role of a node. These include the classical centrality metrics (see section II). Let $L$ be the set of identified leaders. In algorithm 1 this step is achieved by the function $isLeader()$ (line 3). The list $L$ is then reduced by grouping leaders that are estimated to be in the same community. This is the task of the function $computeCommunitiesLeader()$, line 7 in algorithm 1. Let $C$ be the set of identified communities. Each node in the network (a leader or a follower) computes its membership degree to each community in $C$. A ranked list of communities can then be obtained, for each node, where communities with highest membership degree are ranked first (lines 9-13 in algorithm 1). Next, each node will adjust its community membership preference list by merging this with preference lists of its direct neighbors in the network. Different strategies borrowed form the social choice theory can applied here to merge the different preference lists [16]. This step is iterated until stabilization of obtained ranked lists at each node. The convergence towards a stable state is function of the applied voting scheme. Lastly, each node will be assigned to top ranked communities in its final obtained membership preference list.

The local voting process intends to ensure local homogeneity in nodes membership to different communities. Notice that the algorithm is designed as a general framework that allow testing different working hypothesis: How to select leader ? How to compute community membership ? And how to merge preferences linked nodes ? Next we describe choices made for the first concrete implementation of this general framework.

C. Implementation issues

We give next some details about the implementation of each of the main steps of the proposed algorithm.

1) Function $isLeader()$: One simple idea to distinguish leaders from followers nodes is to compare nodes centralities. Actually, leader nodes are expected to have higher centrality (whatever the centrality is) than follower nodes. We propose to apply two basic centralities : the degree centrality and betweenness centrality (see equations 1, 2). The first is local computed metric while the second capture global proprieties of the network (see equations 1, 2). A node is a leader if
Algorithm 1 LICOD algorithm

Require: \( G = (V,E) \) a connected graph

1: \( L \leftarrow \emptyset \) \{set of leaders\}
2: \textbf{for} \( v \in V \) \textbf{do}
3: \textbf{if} \( \text{isLeader}(v) \) \textbf{then}
4: \( L \leftarrow L \cup \{v\} \)
5: \textbf{end if}
6: \textbf{end for}
7: \( C \leftarrow \text{computeCommunitiesLeader}(L) \)
8: \textbf{for} \( v \in V \) \textbf{do}
9: \textbf{for} \( c \in C \) \textbf{do}
10: \( M[v,c] \leftarrow \text{membership}(v,c) \) \{see equation 3\}
11: \textbf{end for}
12: \( P[v] = \text{sortAndRank}(M[v]) \)
13: \textbf{end for}
14: \textbf{repeat}
15: \textbf{for} \( v \in V \) \textbf{do}
16: \( P^*[v] \leftarrow \text{rankAggregate}_{x \in \Gamma(v)} \cdot \text{SPath}(v,x) \cdot P[x] \)
17: \( P^*[v] \leftarrow P^*[v] \)
18: \textbf{end for}
19: \textbf{until} Stabilization of \( P^*[v] \forall v \)
20: \textbf{for} \( v \in V \) \textbf{do}
21: \( P[v] = P^*[v] \)
22: \textbf{for} \( c \in P[v] \) \textbf{do}
23: \textbf{if} \( |M[v,c] - M[v,P[0]]| \leq \epsilon \) \textbf{then}
24: \( COM(c) \leftarrow COM(c) \cup \{v\} \)
25: \textbf{end if}
26: \textbf{end for}
27: \textbf{end for}
28: \textbf{return} \( C \)

This centrality is greater or equal to \( \sigma \in [0,1] \) percent of its neighbors centralities. The rational behind introducing the \( \sigma \) parameter is to be able to recover leaders connected to other leaders. Notice that the number of leaders will depend on the value of the threshold \( \sigma \). More \( \sigma \) is high fewer the leaders are.

2) computeCommunitiesLeaders: Two leaders are grouped in a same community if the ratio of common neighbors to the total number of neighbors is above a given threshold \( \delta \in [0,1] \). The couple \( \sigma, \delta \) determines somehow, the number of communities detected by the algorithm.

