Expert Systems With Applications 54 (2016) 121–135

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A consensus graph clustering algorithm for directed networks

Camila Pereira Santos, Desiree Maldonado Carvalho, Mariá C.V. Nascimento[*]

Instituto de Ciência e Tecnologia, Universidade Federal de São Paulo (UNIFESP), Av. Cesare M. G. Lattes, 1201, Eugênio de Mello, São José dos Campos, SP 12247-014, Brazil

A R T I C L E I N F O

Keywords:
Heuristic method
Community detection
Directed networks

A B S T R A C T

Finding groups of highly related vertices in undirected graphs has been widely investigated. Nevertheless, a very few strategies are specially designed for dealing with directed networks. In particular, strategies based on the maximization of the modularity adjusted to overcome the resolution limit for directed networks have not been developed. The analysis of the characteristics of the clusters produced by these approaches is highly important since among the most used strategies for detecting communities in directed networks are the modularity maximization-based algorithms for undirected graphs. Towards these remarks, in this paper we propose a consensus-based strategy, named ConClus, for providing partitions for directed networks guided by the adjusted modularity measure. In the computational experiments, we compared ConClus with benchmark strategies, including Infomap and OSLOM, by using hundreds of LFR networks. ConClus outperformed Infomap and was competitive with OSLOM even for graphs with high mixture index and small-sized clusters, to which modularity-based algorithms have limitations. ConClus outperformed all algorithms when considering the networks with the highest average and maximum in-degrees among the networks used in the experiments.

1. Introduction

Detecting communities in networks, also known as graph clustering, plays an important role in pattern recognition research area. Roughly, it enables the identification of groups of highly related vertices in a graph, also known as clusters. It is a relevant issue, for example, to look into the communities that represent the functional activities of the brain, known as brain networks (Park & Friston, 2013). One reason is that, in some surgeries, this knowledge might enable a better assessment about the areas of the brain related to motor skills. Regardless the distance between two regions of the brain, they might be strongly related according to the functional activities.

In spite of most community detection strategies being designed for undirected networks, several applications to which community detection is highly relevant are better modeled in directed networks. We may cite, for example, social, informational, biological and neuroscience networks. For defining communities in these networks, the most employed approach consists in ignoring the arc directions of the networks to make use of strategies designed for undirected graphs.

However, Malliaros and Vazirgiannis (2013) point out that important characteristics of the network might be lost with this approach, the reason why arc directions should be considered. One reason is the non-existence of reciprocal relationship between vertices, created after ignoring the arc directions. For example, in citation networks, networks of scientific papers, the links are obviously directed and without symmetric arcs, since it is rare an article to cite and to be cited by the same paper. Consequently, to detect communities by ignoring the arcs directions could lead to communities different from the expected for a correct analysis.

Additionally, the uncertainty about the clustering structure of undirected networks has led the proposal of many new algorithms for detecting communities. Consequently, to determine which algorithm to adopt for general applications is hard, as pointed out in Lancichinetti and Fortunato (2009). Lancichinetti and Fortunato (2009) assess the quality of a number of community detection algorithms for undirected networks to attest which of them have a good performance. They performed the experiments using artificial graphs, known as LFR networks (Lancichinetti, Fortunato, & F, 2008), whose expected partitions are known. According to the experiments carried out by the authors, Infomap (Rosvall, Bergstrom, 2010) appears as the best algorithm since it outperformed all tested strategies, including the modularity maximization-based algorithms as, e.g., the Louvain method (Blondel, Guillaume, Lambiotte, & Lefebvre, 2008).

[*] Corresponding author. Tel.:+55 12 33099595; fax.: +55 12 3309 9500.
E-mail addresses: camila.santos@unifesp.br (C.P. Santos), dmcarvalho@unifesp.br (D.M. Carvalho), mcv.nascimento@unifesp.br (M.C.V. Nascimento).

http://dx.doi.org/10.1016/j.eswa.2016.01.026
0957-4174/© 2016 Elsevier Ltd. All rights reserved.

It is worth to underline that modularity maximization-based algorithms, extensively adopted for the task of providing clusterings, are also known by their tendency to fail in detecting partitions with numerous small-sized clusters. As a consequence, these strategies tend to merge communities that represent individual groups (Fortunato & Barthélemy, 2007). A promising alternative for the modularity measure suggested in Reichardt and Bornholdt (2006) is the target of the study of this paper. Reichardt and Bornholdt (2006) proposed to fine-tune modularity through the inclusion of a parameter, here called resolution parameter. In Carvalho, Resende, and Nascimento (2014), the authors firmly establish the connection between some graph characteristics and the resolution parameter, for automatically adjusting it. For this, they proposed the use of a neural network, trained according to the topology of the graph. For each input graph, the output of the neural network is an interval of values, expected to be the most suitable options for defining the resolution parameter.

Among these algorithms for undirected networks, some already have its adaptation to tackle directed networks as, e.g., Infomap. More recently, Lancichinetti et al. (2011) proposed the Order Statistics Local Optimization Method (OSLOM), that outperformed Infomap in both undirected and directed LFR networks.

Bearing in mind the discussion outlined, this paper presents:

- A robust consensus clustering, named ConClus, based on arc contractions with a memory mechanism resulting in a strategy that unifies both diversification and intensification paradigms to detect communities in directed networks;
- A study about the performance of this modularity-based algorithm with the resolution parameter adjusted by a neural network trained according to the topology of a number of directed LFR networks;
- An experimental analysis of ConClus using 600 LFR networks with different sizes (from 1000 to 5000 nodes), mixture degrees (from 0.1 to 0.8), community sizes (small and large) and average/maximum in-degrees (20/50 and 40/100);
- A comparative analysis of the results achieved by ConClus with those obtained by the benchmark community detection algorithms: OSLOM, Infomap and the Label Propagation (LP);
- The competitive results of ConClus considering directed LFR networks with average/maximum in-degrees 20/50 in comparison to OSLOM and its better performance over Infomap and LP;
- The results indicating that ConClus outperformed all algorithms considering directed LFR networks with average/maximum in-degrees 40/100;
- A case study with real networks showing that ConClus achieved very accurate results for an undirected network and a directed network.

The remaining of this paper is organized as follows. Section 2 shows a brief review of related works about community detection algorithms in directed networks. Section 2.2.1 presents a comprehensive discussion about the modularity measure and the resolution limit focusing on directed networks. Section 3 presents the proposed strategy. Section 4 shows the computational experiments with real and artificial directed networks. To sum up, Section 5 presents the final remarks and directions of future works.

2. Related works

This section briefly reviews the main approaches for directed networks. The reader interested in a detailed survey in this topic, we indicate the reading of Malliaros and Vazirgiannis (2013). The most recent references are underlined in this paper. Before going into detail about the literature review, this section starts presenting some basic graph theory definitions to be used throughout the paper.

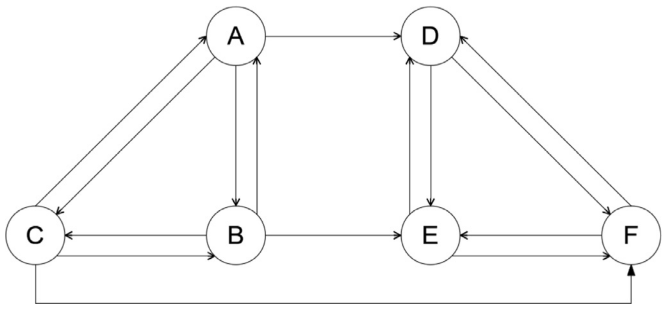


Fig. 1. A directed network composed by two natural communities.

2.1. Basic terminology and background

In this paper, a directed graph $G = (V(G), E(G))$ is represented by a set of vertices or nodes, $V(G)$, and a set of arcs, $E(G)$, where each arc $e := (v_i, v_j) \in E(G)$ is associated with an ordered pair of vertices of G. Additionally, a given arc $(v_i, v_j) \in E(G)$ has as ends the vertices v_i and v_j, where v_i is called the tail, v_j is called the head of the arc and $i, j \in \{1, 2, \ldots, |V(G)|\}$. The number of vertices and arcs of G are denoted in this paper by $n(G)$ and $m(G)$, respectively. The degree of a vertex v_i from G, $d_G(v_i)$, corresponds to the number of times v_i is an end vertex. The in-degree and out-degree of a vertex v_i from G, here called, respectively, $d_G^-(v_i)$ and $d_G^+(v_i)$, correspond to the number of times that a vertex v_i appears as an arc head and arc tail in G. A graph induced by a set of vertices $X \subseteq V$ is denoted by $G[X]$. $\mathcal{N}^-(v_i)$ and $\mathcal{N}^+(v_i)$ are the sets of vertices where v_i is an arc head and tail, respectively. Let $e' \cap e$, where e and $e' \in E(G)$, be the coincident end-vertices of the arcs e and e'.

The pattern recognition in graphs may be performed by identifying their groups of highly related vertices. For this, one way is to find communities through graph clustering algorithms. Among them, we underline those guided by evaluation measures that quantify the clustering quality. The definition of a clustering relies on the k-way partition of the vertex set. Let $\mathcal{C} = \{V_1, V_2, \ldots, V_k\}$, with $1 \le k \le n$, be a k-way partition of $V(G)$. The induced graph $G[\mathcal{C}] = (V(G), E(G[\mathcal{C}]))$, where $E(G[\mathcal{C}]) := \bigcup_{i=1}^{k} E(G[V_i])$ defines a graph clustering.

2.2. Community detection in directed networks

Malliaros and Vazirgiannis (2013) present in their survey a good overview of the existing approaches for detecting communities in directed networks. Although the relevance of the topic, they highlight the lack of a consensual general definition for this problem. In interpreting the problem as detecting a group of highly related vertices, what would be "highly related vertices"? What type of relations are expected inside the communities? To formally answer these questions is the first challenge the authors point up as suggestions for future works.

Consequently, it is common to approach the community detection in directed networks by simply ignoring arc directions. However, there is a strong evidence that, depending on the networks, this approach might fail in describing important characteristics of the reciprocity of the network links. Figs. 1 and 2 display an example of a directed network that, if having its arc directions ignored, algorithms may produce incorrect communities. Fig. 1 presents the directed network composed by two communities: {A, B, C} and {D, E, F}. However, in the network obtained by ignoring arc directions, presented in Fig. 2, it is not clear whether there is one cluster or the two original clusters, even being the expected communities maximal cliques.

