Vietnam J Comput Sci (2014) 1:241–256
DOI 10.1007/s40595-014-0025-6

REGULAR PAPER

LICOD: A Leader-driven algorithm for community detection in complex networks

Zied Yakoubi · Rushed Kanawati

Received: 30 November 2013 / Accepted: 6 June 2014 / Published online: 14 September 2014

Abstract Leader-driven community detection algorithms (LdCD hereafter) constitute a new trend in devising algorithms for community detection in large-scale complex networks. The basic idea is to identify some particular nodes in the target network, called *leader nodes*, around which local communities can be computed. Being based on local computations, they are particularly attractive to handle large-scale networks. In this paper, we describe a framework for implementing LdCD algorithms, called LICOD. We propose also a new way for evaluating performances of community detection algorithms. This consists on transforming data clustering problems into a community detection problems. External criteria for evaluating obtained clusters can then be used for comparing performances of different community detection approaches. Results we obtain show that our approach outperforms top state of the art algorithms for community detection in complex networks.

Keywords Complex networks · Community detection · Leader-driven algorithms · Task-based evaluation

1 Introduction

Research in mining and analyzing large-scale complex networks has been boosted recently after discovering that much of complex networks extracted form natural and artificial systems share a set of non-trivial characteristics that distinguish them from pure random graphs. Basic topological characteristics of complex networks are: low separation degree (or what is better known as small-world feature [37]), power-law distribution of node's degrees [75], and high clustering coefficient [46]. As a consequence of these basic topological features, almost all real-world complex networks exhibit a mesoscopic level of organization, called *communities* [58]. A community is loosely defined as a connected subgraph whose nodes are much linked with one each other than with nodes outside the subgraph. 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 underlaying communities. The semantic interpretation of a community depends on the type of the analyzed graph. In a metabolic network, a community would express a biological function in a cell [26]. In a network of transactions in an e-commerce site, this would express a set of similar customers [6]. Considering the web as a complex network, a community would be a set of pages dealing with a same topic [20].

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 [25].

A quite big number of algorithms have been proposed for detecting communities in complex networks. Recent interesting survey tidies on this topic can be found in [21,66,83]. A quick review of the scientific literature allows to distinguish three different, but related problems:

- *Disjoint communities detection*: The goal here is to compute a partition of the graph node's set. One node can belong to only one community at once. Most of the work in the area of community detection deals with this problem [21].

Z. Yakoubi · R. Kanawati (✉)
LIPN, CNRS UMR 7030, University Paris Nord, Villetaneuse, France
e-mail: rushed.kanawati@lipn.univ-paris13.fr

- *Overlapping communities detection*: The goal is to compute soft clustering of the graph node's set where a node can belongs to several communities at once [61,64,77,87, 88].
- *Local community identification*: The goal here is to compute the community of a given node rather than partitioning the whole graph into communities. This can be useful in different settings, namely in the area of recommender systems [5,11,13,33].

Both problems, disjoint and overlapping community detection are NP-hard [10]. Different heuristics have been proposed to compute sub-optimal partitions. Most popular methods are based on applying greedy optimisation approaches of a graph partition quality measure [7,23,73]. The most applied graph partition criteria are the modularity initially introduced in [23]. However, some recent studies has pointed out some serious limitations of modularity optimization-based approaches [24,40]. These limitations have boosted the research for alternative approaches for community detection. Emergent approaches include label propagation approaches [71] and seed-centric ones [34]. The basic idea of seed-centric approaches is to select a set of nodes (i.e. seeds) around which communities are constructed. Being based on local computations, these approaches are very attractive to deal with large-scale and/or dynamic networks. One special case of seeds is to select nodes that are likely to act as leaders of their communities [36,76]. In this work, we propose a general framework for implementing Leader-driven community detection algorithms (LdCD hereafter) called LICOD. The approach we develop here is an extension of the work presented in [32]. Major enhancements are about transforming LICOD into a framework for implementing LdCD algorithms as described in Sect. 4. Another major new contribution concerns the evaluation process. Actually, since LdCD algorithms are not based on maximizing an objective function (i.e. the modularity), it is unfair to use the later criteria to compare these algorithms with popular modularity-guided approaches. One idea to provide fair evaluation criteria for different community detection algorithms is *task-oriented evaluation*. This can be conducted by evaluating how good are computed communities for realizing a given dependent task. In this paper, we propose using data clustering task for that purpose. The idea is to transform classical clustering benchmarks into a community detection problem. Algorithms can then be evaluated using classical extrinsic clustering evaluation metrics [52].

To sum up, main contributions of this paper are the following:

- Proposing LICOD, a general framework form implementing LdCD algorithms.
- Introducing task-oriented evaluation of community detection algorithms and providing an approach for evaluating different community detection algorithms on data clustering tasks.

The remainder of this paper is organized as follows. Next in Sect. 2, we provide basic notations used in this paper. In Sect. 3, we review briefly major approaches for community detection algorithms as well as evaluation approaches. The LICOD approach is detailed in Sect. 4. Next, in Sect. 5, experimentation on both small benchmark networks and applying the proposed task-oriented evaluation approach are described. The clustering-oriented evaluation approach is described in Sect. 5.2. Obtained results are provided and commented. Finally, we conclude in Sect. 6.

2 Definitions and notations

In this study, we only consider simple unweighted, undirected graphs. A graph G is defined by a couple: $G = \langle V, E\rangle$ where $V = \{v_1 \ldots, v_n\}$ is a set of nodes (a.k.a actors, sites, vertices) and $E \subseteq V \times V$ is a set of links (a.k.a ties, arcs, or relationships). We denote by $n_G = |V|$ (reps. $m_G = |E|$) the number of nodes (reps. links) of graph G. The set of direct neighbors of a node $v \in V$ is given by the function $\Gamma(v)$. The number of direct neighbors of a node is the node's degree and is denoted by $d_v = |\Gamma(v)|$. The density of a graph G is given by the ratio of the number of existing links to the number of potential links. This is given by: $d(G) = \frac{2\times m_g}{n_g\times(n_g-1)}$. We denote by A the adjacency matrix of graph G. We have $A_{ij} = 1$ (resp. $A_{ij} = 0$) if nodes $v_i, v_j \in V$ are linked (resp. unlinked).

3 Related work

In this section, we provide a brief survey on both following topics related to the contributions of this paper: community detection algorithms and community evaluation approaches.

3.1 Community detection approaches

We focus in this study on approaches that aim to compute a partition, or disjoint communities of a complex network. A wide variety of different approaches have been proposed so far. Some comprehensive survey studies are provided in [21,66,83]. Here, we propose to classify existing approaches into four classes: Group-based approaches, network-based approaches, propagation-based approaches and seed-centric ones. Next we briefly review each of these identified classes.

3.1.1 Group-based approaches

These are approaches based on identifying groups of nodes that are highly connected or share some strong connection patterns. Some relevant connection patterns are the following:

- *High mutual connectivity*: a community can be assimilated to a maximal *clique* or to a γ-quasi-clique. A subgraph G is said to be γ-quasi-clique if $d(G) \leq \gamma$. Finding maximal cliques in a graph is known to be a NP-hard problem. Generally, cliques of reduced size are used as seeds to find larger communities. An example is the clique percolation algorithm [1,82]. Such approaches are relevant for networks that are rather dense.
- *High internal reachability*: One way to relax the constraint of having cliques or quasi-cliques is to consider the internal reachability of nodes within a community. Following this, a community core can be approximated by a maximal k-clique, k-club or k-core subgraph. A k-clique (resp. k-club) is a maximal subgraph in which the longest shortest path between any nodes (resp. the diameter) is $\leq k$. A k-core is a maximal *connected* subgraph in which each node has a degree $\geq k$. In [86], authors introduce the concept of *k-community* which is defined as a connected subgraph $G' = \langle V' \subset V, E' \subset E \rangle$ of a graph G in which for every couple of nodes $u, v \in V'$ the following constraint holds: $|\Gamma_G(v) \cap \Gamma_G(u)| \geq k$. The computational complexity of k-cores and k-communities is polynomial. However, these structures do not correspond to all the community, but are rather used as seeds for computing communities. An additional step for adding non-clustered nodes should be provided. In [67], authors propose to compute k-cores as mean to accelerate computation of communities using standard algorithms, but on size-reduced graphs.

