Information Sciences 424 (2018) 204–223

Contents lists available at ScienceDirect

Information Sciences

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

Community extraction and visualization in social networks applied to Twitter

Youcef Abdelsadek[a,*], Kamel Chelghoum[a], Francine Herrmann[a], Imed Kacem[a], Benoît Otjacques[b]

[a] Laboratoire de Conception, Optimisation et Modélisation des Systèmes, LCOMS EA 7306, Université de Lorraine, Metz, France
[b] e-Science Research Unit, Environmental Research and Innovation, Luxembourg Institute of Science and Technology, Belvaux, Luxembourg

ARTICLE INFO

Article history:
Received 7 September 2016
Revised 28 June 2017
Accepted 3 September 2017
Available online 14 September 2017

Keywords:
Social network analysis
Community detection
Interactive visualization
Visualization tool
Twitter

ABSTRACT

Nowadays, social network analysis attracts more interest from the scientific community. However, it becomes trickier to analyse the generated data by the social networks due to their complexity, which hides the underlying patterns. In this work we propose an approach for social media analysis, especially for Twitter's network. Our approach relies on two complementary steps: (i) a community identification based on a new community detection algorithm called *Tribase*, and (ii) an interactive community visualization, which provides gradual knowledge acquisition using our visualization tool, called *NLCOMS*. In order to assess the proposed approach, we have tested it on real-world data of the ANR InfoRSN project. This project is related to information propagation and community detection in Twitter's network, more precisely on a collection of tweets dealing with media articles. The results show that our approach allows us to visually reveal the community structure and the related characteristics.

1. Introduction

The use of social networks is exponentially growing in our society, having as a consequence a deep change in the manners that people react to events and interact with each other. Social network analysis [56], SNA, is the field studying these social behaviors. It has been increasingly popular in the last decade with the ubiquitous use of social networks. SNA relies on the use of networks to investigate social interactions [33]. In many domains, like in biology, computer science and economy, a set of interacting entities leads to complex systems [34] with hidden properties. These systems can be modelled as graphs, where the nodes and the edges respectively represent the entities and the relationships between them. For example, in sociology, they try to understand the friendship in a social blog, the phone calls between customers of a cellphone operator [4] or also the exchanged e-mails within an institution [51]. In some cases, a numerical value can be assigned to the binary edges. These values express either a strength or a shared quantity between two actors, called weighted graphs. For example, the weight in the e-mail network could be the number of e-mails exchanged between two employees. In this context, an efficient network analysis, should take into account the edge weight in the devised approach. In this paper, we use edge weighted graphs to model the "retweet", which is a well known relation on Twitter. It occurs when a Twitter user

* Corresponding author.
E-mail addresses: youcef.abdelsadek@univ-lorraine.fr (Y. Abdelsadek), kamel.chelghoum@univ-lorraine.fr (K. Chelghoum), francine.herrmann@univ-lorraine.fr (F. Herrmann), imed.kacem@univ-lorraine.fr (I. Kacem), benoit.otjacques@list.lu (B. Otjacques).

http://dx.doi.org/10.1016/j.ins.2017.09.022
0020-0255/

republishes an original tweet of another Twitter user. Therefore, the edge weight corresponds to the number of times where a retweet is observed between two Twitter users.

Furthermore, an important objective in social network analysis is to reveal some semantic aspects behind the network topology. This objective can be reached by identifying the highly intra-connected groups of nodes or communities which are in the opposite poorly inter-connected with each other, known as community detection problem [11]. This problem is very challenging and can be met in several domains. For example, it can be met in sociology [13] where the aim is to study the common characteristics of social groups and what makes peoples group together in a community. To detect such communities, the approach that we propose in this paper uses at its first step a community detection algorithm for weighted graphs based on a collection of triangles to build the skeleton of the community structure as starting point. Indeed, triangles are an important structure in social networks, which reflects the closeness of a community [40]. This closeness is expressed by the number of closed triads over the total number of triads, where a triad is a connected triplet of nodes. The closer to 1 is this value, the higher the probability for a given node to have connected neighbors (i.e., it belongs to a triangle). Next, the algorithm repeatedly compares the intra-community and inter-community weights between groups allowing dominant communities to increase.

Community detection is the preliminary step to grasp the underlying semantic and structural information in the network. Next comes the visualization issue. What is the most appropriate visualization for the detected communities and the hidden information? As examples, a node-link diagram [4] is used for community depiction whereas word cloud [57] is used to visualize nodes attributes. In this work, additionally to node-link representation, circle packing is used to visualize the detected communities. Also, synchronous and coordinated views help the expert user to build his/her own ideas about communities characteristics, like bar chart, partition layout and word cloud.

The remaining part of this paper is organized as follows. In Section 2, we summarize the related works to the context of this paper. In Section 3, we introduce our community detection algorithm for weighted networks, called *Tribase* and we compare it with five community detection algorithms identified in the literature. The community visualization aspect and our visualization tool *NLCOMS* are presented in Section 4. Section 5 discusses the case study used to assess the approach proposed in this paper. Finally, Section 6 concludes the paper and suggests further improvements of this work.

2. Related works

2.1. Community detection

In this section, we present a non-exhaustive list of community detection algorithms. As the number of existing algorithms is huge, we try to list a representative subset which covers the most known techniques. Globally speaking, two main methods are available to extract communities in graphs. They are presented in the following subsections.

2.1.1. Structure based algorithms

The first method relies on the structure of the graph and its characteristics to extract communities. It can be divided into two families of algorithms.

The first family is the divisive family or the top-down strategy. Here, the whole graph is initially considered as a unique community and a divisive algorithm tries iteratively to prune the inter-community edges. As an example, we refer to the algorithm of Newman and Girvan [16,38]. This algorithm uses the edge betweenness centrality to detect such inter-community edges, where the edge betweenness centrality metric counts the number of short-paths between all couples of nodes to which this edge belongs. Thus, the edges are iteratively removed in a decreasing order of their betweenness centrality, driven by the idea that inter-community edges belong to many short paths allowing to cross from a community to another. The weakness of this algorithm is the heavy computation needed for the edge betweenness centrality metric. As an improvement, another divisive algorithm is proposed by Radicchi et al. [44] where a more local and less time consuming metric (i.e., the edge clustering coefficient) is used. It is based on the triplets of nodes and it quantifies how clustered the graph is.

The second family refers to the agglomerative algorithms or the bottom-up strategy. Our community detection algorithm can be categorised as an agglomerative algorithm. Unlike the divisive algorithms, an agglomerative algorithm starts with v communities, where v represents the number of nodes in the graph, which means that initially each node forms a community. Next, the aim is to merge communities with respect to an objective function or a metric. A widely used metric in the literature [8,36], is the modularity φ of Newman and Girvan, formulated in Eq. (1). In weighted graphs, it expresses whether the weights of the edges inside communities are greater than a random edge distribution with the same properties. However, it has been shown that modularity-based algorithms suffer from some limits [12,26]. For example, it is hard to detect small communities, also maximizing the modularity yield to split the big communities. In order to overcome the first limit, another version of the modularity is proposed in [48] where a new parameter is added to adjust the community detection resolution.

$$\varphi = \frac{1}{2M}\sum_{i=1}^{v}\sum_{j=1}^{v}\left(e^{w}_{n_i,n_j} - \frac{D_{n_i}D_{n_j}}{2M}\right)\delta\left(c_{n_i}, c_{n_j}\right) \tag{1}$$

where:

Table 1
The computational complexities of the above community detection algorithms.

Algorithm	Complexity
Schlitter et al. [51]	$\mathcal{O}(v+m)$
Newman and Girvan [16]	$\mathcal{O}(m^2v)$ on sparse graphs
Radicchi et al. [44]	$\mathcal{O}(v^2)$ on sparse graphs
Clauset et al. [8]	$\mathcal{O}(v\log^2 v)$ on sparse graphs
Blondel et al. [4]	linear in m
Pons and Latapy [42]	time $\mathcal{O}(mv^2)$ and space $\mathcal{O}(v^2)$ in the worst case
Raghavan et al. [45]	near-linear ($\mathcal{O}(m)$ for each iteration)
Rosvall and Bergstrom [50]	$\mathcal{O}(m)$
Ronhovde and Nussinov [49]	$\mathcal{O}(m^{1.3}\log v)$

- $M = \sum_{i=1}^{v}\sum_{j=i+1}^{v} e^{w}_{n_i,n_j}$, with v the number of nodes and $e^{w}_{n_i,n_j}$ the weighted edge linking n_i and n_j
- c_{n_i} is the community of node n_i,
- $\delta\left(c_{n_i}, c_{n_j}\right) = 1$, if $c_{n_i} = c_{n_j}$, 0 otherwise,
- D_{n_i} is the weighted graph degree of node n_i defined as follows: $D_{n_i} = \sum_{j=1}^{v} e^{w}_{n_i,n_j}$.

The aforementioned community definition with more edges within communities than outside is still valid in the weighted case. Indeed, exchanging each weighted edge by as much edges having 1 as weight value leads to a multi-graph [37]. As an example of an agglomerative algorithm, we can refer to the algorithm of Blondel et al. [4], which provides better compromise between the accuracy of the estimated modularity value and complexity than the greedy approaches, like in Clauset et al. [8]. In the algorithm of Blondel et al., a merging step is achieved until a local maximum of modularity is reached; no merging can improve the modularity. In the second phase, each detected community is replaced by a meta-node. These two phases are iteratively performed until the modularity cannot be increased anymore. However, it has been shown in [25] that the performance of the algorithm for large graphs is not as good as for small graphs. Another agglomerative algorithm is the algorithm of Pons and Latapy [42] which uses random walks of a predefined length (i.e., the length of the walk is a parameter of the algorithm) to iteratively merge communities. The algorithm is motivated by the fact that, if a random walk is done on a graph, then this walk has a great probability to remain in the same community. However, one among its drawbacks is the large amount of memory needed, especially for large graphs.

2.1.2. Similarity based algorithms

The second method consists in computing a similarity or a distance function between each couple of nodes. Eq. (2) is an example of distance function, noted as $dist_{n_i,n_j}$, where $e^{w}_{n_i,n_j}$ represents the positive value of the weighted edge between the i^{th} and j^{th} nodes and *Const* represents a constant. Then, a clustering algorithm is applied, like in [51]. However, it is well-known that clustering techniques are highly sensitive to the choice of the similarity function.

$$dist_{n_i,n_j} = \begin{cases} 0, & \text{if } n_i = n_j \\ \dfrac{1}{e^{w}_{n_i,n_j}}, & \text{if } e^{w}_{n_i,n_j} > 0 \\ Const, & \text{if } e^{w}_{n_i,n_j} = 0 \end{cases} \tag{2}$$

We point out that there exist other approaches. For example, the authors of [45] propose a fast algorithm, which is based on an iterative node labeling process. Indeed, each node has to choose the most used label in its neighboring. The algorithm stops when the process leads to a consensus on one label for each community. In [50], a theoretical approach is used to detect communities. The method decomposes the network into modules by compressing information flows. These are highlighted in a map relying on probability flow of random walks. Additionally, Potts model based technique is proposed by Ronhovde and Nussinov [49] for multi-resolution graph structure. They minimize the Hamiltonian Potts spin model where the node-community belonging is represented by the spin state.