As an attempt to overcome this misinterpretation of the arc directions, another approach for dealing with a directed network is

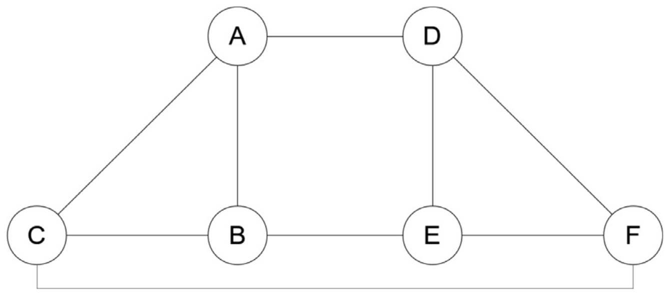


Fig. 2. Network transformation to an undirected graph.

to map its elements into *XY*-bipartite undirected graphs that preserve the meaning of the orientation of the arcs (Malliaros & Vazirgiannis, 2013). One of the groups of the bipartition must contain all vertices with positive out-degree. The other group, then, must include all vertices with positive in-degree. It is very likely the need for duplicating vertices to participate in both sets. Accordingly, a community detection algorithm for undirected bipartite networks can be employed to find communities in the corresponding *XY*-bipartite undirected graph. According to Malliaros and Vazirgiannis (2013), however, it is not clear if the bipartite undirected graph obtained preserves all the characteristics of the corresponding directed network.

The Infomap algorithm, introduced by Rosvall and Bergstrom (2007), is among the strategies specially proposed to detect communities in directed networks. The key idea behind this strategy relies on the premise of how a cartographer represents a map with compressed spatial information but keeping the most important details. As a consequence, the authors presented a measure, named map equation, that favors communities “compressed” into groups whose information flow inside communities is more natural (rapid and regular). For such, the authors introduce a code to describe the random walking process within the network. Therefore, the interesting duality between the community detection and the compression problem is approached by a greedy search combined with a simulated annealing strategy and with a heat-bath algorithm.

Lancichinetti et al. (2011) developed the algorithm Order Statistics Local Optimization Method (OSLOM) for detecting communities in both undirected and directed networks. OSLOM works by evaluating the statistical significance of the communities. The statistical significance measures the probability of a vertex to have neighbors in a given cluster. The order statistics relies on the inclusion of vertices more strongly related to the vertices of the communities. In the case of directed networks, the score of a vertex is calculated according to the probability of it having outgoing arcs incident to vertices of the given cluster times the probability of arcs from the cluster to have *i* as the head. The experiments performed by Lancichinetti et al. (2011) with directed networks show that OSLOM achieves better results than Infomap.

There are a very few community detection strategies to detect communities in directed networks in the recent literature. More related to the proposed study, Romdhane, Chaabani, Zardi, Group et al. (2013) introduced an intelligent system to perform this task based on ant colony optimization, called ACODIG. According to their experiments, ACODIG obtained better results than the a random walk method suggested in Kim, Son, and Jeong (2010), whose fitness function is based on the modularity measure (Girvan & Newman, 2002) and the PageRank algorithm. These experiments considered two LFR networks with a very low mixture degree.

The very popular evaluation measure modularity, originally proposed for undirected networks, has been adapted to detect communities in directed networks (Arenas, Duch, Fernández, & Gómez, 2007; Leicht & Newman, 2008). Since a variant of modularity is the fitness function of ConClus, the next section goes into detail about the modularity and existing strategies based on this measure.

2.2.1. Modularity-based algorithms

It is well known that a substantial amount of studies has been published on the graph clustering subject, primarily with the purpose of refining the results of the existing algorithms. Major progress towards the topic started after Girvan and Newman (2002) developed a statistical mechanics study that resulted in the widely employed measure known as modularity (Girvan & Newman, 2002). Of the many ways for defining this measure, Eq. (1) presents a formulation.

$$q(\mathcal{C}) = \frac{1}{m(G)} \sum_{i=1}^{k} [m(G[V_i]) - p(G[V_i])] \tag{1}$$

where $p(G[V_i]) = \sum_{\forall v_r \neq v_j \in V_i} \frac{d_G(v_r) d_G(v_j)}{2m(G)} + \sum_{\forall v_r \in V_i} \frac{d_G(v_r)^2}{4m(G)}$ is the expected number of edges between vertices from V_i in a random graph with the same degree sequence as *G*. This formula provides an assessment measure that the higher its value, the better the quality of the partition evaluated. Moreover, the maximum modularity of a partition is 1, whereas the minimum is $-1/2$.

In the literature, it is noteworthy the significant number of algorithms guided by the modularity to provide graph clusterings for undirected networks. As it has been proven that the decision version of the modularity maximization problem is $\mathcal{NP}$-complete (Brandes et al., 2008), heuristics are the best strategies to tackle large scale graphs (Blondel et al., 2008; Fortunato & Barthélemy, 2007).

Leicht and Newman (2008) adapted this measure for directed networks by considering a directed random graph with the same in-degree and out-degree sequences as *G*. In this case, the authors set $p(G[V_i]) = \sum_{\forall v_r, v_j \in V_i} \frac{d^-_G(v_r) d^+_G(v_j)}{m(G)}$. This measure is referred here as directed modularity. Leicht and Newman (2008) also proposed a heuristic based on spectral theory for the maximization of the directed modularity.

Few strategies, however, followed the proposal of the directed modularity. Fortunato and Barthélemy (2007) pointed out a scaling problem in modularity, defined as the resolution limit problem explained next.

Resolution limit Despite the modularity maximization-based algorithms mostly obtain cohesive groups, these strategies do not find the expected clusterings for some types of graphs. Fortunato and Barthélemy (2007) performed a detailed investigation over the modularity measure considering a particular graph topology for which modularity seems to benefit unexpected partitions. In these graphs, the measure might lose important information with regard to the structure of the communities of a given network. This means there are cases in which a partition with a certain number of clusters has a worse modularity than a partition with a smaller number of communities even if the former clustering is that expected.

Consequently, there is the possibility of flaws in identifying communities by modularity maximization-based algorithms depending on the graph scale regarding the size of its natural clusters (Fortunato & Barthélemy, 2007). This condition is the so-called resolution limit. Many studies (Ronhovde & Nussinov, 2010; Traag, Van Dooren, & Nesterov, 2011), in particular, Fortunato and Barthélemy (2007) discuss the issue thoroughly, pointing up the conditions in which the modularity optimization may not provide the expected clustering.

Reichardt and Bornholdt (2006) proposed a variation on the modularity measure, based on the scaling of the expected number of edges between pairs of vertices as an attempt to ensure the detection of small-sized communities in the networks. Eq. (2)

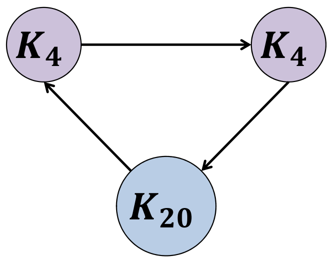


Fig. 3. Ring with three cliques.

presents a formula of this measure.

$$q'(\mathcal{C}) = \frac{1}{m(G)} \sum_{i=1}^{k} [m(G[V_i]) - \lambda p(G[V_i])] \tag{2}$$

For ease of understanding, consider a directed network made up of a ring with three tournaments[1], two of size r and another s-sized, where $s \gg r$. Only one of the vertices of each tournament is an arc head linking to another tournament. In this case, even though the two smallest tournaments clearly represent separate communities, they are regarded as a single community by the modularity maximization problem. Consider $d_G^-(V_i) = \sum_{v \in V_i} d_G^-(v)$ and $d_G^+(V_i) = \sum_{v \in V_i} d_G^+(v)$.

Additionally, it is possible to note that

$$\sum_{\forall v_r, v_j \in V_i} \frac{d_G^-(v_r) d_G^+(v_j)}{m(G)} = \frac{1}{m(G)} \left(\sum_{v_r \in V_i} d_G^-(v_r) d_G^+(V_i) \right)$$
.

As $\sum_{v_r \in V_i} d_G^-(v_r) d_G^+(V_i) = d_G^-(V_i) d_G^+(V_i)$, one can rewrite Eq. (1) as the formulation presented in (3).

$$q'(\mathcal{C}) = \sum_{i=1}^{k} \left[\frac{m(G[V_i])}{m(G)} - \lambda \frac{d_G^-(V_i) d_G^+(V_i)}{(m(G))^2} \right] \tag{3}$$

Let $\mathcal{C}^{(1)}$ denote the clustering in which each tournament represents a different community. Therefore, $\mathcal{C}^{(1)}$ has 3 groups: two r-sized and one with s vertices. Denote $\mathcal{C}^{(1)}$ by $\{V_{r1}, V_{r2}, V_s\}$. Consider $\mathcal{C}^{(2)} = \{V_{2r}, V_s\}$ to be the clustering where the two smallest tournaments are considered a unique community and the s-sized tournament a single community as well. Then, the second clustering possesses two groups: one with $2r$ vertices and another with s vertices. The modularity of $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$ are different only due to their influence whether they are placed together or into separated groups. As $\mathcal{C}^{(1)}$ is the expected clustering, it is necessary that $q'(\mathcal{C}^{(1)}) > q'(\mathcal{C}^{(2)})$. Then,

$$\frac{2}{m(G)} \left(m(G[V_{r1}]) - \lambda \frac{d_G^-(V_{r1}) d_G^+(V_{r1}))}{m(G)} \right)$$
$$> \frac{1}{m(G)} \left(m(G[V_{2r}]) - \lambda \frac{d_G^-(V_{2r}) d_G^+(V_{2r}))}{m(G)} \right)$$

As $m(G[V_{2r}] = 2m(G[V_{r1}]) + 1$, $d_G^+(V_{2r}) = 2d_G^+(V_{r1})$ and $d_G^-(V_{2r}) = 2d_G^-(V_{r1})$, then Eq. (4) holds.

$$m(G) < 2\lambda d_G^-(V_{r1}) d_G^+(V_{r1}) \tag{4}$$

By fixing the value r for the smallest tournaments, it is possible to identify the size of the largest community for which Eq. (4) is addressed. Accordingly, Fig. 3 displays an example following this constraint. The network under consideration is a ring with three tournaments, two of size 4 and one 20-sized.

Consider $\mathcal{C}^{(1)}$ to be the clustering with three groups where each tournament is a community, here denoted as $\mathcal{C}^{(1)}$. Let $\mathcal{C}^{(2)}$ to be the clustering with two groups: one with the tournament with 20 vertices and another with the two 4-sized tournaments placed into a single community. By estimating the modularity of each clustering, there is an undesirable result: $q(\mathcal{C}^{(1)})$ is 0.11495 whereas $q(\mathcal{C}^{(2)})$ is slightly higher than $q(\mathcal{C}^{(1)})$, 0.11750. However, by Eq. (4), it is possible to achieve $q'(\mathcal{C}^{(1)}) > q'(\mathcal{C}^{(2)})$ by using the adjusted modularity with $\lambda > 2.1$.