3) membership\((v,c)\): We propose to measure the membership degree of a node \( v \) to a community \( c \) by the inverse of the minimal shortest path that links \( v \) to one of the leaders of \( c \).

\[
\text{membership}(v,c) = \frac{1}{\min_{x \in \text{COM}(c)} \text{SPath}(v,x)) + 1} \quad (3)
\]

It is easy to see that the previous function takes values in the range \([0,1]\). The diameter of a graph is the maximum of the shortest path between any pair of nodes. Notice also that for a community \( c \), the membership degree of all its leaders is equal to 1.

4) Rank merging: Different social choice or voting algorithms can be applied to compute \( P^*_v \) the final community preference membership list [16]. In the current version, we use two voting schemes:

- Borda vote [17]. This is a simple position-based voting scheme. Let \( k \) be the number of detected communities (i.e. \( k = |C| \)). In each preference list, the top ranked community is attributed weight \( k \) the second ranked community is attributed weight \( k-1 \) and so forth till attributing one to the last ranked community. The merging function of all votes consists simply on ranking communities in function of the sum of weights they get from each voter (the node and its direct neighbors).

- Local kemenisation approach [18]. This approach guarantees that the output of the merging process satisfies the extended Condorcet principle. This principle states that: if there is a partition \( C,D \) of the total list of candidates (i.e communities) \( C \) such that \( \forall x \in C, \forall y \in D \) if the majority of voters (i.e. the local node and its direct neighbors) prefers \( x \) to \( y \), then \( x \) is ranked above \( y \). Details about this method are out of the scope of this paper. Interested reader can consult [18].

5) Community assignment: A node \( v \) is assigned to top ranked communities in the final community preference list \( P^*_v \). As showed in lines 22-26 of algorithm 1, a node is assigned simultaneously to communities for which its membership is \( \epsilon \)-far from the membership degree to the top ranked community. The \( \epsilon \) threshold controls the degree of desired overlapping in identified communities. However, putting \( \epsilon \) to 0 may still results in having overlapping communities since for a given node different communities may have the same membership degree.

IV. EXPERIMENTAL RESULTS

We measure the performances of the proposed algorithm by applying it to three classical small networks: The Zachary Karate Club [19], the Strike network, and the NACA football Bowl subdivisions games schedule dataset. All three datasets are available on the pajek web site\(^2\). Table I gives essential characteristics of these datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Nodes</th>
<th># edges</th>
<th># Real Communities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zachary</td>
<td>34</td>
<td>78</td>
<td>2</td>
</tr>
<tr>
<td>Strike</td>
<td>24</td>
<td>38</td>
<td>3</td>
</tr>
<tr>
<td>Football</td>
<td>115</td>
<td>613</td>
<td>11</td>
</tr>
</tbody>
</table>

While these networks are rather small, compared to the scale of on-line social networks, they present the advantage of being analyzed by experts in order to partition these into real communities.

When a ground truth about a community structure is available, classical metrics for evaluating performances of clustering algorithms can be applied. We use here two most used clustering evaluation measures: Purity and Rand index.

Let \( P = \{p_1,p_2,\ldots,p_k\} \) be the output of a community detection algorithm applied to graph \( G = (V,E) \), and let \( R = \{r_1,r_2,\ldots,r_n\} \) be the real community structure. The

\(^2\)http://vldao.fmf.uni-lj.si/pub/networks/pajek
purity of $\mathcal{P}$, with respect to $\mathcal{R}$ is then given by:

\[
purity(\mathcal{P}, \mathcal{R}) = \frac{1}{|\mathcal{V}|} \sum_{j=1}^{k} \max_{i} (|p_j \cap r_i|)
\]

(4)

The above equation computes simply the mean local purity of each computed community ($p_j$) as compared to the real community with which it has the highest intersection. The purity takes values in the interval $[0, 1]$ where 1 express the fact that computed communities match exactly real ones. The purity measure captures, in some how, the precision of the computed community.