3.1.2 Network-based approaches

These approaches consider the whole connection patterns in the network. Historical approaches include classical clustering algorithms. The adjacency matrix can be used as a similarity one, or topological similarity between each couple of nodes can also be computed. Spectral clustering approaches [59] and hierarchical clustering approaches can then be used [70]. Usually the number of clusters to be found should be provided as an input for the algorithm. Another drawback of spectral clustering is its high computation complexity which might be cubic on the size of the input dataset. Some distributed implementations of these approaches are proposed to provide efficient implementations [85]. More popular network-based approaches are those based on optimizing a quality metric of graph partition. Different partition quality metrics have been proposed in the scientific literature. The *modularity* is the most widely used one [58]. This is defined as follows. Let $\mathcal{P} = \{C_1, \ldots, C_k\}$ a partition of the node's set V of a graph. The modularity of the partition $\mathcal{P}$ is given by:

$$Q(\mathcal{P}) = \sum_{c \in \mathcal{P}} e(\mathcal{C}) - a(\mathcal{C})^2 \qquad (1)$$

where $e(\mathcal{C}) = \frac{\sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}} A_{ij}}{2 \times m_G}$ is the fraction of links inside the community $\mathcal{C}$, and $a(\mathcal{C}) = \frac{\sum_{i \in \mathcal{C}} \sum_{j \in V} A_{ij}}{2.m_G}$ is the fraction of links incident to a node in $\mathcal{C}$. The computing complexity of Q is $(O)(m_G)$ [23]. Some recent work has extended the definition to bipartite and multipartite graphs [18,48,56] and even for multiplex and dynamic graphs [39,55]. Different heuristic approaches have been proposed for computing partitions that maximize the modularity. These can be classified into three main classes:

- *Agglomerative approaches*: These implement a bottom-up approach where an algorithm starts by considering each single node as a community. Then, it iterates by merging some communities guided by some quality criteria. The *louvain* algorithm [7] is one very known example of such approaches. The algorithm is composed of two phases. First, it looks for small communities by optimizing modularity in a local way. Second, it aggregates nodes of the same community and builds a new network whose nodes are the communities. Two adjacent communities merge if the overall modularity of the obtained partition can be enhanced. These steps are repeated iteratively until a maximum of modularity is reached. The computing complexity of the approach is empirically evaluated to be $\mathcal{O}(nlog(n))$.
- *Separative approaches*: These implement a top-down approach, where an algorithm starts by considering the whole network as a community. It iterates to select ties to remove 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 [58]. The algorithm is based on the simple idea that a tie linking two communities should have a high betweenness centrality. This is naturally true since an inter-community tie would be traversed by a high fraction of shortest paths between nodes belonging to these different communities. Considering the whole graph G, the algorithm iterates for m_G times, cutting at each iteration the tie with the highest betweenness centrality. This allows to build a hierarchy of communities, the root of which is the whole graph and leafs are communities composed of isolated nodes. Partition of highest modularity is returned as an output. The algorithm is simple to implement and has the advantage to discover automatically the best number of communities to identify. However, the computation complexity is rather high: $\mathcal{O}(n^2 \cdot m + (n)^3 log(n))$. This is prohibitive to apply to large-scale networks.

- *Other optimization approach*: Other classical optimization approaches can also be used for modularity optimization such as applying genetic algorithms [31,47,68], evolutionary algorithms [29] or multi-objective optimization approaches [69].

All modularity optimization approaches make implicitly the following assumptions:

- The best partition of a graph is the one that maximize the modularity.
- If a network has a community structure, then it is possible to find a precise partition with maximal modularity.
- If a network has a community structure, then partitions inducing high modularity values are structurally similar.

Recent studies have showed that all three above-mentioned assumptions do not hold. In [24], authors show that the modularity function exhibits extreme degeneracies: it namely accepts an exponential number of distinct high scoring solutions and typically lacks for a clear global maximum. In [40], it has been shown that communities detected by modularity maximization have a resolution limit. These serious drawbacks of modularity-guided algorithms have boosted the research for alternative approaches. Some interesting emerging approaches are label propagation approaches [71] and seed-centric ones [34].

3.1.3 Propagation-based approaches

Even the top fast algorithm, the *louvain* approach, has a computation complexity that becomes costly for very large-scale networks that can be composed of millions of nodes as it is frequently the case when considering online social networks today. In addition, studied complex networks are very dynamic. A low complexity incremental approaches for community detection are then needed. Label propagation approaches constitute a first step in that direction [71,89]. The underlaying idea is simple: each node $v \in V$ in the network is assigned a specific label l_v. All nodes update in a synchronous way their labels by selecting the most frequent label in the direct neighborhood. In a formal way, we have:

$$l_v = \arg\max_l |\Gamma^l(v)|$$

where $\Gamma^l(v) \subseteq \Gamma(v)$ is the set of neighbors of v that have the label l. Ties situations are broken randomly. The algorithm iterates until reaching a stable state where no more nodes change their labels. Nodes having the same label are returned as a detected community. The complexity of each iteration is $\mathcal{O}(m)$. Hence, the overall computation complexity is $\mathcal{O}(km)$ where k is the number of iterations before convergence. Study reported in [45] shows that the number of iterations grows in a logarithmic way with the growth of n; the size of the target network. In addition to its low computation complexity, the label propagation algorithm can readily be distributed allowing hence handling very large-scale networks [62,78, 92]. While the algorithm is very fast, it suffers from two serious drawbacks:

- First, there is no formal guarantee of the convergence to a stable state.
- Lastly, it lacks for robustness, since different runs produce different partitions due to random tie breaking.

Different approaches have been proposed in the literature to cope with these two problems. Asynchronous, and semi-synchronous label updating have been proposed to hinder the problem of oscillation and improve convergence conditions [14,71]. However, these approaches harden the parallelization of the algorithm by creating dependencies among nodes and they increase the randomness in the algorithm making the robustness even worse. Different other approaches have been developed to handle the problem of label propagation robustness. These include *balanced label propagation* [81], *label hop attenuation* [44] and *propagation preference*-based approaches [49]. Another interesting way to handle the instability of label propagation approaches consists simply on executing the algorithm k times and apply an ensemble clustering approach on the obtained partitions [33,41,63,74].

3.1.4 Seed-centric approaches

The basic idea underlaying seed-centric approaches is to identify some particular nodes in the target network, called *seed nodes*, around which local communities can be computed [32,65,76]. Algorithm 1 presents the general outlines of a typical seed-centric community detection algorithm. We recognize three principal steps:

1. Seed computation.
2. Seed local community computation.
3. Community computation out from the set of local communities computed in the previous step.

Algorithm 1 General seed-centric community detection algorithm

```
Require: $G =< V, E >$ a connected graph,
1: $\mathcal{C} \leftarrow \emptyset$
2: $S \leftarrow$ compute_seeds(G)
3: for $s \in S$ do
4:     $C_s \leftarrow$ compute_local_com(s,G)
5:     $\mathcal{C} \leftarrow \mathcal{C} + C_s$
6: end for
7: return compute_community($\mathcal{C}$)
```

Leader-driven algorithms constitute a special case of seed-centric approaches. Nodes of a network are *classified* into two (eventually overlapping) categories: leaders and followers. Leaders represent communities. An assignment step is applied to assign *followers* nodes to most relevant communities. Different algorithms apply different node classification approaches and different node assignment strategies. Three different LdCD algorithms have been proposed almost simultaneously in three different works [32,36]. Next, we present briefly the first two cited algorithms.

In [36] authors propose an approach directly inspired from the *K-means* clustering algorithm [27]. The algorithm requires as input the number k of communities to identify. This is clearly a major disadvantage of the approach that authors of the approach admit. k nodes are selected randomly. Unselected nodes are labeled as followers. Leaders and followers form hence exclusive sets. Each leader node represents a community. Each follower nodes is assigned to the most nearby leader node. Different levels of neighborhood are allowed. If no nearby leader is found the follower node is labeled as *outlier*. When all flowers nodes are handled. The algorithm computes a new set of k leaders. For each community, the most *central* node is selected as a leader. The process is iterated with the new set of k leaders until stabilization of the computed communities. The convergence speed depends on the quality of initially selected k leaders. Different heuristics are proposed to improve the selection of the initial set of leaders. The best approach according to experimentation is to select the top k nodes that have the top degree centrality and that share little common neighbors.

The algorithm proposed in [76] is much similar to our approach. It starts by computing the *closeness* centrality of all nodes. The closeness centrality of a node v is given by the inverse of the average distance to all other nodes in the network. Leaders will be any node whose closeness centrality is less than at least one of its neighbors. This heuristics results in a huge set of leaders. The list of leaders is sorted in decreasing order of closeness centrality. The list is then parsed assigning to each leader direct followers that are not already assigned to another leader. At the end, leaders that are not followed by any node are assigned to the community to which belong the majority of its direct neighbors.

3.2 Community evaluation approaches

The problem of performances evaluation of community detection algorithms still to be an open problem in spite of the huge amount of work in this area. Existing approaches can be divided into three main types:

1. Evaluation on networks for which a ground-truth decomposition into communities is known.
2. Evaluation in function of the topological features of computed communities.
3. Task-driven evaluation.

Next, we detail these different approaches.

3.2.1 Ground-truth comparison approaches

Networks with ground-truth partitions can be obtained by one of the following ways:

- *Annotation by experts*: For some networks representing real systems, experts in the system field have been able to define the community structure. Examples of such networks are given in Sect. 5.1. In general, these networks are rather very small (allowing hence to be handled by experts) and the defined community structure is usually given by a partition of the studied graph with no overlapping among defined communities.
- *Network generators use*: The idea here is to generate artificial networks with predefined community structure. Some early work in this area is the Girvan–Newman benchmark graph [23]. A more sophisticated generator is proposed in [42] where the user can control different parameters of the network including the size, the density, the degree distribution law, the clustering coefficient, the distribution of communities size as well as the separability of the obtained communities. While the approach is interesting, generated networks are not guaranteed to be similar enough to real complex networks observed in real-world applications.
- *Implicit community definition* : This approach is based on inferring the community structure in a graph applying simple rules taking usually the semantic of ties into account. For example in [90] authors define a community in the *Live journal* social network as groups of fans of a given artist. Communities in a co-authorship of scientific publications are taken to be authors participating in a same venue! The relevance of proposed rules seems to be questionable.