Bearing in mind this adjusted modularity, next section goes into detail about the proposed strategy.

3. The proposed solution method

The vast majority of the proposed solution methods for either solving the modularity maximization problem in the optimality or for heuristically approaching it requires a high computational cost, mainly for large scale graphs. Additionally, if strictly guided by the modularity measure, the resolution limit influences negatively the quality of the communities with relatively small sizes. In line with this observation, we propose a strategy based on a coarsening strategy, by carefully taking into account the data structures for its implementation and based on the adjusted modularity.

Multi-level algorithms are strategies initially proposed to tackle the graph partitioning problem (Noack & Rotta, 2009). In its original form, a multi-level algorithm is composed by three phases: the coarsening, the partitioning and the refinement. During the coarsening phase, the graph is successively contracted until it reaches a certain size. After that, a clustering is found for the most coarsened graph, characterizing the initial solution for the problem. Since this graph is much smaller than the original one, a low computational effort is required to achieve this solution. Even though the partitioning phase is not mandatory, before the last phase of the multi-level strategy, the initial clustering must somehow be determined. Mainly for the community detection problem, some authors prefer to define each vertex from the coarsened graph, the supernodes, as a cluster. Finally, the refinement phase projects the coarsened graph back to its original state and simultaneously applies a local search to the intermediary clusterings.

In this paper, we employ one of the phases of the multi-level strategies, the coarsening, to introduce our framework. We do not make use of the refinement phase, because of the high computational cost it might have. Preliminary experiments indicated that the local search was not able to significantly improve our solutions, just considerably raising the computational time. The coarsening phase was adapted for directed graphs.

The main idea behind the method is, for each iteration, to find for every end-vertex, the arc e that if contracted produces the best increase in modularity, $\Delta(e)$. This arc e is inserted in a list S sorted in a decreasing order of $\Delta(e)$. After that, a Restrict Candidates List (RCL) must be created with the $\alpha|S|$ first arcs. Then, from this RCL, an arc is randomly chosen to be contracted. This process repeats until the modularity of the clustering defined by the supernodes of the most coarsened graph (projecting the same cluster to every vertex that belongs to a supernode) cannot be further improved. Algorithm 1 presents the proposed heuristic based on the coarsening proposed in this paper.

The input data of Algorithm 1 are the graph G, from which we will determine the communities; the maximum number of iterations of the method, *maxIter*; a parameter to control when a step called permanent coarsening will happen, *iterContr*; and the parameter α to control the size of the RCL. Each iteration of the algorithm produces a clustering after the function Coarsening Phase is applied to the graph G. Then, the set storing the best clusterings, $\mathcal{E}$, set as empty in the beginning of the algorithm, is accordingly updated if it addresses the requirements. At each *iterContr* iterations, the recurrent contractions are evaluated and considered to become permanent coarsening.

[1] A tournament is an orientation of a complete undirected graph.

Algorithm 1: Proposed heuristic.

```
Data: graph $G = (V, E)$, $maxIter$, $iterContr$, $\alpha$
Result: clustering $\mathcal{C}$
$\mathcal{E} \leftarrow \emptyset$
for $iter$:=1 to $maxIter$ do
    $\mathcal{C}$ := Coarsening Phase($G$,$\alpha$)
    updateEliteSet($\mathcal{E}, \mathcal{C}$) if $iter$ mod $iterContr$ = 0 then
        $G$ := Permanent Coarsening( $\mathcal{E}$)
    end
end
```

In the next sections, each of the aforementioned functions of the proposed heuristic is thoroughly explained.

3.1. Coarsening clustering

Coarsening is a strategy applied to graphs with the purpose of reducing them to a more manageable size. Accordingly, the primary procedure to achieve this goal is the iterative arc contractions of the graph. Following, a definition of the arc contraction of a directed graph is presented.

Definition 1. The contraction of an arc $e = (v_i, v_j)$ of a graph G_n with n vertices, G_n/e, corresponds to producing a graph minor of G_n, $G_{n-1} := (V_{n-1}, E_{n-1})$ such that: $V_{n-1} := V_n \backslash \{v_i, v_j\} \cup \{v_{i,j}\}$ and $E_{n-1} := \{e' \in E_n : e' \cap e = \emptyset\} \cup \{(v_{i,j}, v_e) : \exists (v_e, u) \in E_n, u \in \{v_i, v_j\}\} \cup \{(v_e, v_{i,j}) : \exists (u, v_e) \in E_n, u \in \{v_i, v_j\}\}$.

It is worth mentioning that, according to Definition 1, the vertices v_i and v_j to be contracted will be replaced by a supernode $v_{i,\,j}$.[2] This means that every vertex that is adjacent to the vertex v_i or v_j is also adjacent to the supernode $v_{i,\,j}$, respecting the direction of the arc. If both v_i and v_j are the head (or tail) of an arc with vertex v_t as its tail (or head), then instead of adding multiple arcs, we consider a weight on the corresponding arc to measure the relationship between the supernode $v_{i,\,j}$ and vertex v_t. For such, consider the weight function $w : E(G) \rightarrow \mathbb{Z}$ over the arcs of G. If G is an unweighted graph, for contracting it, we relate to every arc of $E(G)$ a unitary weight. Then, when an arc (v_i, v_j) is contracted and both v_i and v_j are tail (or head) of arcs with v_t, the weight of this arc will be $w(v_i, v_t) + w(v_j, v_t)$ (or $w(v_t, v_i) + w(v_t, v_j)$) after the contraction of (v_i, v_j).

Bearing the arc contractions in mind, the coarsening phase of a multi-level algorithm consists in the successive arc contractions of a graph G producing a sequence of coarsened graphs. In this paper, we denote this sequence as $(G_n, G_{n-1}, G_{n-2}, \ldots, G_k)$, whose graphs are sorted with regard to their decreasing number of vertices and arcs. This sequence starts from G_n, that is the original graph G with n vertices, and halts with the graph G_k that is known as the *base graph* with k vertices/supernodes (Blum, Puchinger, Raidl, & Roli, 2011). Apart from the first graph of the sequence, a graph of this sequence, suppose G_i, is the result from an arc contraction from the immediate previous graph of the sequence, i.e., from the graph G_{i+1}, $k \leq i \leq n$.

A strategy for the coarsening phase of a multi-level strategy based on the modularity maximization problem was proposed in Noack and Rotta (2009). In this strategy, designed for undirected graphs, the authors suggest considering the gain with regard to the modularity of the partition resulted by the inclusion of the pair of vertices/supernodes to be contracted in the same cluster. Therefore, initially, the starting clustering is the partition where every vertex is an isolated cluster. In line with this, the authors proposed a greedy strategy that for the contraction chooses the edge, among all existing edges of the graph, that produces the highest modularity gain if contracted in that iteration. When the coarsening phase is finished, instead of partitioning the base graph, the authors consider each supernode/vertex of G_k as a cluster. This greedy strategy worked very well in the experiments performed by the authors.

In this paper, we propose a coarsening based on the strategy introduced in Noack and Rotta (2009) and adapted for being semi-greedy. A pseudocode of this algorithm is presented in Algorithm 2.

Algorithm 2: Coarsening phase.

```
Data: graph $G_n = (V_n, E_n)$, random seed $\alpha$
Result: graph $G_k = (V_k, E_k)$
$Q$:=Modularity($V_n$)
bestNeighbor := DefineBestNeighbors($G_n$)
maxList := CreateMaxList(bestNeighbor)
$G := G_n$

while $\exists e \in$ Maxlist do
    $(v_j, v_t)$ := ChooseContraction(maxList, $\alpha$) Contract
    Arc($G, v_j, v_t, Q$) UpdateBestNeighbor(bestNeighbor,$v_j$)
    UpdateMaxList(maxList, $v_j$)
end
return $G$
```

In Algorithm 2, the function Modularity(V_n) provides the modularity of the singletons of G_n and requires $O(n)$ time. The function DefineBestNeighbors(G_n) defines, for each vertex of G_n, the best neighbor for contraction into a supernode in $O(m)$ time. The function CreateMaxList(bestNeighbor) produces a list, the maxList, with the arcs defined in bestNeighbor, decreasingly sorted according to their modularity gain. This list must contain only arcs whose contraction results in a positive modularity gain. This function requires $O(n)$ time.

Then, an arc (v_j, v_t) from maxList restricted by a factor α is chosen to be contracted. The function Contract Arc(G_i, v_j, v_t, Q) is responsible for performing the arc contraction and all necessary updates, in particular, of the modularity value Q. It is presented in Algorithm 3. The functions UpdateBestNeighbor(bestNeighbor,v_j) and UpdateMaxList(maxList, v_j) are responsible for the updating of the sets bestNeighbor and maxList, respectively. The employed stop criterion for the arc contractions is when the algorithm does not find any arc whose contraction (cluster merging) will provide an increase in the modularity value.

Concerning the complexity of the whole coarsening process, consider that each arc contraction reduces the number vertices of the graph by one, i.e., $|V(G_i)| = |V(G_{i+1})| - 1$, for $i = n, n-1, \ldots, 2$. Thus, the complexity of all the contractions performed during the coarsening phase is given by the series $T(n) = \sum_{i=2}^{n} n_i \log n_i$, where $n_i = V(G_i)$. Because $n_i = i$, for $i = n, n-1, \ldots, 2$, $T(n)$ can be rewritten as $T(n) = \sum_{i=2}^{n} (i \log i)$. By considering that $\log n < \sqrt{n}$ asymptotically holds, it can be proved, by mathematical induction, that $T(n) = O(n^2)$. Therefore, the overall complexity of the coarsening algorithm is $O(n^2)$.

In Algorithm 3, we describe how the contraction of an arc (v_j, v_t) works. In our implementation, instead of explicitly creating a supernode, the tail vertex v_j is replaced by the supernode produced with the contraction of (v_j, v_t).

The modularity of the supernodes after the contraction of (v_j, v_t) and the in-/out-degree of vertex v_j, which now will represent the supernode, are updated (Update(G, Q)). These operations require $O(1)$ time. The functions RemoveArc(v_j, v_t) and RemoveArc(v_t, v_j) are responsible for the removal of arc (v_j, v_t) and, possibly, arc

[2] A set of vertices that were contracted together (Dhillon, Guan, & Kulis, 2005) is called a supernode.