The **Rand index** is proposed in order to take into account both precision and recall of the outcome of a community detection algorithm [20]. This is computed as follows: Let $a, b, c$ and $d$ be the number of pairs of nodes that are respectively in the same community according to $\mathcal{P}$ and $\mathcal{R}$, in a same community according to $\mathcal{P}$ but in different communities according to $\mathcal{R}$, in different communities as given by $\mathcal{P}$ but in a same community as given by $\mathcal{R}$, and in different communities according to both $\mathcal{P}$ and $\mathcal{R}$. The basic Rand Index is then given by:

\[
rand(\mathcal{P}, \mathcal{R}) = \frac{a + d}{a + b + c + d}
\]

(5)

Different variations of this index are also proposed in the literature [21].

For each network we’ve applied the proposed algorithm by changing the configuration parameters as follows:

- Centrality metrics = [Degree centrality, Betweenness centrality]
- Voting method = [Borda, Local kemenisation]
- $\sigma \in [0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$
- $\delta \in [0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$
- $\epsilon \in [0.0, 0.1, 0.2]$

For each configuration we compute the number of found communities as well as the purity and the rand index. Next graphics show the variations of these metrics, for each dataset, with the variation of $\sigma$. We’ve omitted to show the results with different values of $\delta$ since on these datasets the $\delta$ value has showed negligible impact on obtained results. On each figure, we plot four graphics showing the variation of a metric for each of the possible four configurations depending on the choice of the used centrality and the voting method. Dashed lines (resp. circles, triangles) show curves for $\epsilon$ equal to 0.0 (resp. 0.1, 0.2). For figures showing the number of communities we trace also a line showing the real number of communities.

As expected the number of identified communities diminish with the increase of $\sigma$. The use of the betweenness centrality accelerate slightly the convergence for the right value to obtain. Local Kemeney voting method outperforms Borda in the case of the football network only and gives comparable results for the first two networks.

Increasing $\epsilon$ results in diminishing the rand index. This can be explained by the fact that high value of $\epsilon$ increases the overlapping degree of obtained communities while real communities we have here are all disjoint.

The best results are obtained for $\sigma$ around 0.7, 0.8. This argues for the validity the idea of introducing the $\sigma$ threshold and not to consider extreme cases where a node is qualified as a leader if it has the highest centrality in its direct neighborhood. While curves are similar for the two first networks (Zachary and Strike) these have a different dynamic in the third case (football dataset). This suggests that real social interaction networks (as the first two) may differ from other types of complex networks (ex. game schedule). The choice of a configuration of the proposed algorithm in function of the properties of the target network constitute one interesting topic to cope with.

![Fig. 1. Zachary Dataset: Community Number vs. $\sigma$.](image1)

![Fig. 2. Zachary Dataset: Purity vs. $\sigma$.](image2)

We also compared the results of our algorithm with results obtained by well known algorithms: The Newman-Girvan algorithm [1] and the Louvain algorithm [3]. Table II gives obtained results on both datasets.

These preliminary results shows clearly that the modularity metric does not correspond to the best decomposition into


<table>
<thead>
<tr>
<th>Approach</th>
<th>Zacahary dataset</th>
<th>Strike dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># Communities</td>
<td>Purity</td>
</tr>
<tr>
<td>Newman</td>
<td>2</td>
<td>0.97</td>
</tr>
<tr>
<td>Louvain</td>
<td>4</td>
<td>0.64</td>
</tr>
<tr>
<td>Our approach</td>
<td>$\sigma = 0.8$</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Fig. 3. Zachary Dataset: Rand vs. $\sigma$

Fig. 4. Strike Dataset: Community Number vs. $\sigma$

Fig. 5. Strike Dataset: Purity vs. $\sigma$

Fig. 6. Strike Dataset: Rand vs. $\sigma$

![Comparison of performances of different community detection algorithms](image)

V. CONCLUSION

In this work we have proposed a new approach for community detection in complex networks. The approach is based on identifying a set of community leaders then assigning nodes to these leaders. The approach is constructed to allow detection of overlapping communities. The complexity of the approach depends on the complexity of the applied centrality metric used to identify leader nodes. Results obtained on small benchmark social network argue for the capacity of the approach to detect real communities. Future developments we are working on include: testing the algorithm on large scale networks, develop a full distributed self-stabilizing version exploiting the fact that major part of computations are made in a local manner and finally adapting the approach for multi-partite networks.
REFERENCES