When a ground-truth community structure is available, classical external clustering evaluation indices can be used to evaluate and compare community detection algorithms. Different clustering comparison or similarities functions have been proposed in the literature [2]. In this work, we apply two widely used indices: the Adjusted Rand Index (ARI) [30] and the Normalized Mutual Information (NMI) [79].

The ARI index is based on counting the number of pairs of elements that are clustered in the same clusters in both compared partitions. Let $P_i = \{P_i^1, \ldots, P_i^l\}$, $P_j = \{P_j^1, \ldots, P_j^k\}$ be two partitions of a set of nodes V. The set of all (unordered) pairs of nodes of V can be partitioned into the following four disjoint sets:

- S_{11} = {pairs that are in the same cluster under P_i and P_j}
- S_{00} = {pairs that are in different clusters under P_i and P_j}
- S_{10} = {pairs that are in the same cluster under P_i but in different ones under P_j}
- S_{01} = {pairs that are in different clusters under P_i but in the same under P_j}

Let $n_{ab} = |S_{ab}|, a, b \in \{0, 1\}$, be the respective sizes of the above defined sets. The rand index initially defined in [72] is simply given by:

$$\mathcal{R}(P_i, P_j) = \frac{2 \times (n_{11} + n_{00})}{n \times (n-1)}$$

In [30], authors show that the expected value of the Rand Index of two random partitions does not take a constant value (e.g. zero). They proposed an adjusted version which assumes a generalized hypergeometric distribution as null hypothesis: the two clusterings are drawn randomly with a fixed number of clusters and a fixed number of elements in each cluster (the number of clusters in the two clusterings need not be the same). Then the ARI is the normalized difference of the Rand Index and its expected value under the null hypothesis. It is defined as follows:

$$\mathrm{ARI}(P_i, P_j) = \frac{\sum_{x=1}^{l}\sum_{y=1}^{k}\binom{|P_i^x \cap P_j^y|}{2} - t_3}{\frac{1}{2}(t_1+t_2) - t_3} \qquad (2)$$

where:

$$t_1 = \sum_{x=1}^{l}\binom{|P_i^x|}{2}, \quad t_2 = \sum_{y=1}^{k}\binom{|P_j^y|}{2}, \quad t_3 = \frac{2t_1t_2}{n(n-1)}$$

This index has expected value zero for independent clusterings and maximum value 1 for identical clusterings.

Another family of partitions comparisons functions is the one based on the notion of mutual information. A partition P is assimilated to a random variable. We seek to quantify how much we reduce the uncertainty of the clustering of randomly picked element from V in a partition P_j if we know P_i. The Shannon's entropy of a partition P_i is given by:

$$H(P_i) = -\sum_{x=1}^{l}\frac{|P_i^x|}{n}\log_2\left(\frac{|P_i^x|}{n}\right)$$

Notice that $\frac{|P_i^x|}{n}$ is the probability that a randomly picked element from V be clustered in P_i^x. The mutual information between two random variables X, Y is given by the general formula:

$$\mathrm{MI}(X, Y) = H(X) + H(Y) - H(X, Y) \qquad (3)$$

This can then be applied to measure the mutual information between two partitions P_i, P_j. The mutual information defines a metric on the space of all clusterings and is bounded by the entropies of involved partitions. In [79], authors propose a normalized version given by:

$$\mathrm{NMI}(X, Y) = \frac{\mathrm{MI}(X, Y)}{\sqrt{H(X)H(Y)}} \qquad (4)$$

Another normalized version is also proposed in [22]. Other similar information-based indices are also proposed [52,60].

3.2.2 *Topological measures for community evaluation*

Two types of topological measures can be used to evaluate the *quality* of a computed community structure:

- Global measures that evaluate the quality of the computed partition as a whole. The modularity Q defined in [57] (see formula 1) is the most applied measure. Other modularity measures have also been proposed [51,54]. However, the different modularity limitations discussed earlier (see Sect. 3.1.2) hinder the utility of using it as an evaluation metric.
- Local topological measures. A number of local topological measures have been proposed to evaluate the quality of a given community. Most are used in the context of identifying ego-centered communities [4,11]. In [90], authors present an interesting survey on these measures. Let $f(c)$ be a community evaluation measure. The quality of a partition is then simply given by:

$$Q(\mathcal{C}) = \frac{\sum_i f(S_i)}{|\mathcal{C}|} \qquad (5)$$

3.2.3 *Task-driven evaluation*

The principle of task-driven evaluation is the following: Let T be a task where community detection can be applied. Let $per(T, Algo^x_{com})$ be a performance measure for T execution applying the community detection algorithm $Algo^x_{com}$. We can then compare performances of different community detection algorithms by comparing induced $per(T, Algo^x_{com})$ values. In [66], authors propose to use the recommendation task for evaluating purposes. In this work, we propose using the data clustering as an evaluation task.

4 The LICOD approach

4.1 Informal presentation

The basic idea underlaying the proposed algorithm is that a community is composed of two types of nodes: *Leaders* and

Followers. Algorithm 2 sketches the general outlines of the proposed approach. The algorithm functions as follows:

1. First, it searches for nodes in the network that are likely to be leaders in a community. Different node ranking metrics can be used to estimate the role of a node. These include the classical centrality metrics. Let $\mathcal{L}$ be the set of identified leaders. In Algorithm 2, this step is achieved by the function $isLeader()$ (line 3).
2. The list $\mathcal{L}$ 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 2. Let $\mathcal{C}$ be the set of identified communities.
3. Each node in the network (a leader or a follower) computes its membership degree to each community in $\mathcal{C}$. 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 2).
4. 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. This step is iterated until stabilization of obtained ranked lists at each node. The convergence towards a stable sate is function of the applied voting scheme.
5. 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 allows testing different working hypothesis: How to select leader? How to compute community membership? And how to merge preferences of linked nodes? Next we describe possible choices for implementing each step.

Algorithm 2 LICOD algorithm

```
Require: $G =< V, E >$ a connected graph
1: $\mathcal{L} \leftarrow \emptyset$ {set of leaders}
2: for $v \in V$ do
3:   if $isLeader(v)$ then
4:     $\mathcal{L} \leftarrow \mathcal{L} \cup \{v\}$
5:   end if
6: end for
7: $\mathcal{C} \leftarrow computeComumunitiesLeader(\mathcal{L})$
8: for $v \in V$ do
9:   for $c \in \mathcal{C}$ do
10:     $M[v, c] \leftarrow membership(v, c)$ {see equation 6}
11:   end for
12:   $P[v]$ = sortAndRank($M[v]$)
13: end for
14: repeat
15:   for $v \in V$ do
16:     $P^*[v] \leftarrow \mathbf{rankAggregate}_{\mathbf{x}\in\{\mathbf{v}\}\cap\Gamma_{\mathbf{G}}(\mathbf{v})}\mathbf{P[x]}$
17:     $P[v] \leftarrow P^*[v]$
18:   end for
19: until Stabilization of $P^*[v] \forall v$
20: for $v \in V$ do
21:   /* assigning v to communities */
22:   for $c \in P[v]$ do
23:     if $|M[v, c] - M[v, P[0]]| \leq \epsilon$ then
24:       $COM(c) \leftarrow COM(c) \cup \{v\}$
25:     end if
26:   end for
27: end for
28: return $\mathcal{C}$
```

4.2 Implementation issues

The LICOD algorithm is implemented using the *igraph* graph analysis toolkit [15]. We give next some details about the implementation of each of the main steps of the proposed algorithm.

4.2.1 Function $isLeader()$

One simple idea to distinguish leaders from follower nodes is to compare nodes centralities. Actually, leader nodes are expected to have higher centrality (whatever the centrality is) than ordinary nodes. Different centrality measures can be used. In our experiments, we have tested the following two basic centralities:

Degree centrality (denoted dc): This is given by the proportion of nodes directly connected to the target node. Formally, the degree centrality of a node v is given by: $\mathrm{dc}(v) = \frac{d_G(v)}{n_G-1}$. The computation complexity is $\mathcal{O}(n_G)$.
Betweenness centrality $\mathrm{BC}(v)$: The is given by the fraction of all-pairs shortest paths that pass through the target node. Formally, the betweenness centrality of a node v is given by $\mathrm{BC}(v) = \sum_{s,t\in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$ 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. The best known algorithm for computing this centrality has a computation complexity $\mathcal{O}(n_G.m_G + (n_G)^2 \log(n_G))$ [9].

The first centrality is local-computed metric while the later captures global proprieties of the network. A node is identified as a leader if its centrality is greater or equal to $\sigma \in [0, 1]$ percent of its neighbors centralities. The rational behind introducing the σ 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 σ. More σ is high fewer are the leaders.