Algorithm 3: Contract Arc.

```
Data: graph $G = (V, E)$, arc $(v_j, v_t)$, modularity $Q$
Result: graph $G$
Update(G,Q)
RemoveArc($v_j, v_t$)
RemoveArc($v_t, v_j$)
forall the $k \in \mathcal{N}^+(v_t)$ do
    RemoveArc($v_t, k$)
    if $k \in \mathcal{N}^+(v_j)$ then $w(v_j, k) := w(v_j, k) + w(v_t, k)$ ;
    else InsertArc($v_j, k, w(v_t, k)$) ;
end
forall the $k \in \mathcal{N}^-(v_t)$ do
    RemoveArc($k, v_t$)
    if $k \in \mathcal{N}^-(v_j)$ then $w(k, v_j) := w(k, v_j) + w(k, v_t)$ ;
    else InsertArc($k, v_j, w(k, v_t)$) ;
end
RemoveVertex($G, v_t$)
return $G$
```

(v_t, v_j) from $E(G)$ and requires $O(\log n)$. Consequently, all arcs (v_t, k), for $k \in \mathcal{N}^+(v_t)$, are eliminated from the set $\mathcal{N}^+(v_t)$. Additionally, if an arc $(v_j, k) \notin E(G)$, it is created with weight $w(v_t, k)$ (InsertArc(v_j, k, $w(v_t, k)$)). If $(v_j, k) \in E(G)$, the value $w(v_j, k)$ is incremented by $w(v_t, k)$. Similarly, analogous operations are performed concerning arcs (k, v_t), for $k \in \mathcal{N}^-(v_t)$. These last updates require $O(n \log n)$ time. Other updates involving the maxList also have to be addressed.

3.1.1. Permanent coarsening

In this paper, we introduce a permanent clustering strategy for providing a memory mechanism for the proposed algorithm, as well as to reduce the size of the original graph. Here, the main goal of this strategy is to preserve the good components of the solutions found along the iterations and diminishes the overall computational time.

For the Permanent Coarsening (PC), we kept an elite set of solutions, represented by $\mathcal{E}$. This set contains the best clusterings found up to the current iteration of the algorithm. Starting from a specific iteration of the consensus strategy, the PC algorithm is applied to the graph at each certain number of iterations. The strategy is considerably ease of understanding: it permanently coarsens into a supernode those pairs of vertices that were grouped in the same cluster in at least 50% of $\mathcal{E}$.

Even though the PC strategy requires $O(n^2)$, it is worthy to highlight that this strategy significantly decreases the overall complexity of the proposed algorithm, since it diminishes the size of the original graph.

3.2. Consensus strategy

Consensus clustering is a type of strategy which aggregates information of distinct solutions. In Topchy, Jain, and Punch (2005), the authors mention that one way of performing this task is by combining components of different clusterings obtained by a single clustering algorithm. This strategy enables a consensus clustering which suits better the community structure of a graph. Lancichinetti and Fortunato (2012) employ a consensus strategy to find robust partitions from a number of executions of a same community detection technique. They considered benchmark algorithms to perform the consensus analysis, as OSLOM, the Louvain method, and Infomap algorithms. In experiments carried out by the authors with benchmark networks, they attested a better robustness of the methods with the consensus strategy.

In line with this concept, this paper presents a consensus clustering by generating different partitions with the introduced coarsening algorithm, but with different values of λ. For such, we specify a pool of values for λ using the results provided by the neural network proposed in Carvalho et al. (2014). Carvalho et al. (2014) presented a semi-supervised learning algorithm to define the most suitable intervals of values for the parameter λ of the adjusted modularity (Reichardt & Bornholdt, 2006). The multi-layer perceptron defines for a given network, appropriately trained with LFR networks, in which of 8 intervals of values λ belongs to: [1, 1.5], [1.6, 2], [2.1, 2.5], [2.6, 3], [3.1, 3.5], [3.6, 4], [4.1, 4.5] or [4.6, 5]. For designing the neural network, the topological traits of the input graphs are considered by an analysis of the proportion between the number of 4-sized motifs and the size of the network (Meira, Máximo, Fazenda, & Da Conceição, 2014).

Then, after finding the interval for a given graph, ConClus generates a set of partitions taking specific values within the interval. Let us consider $[a_i, b_i]$ the interval provided by the neural network. We select the following values for λ: a_i, a_i+0.1, a_i+0.2, a_i+0.3,a_i+0.4 and b_i. Taking the set of clusterings into account, a consensus partition is produced by the analysis of the pairwise relation of the vertices with regard to the clustering set. In this case, ConClus employs the criterion of gathering into the same cluster those vertices grouped together in, at least, 50% of the partitions from the set of clusterings. Since a small number of clusterings are required to perform the consensus clustering, the asymptotic complexity of the method is $O(n^2)$.

In the next section, we show the computational experiments performed with ConClus.

4. Computational experiments

This section presents the analysis of the experiments carried out to evaluate the performance of ConClus. After preliminary experiments to determine the parameters of the algorithm, we employed the following: the parameter α was 0.5; the maximum number of iterations of the algorithm *(maxIter)* was 30; and the number of iterations of the permanent clustering strategy evaluations (*iterContr*) was 3. Moreover, this paper shows the results of ConClus with α is 0, i.e. when the coarsening algorithm is greedy. This version of ConClus is referred as G-ConClus. All experiments were run in a machine with Intel Xeon E5-1620 3.7-GHz processor and 32GB of main memory.

In the first experiment, we used two real benchmark networks: *afootball* (Girvan & Newman, 2002) and *polBlogs* (Adamic & Glance, 2005). The second experiment was similar to the methodology described in Lancichinetti et al. (2011). In both experiments, the results achieved by ConClus were compared with those obtained by the Label Propagation (LP) algorithm (Raghavan, Albert, & Kumara, 2007), Infomap (Rosvall & Bergstrom, 2007) and OSLOM (Lancichinetti et al., 2011). The only algorithm that does not take into account the arc directions is the LP algorithm. Roughly, LP is a neighborhood-based strategy that labels a given vertex according to the majority of the labels of its neighborhood. The three algorithms are extensively used for detecting communities in networks, reportedly efficient in this task according to the most recent literature. Moreover, they are among the few available implementations of algorithms that detect communities in directed networks.

4.1. Experiment I

The first real dataset considered in this experiment contains information about matches of the division I-A of American college football teams occurred in 2000, fall season. The resulting

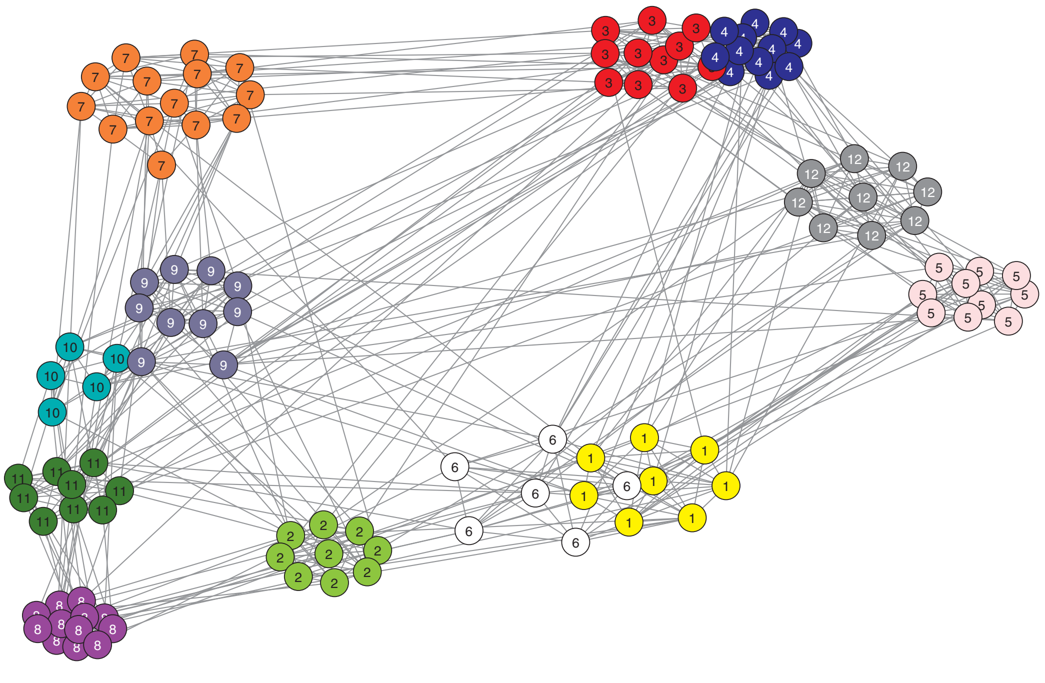


Fig. 4. The *afootball* benchmark graph with its vertices marked according to the communities found by ConClus. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

unweighted undirected network has its vertices representing the football teams, whereas the existence of an edge between vertices corresponds to whether or not the two teams faced each other in the season. It is natural that the communities in this network are in agreement with the 12 conferences of the season. Accordingly, teams from the same conference are supposed to belong to the same community. This data, compiled by Girvan and Newman (2002), is characterized by highly connected groups.

Fig. 4 illustrates *afootball* network using both different numbers and colors to identify the communities obtained by ConClus. The neural network provided the interval [2.6, 3] to determine the λ value. ConClus found a partition with exactly 12 groups. To compare this partition with that resulting from the conferences, we used the measure known as Normalized Mutual Information (NMI) (Danon, Diaz-Guilera, Duch, & Arenas, 2005). NMI is an information theory measure that assesses the correlation between two variables, in this case, the vertex partitions. The closer its value is to 1, the stronger the correlation of the two variables. The corresponding NMI value was 0.92419.

G-ConClus, LP, Infomap and OSLOM found communities whose NMI values in relation to the expected partition were, respectively, 0.88580, 0.91085, 0.92419 and 0.91568. As this network is unweighted, we also considered the Louvain method (Blondel et al., 2008), a benchmark modularity maximization heuristic. The corresponding NMI value was 0.89032. Therefore, both ConClus and Infomap presented the highest NMI values. Infomap was the algorithm that, in the experiments in Lancichinetti and Fortunato (2009), outperformed a number of benchmark community detection algorithms.

The second real network, named *polBlogs* (Adamic & Glance, 2005), represents blogs about US politics and their hyperlinks. It is an unweighted directed network, where the vertices correspond to the sites of the blogs whilst the arcs refer to the hyperlinks between them. The blogs are separated into two categories, the expected communities: liberal and conservative. This network has 1490 vertices, 266 of which are unconnected that mean blogs without hyperlinks pointing to and by them. They were removed from the network to eliminate unnecessary noise. It is worth mentioning, however, that all algorithms in the experiment interpreted these blogs as isolated communities.