For more details, we refer to [25,32] where comparative studies of community detection algorithms are conducted whereas Table 1 presents the complexities of the above community detection algorithms.

2.2. Community visualization and tools

In this section, we make a non-exhaustive survey of related visualization methods and tools. There is a bunch of existing visualization techniques and tools, each of which is more or less appropriate depending on the analytic task. For the readers interested by a more detailed survey on graph drawing visualizations and software, we refer to [20,21,28], and more recently to [55].

2.2.1. Graph and community visualization

Various graph representations exist in the literature. These representations can be divided into three major categories.

First, node-link diagrams is the most popular representation of graphs. The nodes are depicted by items (e.g., small circles, small rectangles, crosses... etc.) whereas the relationship between two nodes is represented by a line between the related point-like items. The second category is the matrix-based representation. In this case the graph is visualized by a two-dimensional array. Row and columns depicts the nodes and the cells depict the edges. The third category uses the two aforementioned categories together. Indeed, they can be used in multiple views, like in [18], as well as in one single view by superposing node-link diagram on a matrix grid, like in [41]. The denser part of the graph may also be depicted by matrix representations, like in [19].

Each representation has its strengths and drawbacks depending on the visual task that has to be carried out [15,30]. Choosing a representation instead of another is crucial and has a great impact on the visualization efficiency. For example, with matrix-based representation it would be difficult to follow a path from a source node to a target node. The node-link diagram representation allows us to perform this task more easily and more quickly. The path detecting task is an important task in social networks to verify whether two nodes of the network communicate via intermediate nodes. In this work, the node-link diagram representation is used and a force-directed algorithm [23] is used as graph drawing algorithm. In this context, nodes are as electrically charged particles which try to repulse each others while the edges keep them closer. The system becomes stable when the force reaches an equilibrium. This algorithm is usually very efficient for producing nice drawings which meet the aesthetic criteria of graph drawing [43]. However, node-link diagrams has shown some limits when it deals with large graphs, like visual clutter due to node-edge superposition and edge crossing issue. In order to prevent these issues, circle packing has been chosen to visualize the detected communities.

2.2.2. Visualization tools

In the literature, we can find a large variety of tools for graph visualization and for analyzing networks. We present here the most relevant ones for our work. Gephi [2] is an exploratory graph data analysis tool and it enables user interactions (e.g., dynamic filtering) and graph manipulation (e.g., node-link layout setting). Additionally, standard statistics for social network analysis are provided, like the modularity. Graphviz [10] describes a graph drawing software which provides several layout programs with features for the graph layout. Meerkat [6] proposes a social analysis tool which enables network visualization, automatic community detection and identification of individuals representing the leaders in the underlying communities. Netlytic [17] is a social network analyser, which catches data from social media sites (i.e., online discussions), like Twitter. Netlytic allows us to discover most frequent topics and emerging themes of discussions, in addition to network visualization. NetworKit [54] is a toolkit for large-scale network analysis including techniques like parallelism and scalable algorithms for related social network analysis problems. NetworkX [35] is a package for manipulating complex graph structure. It handles edge weighted graphs and provides classical graph generators. NetworkX as so many others tools uses standard graph drawing algorithms and node-link diagram. NodeXL [53] is a template for Microsoft Excel which allows us to visualize and analyse networks imported via spreadsheets or directly linked to social networks, like Twitter. NodeXL provides automatic node grouping and makes the groups visually detectable relying on node shapes and colors. An interesting and widely used library is SNAP [31]. The latter is a general purpose network analysis which is also available through NodeXL. It manipulates large graphs and calculates structural properties. Last and not least, we can cite the igraph package [9], which is a collection of network analysis tools. Its library provides manipulation, visualization of networks and also calculation of different networks metrics.

3. Community extraction via *Tribase*

In this section, we present a new community detection algorithm, called *Tribase*. The main idea of *Tribase* is to select a collection of triangles as a skeleton of the communities structure. The motivations behind this are twofold. First, triangles play an important role in the communities structure and they enable to capture the overall community organization in the graph, especially in social networks [14]. As example, we can mention the friendship graph where there are more chances that two friends share a common friend. Second, topologically speaking, triangles are the smallest group of interconnected nodes after edges and they are considered in various networks metrics. As an example, we refer to the global clustering coefficient of [40] and the 3-cycle cut ratio of [22] for directed networks. After this preliminary step, the most attractive communities are iteratively merged until a community-like structure is reached. Our algorithm can be categorized as an agglomerative algorithm since it is a bottom-up approach. Knowing that the size of the communities in a real-world context is heterogeneous and due to the above modularity limits, the modularity is not used as a metric in our algorithm.

3.1. Definitions and notations

Before introducing more in detail *Tribase*, we give some definitions and notations. Let $G = (N, E, E^w)$ be a graph where N, E and E^w denote respectively the set of nodes of size v, the set of edges of size m and the weights of the edges. The set of communities is $Cs = \{c_1, c_2, \ldots, c_q\}$ where $q \leq v$. Furthermore, the community degree can be interpreted as the global connectivity degree of a given community, which is defined by $CD_{c_g} = \sum_{n_i \in c_g} D_{n_i}$. The community intra-weight $intra_{c_g}$ and inter-weight $inter_{c_g,c_h}$ can be interpreted respectively as the local community weight and the shared weight between two communities. Additionally, we introduce two comparison functions defined below, α and β. These compare community weights and return a Boolean indicating whether the inequalities are meet.

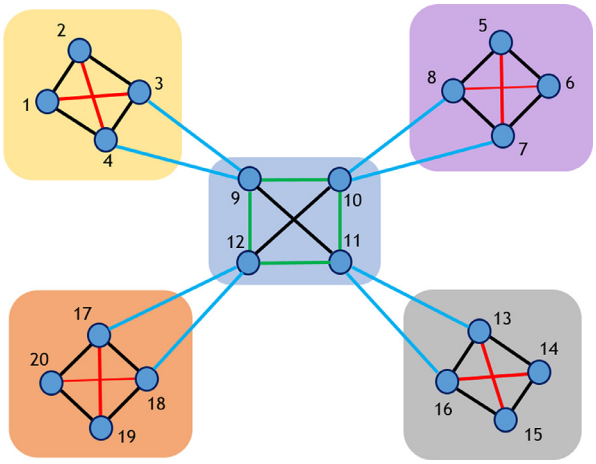


Fig. 1. Graph example, with 1, 2, 3 and 4 as edge weights for respectively, black, blue, green and red edges. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- $intra_{c_g} = \sum_{n_i \in c_g} \sum_{n_j \in c_g} \frac{e^w_{n_i,n_j}}{2}$,
- $inter_{c_g,c_h} = \sum_{n_i \in c_g} \sum_{n_j \in c_h} e^w_{n_i,n_j}$,
- $\alpha\left(c_g, inter_{c_g,c_h}\right) = \begin{cases} true, & \text{if } inter_{c_g,c_h} \geqslant intra_{c_g} \\ false, & \text{otherwise} \end{cases}$
- $\beta\left(c_g, inter_{c_g,c_h}\right) = \begin{cases} true, & \text{if } inter_{c_g,c_h} \geqslant (CD_{c_g} - intra_{c_g} - inter_{c_g,c_h})\Omega \\ false, & \text{otherwise} \\ & \text{such as } 0 \leq \Omega \leq 0.5 \end{cases}$

Two communities c_g and $c_h \in Cs$ are adjacent if $inter_{c_g,c_h} \neq 0$. What makes a community dominant is its *intra* and its *CD*, the greater are these values, the more dominant is the community. On the one hand, *intra* value can be interpreted as the force that repulses the other communities in the iterative merging process. On the other hand, *CD* value can be seen as the force that absorb the adjacent communities.

The objective of the unweighted Maximum Triangle Packing problem (MTP for short) is to find the largest collection of pairwise node-disjoint triangles (i.e., cliques of size 3) of a graph among all possible triangles, we refer to [1] for algorithms addressing the MTP problem. A weighted version of the MTP can be formulated as the linear program [3]. We point out that there exists another formulation of the weighted MTP in [7]:

$$\begin{aligned} Maximize \quad & \sum_{j=1}^{T} k^w_j x_j \\ s.t. \quad & \sum_{j=1}^{T} B_{ij}.x_j \leq 1, \qquad \forall i \in \{1, \ldots, v\} \\ & x_j \in \{0, 1\}, \qquad \forall j \in \{1, \ldots, T\} \end{aligned} \tag{3}$$

where:

- x_j are the decision variables,
- k^w_j is the triangle weight which is equal to the sum of edge weights within the j^{th} triangle,
- $B \in \{0, 1\}^{v \times T}$ is the node-triangle belonging matrix.

For instance, let us consider the graph in Fig. 1. A fast solution for the weighted MTP can be obtained by selecting the triangles in a decreasing order with respect to their triangle weight values k^w_j. Thus, we obtain the following result: $\{(n_1, n_3, n_4), (n_6, n_7, n_8), (n_9, n_{10}, n_{11}), (n_{13}, n_{14}, n_{16}), (n_{17}, n_{18}, n_{20})\} = 6 + 6 + 7 + 6 + 6 = 31$, is returned as solution for the weighted MTP problem because, $k^w(n_9, n_{10}, n_{11}) > k^w(n_1, n_3, n_4) \geqslant k^w(n_6, n_7, n_8) \geqslant k^w(n_{13}, n_{14}, n_{16}) \geqslant k^w(n_{17}, n_{18}, n_{20})$.

3.2. Tribase *algorithm*

In this section, we introduce our algorithm for community detection in weighted graphs as illustrated in Algorithm 1. The global idea behind *Tribase* is to consider, in a first step, 3-sized cycles as skeleton for to catch the overall community

Algorithm 1 *Tribase* algorithm.

Input: the graph G, Ω and *userSortChoice*;
Output: a collection of communities *Cs*.

```
BEGIN
var Cs ← {c_g}^v, c_g ← {n_g}, ∀g ∈ {1...v};
inter_{c_g} ⟵ 0, CD_{c_g} ⟵ D_{n_g}, ∀g ∈ {1...v};
triangles ⟵ findTriangles(G);
for each tri in triangles do
  Cs ← Cs \ nodes(tri);
  Cs ← Cs ∪ {c_{|Cs|+1}} where c_{|Cs|+1} ← {nodes(tri)};
  Compute intra_{c_{|Cs|+1}} and CD_{c_{|Cs|+1}};
end for
Cs ← sortInitialCommunities(Cs, userSortChoice);
do
  for each c_g in Cs do
    var sortedCs ⟵ sortAdjacentCommunities(c_g, Cs, G);
    for each c_h in sortedCs do
      if ((α(c_g, inter_{c_g,c_h}) and β(c_g, inter_{c_g,c_h}))
        or (α(c_h, inter_{c_g,c_h}) and β(c_h, inter_{c_g,c_h})))
        then
        c_g ← c_g ∪ c_h;
        intra_{c_g} ← intra_{c_g} + intra_{c_h} + inter_{c_g,c_h};
        Cs ← Cs \ c_h;
      end if
    end for
  end for
while a merging is possible
Return Cs;
END.
```

organisation in the graph. In the second step, the weights of adjacent communities are pairwise compared in order to allow dominant communities to gain in size. This second step is iteratively repeated until no community merging is possible.