4.2.2 Function $computecommunitiesleaders$

Two leaders are grouped in the 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 σ, δ determines in somehow, the number of communities detected by the algorithm.

4.2.3 Function $memebership(v, c)$

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.

$$membership(v, c) = \frac{1}{(min_{x \in COM(c)} SPath(v, x)) + 1} \tag{6}$$

It is easy to see that the previous function takes values in the range $\left[\frac{1}{Diameter(G)}, 1\right]$. 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 of all its leaders is equal to 1.

4.2.4 Rank aggregation approaches

Let S be a set of elements to be ranked by a set of m rankers. We denote by S^{r_i} the ranking provided by ranker r_i. $\{S^{r_1}, \dots, S^{r_m}\}$ is a set of all ranks provided by the m rankers. Notice that each list S^{r_i} represents a permutation of elements of S. An optimal ensemble ranking approach seeks for a permutation σ that has the minimum number of pairwise disagreements with all input ranks S^{r_i} [3,12,19,80]. The *Kendall Tau* distance computes the pairwise disagreement between two ranks defined over the same set of elements S. This is formally defined as follows:

$$\mathcal{K}(\pi, \sigma) = \sum_{x,y \in S} d_{\pi,\sigma}(x, y) \tag{7}$$

where:

$$d_{\pi,\sigma}(x, y) = \begin{cases} 0 & \text{if } \pi \text{ and } \sigma \text{ rank } x \text{ and } y \text{ in the same order} \\ 1 & \text{otherwise} \end{cases}$$

This problem has been extensively studied in the context of social choice algorithms [3]. Early work tackling this problem goes back the French revolution epoch with the work of Borda [8] and Marquis de Condorcet [16] striving to define a fair election rule. Rank aggregation approaches can be classified into two classes: position-based approaches and order-based ones [12].

One well-known position-based method is Borda's method [8]: A Borda score is computed for each element in the lists. For a set of complete ranked lists $L = [L_1, L_2, L_3, \dots, L_k]$, the Borda's score of an element i and a list L_k is given by: $B_{L_k}(i) = \{\text{count}(j) | L_k(j) < L_k(i) \,\&\, j \in L_k\}$. The total Borda's score for an element is then: $B(i) = \sum_{t=1}^{k} B_{L_t}(i)$. Elements are sorted in function of their total Borda score with random selection in case of ties.

Kemeny approaches are well-known order-based approaches. A Kemeny optimal aggregation [35] is an aggregation that has the minimum number of <div> pairwise disagreement as computed by the *Kendall tau* distance [43]. Computing an optimal Kemeny aggregation is NP-hard starting from a list of four candidates. Different approximate Kemeny aggregation approaches have been proposed in the literature. The basic idea of all proposed approximate Kemeny aggregation is to sort the candidate list, using standard sorting algorithms, but using a non-transitive comparison relationship between candidates. This relation is the following: s_i is preferred to s_j, noted $s_i \succ s_j$, if the majority of rankers ranks s_i before s_j. Since the $\succ$ relation is not transitive, different sorting algorithms will provide different rank aggregations with different proprieties. In [19] authors propose a *local Kemeny* aggregation applying a bubble sort algorithm. In [53] authors propose an *approximate Kemeny* aggregation applying quick sort algorithm .

4.2.5 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 2, a node is assigned simultaneously to communities for which its membership is ϵ-far from the membership degree to the top-ranked community. The ϵ threshold controls the degree of desired overlapping in identified communities. However, putting ϵ to 0 may still results in having overlapping communities since for a given node different communities may have the same membership degree.

5 Experimentation

5.1 Evaluation on benchmark networks

In a first experiment, we evaluate the proposed approach on a set of four widely used benchmark networks for which a ground-truth decomposition into communities is known. These networks are the following:

Zachary's karate club This network is a social network of friendships between 34 members of a karate club at a US university in 1970 [91]. Following a dispute, the network was divided into two groups between the club's administrator and the club's instructor. The dispute ended in the instructor creating his own club and taking about half of the initial club with him. The network can hence be divided into two main communities.

Dolphins social network This network is an undirected social network resulting from observations of a community of 62 dolphins over a period of 7 years [50]. Nodes represent dolphins and edges represent frequent associations between dolphin pairs occurring more often than expected by chance. Analysis of the data revealed two main groups.

American college football dataset This dataset contains the network of American football games [23]. The 115 nodes represent teams and the edges represent games between 2 teams. The teams are divided into 12 groups containing around 8–12 teams each and games are more frequent between members of the same group. Also teams that are geographically close but belong to different groups are more likely to play one another than teams separated by a large distance. Therefore, in this dataset groups can be considered as known communities.

American political books This is a political books co-purchasing network. Nodes represent books about US politics sold by the online bookseller *Amazon.com*. Edges represent frequent co-purchasing of books by the same buyers, as indicated by the "customers who bought this book also bought these other books" feature on Amazon. Books are classified into three disjoint classes: liberal, neutral or conservative. The classification was made separately by Mark Newman based on a reading of the descriptions and reviews of the books posted on Amazon.

Table 1 Basic topological characteristics of selected benchmark networks

Dataset	# Nodes	# Edges	# Real communities
Zachary	34	78	2
Football	115	616	11
US Politics	100	411	2
Dolphin	62	159	2

Figure 1 shows the structure of the selected networks with real communities indicated by the color code. Table 1 gives the basic characteristics of these networks.

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

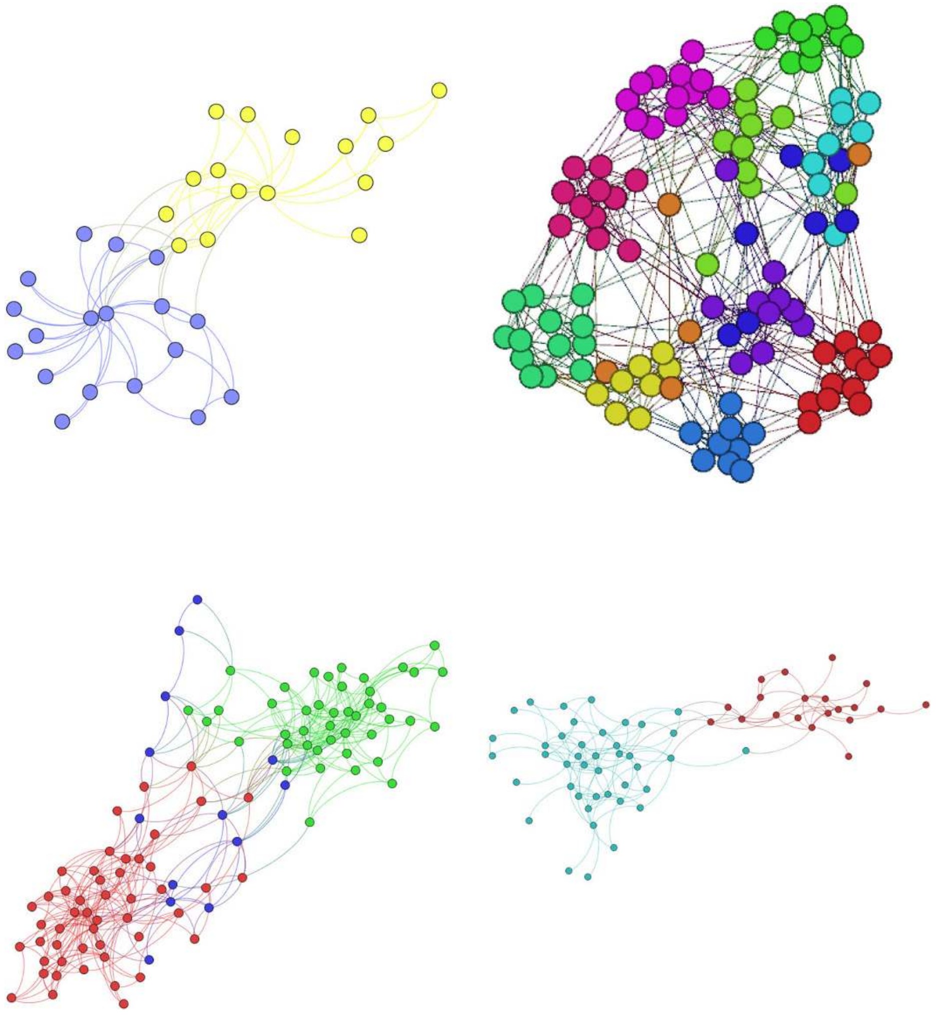
Fig. 1 Real community structure of the four selected benchmark networks. Zachary Karate Club Network [91], Collegae football network [23], US Politics books network [38], Dolphins social network [50]