In this case, the neural network provided an overestimated interval for λ, [1.0, 1.5]. The NMI between the expected partition and that from ConClus with this interval was 0.45763. As the number of expected communities is low, this interval, perhaps, is not the most suitable for this network. For this reason, we ran ConClus considering the interval [0.6, 1] and the NMI was significantly better, 0.66928. Fig. 5 displays the *polBlogs* network and its vertices labeled according to the communities with the highest NMI obtained by ConClus. As this network is large-sized, the communities are collapsed into supernodes for a better visualization. The diameters of the vertices are proportional to the number of vertices in the communities. The thickness of the arcs corresponds to the amount of arcs between vertices of different communities. However, they are not in real scale primarily due to those communities with very few vertices. The two largest communities (labeled as 1 and 3) have most of the blogs correctly identified. The community with label 1 is mostly composed by liberal blogs, whereas community 3, by conservative blogs.

Concerning the other strategies, the NMI values of the communities obtained by G-ConClus, OSLOM, Infomap and LP were, respectively, 0.67888, 0.57208, 0.43547 and 0.38534. All NMI values correspond to the average of five independent executions.

In the next section, we present the results from the experiments performed with artificial datasets.

4.2. Experiment II

This experiment considers the software introduced in (Lancichinetti et al., 2008) to produce artificial directed networks with communities of different sizes. It is worth mentioning that we consider networks with small-sized, named *S*, and large-sized communities, referred as *L*. The key difference related to the community structure of the networks from the same set is the

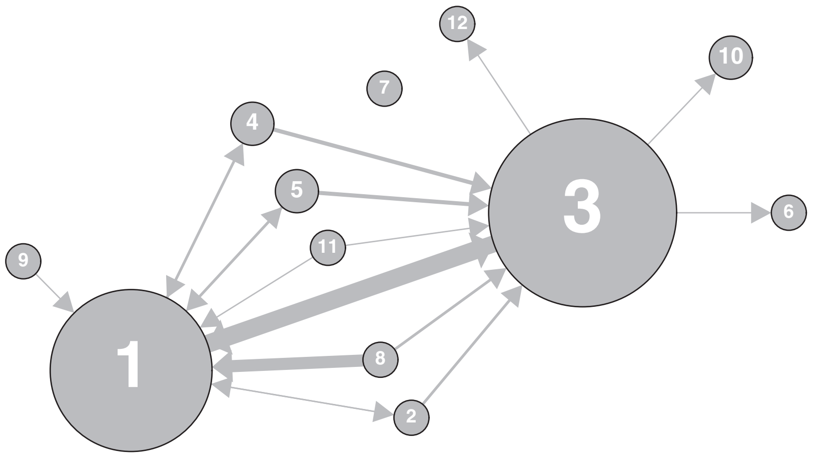


Fig. 5. The *polBlogs* benchmark network with its vertices marked according to the communities found by ConClus.

mixing degree between communities, referred as μ. The higher its value is, the more fused the communities are.

Both sets S and L include 5 subsets of 40 directed networks each of which with 1000, 2000, 3000, 4000 and 5000 vertices, and 5 different networks for each μ totalizing 400 networks. Again, following the suggestions provided in Lancichinetti et al. (2011), the values of the parameters average in-degree, maximum in-degree, negative exponent for degree sequence and negative exponent for the community size distribution of the generated networks have the respective values: 20, 50, 2 and 1.

On the one hand, the remaining parameters of the networks in the set S are: the minimum value for the community size, that is 10; and the maximum value for the community size, that is 50. On the other hand, for the set L, the minimum community size is 20 and the maximum community size is 100.

The measure that assesses how close the communities acquired by the algorithms are in relation to the expected partitions is, again, the NMI (Danon et al., 2005). Figs. 6–15 summarize the results of the performed experiment for networks with small and large-sized communities.

Each figure displays the average results of all algorithms considered in the experiment for a given set of networks (S or L) and a given number of vertices. Accordingly, the left figures indicate the average NMI values obtained by each μ. The reported NMI values are the average of 5 independent executions, for all algorithms. For Infomap, we considered 100 trials for each of the 5 independent executions. This number was suggested by the authors of the algorithm. The right figures indicate the average running times of the algorithms in the independent executions. The only average running times not reported in the graphics are those from LP, whose employed implementation is in *R* language. However, to be fair, we can highlight that LP, so as Infomap, is estimated to be of the linear time order.

The results of the experiments indicate that the algorithms achieved the highest NMI values for low values of μ. This behavior is expected since the network communities with higher μ are weakly defined. On the one hand, according to the presented results, ConClus and OSLOM had a very consistent NMI curve considering the different mixture parameters. Even for $\mu = 0.8$, one may notice NMI values larger than 0.65. On the other hand, both the communities provided by Infomap and LP had NMI 0.0 for networks with high mixture degrees. Both strategies placed all vertices into a single community for most of the networks with high mixture degree. In spite of that, Infomap is faster than both ConClus and OSLOM. It is expected that the running time of LP is as low as the computational time of Infomap.

For networks from the set S and with 1000 and 2000 vertices, ConClus achieved the highest NMI values in comparison to all tested algorithms. However, in detecting communities in networks with 1000 vertices and μ equals to 0.8, from the set L, ConClus obtained a significantly higher NMI value in comparison to the other algorithms. Considering this set networks, OSLOM outperformed the other algorithms for μ equals to 0.6 and 0.7. In most of these instances, except for networks from the set S with 1000 vertices, ConClus is faster than OSLOM in detecting communities in networks with $\mu \leq 0.7$, but slower in networks with $\mu = 0.8$.

OSLOM achieved the highest average NMI values in networks with both small and large-sized communities of 3000, 4000 and 5000 vertices. However, OSLOM was slightly better than ConClus in networks from the set S. G-ConClus obtained results slightly worse than ConClus. However, its computational time was significantly inferior, comparable to the running time of Infomap.

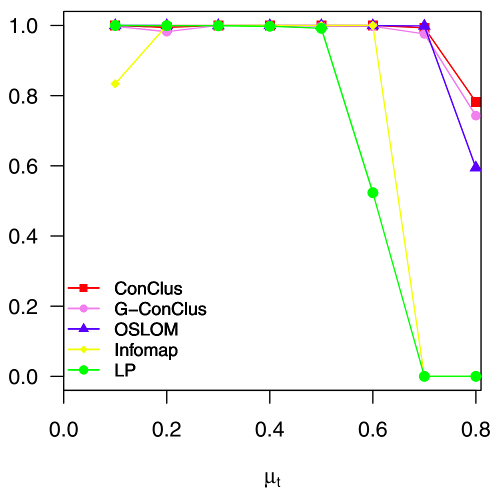
It is worth to point out that for generating the networks using the software proposed in Lancichinetti et al. (2011), we kept the key parameters (average in-degree and communities sizes) with fixed values, independently of the number of vertices. Therefore, networks with 1000 vertices have the same parameter values as networks with 5000 vertices. Consequently, the resulting networks with fewer vertices are denser than those generated with a higher number of vertices. In line with this, by the results of the experiments, it is possible to observe that ConClus performs better with dense networks. To empirically confirm this claim, we generated another set with denser directed networks with large-sized communities. The difference in generating this set and the set L is on the value of the parameters average and maximum in-degrees, set as 40 and 100, respectively. Figs. 16–20 present the results of the algorithms using these networks.

The results with these 200 networks clearly indicate that ConClus outperformed all other algorithms, despite the significantly higher computational time.

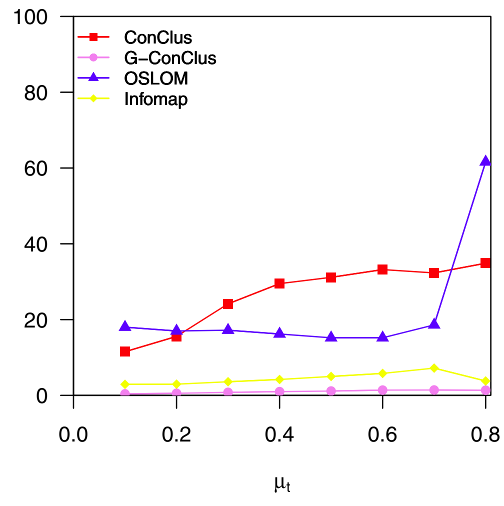
5. Final remarks

This paper presents a study about the community detection in directed networks. Even though the intense research interested in identifying communities in undirected networks, the subject for directed networks remains a challenge. A few algorithms are specially designed to identify communities in these networks.

In line with this motivation, this paper presents a community detection algorithm for directed networks, here named ConClus. Since the modularity measure is an outstanding measure for

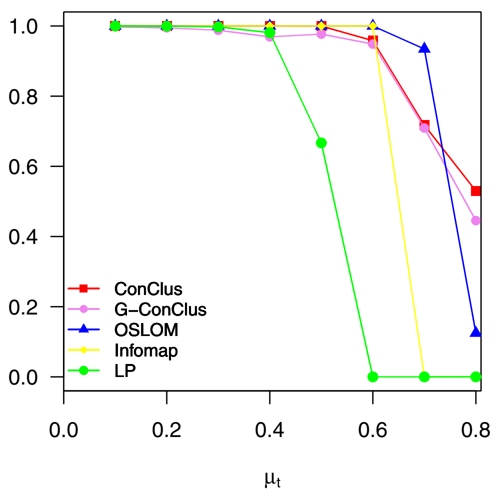


(a) NMI values by mixing parameter

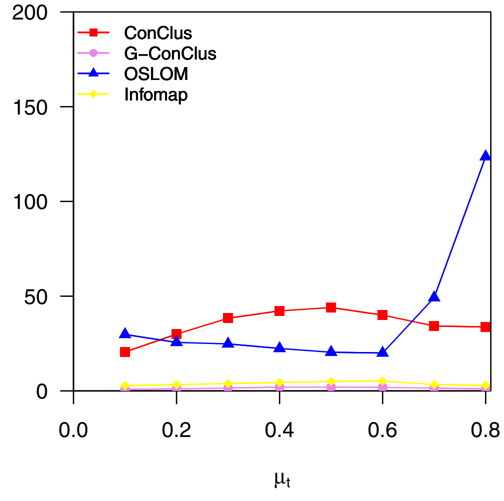


(b) Running times by mixing parameter

Fig. 6. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the set *S* with 1000 vertices.