For better understanding of *Tribase* algorithm, we consider the example in Fig. 1, with $\Omega = 0.1$ and *userSortChoice* ← *intra*. The initialization step yields to $Cs \leftarrow \{(n_i)\}$ for each $i \in [1, 20]$. After *findTriangles*() and *sortInitialCommunities*() return $Cs \leftarrow \{(n_9, n_{10}, n_{11}), (n_1, n_3, n_4), (n_6, n_7, n_8), (n_{13}, n_{14}, n_{16}), (n_{17}, n_{18}, n_{20}), (n_2), (n_5), (n_{12}), (n_{15}), (n_{19})\}$. Then, the adjacent communities are pairwise compared allowing dominant communities to gain in members. The first community considered is (n_9, n_{10}, n_{11}), the latter is compared with $\{(n_1, n_3, n_4), (n_6, n_7, n_8), (n_{13}, n_{14}, n_{16}), (n_{12})\}$. The merging condition is not met with the communities $\{(n_1, n_3, n_4), (n_6, n_7, n_8), (n_{13}, n_{14}, n_{16})\}$, because $\alpha((n_9, n_{10}, n_{11}), inter_{(n_9,n_{10},n_{11}),(n_1,n_3,n_4)}) =$ false; $4 \not\geq 7$ and also $\alpha((n_1, n_3, n_4), inter_{(n_9,n_{10},n_{11}),(n_1,n_3,n_4)}) =$ false; $4 \not\geq 6$. The same condition occurs for the communities $\{(n_6, n_7, n_8), (n_{13}, n_{14}, n_{16})\}$. However, the merging condition is met with (n_{12}), because $\alpha((n_9, n_{10}, n_{11}), inter_{(n_9,n_{10},n_{11}),(n_{12})}) =$ true; $7 \geq 7$ and also $\beta((n_9, n_{10}, n_{11}), inter_{(n_9,n_{10},n_{11}),(n_{12})}) =$ true; $7 \geq (12 \times 0.1)$, thereby (n_9, n_{10}, n_{11}) and (n_{12}) are merged leading to $Cs \leftarrow \{(n_9, n_{10}, n_{11}, n_{12}), (n_1, n_3, n_4), (n_6, n_7, n_8), (n_{13}, n_{14}, n_{16}), (n_{17}, n_{18}, n_{20}), (n_2), (n_5), (n_{15}), (n_{19})\}$. The next community to be considered is (n_1, n_3, n_4). The latter is compared with the adjacent communities $\{(n_9, n_{10}, n_{11}, n_{12}), (n_2)\}$. The merging condition is not met with $(n_9, n_{10}, n_{11}, n_{12})$, in contrast to (n_2) where $\alpha((n_1, n_3, n_4), inter_{(n_1,n_3,n_4),(n_2)}) = \beta((n_1, n_3, n_4), inter_{(n_1,n_3,n_4),(n_2)}) =$ true; $6 \geq 0$. The same happens when the remaining communities to be considered $\{(n_6, n_7, n_8), (n_{13}, n_{14}, n_{16}), (n_{17}, n_{18}, n_{20})\}$ are compared. Thus, *Tribase* algorithm ends with $Cs \leftarrow \{(n_1, n_2, n_3, n_4), (n_5, n_6, n_7, n_8), (n_9, n_{10}, n_{11}, n_{12}), (n_{13}, n_{14}, n_{15}, n_{16}), (n_{17}, n_{18}, n_{19}, n_{20})\} = \{c_1, c_2, c_3, c_4, c_5\}$ which are, respectively, colored in Fig. 1 in yellow, purple, blue, grey and orange.

In order to avoid re-computing *inter* at each community comparison, an adjacency list of communities with the corresponding *inter* is stored. This adjacency list is updated after each community merging. For instance, the adjacency list of the graph in Fig. 1 is presented in [4].

$$
\begin{aligned}
&c_1 :\longrightarrow \{c_3 : 4\}\\
&c_2 :\longrightarrow \{c_3 : 4\}\\
&c_3 :\longrightarrow \{c_1 : 4\} \longrightarrow \{c_2 : 4\} \longrightarrow \{c_4 : 4\} \longrightarrow \{c_5 : 4\}\\
&c_4 :\longrightarrow \{c_3 : 4\}\\
&c_5 :\longrightarrow \{c_3 : 4\}
\end{aligned}
\tag{4}
$$

The trivial triangles computation yield to a complexity of $\mathcal{O}(v^3)$. This complexity can be decreased using the Laptay's matrix multiplication algorithm [29] leading to a complexity of $\mathcal{O}(v^z)$, such as $z < 2.376$. The complexity of each iteration in

the merging process is $\mathcal{O}(p_i^2)$ in the worst case (i.e., fully adjacent communities), where p_i corresponds to the number of communities in each i^{th} iteration. Thus, the global computational complexity of the algorithm is $\mathcal{O}(v^z + rp^2)$ in the worst case, where $z < 2.376$, and r the number of iteration of the merging process.

3.3. Comparison study

In this section, we report the results of a comparative study of *Tribase* algorithm with five algorithms of the literature on instances generated by the LFR benchmark. The LFR generation scheme uses power laws for degrees and community size distributions with μ_t and μ_w as mixing parameters which, respectively, denote the proportion between the number of edges within and outside communities and the proportion of the edge weight within and outside communities. For example, a specific node will have $1 - \mu_t$ edges within its community and μ_t edges outside of its communities. For further details about the LFR benchmark we refer to [27], and to [24] for the weighted version. The number of vertices for the generated instances are: 1000 and 5000. The community sizes are in [20, 100] for 1000 nodes and in [20, 500] for 5000 nodes. The average node degree is set to 20, whereas the maximum node degree is 50 for 1000 nodes and 100 for 5000 nodes. In order to assess the scalability of our approach, we consider larger instances of 10000 and 50000 nodes, whereas the maximum node degree is 200 and the community sizes are in [20, 1000]. For each μ_t and μ_w ten instances are generated. Additionally, for all the instances, we set *userSortChoice* and Ω of the *Tribase* algorithm, respectively, to *intra* and 0.1. The average values of the medium-sized and large-sized instances are presented in Tables 2 and 3.

Hereafter, we give some details on the experimental environment. All the tests are done on Intel Core i7-4800MQ CPU (2.7 GHz, 2.7 GHz), with 8 GB of RAM, under Windows 8.1 OS. As an instance, Fig. 2 is generated with the LFR benchmark where $v = 5000$ and $m \approx 47600$. These data are depicted in Fig. 2 by exploiting the *NLCOMS* tool (cf. Section 4).

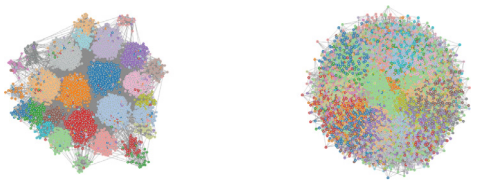
The comparison of the *Tribase* algorithm is conducted with five community detection algorithms [4,8,42,45,50] previously mentioned (cf. Section 2) and respectively denoted by, *label*, *greedy*, *walk*, *infomap* and *Louvain*. The reasons for which we chose these algorithms are twofold. First, they consider community detection in the weighted context, which is not the case of all the algorithms in the related work section. Second, they cover several techniques of the community detection domain. The implementations of these algorithms are available in the igraph library [9] and are used in this comparison

Table 2
Characteristics of the generated instances with 1000 and 5000 nodes.

(μ_t, μ_w)	v	m	T
(0.1, 0.1)	1000 (5000)	9776 (47515.5)	25836.9 (89551.9)
(0.2, 0.2)	1000 (5000)	9757, 5 (47600.3)	18610.9 (61258.6)
(0.3, 0.3)	1000 (5000)	9614, 7 (47666.6)	12582.3 (43437.9)
(0.4, 0.4)	1000 (5000)	9824, 3 (47791.3)	8853.6 (33305.6)
(0.5, 0.5)	1000 (5000)	9785, 7 (47570.6)	6101.8 (20196.3)
(0.6, 0.6)	1000 (5000)	9674, 3 (47683.5)	3765.7 (13140)
(0.7, 0.7)	1000 (5000)	9791, 6 (47925.7)	2754.6 (9221.7)
(0.5, 0.1)	1000 (5000)	9790, 3 (47991)	5916.1 (20627)
(0.5, 0.2)	1000 (5000)	9671, 2 (47538.2)	5454.1 (20655.5)
(0.5, 0.3)	1000 (5000)	9807, 2 (47854.4)	5935.2 (19542)
(0.5, 0.4)	1000 (5000)	9745, 7 (47711.1)	5584.5 (20510.5)
(0.5, 0.5)	1000 (5000)	9722, 9 (47863.6)	5908.7 (20174.6)
(0.5, 0.6)	1000 (5000)	9846, 1 (47644.2)	5756.6 (19155.3)
(0.5, 0.7)	1000 (5000)	9741, 6 (47494)	5705 (20110.6)

Table 3
Characteristics of the generated instances with 10000 and 50000 nodes.

(μ_t, μ_w)	v	m	T
(0.1, 0.1)	10000 (50000)	102359, 3 (509770, 7)	245673, 6 (2267897, 5)
(0.2, 0.2)	10000 (50000)	102097, 5 (510147, 8)	184417, 6 (909905, 4)
(0.3, 0.3)	10000 (50000)	101544, 7 (509971, 2)	132661, 1 (646199, 2)
(0.4, 0.4)	10000 (50000)	102302, 7 (508295, 8)	91671, 5 (439789, 1)
(0.5, 0.5)	10000 (50000)	101511, 2 (508653, 1)	63539, 7 (265590)
(0.6, 0.6)	10000 (50000)	102393, 7 (508419, 9)	45834, 5 (158937, 7)
(0.7, 0.7)	10000 (50000)	101807, 3 (509993, 1)	32736, 8 (85318, 1)
(0.5, 0.1)	10000 (50000)	101259, 2 (463792, 4)	64413, 9 (277023, 5)
(0.5, 0.2)	10000 (50000)	102015, 1 (508332, 5)	65614, 6 (267949)
(0.5, 0.3)	10000 (50000)	101822, 6 (509946, 3)	64947, 4 (271302, 9)
(0.5, 0.4)	10000 (50000)	102712, 9 (510413, 2)	67966, 4 (279692, 4)
(0.5, 0.5)	10000 (50000)	101265, 3 (509787, 6)	64117, 7 (285310)
(0.5, 0.6)	10000 (50000)	101473, 2 (511505, 1)	63962, 6 (276371, 5)
(0.5, 0.7)	10000 (50000)	101882, 4 (509283)	64275 (276641, 4)



(a) (b)

Fig. 2. LFR instances with $v = 5000$ and $m \approx 47600$, (a) $\mu_t = \mu_w = 0.1$, (b) $\mu_t = \mu_w = 0.4$.

study. Furthermore, one have to compare the ground-truth communities of the LFR benchmark and the proposed community partition by the algorithms. To do so, we use the Rand Index of [46], noted here *RI* and expressed by Eq. (5).

$$RI(Cs_1, Cs_2) = \frac{m_{1,1} + m_{0,0}}{m_{1,1} + m_{1,0} + m_{0,1} + m_{0,0}} \tag{5}$$

where $m_{1,1}$ and $m_{0,0}$ represent how often Cs_1 and Cs_2 are in agreement to cluster couples of nodes, whereas the disagreement case is represented by $m_{1,0}$ and $m_{0,1}$. Closer values of *RI* to 1 more an algorithm returns community partition, which matches the ground-truth in terms of nodes couple classification. Furthermore, we use the correlation coefficient (*CC* for short) to evaluate whether the returned number of communities by an algorithm matches the optimal number of communities of the ground-truth benchmarks.