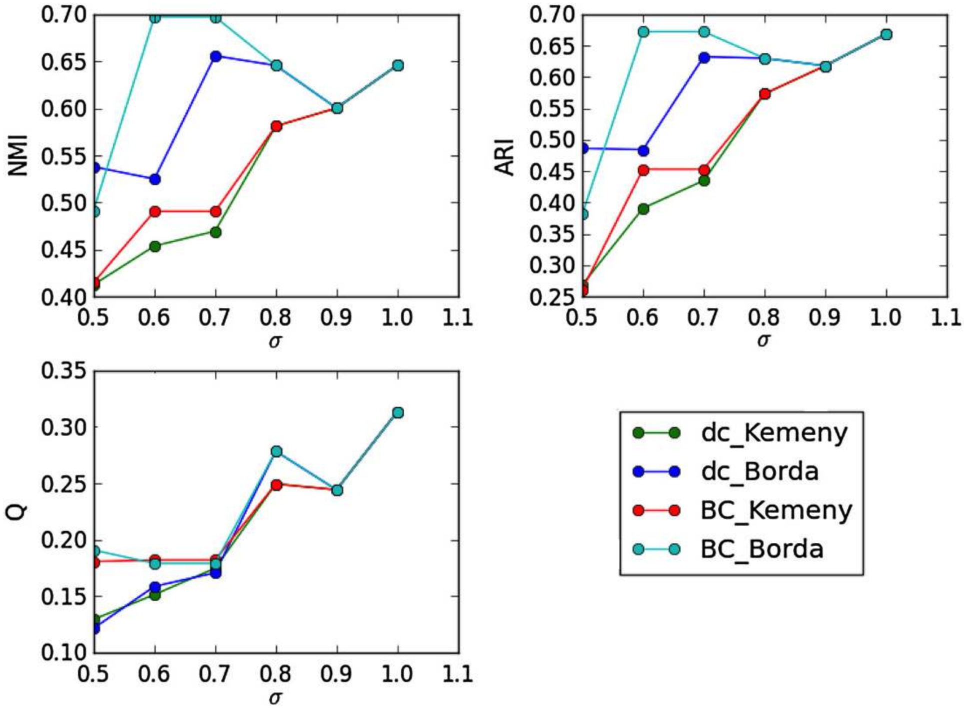

Fig. 2 Performance of applying LICOD to Zachary Karate club network in function of σ in terms of NMI, ARI and the modularity Q

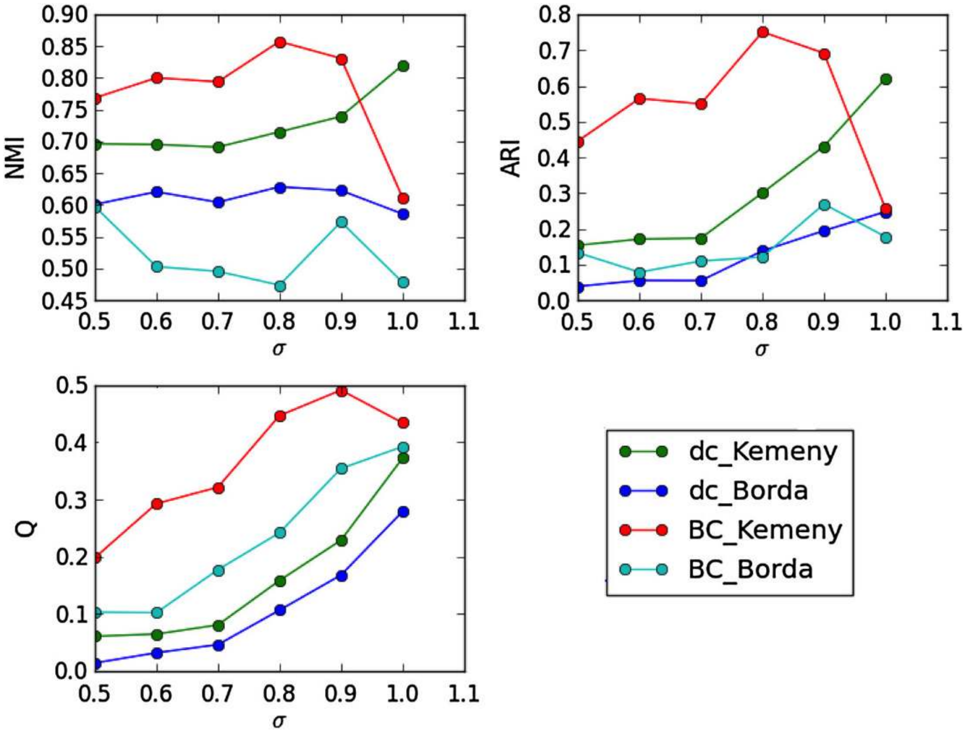

Fig. 3 Performance of applying LICOD to American college football network in function of σ in terms of NMI, ARI and the modularity Q

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

For each configuration, we compute the NMI, ARI and the modularity Q. Figures 2, 3, 4 and 5 show the variations of these metrics, for each dataset, with the variation of σ. We have omitted to show the results with different values of δ since on these datasets the δ value has showed negligible impact on obtained results. The same

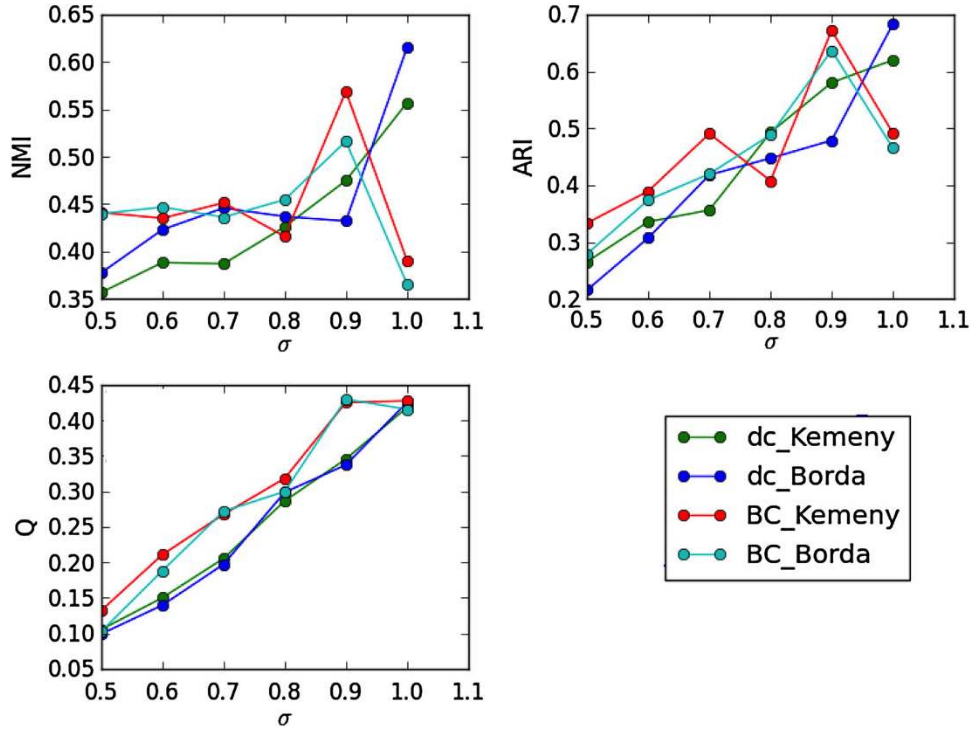

Fig. 4 Performance of applying LICOD to American political books networks in function of σ in terms of NMI, ARI and the modularity Q

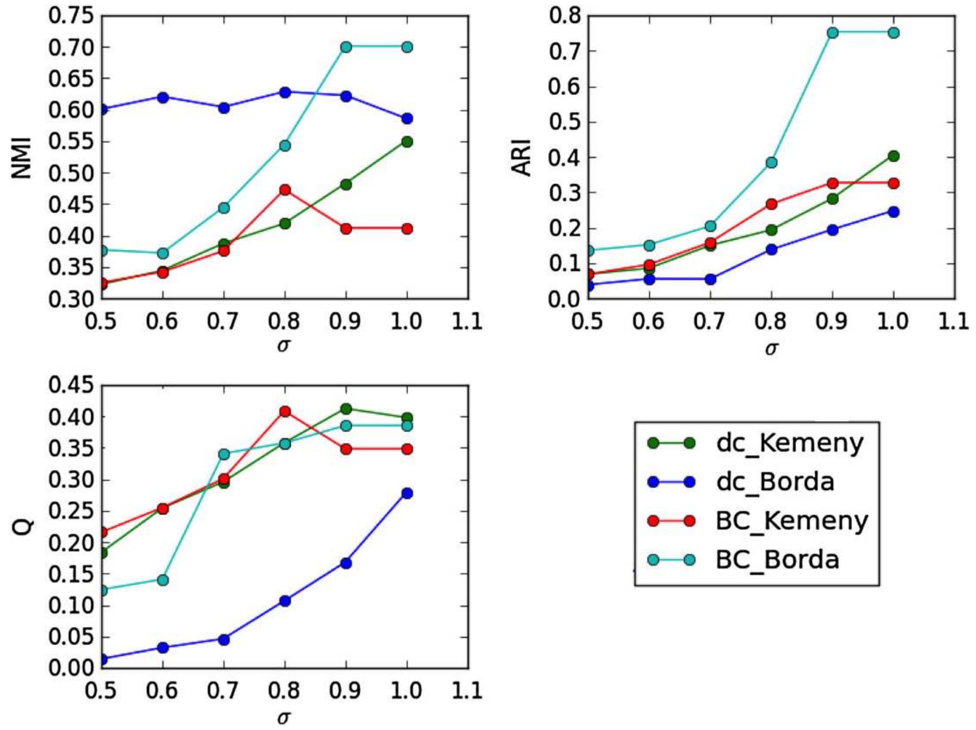

Fig. 5 Performance of applying LICOD to dolphins social network in function of σ in terms of NMI, ARI and the modularity Q

effect was observed for the ϵ parameter. On each figure, we plot four graphics showing the variation of NMI, ARI and Q, for each of the possible four configurations depending on the choice of the used centrality and the voting method.