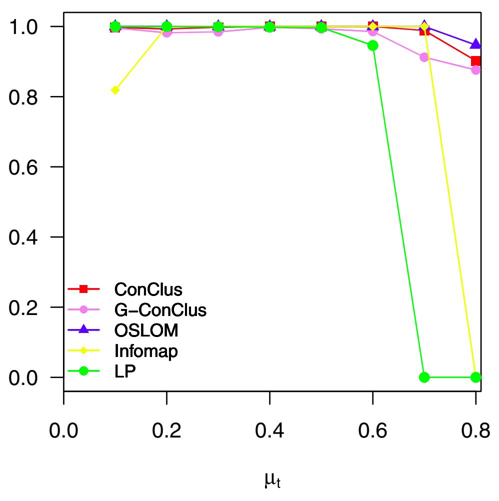


(a) NMI values by mixing parameter

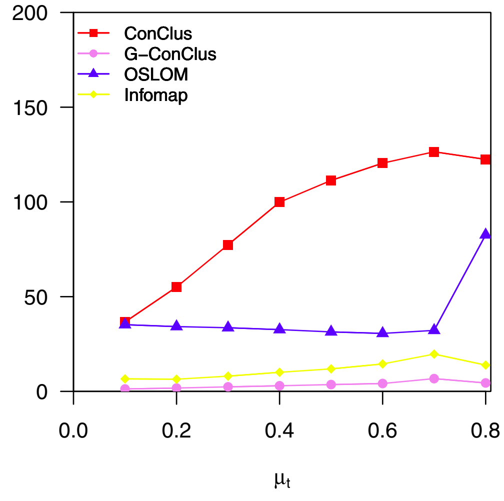


(b) Running times by mixing parameter

Fig. 7. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the set *L* with 1000 vertices.

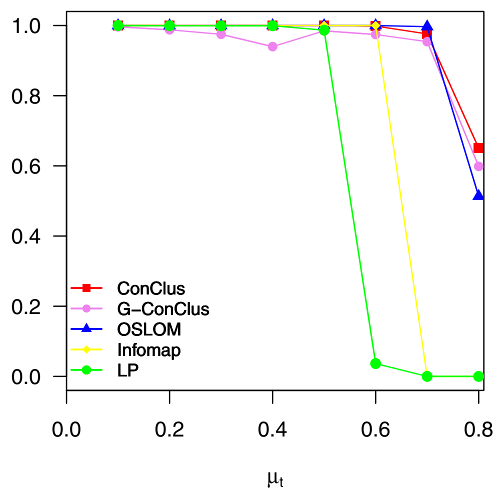


(a) NMI values by mixing parameter

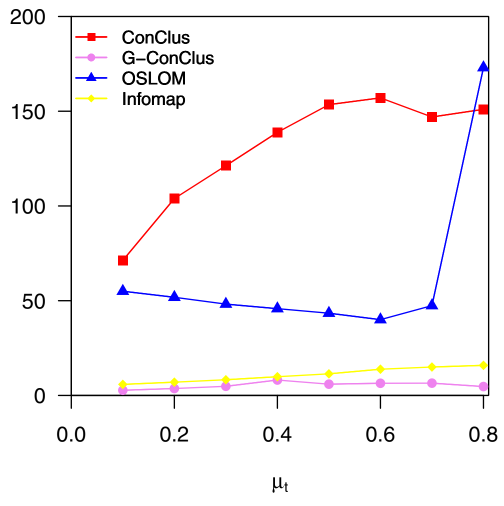


(b) Running times by mixing parameter

Fig. 8. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the set *S* with 2000 vertices.

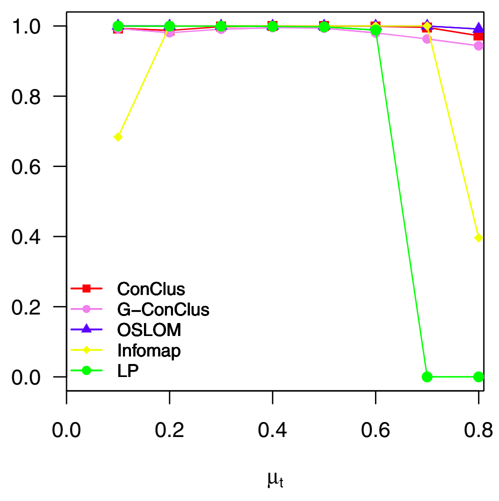


(a) NMI values by mixing parameter

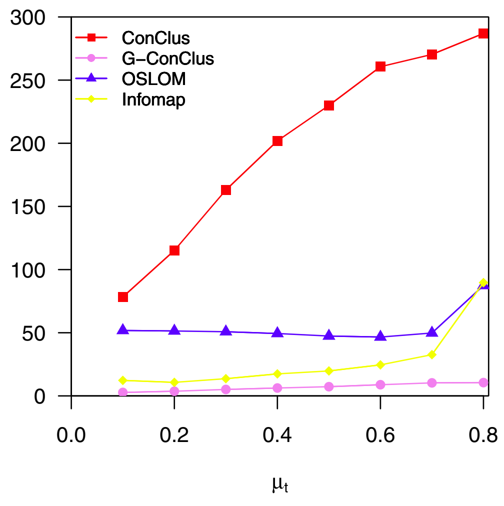


(b) Running times by mixing parameter

Fig. 9. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the set *L* with 2000 vertices.

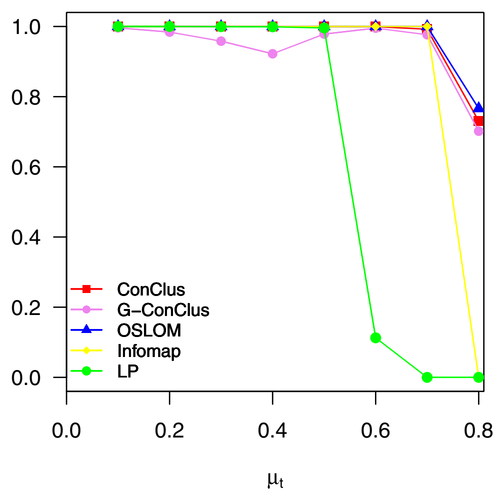


(a) NMI values by mixing parameter

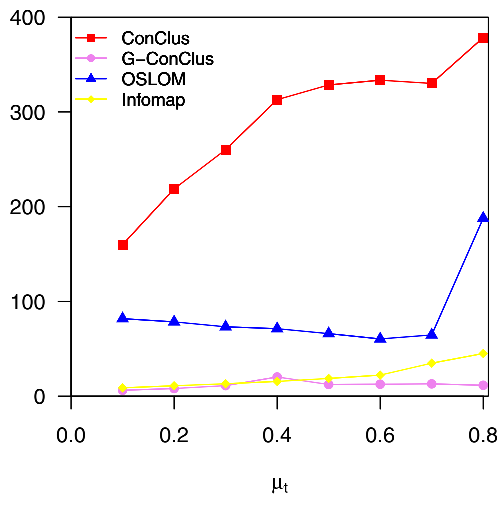


(b) Running times by mixing parameter

Fig. 10. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the set *S* with 3000 vertices.

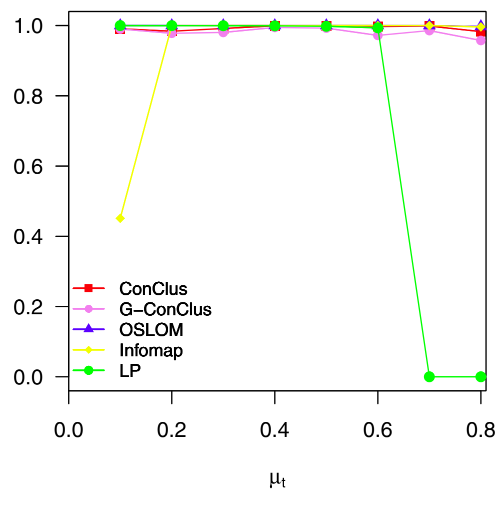


(a) NMI values by mixing parameter

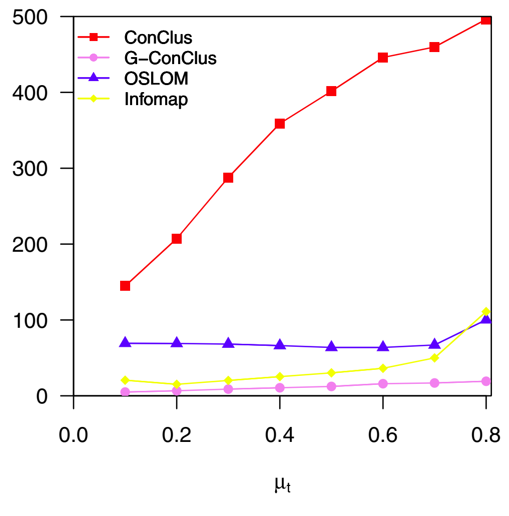


(b) Running times by mixing parameter

Fig. 11. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the set *L* with 3000 vertices.

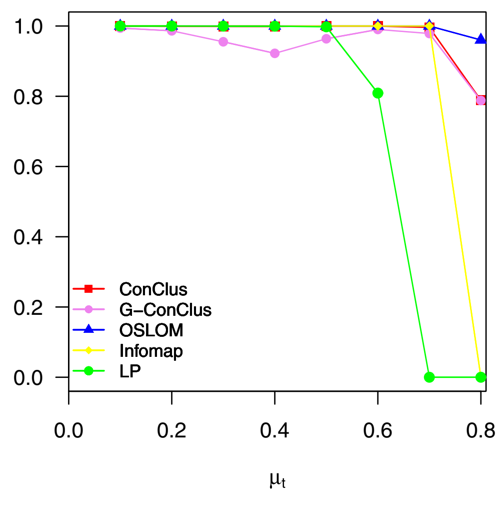


(a) NMI values by mixing parameter

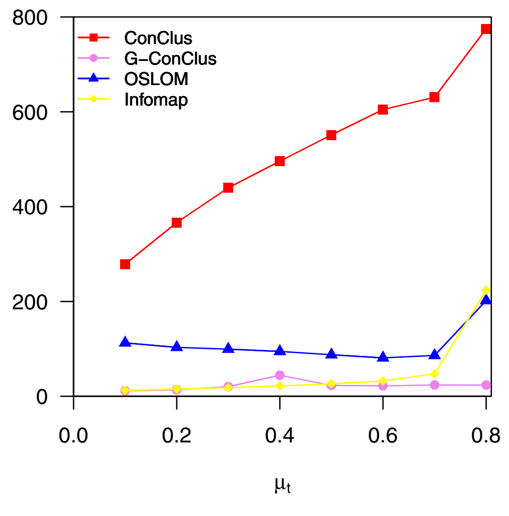


(b) Running times by mixing parameter

Fig. 12. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the set *S* with 4000 vertices.