Fig. 3 shows, the obtained *RI* for the instances with 1000 and 5000 nodes where $\mu_t = \mu_w$ and $\mu_t = 0.5$. As in [25], the instances with $\mu_t = 0.5$ assume that there is as much edges within a community as outside a community for each node, which means that there is no topological community dominance. In this context, a community detection algorithm devised for weighted networks has to rely only on weights (i.e., in this case, varying μ_w to conduct an evaluation) to identify communities. From Fig. 3, we remark that *Tribase* algorithm is very competitive and it has a high *RI* values up to 0.5. After this limit value it becomes trickier to identify the original communities for almost all the algorithms, because there are more inter-community edges than intra-communities edges. We notice also that even though there is a balance between the number of inter-community and intra-communities edges, the aforementioned observations are still valid. However, the *greedy* algorithm seems to be not as good as the other algorithms while *Louvain* algorithm provides very good results.

Regarding the *CC* values, Fig. 4 shows, the obtained *CC* for the instances with 1000 and 5000 nodes where $\mu_t = \mu_w$ and $\mu_t = 0.5$. From this figure, we note that up to the limit 0.5 *Tribase* algorithm finds the optimal number of communities for all the instances. Furthermore, the *greedy* algorithm is very far from the optimal number of communities especially when $\mu_t = \mu_w$. However, reaching the mixing parameter 0.4 in Fig. 4(c), *Louvain* and *infomap* algorithms miss some optimal communities.

Additionally, the modularity and the running time for the medium-sized instances are presented respectively in Figs. 5 and 6. We observe that the computation time is acceptable for the *Tribase* algorithm while the *infomap* algorithm is time consuming because of simulated annealing approach, which could makes the algorithm quite slow. Finally, the modularity values are almost the same for all the algorithms with some exceptions when $\mu_t > 0.5$ and for the *greedy* algorithm.

In order to evaluate the scalability of the proposed algorithm *Tribase*, we consider larger instances with 10000 and 50000 nodes. We decide to not include the *greedy* algorithm in this comparison due to the poor results obtained for 1000 and 5000 nodes. Additionally, preliminary tests shows that *walk* algorithm seems to be space memory consuming using the above experimental environment. As a consequence, we choose to not include it too in this comparison with large-sized instances.

Fig. 7 shows the obtained *RI* for the instances with 10, 000 and 50, 000 nodes, where $\mu_t = \mu_w$ and $\mu_t = 0.5$. From Fig. 7, we remark that *Tribase* algorithm is still very competitive and it shows interesting results. We observe also that the *label* algorithm after the limit 0.5 returns bad results, because it is difficult to be efficient when there are more inter-community edges than intra-communities edges.

Concerning the *CC* values, Fig. 8 shows the obtained *CC* for the instances with 10, 000 and 50, 000 nodes where $\mu_t = \mu_w$ and $\mu_t = 0.5$. From this figure, we can say that the *infomap* algorithm is far from the optimal number of communities when $v = 10,000$. This observation is valid also for the *Louvain* algorithm when $v = 50,000$. However, our proposed algorithm seems to catch the community organisation very well up to 0.4. These results show that using a collection of triangles in the first step of the community detection process can yield to interesting results.

Furthermore, the modularity and the running time for the large-sized instances are presented respectively in Figs. 9 and 10. These figures show that the *infomap* algorithm still time consuming. Concerning the *Tribase*, one could say that it provides a very good compromise between efficiency and computation time. Finally, the modularity values are almost the same for all the algorithms with some exceptions when $\mu_t > 0.5$.

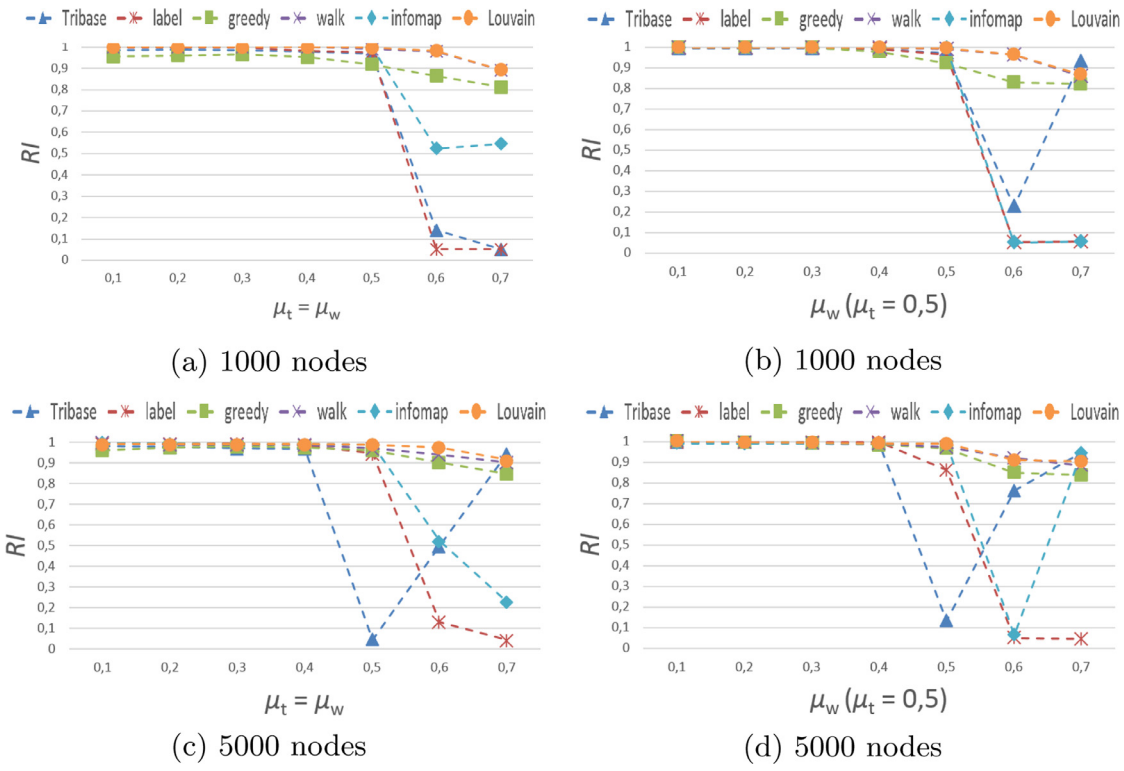


Fig. 3. The obtained *RI* for instances with 1000 and 5000 nodes.

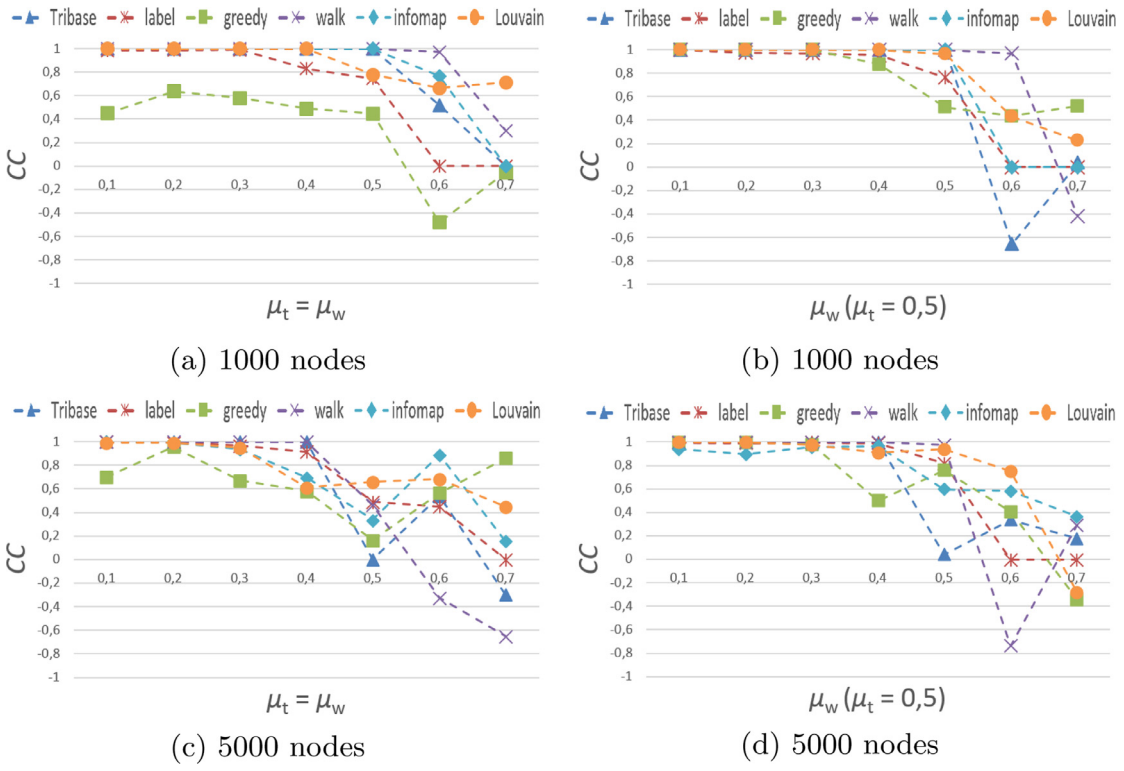


Fig. 4. The obtained *CC* for instances with 1000 and 5000 nodes.

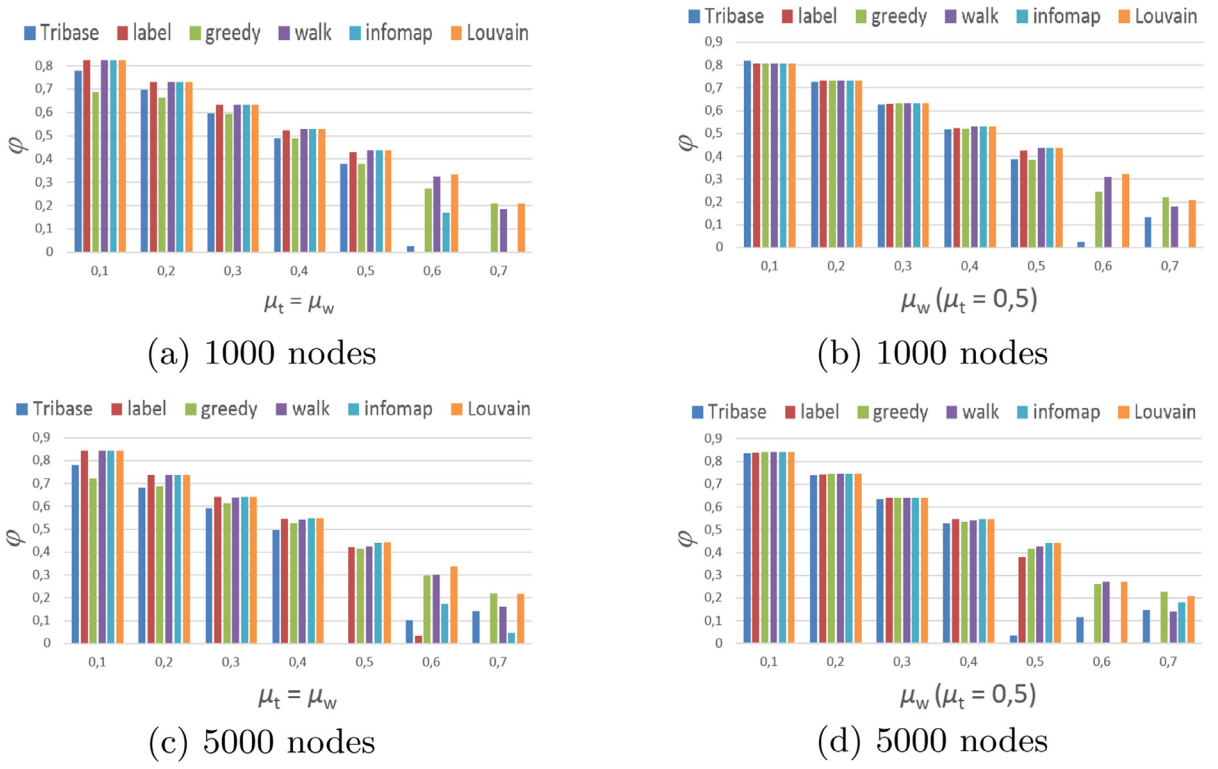


Fig. 5. The obtained φ for instances with 1000 and 5000 nodes.