These results show that the use of the betweenness centrality accelerate slightly the convergence for the right value to obtain. Local Kemeny voting methods out performs that Borda in the case of the football network only and gives comparable results for the US Politics network. Borda gives

Table 2 Comparison of performances of different community detection algorithms

Dataset	Algorithm	NMI	ARI	Q	# Communities
Zachary	Newman	0.57	0.46	0.40	5
	Louvain	0.58	0.46	0.41	4
	Walktrap	0.50	0.33	0.35	5
	LICOD	**0.60**	**0.62**	0.24	3
Football	Newman	0.87	0.77	0.59	10
	Louvain	0.89	0.80	0.60	10
	Walktrap	0.88	0.81	0.60	10
	LICOD	0.83	0.69	0.49	16
US Politics	Newman	0.55	0.68	0.51	5
	Louvain	0.57	0.55	0.52	4
	Walktrap	0.53	0.65	0.50	4
	LICOD	**0.68**	0.67	0.42	6
Dolphins	Newman	0.55	0.39	0.51	5
	Louvain	0.51	0.32	0.51	5
	Walktrap	0.53	0.41	0.48	4
	LICOD	0.41	0.32	0.35	**2**

Bold values indicate the best score by LICOD

good results only for the Dolphins network using also the betweenness centrality.

Increasing ϵ results in diminishing the NMI and ARI. This can be explained by the fact that high value of ϵ increases the overlapping degree of obtained communities while real communities we have here are all disjoint.

The best results are obtained for σ around 0.8, 0.9. This argues for the validity the idea of introducing the σ 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. We notice that the dynamic curves differ from one network to another, and this is closely related to the specificities of each network. The choice of a configuration of the proposed algorithm in function of the properties of the target network constitutes one interesting topic to cope with.

We also compared the results of our algorithm with results obtained by well-known algorithms: The Newman–Girvan algorithm [58], the WalkTrap algorithm [70] and the *Louvain* algorithm [7]. The configuration adopted for LICOD is the following: Centrality metric is betweenness centrality, Voting method is local Kemeny, $\sigma = \delta = 0.9$, and $\epsilon = 0$. Table 2 gives obtained results on the four datasets.

These results show that LICOD performs better than the other algorithms for both Zachary and US Politics networks. It also gives competitive results in the other two networks. This could be explained by the absence of leaders in these two networks, which makes the communities detection task more difficult.

These results show also that the modularity metric does not correspond to the best decomposition into communities as measured by both NMI and ARI. For instance, the *Louvain* method obtains always the best modularity (even better than the modularity of the ground-truth decomposition), however, it is ranked not first according to NMI . Best results are obtained by our approach for high values of σ.

5.2 Data clustering-driven evaluation

We propose here to use the task of data clustering to apply a task-driven evaluation of community detection algorithms. The basic idea is to transform a data clustering problem into a community detection one. Some earlier work has already applied community detection algorithms to the clustering task [17]. Figure 6 illustrates the overall approach. First, a relative neighborhood graph (RNG), as defined in [84], is constructed over the set of items to cluster. The choice of RNG graph is motivated by the topological characteristics of these graphs that are connexe and sparse. To build an RNG graph, we first compute a similarity matrix between couple of items in the dataset (Fig. 7). This results in a symmetric square matrix of size $n \times n$ where n is the number of items in the dataset. A RNG graph is defined by the following simple construction rule: two points x_i and x_j are connected by an edge if they satisfy the following property:

$$d(x_i, x_j) \leq \max_l \{d(x_i, x_l), d(x_j, x_l)\}, \quad \forall l \neq i, j \tag{8}$$

where $d(x_i, x_j)$ is the distance function. A community detection algorithm is applied on the obtained graph to cluster the

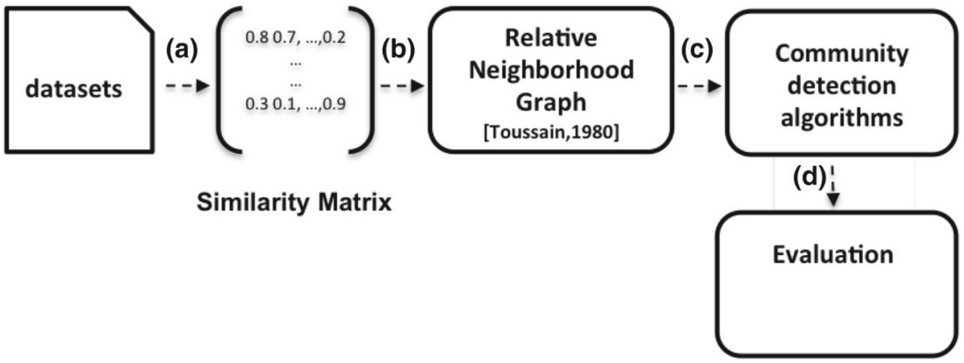

Fig. 6 Applying community detection to data clustering

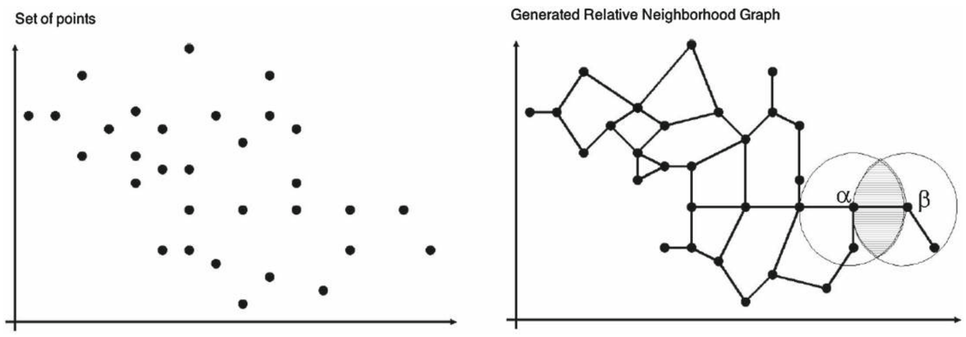

Fig. 7 Example of the generation of a RNG from a cloud of data: α and β are two relatifs neighbors because there is no other node in the intersection of the two circles centered, respectively, in α and β and with radius $d(\alpha, \beta)$

Table 3 Characteristics of used datasets

Dataset	Glass	Iris	Wine	Vehicle	Abalone
#Instances	214	150	178	846	772
#Attributes	10	4	13	18	8
#Classes	7	3	3	4	29

Table 4 Applied basic distance functions

Distance	Formula
Euclidean distance	$dist_{euc}(x, y) = \sqrt{\sum_{i=1}^{n} \lvert x_i - y_i \rvert^2}$
Cosine similarity	$dist_{cos}(x, y) = 1 - \frac{x.y}{\lvert x \rvert \lvert y \rvert}$
Chebyshev distance	$d_{cheb}(x, y) = \max_i(x_i - y_i)$

given examples. Clustering evaluation criteria a-can then be used to compare different algorithms.

We have tested our approach on five classical datasets publicly available from UCI website.[1] The selected datasets are briefly described in Table 3.

We have constructed the different RNG graphs on these datasets using the following classical distance cited in Table 4.

Table 5 shows basic topological characteristics of obtained graphs. We can see that these graphs have some characteristics of real networks such as the small diameter and low density. However, the Chebyshev distance induces dense graphs though the obtained clustering coefficient is also high. We have also obtained graphs with a relatively high transitivity.

Based on these results, we have applied the community detection algorithms on RNG graphs defined by the Cosine distance function. We apply on the above generated graphs four different community detection algorithms: Louvain [7], the Newman–Girvan algorithm, the Walktrap algorithm and LICOD. Results are evaluated in terms of *NMI*, and *ARI* computed in function of the real classes defined in each dataset. We compute also the modularity Q to show that it does not always reflect the true quality of the community. Results given in the Table 6 show that LICOD is ranked first for the two datasets: wine and abalone. It gives competitive results for the other datasets.

[1] http://archive.ics.uci.edu/ml/datasets.html.