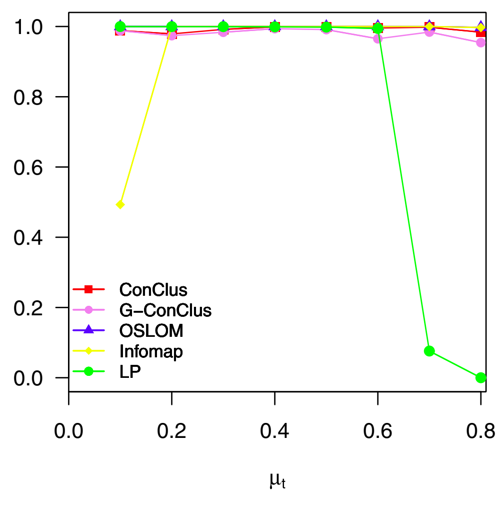


(a) NMI values by mixing parameter

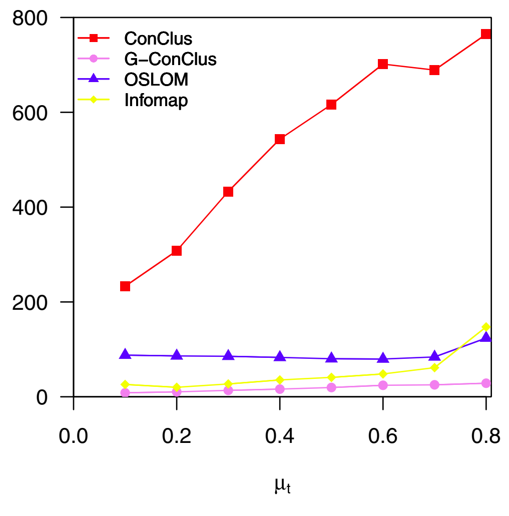


(b) Running times by mixing parameter

Fig. 13. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the set *L* with 4000 vertices.

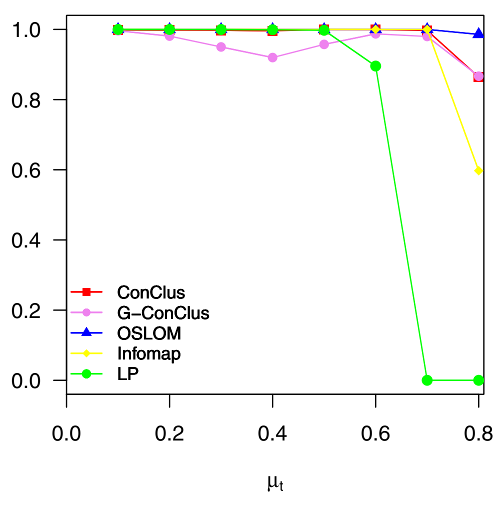


(a) NMI values by mixing parameter

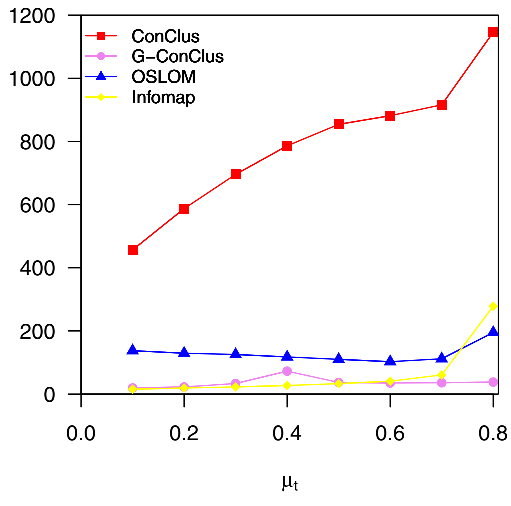


(b) Running times by mixing parameter

Fig. 14. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the set *S* with 5000 vertices.

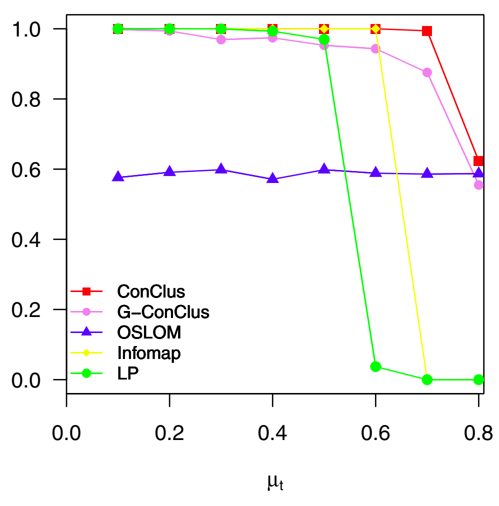


(a) NMI values by mixing parameter

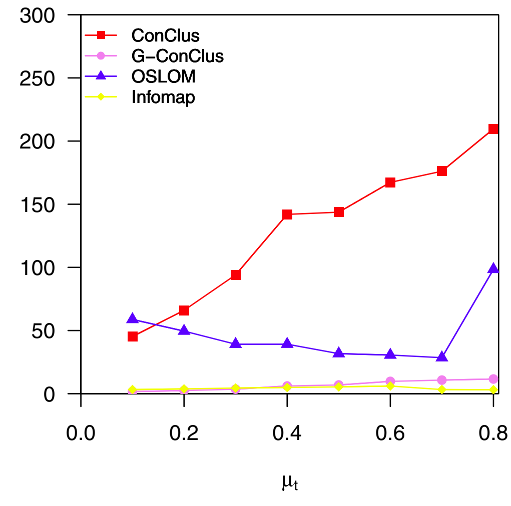


(b) Running times by mixing parameter

Fig. 15. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the set *L* with 5000 vertices.



(a) NMI values by mixing parameter



(b) Running times by mixing parameter

Fig. 16. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the dense set with 1000 vertices and with average and maximum in-degrees, respectively, 40 and 100.

defining the quality of partitions, ConClus relies on this metric as fitness function. A neural network provides intervals of values for defining a parameter known as resolution parameter for adjusting the modularity measure to overcome the so-called resolution limit. Then, after a semi-greedy strategy produces a set of partitions guided by the adjusted modularity, ConClus returns the final partition that is a consensus of the set.

In the computational experiments, we compared the results of ConClus with those from three extensively used algorithms from the literature: OSLOM, Infomap and Label Propagation (LP). Additionally, this paper shows the results of ConClus when α is 0, i.e., considering the greedy version of the coarsening algorithm. This version of ConClus is referred as G-ConClus. ConClus achieved the best results in the first experiment with two real benchmark networks. Concerning a second experiment with 600 directed LFR networks of small and large-sized communities, ConClus presented a very good performance, being robust and effective. It is worth to point up that ConClus outperformed Infomap and LP even in the networks that modularity is reportedly unreliable due to its resolution limit. In comparison to OSLOM, ConClus was very competitive. It achieved more accurate results than OSLOM when considering dense graphs, whereas OSLOM is better for sparse graphs. G-ConClus was slightly worse than ConClus, but with a computational time significantly inferior.

The strongest points of ConClus are the semi-supervised learning phase and the memory stage. The former was fundamental for defining a measure more accurate for detecting communities in LFR networks. The latter, in which the vertices are collapsed into communities due to the frequency they are together in the evaluated partitions, had a major impact on the quality of the results and on the computational time.

ConClus, however, presented a higher computational time than the algorithms involved in the comparisons. Nevertheless, one may consider G-ConClus in applications where the computational times are supposed to be low. G-ConClus, besides being deterministic, outperformed the results of Infomap, the only algorithm that had a computational time competitive with G-ConClus. As a matter of fact, G-ConClus was faster than Infomap. Additionally, it is possible

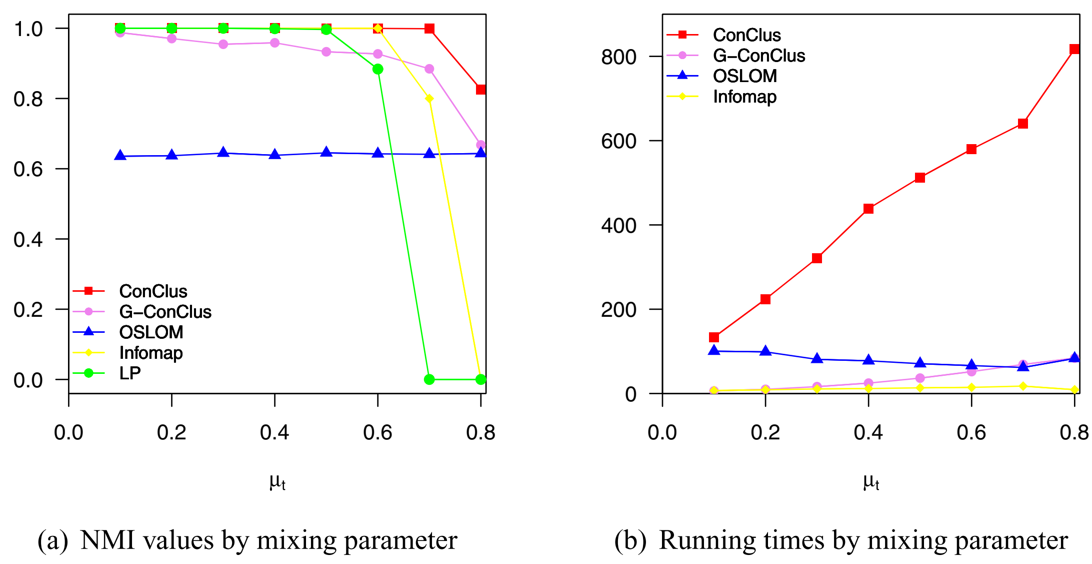


Fig. 17. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the dense set with 2000 vertices and with average and maximum in-degrees, respectively, 40 and 100.

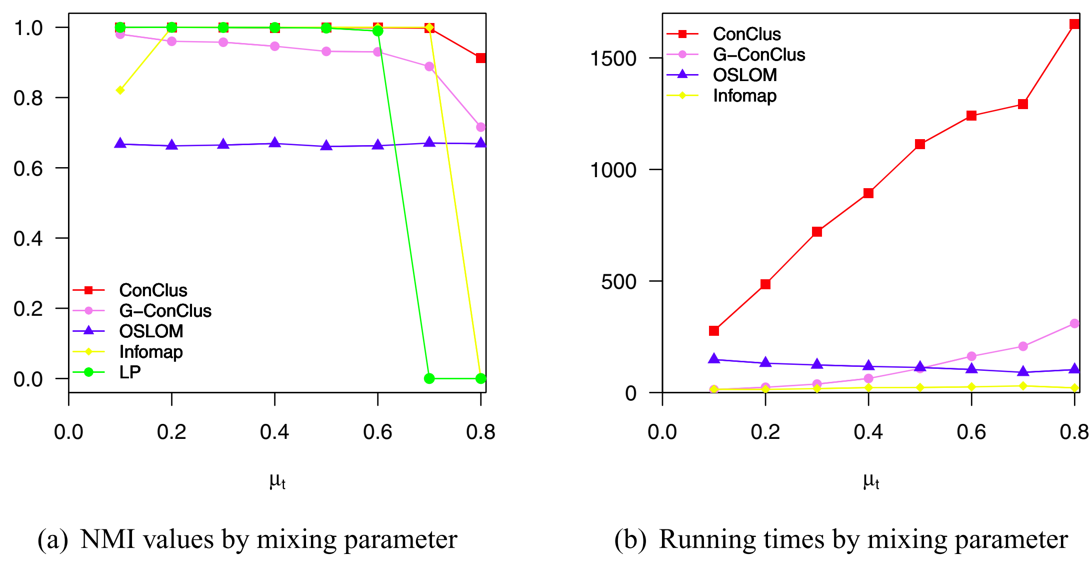


Fig. 18. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the dense set with 3000 vertices and with average and maximum in-degrees, respectively, 40 and 100.