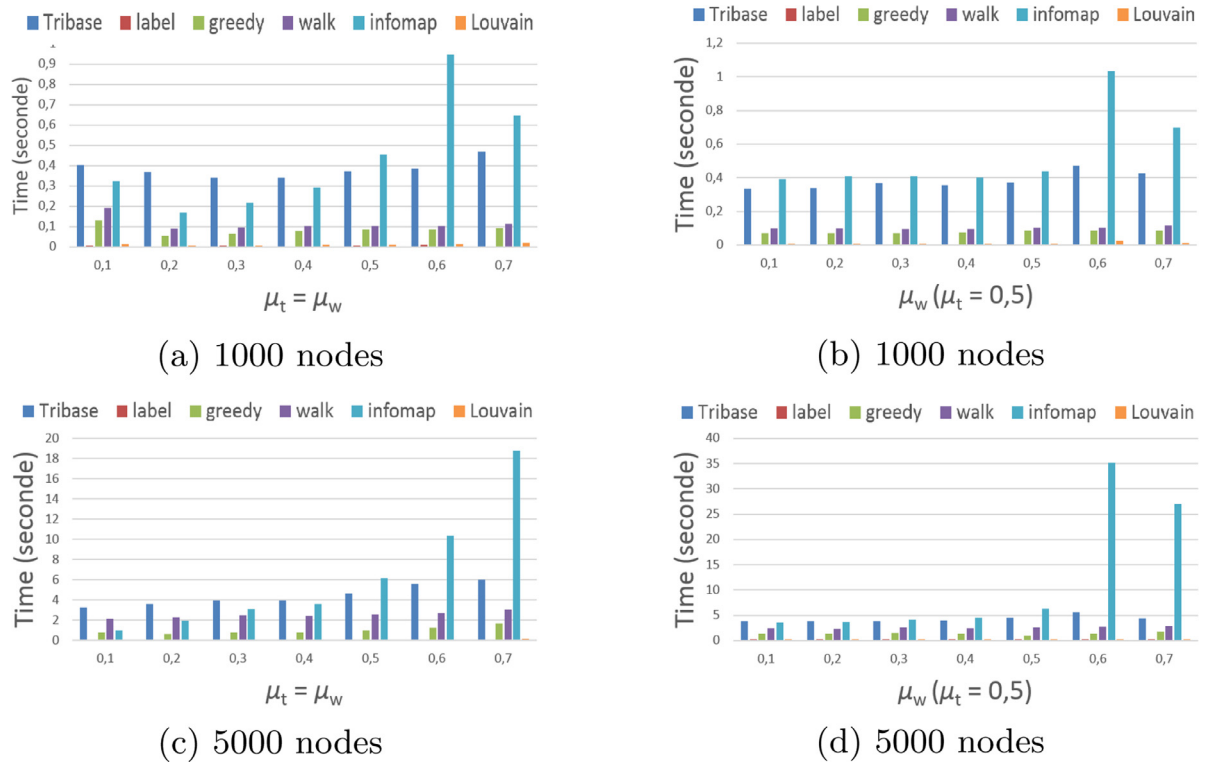


Fig. 6. The computation time for instances with 1000 and 5000 nodes.

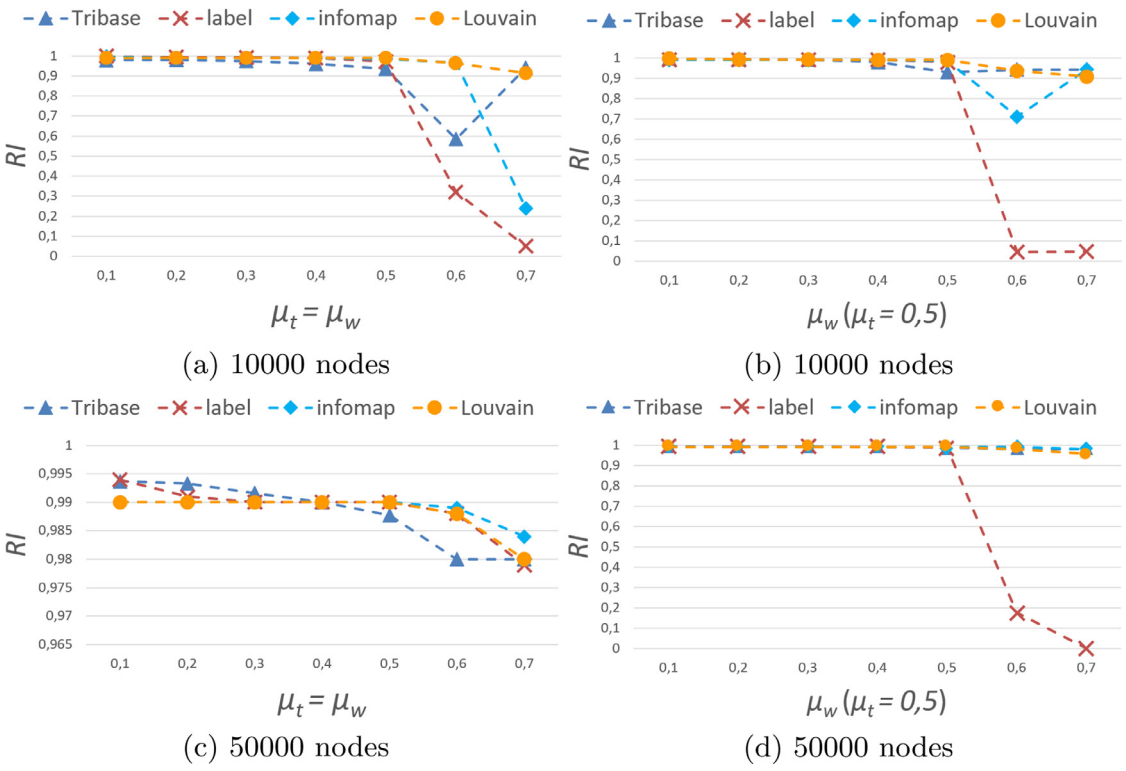


Fig. 7. The obtained *RI* for instances with 10000 and 50000 nodes.

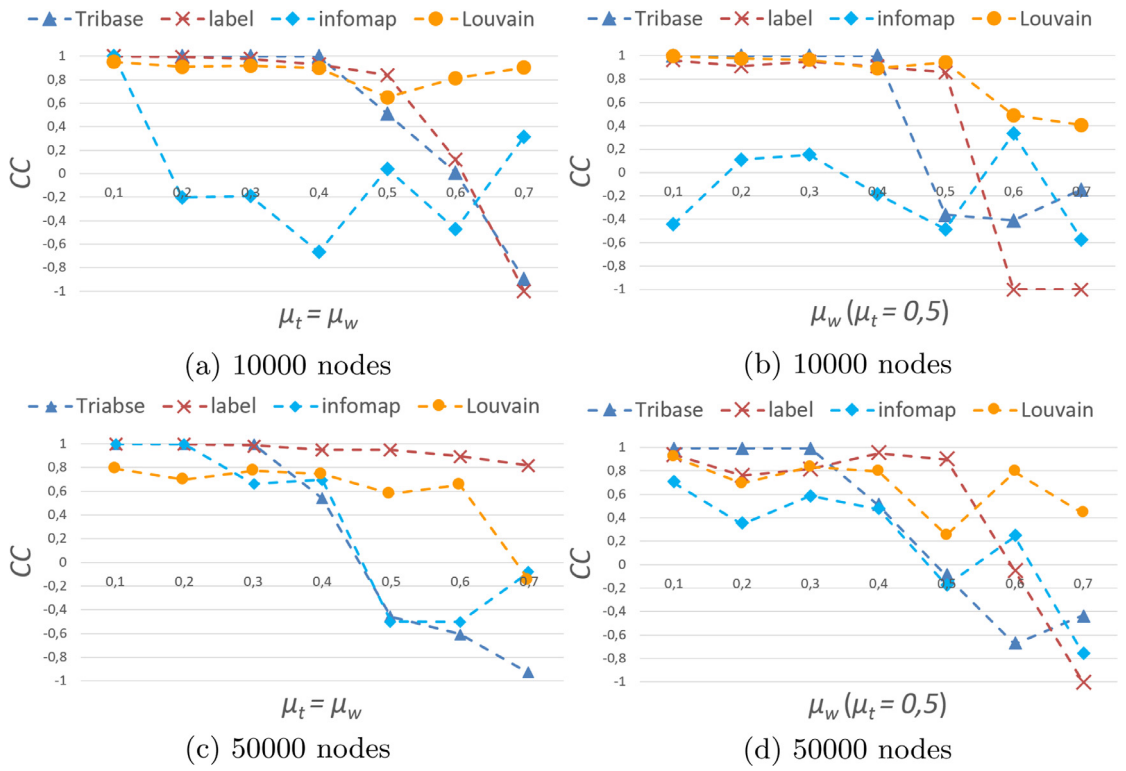


Fig. 8. The obtained *CC* for instances with 10000 and 50000 nodes.