Table 5 Topological characteristics of obtained RNG graphs

Dataset	Feature	Euclidean	Chebyshev	Cosine
Iris	#Edges	382	2,468	426
	Diameter	33	14	25
	Average degree	5.09	32.9	5.68
	Density	0.034	0.220	0.038
	Transitivity	0.055	0.340	0.011
Glass	#Edges	558	7,786	552
	Diameter	21	8	24
	Average degree	5.21	72.76	5.15
	Density	0.024	0.341	0.024
	Transitivity	0.0139	0.252	0.011
Wine	#Edges	380	514	438
	Diameter	102	84	59
	Average degree	4.26	5.77	4.92
	Density	0.024	0.032	0.027
	Transitivity	0	0.178	0
Vehicle	#Edges	2,598	4,072	2,764
	Diameter	63	54	45
	Average degree	6.14	9.62	6.53
	Density	0.007	0.011	0.007
	Transitivity	0.002	0.091	0
Abalone	#Edges	2,542	89,338	2,158
	Diameter	38	22	50
	Average degree	6.58	231.44	5.59
	Density	0.008	0.30	0.007
	Transitivity	0	0.49	0

6 Conclusion

In this work, we contribute to the state of the art on community detection in complex networks by:

- Providing a new efficient algorithm for computing (eventually overlapping) communities.

Table 6 Performance of LICOD vs Louvain, Walktrap, Newman–Girvan algorithms

Dataset	Algorithm	NMI	ARI	Q	#Communities
Iris	Newman	0.66	0.44	0.72	9
	Louvain	0.59	0.40	0.72	8
	Walktrap	0.64	0.47	0.68	12
	LICOD	0.59	0.42	0.64	8
Glass	Newman	0.45	0.21	0.76	11
	Louvain	0.47	0.21	0.75	12
	Walktrap	0.49	0.15	0.73	22
	LICOD	0.46	0.17	0.70	18
Wine	Newman	0.32	0.14	0.79	11
	Louvain	0.31	0.13	0.79	12
	Walktrap	0.32	0.11	0.77	15
	LICOD	**0.34**	**0.21**	0.72	14
Vehicle	Newman	0.23	0.10	0.79	17
	Louvain	0.25	0.11	0.78	14
	Walktrap	0.23	0.06	0.75	32
	LICOD	0.21	0.05	0.65	41
Abalone	Newman	0.34	0.10	0.83	15
	Louvain	0.35	0.10	0.83	19
	Walktrap	0.33	0.08	0.82	21
	LICOD	**0.44**	0.08	0.70	68

- Proposing a new approach for qualitative community evaluation using classical data clustering tasks.

Results obtained on both small benchmark social network and on clustering problems argue for the capacity of the approach to detect real communities. Future developments we are working 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 adapt the approach for *K*-partite and for multiplex networks [28].

References

1. Adamcsek, B., Palla, G., Farkas, I.J., Derényi, I., Vicsek, T.: Cfinder: locating cliques and overlapping modules in biological networks. Bioinformatics **22**(8), 1021–1023 (2006)
2. Aggarwal, C.C., Reddy, C.K. (eds.) Data clustering: algorithms and applications. CRC Press, Boca Raton (2014)
3. Arrow, K.: Social choice and individual values, 2nd edn. Cowles Foundation, New Haven (1963)
4. Bagrow, J.P.: Evaluating local community methods in networks. J. Stat. Mech. **2008**(5), P05001 (2008)
5. Bagrow, J.P., Bollt, E.M.: A local method for detecting communities. Phys. Rev. E **72**, 046108 (2005)
6. Benchettara, N., Kanawati, R., Rouveirol, C.: Supervised machine learning applied to link prediction in bipartite social networks. In: International Conference on Advances in Social Network Analysis and Mining, ASONAM, pp. 326–330 (2010)
7. Blondel, V.D., Guillaume, J.-L., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. Theory Exp. **2008**, P10008 (2008)
8. Borda, J.C.: Mémoire sur les élections au scrutin. Comptes rendus de l'Académie des sciences, traduit par Alfred de Grazia comme Mathematical Derivation of a election system, Isis, vol. 44, pp. 42–51 (1781)
9. Brandes, U.: A faster algorithm for betweenness centrality. J. Math. Sociol. **25**(2), 163–177 (2001)
10. Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z., Wagner, D.: On modularity clustering. IEEE Trans. Knowl. Data Eng. **20**(2), 172–188 (2008)
11. Chen, J., Zaïane, O.R., Goebel, R.: ASONAM. In: Memon, N., Alhajj, R. (eds.) Local community identification in social networks, pp. 237–242. IEEE Computer Society, Athens (2009)
12. Chevaleyre, Y., Endriss, U., Lang, J., Maudet, N.: A short introduction to computational social choice. SOFSEM 2007: Theory and Practice of Computer Science, pp. 51–69 (2007)
13. Clauset, A.: Finding local community structure in networks. Phys. Rev. E. **72**, 026132 (2005)
14. Cordasco, G., Gargano, L.: Label propagation algorithm: a semi-synchronous approach. IJSNM **1**(1), 3–26 (2012)
15. Csardi, G., Nepusz, T.: The igraph software package for complex network research. Int. J. Complex Syst. 1695 (2006)
16. de Condorcet, M.: Essai sur l'application de l'analyse à la probabilité des decisions rendues à la pluralité des voix (1785)
17. de Oliveira, T.B.S., Zhao, L., Faceli, K., de Carvalho, A.C.P.L.F.: Data clustering based on complex network community detection. In: IEEE Congress on Evolutionary Computation, 2008 (CEC 2008), 1-6 June 2008, Hong Kong, pp. 2121–2126 (2008)
18. Du, N., Wang, B., Wu, B., Wang, Y.: Overlapping community detection in bipartite networks. In: IEEE WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2008 (WI-IAT '08), vol 1, 9-12 Dec 2008, pp. 176–179 (2008)
19. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the Web. In: Proceedings of the 10th international conference on World Wide Web (WWW '01). ACM, New York, NY, USA, pp. 613–622 (2001)
20. Flake, G.W., Lawrence, S., Giles, C.L., Coetzee, F.: Self-organization and identification of web communities. IEEE Comput. **35**(3), 66–71 (2002)
21. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010)
22. Fred, A.L.N., Jain, A.K.: Robust data clustering. In: CVPR. Proceedings of IEEE Computer Society(2), pp. 128–136 (2003)
23. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. PNAS **99**(12), 7821–7826 (2002)
24. Good, B.H., de Montjoye, Y.A., Clauset, A.: The performance of modularity maximization in practical contexts. Phys. Rev. **E(81)**, 046106 (2010)
25. Grabowski, S., Bieniecki, W.: Tight and simple web graph compression. CoRR, abs/1006.0 (2010)
26. Guimerà, R., Sales-Pardo, M., Amaral, L.A.N.: A network-based method for target selection in metabolic networks. Bioinformatics **23**(13), 1616–1622 (2007)
27. Hartigan, J.A., Wong, M.A.: Appl. Stat. **28**, 100–108 (1979)
28. Hmimida, M., Kanawati, R.: A seed-centric algorithm for community detection in multiplex networks. In: First European Conference on Social Network Analysis, Barcelona (2014)