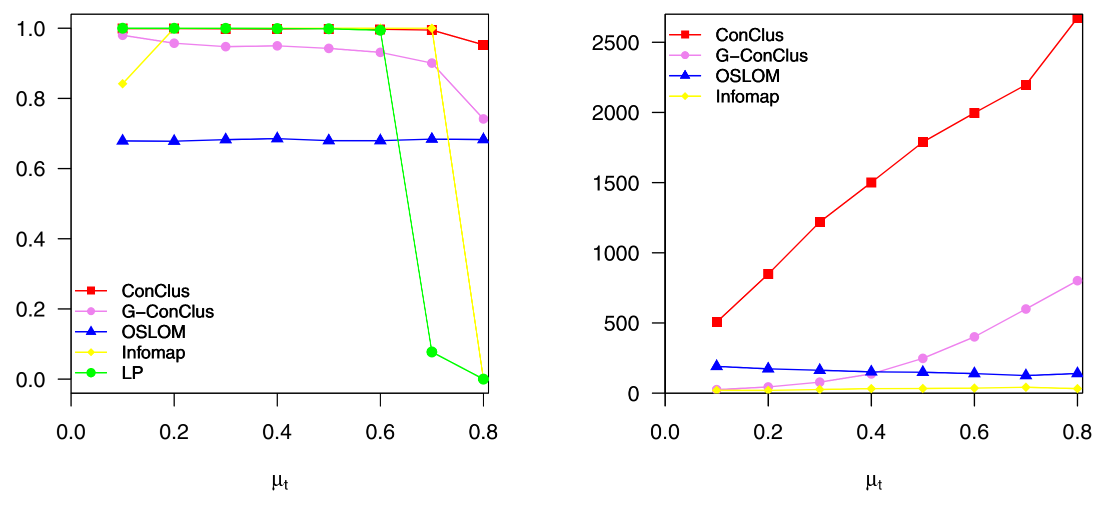
to consider parallelism in the multiple iterations of ConClus, to increase the speediness.

As future research, we suggest the development of frameworks guided by the adjusted modularity, but with a lower asymptotic complexity to detect communities in large scale networks. In line with this, we recommend a distributed version of ConClus, using the local (adjusted) modularity measure, similar to a recent proposal of the Louvain method. This type of framework is highly indicated since it can be used for either dynamic or distributed networks. In adapting ConClus, it is notable its natural parallelization since, during the coarsening iterations, it provides partitions independently, before considering them for the consensual analysis.

Following a more theoretical line, there is a strong potential in studies that perform topological analysis of the networks. In this context, these investigations may provide ways to extract relevant information of the network to determine the resolution parameter. In particular, we strongly recommend the development of effective local metrics, as an attempt to find cheap and distributed algorithms to detect communities in large networks.

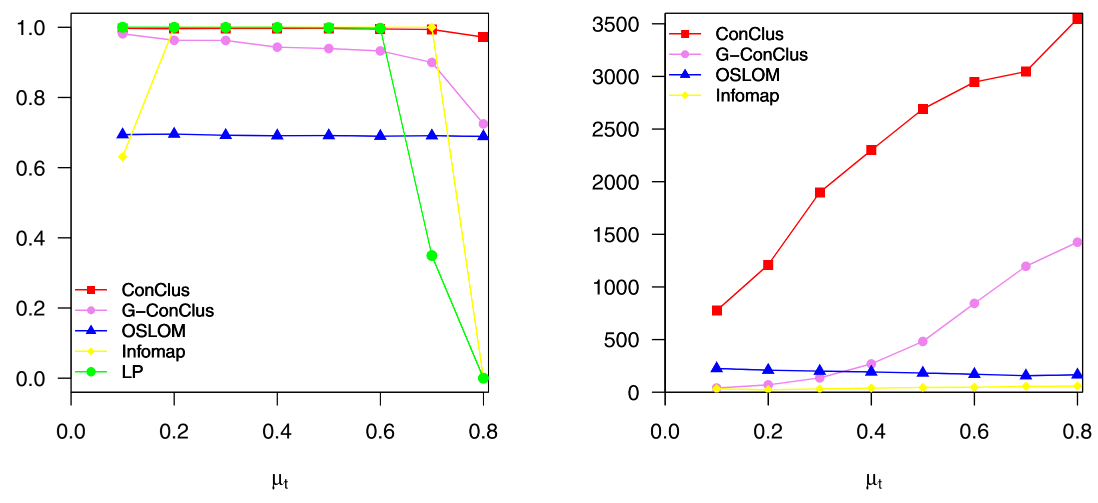
The core of most studies related to intelligent systems treating the community detection problem in networks either introduces new measures or uses consolidated measures, such as modularity, to guide their algorithm that may fail in some case studies. This paper shows that the parameter adjustment by the neural network by evaluating the network topology enhanced significantly the quality of the results achieved by the algorithm. The interval approach gave diversity to the strategy since ConClus used the range to find the consensus partitions from the set acquired by varying the parameter values within the interval. We suggest as future work related to intelligent systems the fine tuning of the resolution parameter according to the graph topology. Furthermore, we recommend the use of a memory mechanism hybridized with a diversification strategy to guide the search for high-quality communities.

To sum up, this paper demonstrated that even though there is evidence about the low quality of the modularity measure due to the resolution limit, its adjusted version guided ConClus that achieved outstanding results. In spite of the training of the



(a) NMI values by mixing parameter (b) Running times by mixing parameter

Fig. 19. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the dense set with 4000 vertices and with average and maximum in-degrees, respectively, 40 and 100.



(a) NMI values by mixing parameter (b) Running times by mixing parameter

Fig. 20. Performance of ConClus in comparison to OSLOM, Infomap and LP using networks from the set with 5000 vertices and with average and maximum in-degrees, respectively, 40 and 100.

neural network employed LFR networks, ConClus was very accurate in detecting communities in the real benchmark networks. The most cohesive groups of vertices were identified by the permanent coarsening strategy enabling ConClus to further investigate the communities of the border vertices.

Acknowledgments

The authors thank Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) for the financial support. We also are indebted to the anonymous reviewers whose comments significantly improved the presentation of the paper.

References

Adamic, L. A., & Glance, N. (2005). The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the 3rd international workshop on link discovery* (pp. 36–43). ACM.

Arenas, A., Duch, J., Fernández, A., & Gómez, S. (2007). Size reduction of complex networks preserving modularity. *New Journal of Physics, 9*(6), 176.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment, 2008*(10), P10008.

Blum, C., Puchinger, J., Raidl, G. R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: a survey. *Applied Soft Computing, 11*, 4135–4151.

Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikolosk, Z., & Wagner, D. (2008). On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering, 20*, 172–188.

Carvalho, D. M., Resende, H., & Nascimento, M. C. V. (2014). Modularity maximization adjusted by neural networks. In *Proceedings of the 21st international conference in neural information processing (ICONIP): 8834* (pp. 287–294). Springer.

Danon, L., Diaz-Guilera, A., Duch, J., & Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment, 2005*(09), P09008.

Dhillon, I., Guan, Y., & Kulis, B. (2005). A fast kernel-based multilevel algorithm for graph clustering. In *Proceedings of the eleventh ACM SIGKDD international conference on knowledge discovery and data mining (KDD)* (pp. 629–634).

Fortunato, S., & Barthélemy, M. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences USA, 104*(1), 36.

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

Kim, Y., Son, S.-W., & Jeong, H. (2010). Finding communities in directed networks. *Physical Review E, 81*(1), 016103.

Lancichinetti, A., & Fortunato, S. (2009). Community detection algorithms: a comparative analysis. *Physical Review E, 80*(5), 056117.

Lancichinetti, A., & Fortunato, S. (2012). Consensus clustering in complex networks. *Scientific Reports, 2* Article number: 336.

Lancichinetti, A., Fortunato, S., & Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Physical Review E, 78*, 046110.[5 pages].

Lancichinetti, A., Radicchi, F., Ramasco, J. J., Fortunato, S., et al. (2011). Finding statistically significant communities in networks. *PloS One, 6*(4), e18961.

Leicht, E. A., & Newman, M. E. (2008). Community structure in directed networks. *Physical Review Letters, 100*(11), 118703.

Malliaros, F. D., & Vazirgiannis, M. (2013). Clustering and community detection in directed networks: a survey. *Physics Reports, 533*(4), 95–142.

Meira, L. A., Máximo, V. R., Fazenda, Á. L., & Da Conceição, A. F. (2014). Acc-motif: accelerated network motif detection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 11*(5), 853–862.

Noack, A., & Rotta, R. (2009). Multi-level algorithms for modularity clustering. In *Proceedings of the 8th international symposium on experimental algorithms, SEA '09: 1* (pp. 257–268).

Park, H.-J., & Friston, K. (2013). Structural and functional brain networks: from connections to cognition. *Science, 342*(6158), 1238411.

Raghavan, U. N., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E, 76*(3 Pt 2), 036106.

Reichardt, J., & Bornholdt, S. (2006). Statistical mechanics of community detection. *Physical Review E, 74*, 016110.

Romdhane, L. B., Chaabani, Y., Zardi, H., Group, M. R., et al. (2013). A robust ant colony optimization-based algorithm for community mining in large scale oriented social graphs. *Expert Systems with Applications, 40*(14), 5709–5718.

Ronhovde, P., & Nussinov, Z. (2010). Local resolution-limit-free potts model for community detection. *Physical Review E, 81*(4), 046114.

Rosvall, M., & Bergstrom, C. T. (2007). An information-theoretic framework for resolving community structure in complex networks. *Proceedings of the National Academy of Sciences USA, 104*, 7327.

Rosvall, M., & Bergstrom, C. T. (2010). Mapping change in large networks. *Plos One, 5*(1), e8694.

Topchy, A., Jain, A., & Punch, W. (2005). Clustering ensembles: models of consensus and weak partitions. In *Proceedings of IEEE transactions on pattern analysis and machine intelligence: 27*(12) (pp. 1866–1881).

Traag, V. A., Van Dooren, P., & Nesterov, Y. (2011). Narrow scope for resolution-limit-free community detection. *Physical Review E, 84*(1), 016114.