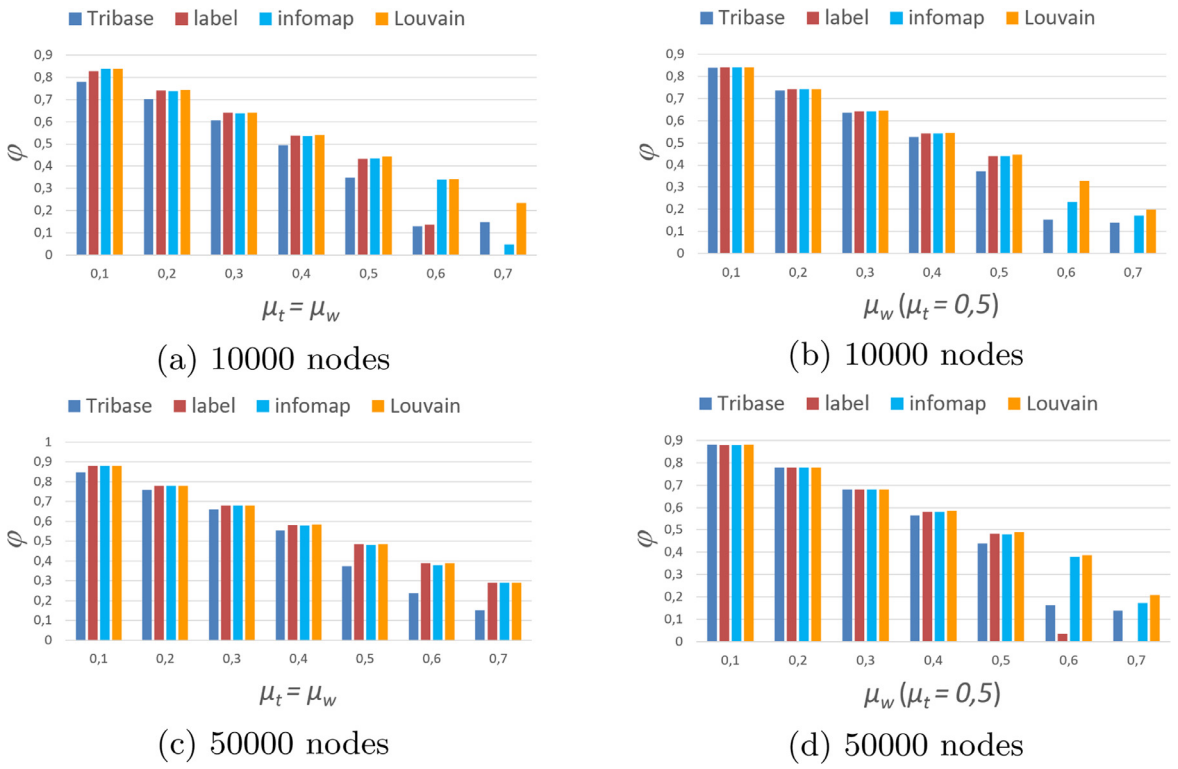


Fig. 9. The obtained φ for instances with 10000 and 50000 nodes.

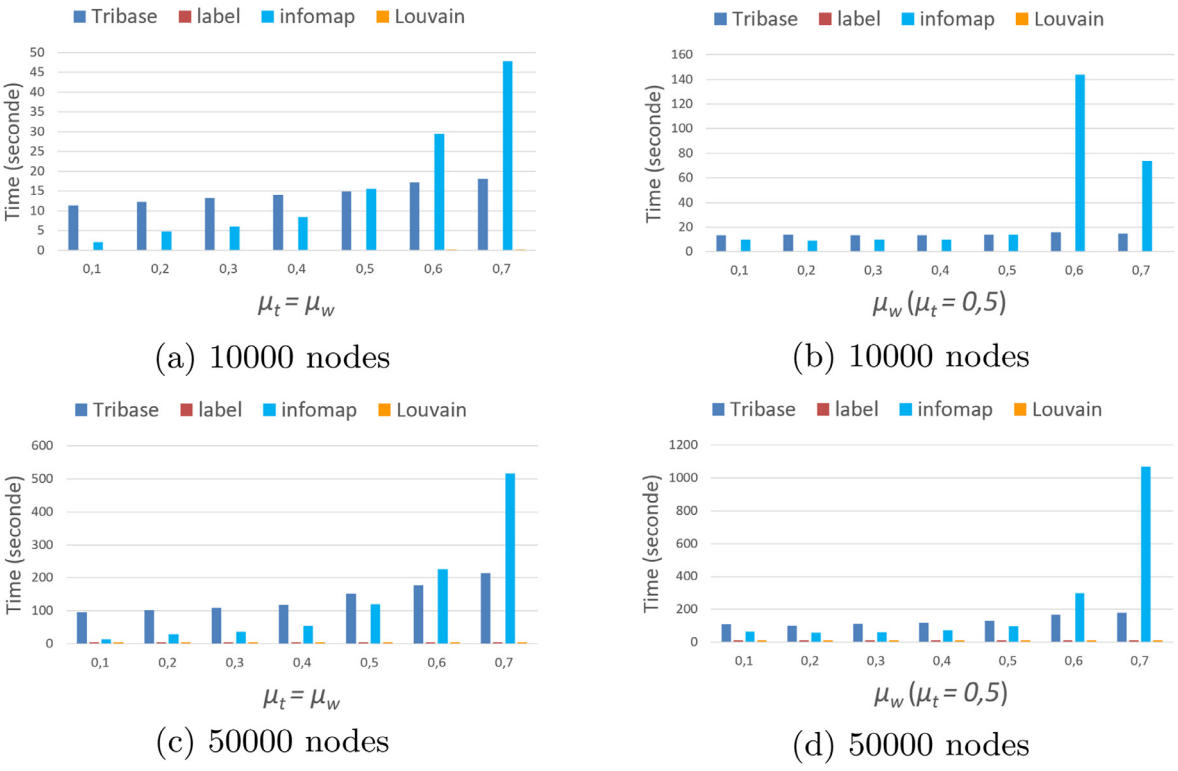


Fig. 10. The computation time for instances with 10000 and 50000 nodes.

4. Community visualization with *NLCOMS*

The second step of the approach proposed in this paper relies on an interactive visualization of the communities that provides gradual knowledge acquisition. Nevertheless, depict a collection of communities with the appropriate visualization is hard as much as the detection process, especially, if additional information to the community structure has to be visualized. Furthermore, we have to avoid misleading user interpretation resulting from inappropriate community visualization even if the community detection process identifies the right communities. To this end, an application called NLCOMS (Node-Link and COMmunitieS) has been implemented for visualizing, analyzing and exploring complex network and their underlying patterns. *NLCOMS* is a visual interactive application for social network analysis based on node-link representation of networks and circle packing for the detected communities. As discussed in the related work section, focusing only on node-link representation, especially for large networks, can lead to visual clutter. With *NLCOMS*, the exploration process is gradually provided relying on the circle packing representation. Fig. 11 shows the circle packing representation of the detected communities by *Tribase* algorithm for the graph of Fig. 2(a).

Indeed, *NLCOMS* is driven by Shneiderman's visual information-Seeking Mantra "Overview First, Zoom and Filter, Then Details-on Demand" [52]. A global view gives to the expert user a sight of the networks structure, next a zoom or a filter on the network (also on the communities) is provided. Details appear on user selection or hovering and effortless interactions (i.e., right and left mouse click) are utilized avoiding pointless extra cognitive load. The interactive visualization steps on which *NLCOMS* relies are presented in Fig. 12.

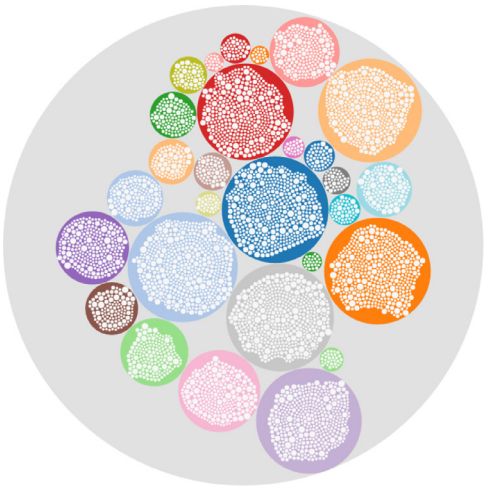


Fig. 11. Circle packing representation using *NLCOMS* for the graph in Fig. 2(a) where the circle size is proportional to the node graph degrees.

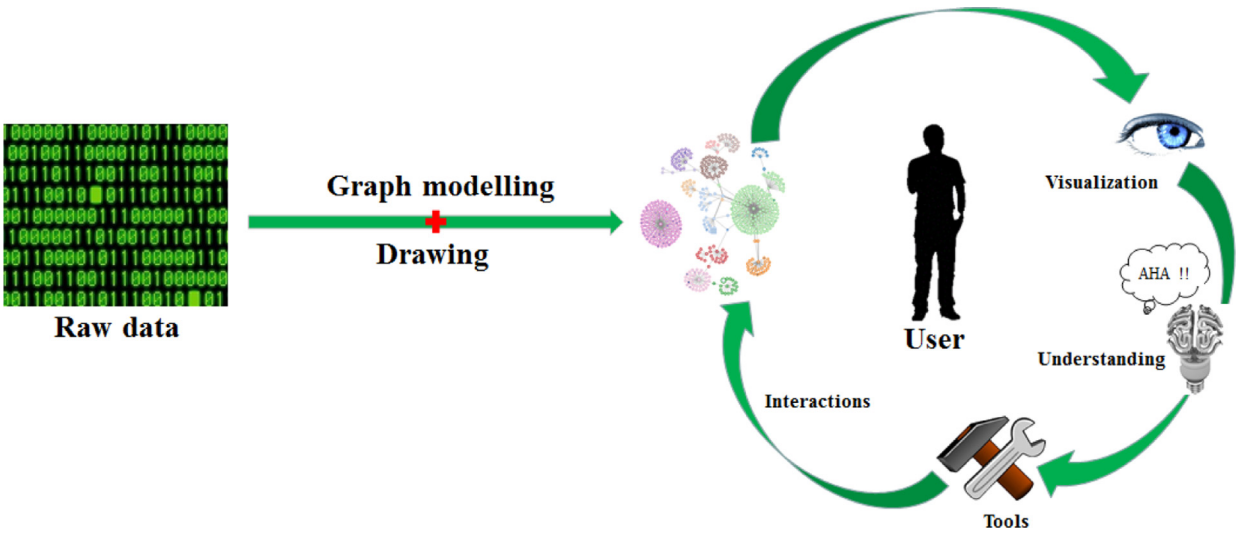


Fig. 12. Interactive visualization steps of *NLCOMS*.

Table 4
ANR-Info-RSN datasets' characteristics.

Datasets	*v*	*m*	*T*	*Thematic*	*Medias source*
Dataset 1	1167	7627	9626	*all*	*all*
Dataset 2	644	6334	16503	*all*	*equipe.fr*
Dataset 3	10000	25185	29385	*all*	*all*

Moreover, *NLCOMS* proposes multiple and synchronized panels, for example, Node-link, communities, Node-link attributes, Community detection, Data filters, Layout parameters, Graph characteristics and Stats views, in which the user can, respectively, visualize the network, visualize the detected communities, visually encode the data attributes, set the *Tribase* algorithm parameters, filter the raw data, set the layout algorithm parameters, survey the related graph characteristics and have access to some statistics on a selected community.

Besides, we also use the visual variables (see Bertin's work [3]), like the node size, the node shape, the edge thickness and the lightness of the inner node colour. Additionally, the node borderline is coloured accordingly to the community nodes' membership. Therefore, one color is associated to each community. Complementary and interactive bar charts help the expert user to build his/her own ideas about communities' characteristics. Finally, arcs' charts are used for each community member to give further information about communities' characteristics.