29. Huang, Q., White, T., Jia, G., Musolesi, M., Turan, N., Tang, K., He, S., Heath, J.K., Yao, X.: PPSN (2). In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) Community detection using cooperative co-evolutionary differential evolution. Lecture Notes in Computer Science, pp. 235–244. Springer, Berlin (2012)
30. Hubert, L., Arabie, P.: Comparing partitions. J. Classif. **2**(1), 192–218 (1985)
31. Jin, D., He, D., Liu, D., Baquero, C.: Genetic algorithm with local search for community mining in complex networks. In: ICTAI (1), pp. 105–112. IEEE Computer Society (2010)
32. Kanawati, R.: LICOD: Leaders identification for community detection in complex networks. In: SocialCom/PASSAT, pp. 577–582 (2011)
33. Kanawati, R.: On applying ensemble ranking to local community identication in complex networks. In: Perner, P. (ed.) Proceedings of the 10th International Conference on Machine Learning and Data Mining (MLDM). Springer, New York (2014)
34. Kanawati, R.. Seed-centric approaches for community detection in complex networks. In: Meiselwitz, G., (ed.) 6th International Conference on Social Computing and Social Media, volume LNCS 8531, pp. 197–208, Crete, Greece. Springer, New York (2014)
35. Kemeny, J.G.: Mathematics without numbers. Daedalus **88**, 571–591 (1959)
36. Khorasgani, R.R., Chen, J., Zaiane, O.R.: Top leaders community detection approach in information networks. In: 4th SNA-KDD Workshop on Social Network Mining and Analysis, Washington D.C. (2010)
37. Kleinberg, J.M.: NIPS. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) Small-world phenomena and the dynamics of information, pp. 431–438. MIT Press, Cambridge (2001)
38. Krebs, V. Political books network. http://www.orgnet.com
39. Lambiotte, R.. Multi-scale modularity in complex networks. In: WiOpt, pp. 546–553. IEEE (2010)
40. Lancichinetti, A., Fortunato, S.: Limits of modularity maximization in community detection. Phys. Rev. E. **84**, 066122 (2011)
41. Lancichinetti, A., Fortunato, S.: Consensus clustering in complex networks. Sci. Rep. **2** (2012)
42. Lancichinetti, A., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Phys. Rev. E **4**, 046110 (2008)
43. Lapata, M.: Automatic evaluation of information ordering: Kendall's Tau. Comput. Linguist. **32**(4), 471–484 (2006)
44. Leung, I.X., Hui, P., Lio, P., Crowcroft, J.: Towards real-time community detection in large networks. Phys. Rev. E. **79**(6), 066107 (2009a)
45. Leung, I.X.Y., Hui, P., Lio, P., Crowcroft, J.: Towards real-time community detection in large networks. Phys. Rev. E **79**(6), 1–10 (2009b)
46. Li, L., Alderson, D., Tanaka, R., Doyle, J. C., Willinger, W.: Towards a theory of scale-free graphs: definition, properties, and implications. Internet Math. **2**(4):431–523 (2005)
47. Li, S., Chen, Y., Du, H., Feldman, M.W.: A genetic algorithm with local search strategy for improved detection of community structure. Complexity **15**(4), 53–60 (2010)
48. Liu, X., Murata, T.: Community detection in large-scale bipartite networks. In: Web intelligence, pp. 50–57. IEEE (2009)
49. Lou, H., Li, S., Zhao, Y.: Detecting community structure using label propagation with weighted coherent neighborhood propinquity. Phys. A Stat. Mech. Appl. **392**(14), 3095–3105 (2013)
50. Lusseau, D., Schneider, K., Boisseau, O.J., Haase, P., Slooten, E., Dawson, S.M.: The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. Behav. Ecol. Sociobiol. **54**, 396–405 (2003)
51. Mancoridis, S., Mitchell, B.S., Rorres, C., Chen, Y.-F., Gansner, E.R.: Using automatic clustering to produce high-level system organizations of source code. In: IWPC, pp. 45–53 (1998)
52. Meila, M.: COLT. In: Schölkopf, B., Warmuth, M.K. (eds.) Comparing clusterings by the variation of information. Lecture Notes in Computer Science, pp. 173–187. Springer, New York (2003)
53. Melville, P., Subbian, K., Meliksetian, E., Perlich, C.: A predictive perspective on measures of influence in networks. In: 5th Annual Machine Learning Symposium, pp. 1–5 (2010)
54. Mirshahvalad, A., Lindholm, J., Derlen, M., Rosvall, M.: Significant communities in large sparse networks. PLoS ONE **7**(3): e33721 (2012). doi:10.1371/journal.pone.0033721
55. Mucha, P.J., Richardson, T., Macon, K., Porter, M.A., Onnela, J.-P.: Community structure in time-dependent, multiscale, and multiplex networks. Science **328**(5980), 876–878 (2010)
56. Murata, T.: Detecting communities from tripartite networks. In: Rappa, M., Jones, P., Freire, J., Chakrabarti, S. (eds.) WWW, pp. 1159–1160. ACM (2010)
57. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. Phys. Rev. E **69**(6), 066133 (2004)
58. Newman, M.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E **69**, 02613:1–022613:15 (2004)
59. Ng, A., Jordan, M., Weiss, Y.: On spectral clustering: analysis and an algorithm. In: Advances in neural information processing systems (NIPS), pp. 849–856, Vancouver, Canada (2001)
60. Nguyen, X.V., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: Danyluk, A.P., Bottou, L., Littman, M.L. (eds.) ICML, volume 382 of ACM International Conference Proceeding Series, p. 135. ACM (2009)
61. Orgaz, G.B., Menéndez, H.D., Camacho, D.: Adaptive k-means algorithm for overlapped graph clustering. Int. J. Neural Syst. **22**(5), 9 (2012)
62. Ovelgönne, M.: ASONAM. In: Rokne, J.G., Faloutsos, C. (eds.) Distributed community detection in web-scale networks, pp. 66–73. ACM, New York (2013)
63. Ovelgönne, M., Geyer-Schulz, A.: Cluster cores and modularity maximization. In: ICDM Workshops, pp. 1204–1213 (2010)
64. Palla, G., Derônyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping modular structure of protein interaction networks. FEBS J. **272**, 434 (2005)
65. Papadopoulos, S., Kompatsiaris, Y., Vakali, A.: A graph-based clustering scheme for identifying related tags in folksonomies. In: DaWak, pp. 65–76 (2010)
66. Papadopoulos, S., Kompatsiaris, Y., Vakali, A., Spyridonos, P.: Community detection in social media—performance and application considerations. Data Min. Knowl. Discov. **24**(3), 515–554 (2012)
67. Peng, C., Kolda, T. G., Pinar, A.: Accelerating community detection by using k-core subgraphs. CoRR, abs/1403.2226 (2014)
68. Pizzuti, C.: Boosting the detection of modular community structure with genetic algorithms and local search. In: Ossowski, S., Lecca, P., (eds.) SAC, pp. 226–231. ACM (2012)
69. Pizzuti, C.: A multiobjective genetic algorithm to find communities in complex networks. IEEE Trans. Evol. Comput. **16**(3), 418–430 (2012b)
70. Pons, P., Latapy, M.: Computing communities in large networks using random walks. J. Graph Algorithms Appl. **10**(2), 191–218 (2006)
71. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E **76**, 1–12 (2007)
72. Rand, W.M.: Objective criteria for the evaluation of clustering methods. J. Am. Stat. Assoc. **66**, 846–850 (1971)
73. Rosvall, M., Axelsson, D., Bergstrom, C.T.: The map equation. Eur. Phys. J. Spec. Top. **13**, 178 (2009)
74. Seifi, M., Guillaume, J.L.: WWW (Companion Volume). In: Mille, A., Gandon, F.L., Misselis, J., Rabinovich, M., Staab, S. (eds.)

Community cores in evolving networks, pp. 1173–1180. ACM, New York (2012)

75. Seshadri, M., Machiraju, S., Sridharan, A., Bolot, J., Faloutsos, C., Leskove, J.: Mobile call graphs: beyond power-law and lognormal distributions. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '08). ACM, New York, NY, USA, pp. 596–604 (2008)
76. Shah, D., Zaman, T.: Community detection in networks: The leader-follower algorithm. In: Workshop on Networks Across Disciplines in Theory and Applications, NIPS (2010)
77. Shi, C., Cai, Y., Fu, D., Dong, Y., Wu, B.: A link clustering based overlapping community detection algorithm. Data Knowl. Eng. **87**, 394–404 (2013)
78. Staudt, C., Meyerhenke, H.: Engineering high-performance community detection heuristics for massive graphs. In: ICPP, pp. 180–189. IEEE (2013)
79. Strehl, A., Ghosh, J.: Cluster ensembles: a knowledge reuse framework for combining multiple partitions. J. Mach. Learn. Res. **3**, 583–617 (2003)
80. Subbian, K., Melville, P.: Supervised rank aggregation for predicting influencers in twitter. In: SocialCom/PASSAT, pp. 661–665. IEEE (2011)
81. Subelj, L., Bajec, M.: Robust network community detection using balanced propagation. Eur. Phys. J. B **81**(3), 353–362 (2011)
82. Sun, P.-G., Gao, L.: A fast iterative-clique percolation method for identifying functional modules in protein interaction networks. Front. Comput. Sci. China **3**(3), 405–411 (2009)
83. Tang, L., Liu, H.: Community detection and mining in social media. Synthesis Lectures on Data Mining and Knowledge Discovery. Morgan & Claypool Publishers, San Rafael (2010)
84. Toussaint, G., Bhattacharya, B.K.: Optimal algorithms for computing the minimum distance between two finite planar sets. Pattern Recognit. Lett. **2**, 79–82 (1983)
85. Tsironis, S., Sozio, M., Vazirgiannis, M.: Accurate spectral clustering for community detection in mapreduce. In: Airoldi, E., Choi, D., Clauset, A., El-Arini, K., Leskovec, J. (eds.) Frontiers of network analysis: methods, models, and applications. Lake Tahoe, NIPS workshop (2013)
86. Verma, A., Butenko, S.: Graph partitioning and graph clustering. In: Bader, D.A., Meyerhenke, H., Sanders, P., Wagner, D. (eds.) Network clustering via clique relaxations: a community based approach. Contemporary Mathematics, pp. 129–140. American Mathematical Society, Providence (2012)
87. Whang, J.J., Gleich, D.F., Dhillon, I.S.: CIKM. In: He, Q., Iyengar, A., Nejdl, W., Pei, J., Rastogi, R. (eds.) Overlapping community detection using seed set expansion, pp. 2099–2108. ACM, New York (2013)
88. Xie, J., Kelley, S., Szymanski, B.K.: Overlapping community detection in networks: the state-of-the-art and comparative study. ACM Comput. Surv. **45**(4), 43 (2013)
89. Xie, J., Szymanski, B.K.: Community detection using a neighborhood strength driven label propagation algorithm. In: Proceedings of IEEE Network Science Workshop (2011)
90. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. In: Zaki, M.J., Siebes, A., Yu, J.X., Goethals, B., Webb, G.I., Wu, X. (eds.) ICDM, pp 745–754. IEEE Computer Society (2012)
91. Zachary, W.W.: An information flow model for conflict and fission in small groups. J. Anthropol. Res. **33**, 452–473 (1977)
92. Zhang, Y., Wang, J., Wang, Y., Zhou, L.: KDD. In: Iv, J.F.E., Fogelman-Soulié, F., Flach, P.A., Zaki, M.J. (eds.) Parallel community detection on large networks with propinquity dynamics, pp. 997–1006. ACM, New York (2009)