5. Application to Twitter's networks

In order to assess the applicability of the proposed approach on real-world data, we consider the community detection in social networks, especially in Twitter. Twitter is a widely used social network allowing the registered users to publish messages on the Internet, called tweets. It also provides online networking by user following and friendship. As a consequence of large use, Twitter attracts increasing interest from the scientific and professional fields, like sociologist, online reporters and communication scientists. In this paper, we consider the data of the French ANR Info-RSN project with the partnership of information and communication scientists of the CREM laboratory of Université de Lorraine (France). In this project 17 millions of tweets are collected, with the objective of providing an explanation on how the information is shared through Twitter and also to detect the underlying communities within Twitter networks. To fulfil this need, we have carried out a pre-processing step on the Info-RSN database in order to build a graph model. In this pre-processing, the couples tweet-retweet are extracted and the expressions with the symbol '@' are collected. Then, we obtain a graph model where nodes are the persons who tweets and the edges represent either re-tweet (person B re-tweet the tweet of person A) or mention (person A is cited by person B in its tweet). *NLCOMS* allows the user to decide which relationship type is represented by the edges. Indeed, only one relationship can be encoded by the edges. Thus, the edge weight represents the number of re-tweets or mentions between two persons. Thereby, the communities induced by re-tweet edges and those induced by mention edges are complementary to investigate the sharing and propagation phenomena. Furthermore, we perform a classification of the tweets by their media source (i.e., from which media they are tweeted) and by the addressed thematic (i.e., which theme the tweet is talking about).

To meet the above objectives, we have decided to offer a visual support to answer to the following set of tasks:

1. What is the community structure of the network?
2. What are the main thematics and medias addressed within a community?
3. How often a member tweet about a thematic or from a given media?
4. Which kind of Twitter's users are within a community?

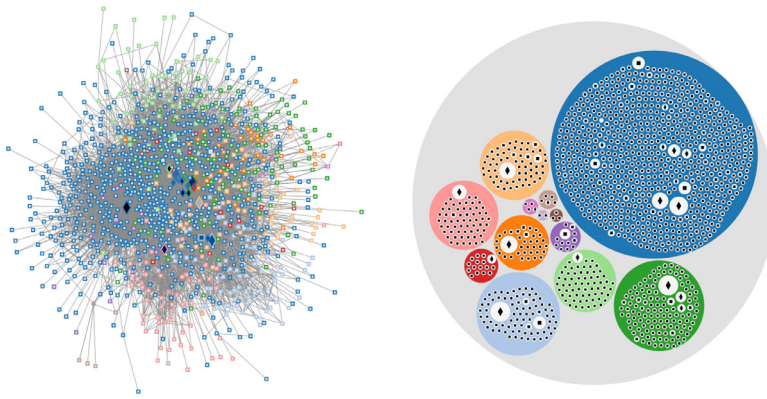
Each of these tasks is respectively handled with the following visual representations:

1. Jointly use node-link diagram and circle packing representations.
2. Provide bar charts for thematic and media proportion within a selected community.
3. Use arcs charts for thematic and media proportion for each community member.
4. Represent each community member type by a symbol - circles for ordinary users, squares for users having reporter profiles, diamonds for users categorized as official reporters of a media, triangles for bots and cross for popular Twitter users (using the number of followers as a metric).

Representative samples from the ANR-Info-RSN project database are analyzed and visualized. Table 4 illustrates the characteristics of these samples.

5.1. Dataset 1

In Dataset 1, we investigate how Twitter's users which are considered as reporters interact. To this end, the communities detected by *Tribase* algorithm and the visualizations proposed by *NLCOMS* are illustrated in Fig. 13.



(a) Node-link representation (b) Circle packing representation

Fig. 13. Dataset 1 of the ANR-Info-RSN samples.

In the latter, the network is dense with communities having star-like shape (Fig. 13(a)). Moreover, the aim of the circle parking representation is to easily distinguish between the multiple communities (Fig. 13(b)), this purpose can sometimes be hard to achieve when the node-link representation suffers from visual clutter resulting from edge crossing. From Fig. 13(b), the main actors are clearly identified where the node size depicts the number of followers of a community member. From there, it becomes straightforward to see that most of the communities contain an official reporter (i.e., diamonds) which are considered as main actors. One would say that the main actors emit the original tweets and the other members propagate them.

Furthermore, in Fig. 14(a), if the expert user selects the light green community, bar charts and arc charts appear to represent respectively thematic (media) proportions within community and thematic (media) proportions for each member. We divided the circles representing community members in two halfs, each of those is devoted to display the arc charts (i.e., right side for media arcs and left side for thematic arcs). Thus, from Fig. 14(c) and (b), we observe that 94% of tweets are tweeted from the media "lesechos.fr" and 25% talks about "economy". Additionally, most of the community members tweets are only from the media "lesechos.fr" (i.e., half blue arcs). For the expert user, the fact of knowing which common factor makes members grouping in one community, in this case the media "lesechos.fr" strengthen the confidence in the automatic community detection process provided by *Tribase* algorithm. Furthermore, the combination of these representations enhances the understanding of community structure by network exploration while grasping gradually and interactively the underlying information.

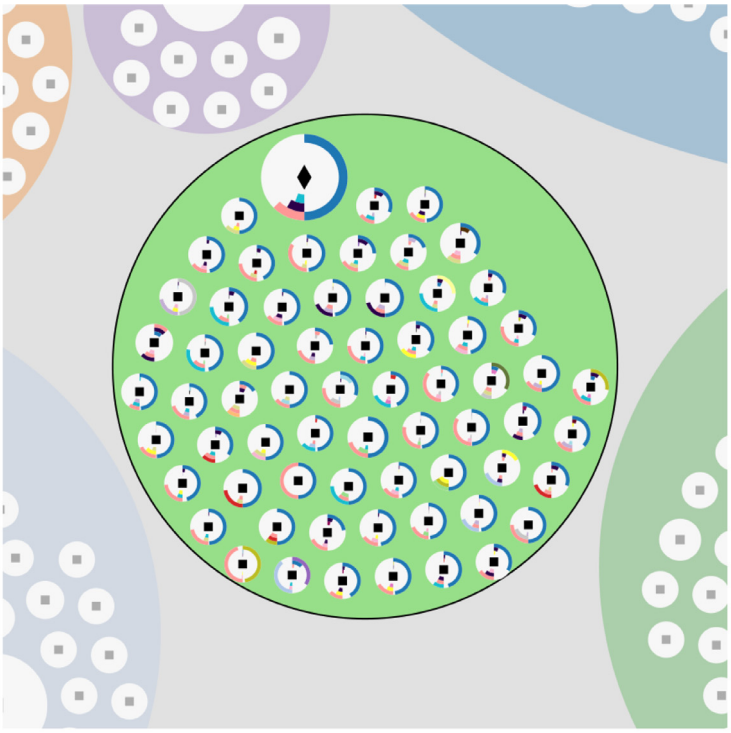
5.2. Dataset 2

Fig. 15 presents the detected communities and the related visualizations for Dataset 2. Unlike to Dataset 1, Dataset 2 does not impose any kind of Twitter's user profiles. Fig. 15(a) shows the induced network by Dataset 2. Furthermore, *NLCOMS* allows us to visually compare the most active members in terms of tweets' publication. For that, the size of the circles in Fig. 15(b) are proportional to the number of tweets of the respective Twitter's users. With this visual encoding the expert user can observe that the most active members are bots represented by triangles in Fig. 16(a).

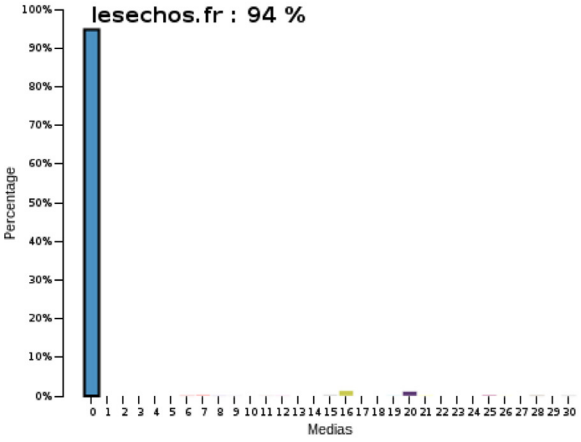
Moreover, relying on the bar charts (Fig. 16(c) and (b)) and the arc charts relative to the light blue community, the expert user can infer that these bots are targeting the media "lequipe.fr". Additionally, the expert user can observe that the majority of the bots address the same thematics in their tweets i.e., "Sport". The second observation consolidates the first one taking into account the fact that "lequipe.fr" is well-known as sport media.

5.3. Dataset 3

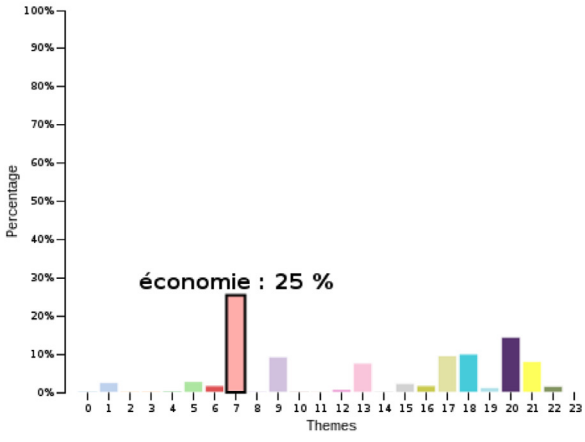
In this dataset, we want to investigate the Twitter users features that permit community organization. To this end, we select as features the number of tweets, the number of retweets, the number of followers, the number of thematics and the number of media for each Twitter user. This first step allows us to apply a K-means algorithm on the whole users (about 570,000 users) using the above features. As a result of the K-means algorithm, 10 clusters are returned ($K = 10$ provides 80% of the inertia value). One among these clusters seems to be interesting because the related features intervals does not overlap with the other clusters features intervals. We consider a sample from this cluster (the most 10,000 active members) and the detected communities are presented in Fig. 17. It shows that important communities emerge with leaders (Fig. 17(b)), (i.e., official media user, like "La Croix"). Indeed, the proposed visualization enables community exploration. Further details on the community members profiles are obtained by hovering and interaction.



(a) Zoom on the selected community of Figure 13

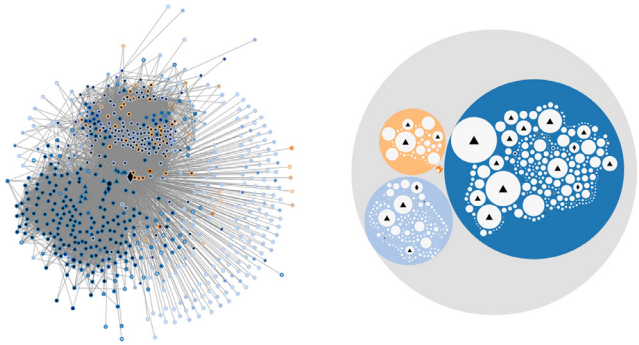


(b) Media bar chart



(c) Thematic bar chart

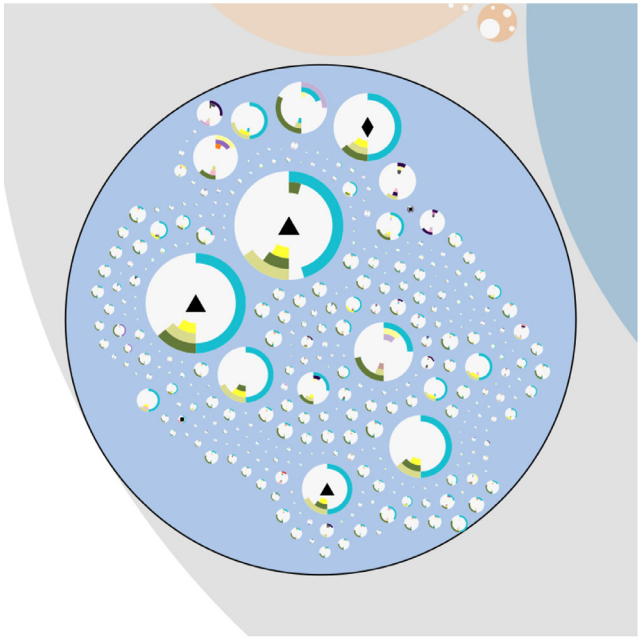
Fig. 14. Community selection and details of Dataset 1 of the ANR-Info-RSN samples.



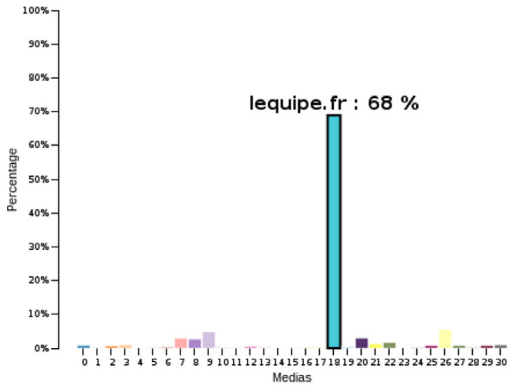
(a) Node-link representation

(b) Circle packing representation

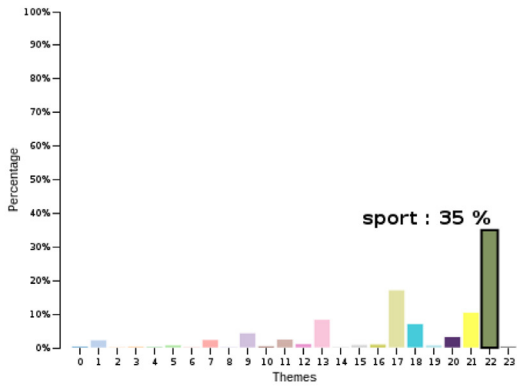
Fig. 15. Dataset 2 of the ANR-Info-RSN samples.



(a) Zoom on the selected community of Figure 15

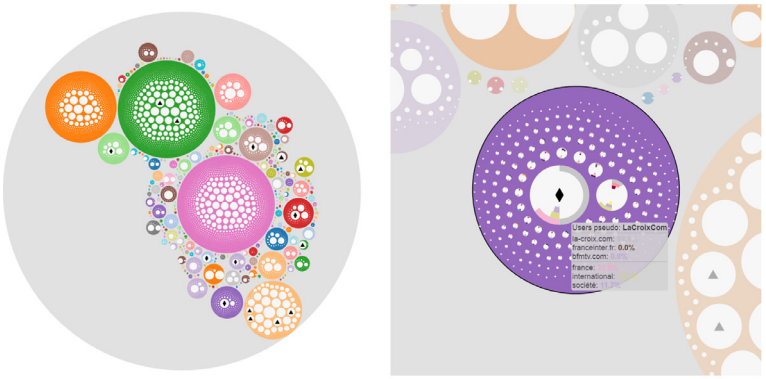


(b) Media bar chart



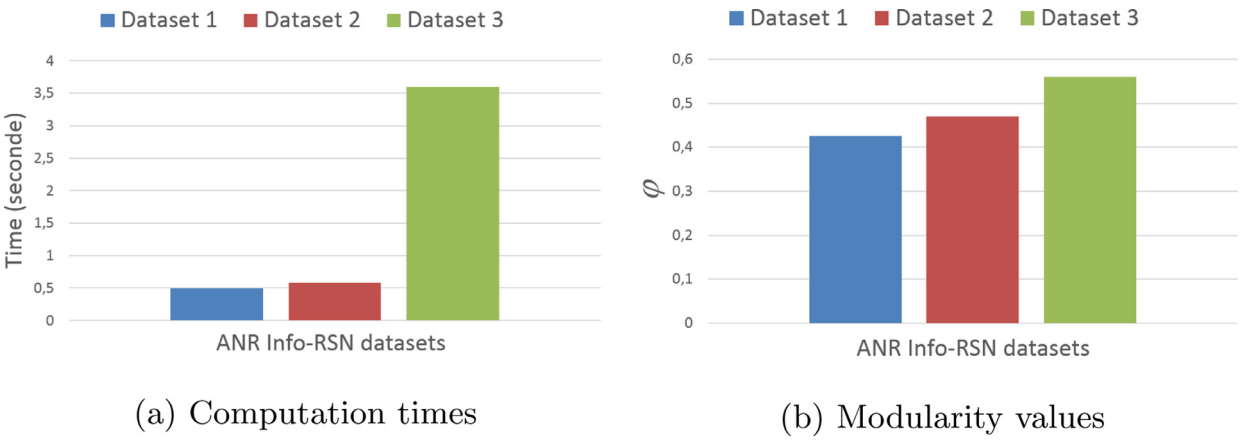
(c) Thematic bar chart

Fig. 16. Community selection and details of Dataset 2 of the ANR-Info-RSN samples.



(a) Circle packing representation (b) Community selection and profile details

Fig. 17. Dataset 3 of the ANR-Info-RSN samples.



(a) Computation times (b) Modularity values

Fig. 18. Results of the ANR-Info-RSN samples.

Finally, Fig. 18 presents the *Tribase* algorithm running times for the ANR Info-RSN datasets and the relative modularity values.

6. Conclusion

In this work, we propose an approach for social media analysis, especially for Twitter's network. Our approach relies on two complementary steps: (i) a community identification based on a new community detection algorithm called *Tribase*, and (ii) an interactive community visualization, which provides gradual knowledge acquisition using our visualization tool, called *NLCOMS*. *Tribase* algorithm uses the triangles obtained by a feasible solution of the weighted MTP problem as a starting point for the community detection. Then, communities are compared allowing dominant communities to gain in size. We carried out a comparison study of *Tribase* algorithm with algorithms of the literature on the LFR benchmark. The results show that the proposed algorithm is competitive and it was able to identify the underlying communities where a community structure exists, even only based on community weights. Furthermore, the second step of our approach uses an interactive visualization approach relying on *NLCOMS* to reveal the structure and the underlying semantic properties within communities. Finally, the proposed approach is assessed on real-world data of the Info-RSN ANR project, which deals with Twitter data. The results show that our approach allows the expert users to visually reveal the community structure and the related characteristics.

As perspectives, it is worthy to consider the dynamic context where the aim is not only to distinguish the interconnected communities at each time-point but also to devise an algorithm which takes advantage from the previous time-points [39], and the related visualization, which permits to depict the evolution of communities structure [47]. Additionally, we plan to address community detection and visualization in multi-layered graphs [5]. In these context, several types of relationship exist, each of them is depicted by a layer of the layered graph. For example, the retweet, the mention and the follow relationships can be investigated simultaneously.

Acknowledgment

This research has been supported by the Agence Nationale de la Recherche (ANR, France) during the Info-RSN Project (ANR-13-SOIN-0008). We would like to thank Prof. Arnaud Mercier and Prof. Nathalie Pignard-Cheynel for their feedbacks regarding the usability of *NLCOMS* as expert users. Finally, we would also like to thank the anonymous reviewers for the constructive comments and for their suggestions, which undoubtedly improve this publication.

References

[1] Y. Abdelsadek, F. Herrmann, I. Kacem, B. Otjacques, Branch-and-bound algorithm for the maximum triangle packing problem, Comput. Ind. Eng. 81 (C) (2015) 147–157, doi:10.1016/j.cie.2014.12.006.
[2] M. Bastian, S. Heymann, M. Jacomy, Gephi: An open source software for exploring and manipulating networks, 2009 http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154.
[3] J. Bertin, Sémiologie Graphique : Les Diagrammes, Les Réseaux, Les Cartes, Mouton, Paris, 1967.
[4] V. Blondel, J. Guillaume, R. Lambiotte, E. Mech, Fast unfolding of communities in large networks, J. Stat. Mech (2008) P10008.
[5] P. Bródka, T. Filipowski, P. Kazienko, An Introduction to Community Detection in Multi-layered Social Network, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 185–190 10.1007/978-3-642-35879-1_23.
[6] J. Chen, J. Fagnan, R. Goebel, R. Rabbany, F. Sangi, M. Takaffoli, E. Verbeek, O. Zaiane, Meerkat: community mining with dynamic social networks, in: 2010 IEEE International Conference on Data Mining Workshops, IEEE, 2010, pp. 1377–1380.
[7] Z.-Z. Chen, R. Tanahashi, L. Wang, An improved randomized approximation algorithm for maximum triangle packing, Discrete Appl. Math. 157 (7) (2009) 1640–1646, doi:10.1016/j.dam.2008.11.009.
[8] A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (2004) 066111. http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0408187.
[9] G. Csardi, T. Nepusz, The igraph software package for complex network research, Int. J. Complex Systems (2006) 1695. http://igraph.org.
[10] J. Ellson, E.R. Gansner, E. Koutsofios, S.C. North, G. Woodhull, Graphviz and dynagraph static and dynamic graph drawing tools, in: Graph Drawing Software, Springer-Verlag, 2003, pp. 127–148.
[11] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (2010) 75–174, doi:10.1016/j.physrep.2009.11.002. http://www.sciencedirect.com/science/article/pii/S0370157309002841.
[12] S. Fortunato, M. Barthelemy, Resolution limit in community detection, Proc. Nat. Acad.f Sci. 104 (1) (2007) 36–41.
[13] L. Freeman, The Development of Social Network Analysis: A Study in the Sociology of Science, BookSurge Publishing, 2004.
[14] A. Friggeri, G. Chelius, E. Fleury, Triangles to capture social cohesion, CoRR abs/1107.3231 (2011). http://dblp.uni-trier.de/db/journals/corr/corr1107.html#abs-1107-3231.
[15] M. Ghoniem, J.-D. Fekete, P. Castagliola, A comparison of the readability of graphs using node-link and matrix-based representations., in: M.O. Ward, T. Munzner (Eds.), INFOVIS, IEEE Computer Society, 2004, pp. 17–24. http://dblp.uni-trier.de/db/conf/infovis/infovis2004.html#GhoniemFC04.
[16] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, PNAS 99 (12) (2002) 7821–7826.
[17] A. Gruzd, Netlytic: software for automated text and social network analysis, 2016 http://Netlytic.org.
[18] N. Henry, J.-D. Fekete, Matrixexplorer: a dual-representation system to explore social networks., IEEE Trans. Vis. Comput. Graph. 12 (5) (2006) 677–684. http://dblp.uni-trier.de/db/journals/tvcg/tvcg12.html#HenryF06.
[19] N. Henry, J.-D. Fekete, M.J. McGuffin, Nodetrix: a hybrid visualization of social networks, IEEE Trans. Vis. Comput. Graph 13 (6) (2007) 1302–1309, doi:10.1109/TVCG.2007.70582.
[20] I. Herman, G. Melancon, M. Marshall, Graph visualization and navigation in information visualization: a survey, IEEE Trans. Vis. Comput. Graph 6 (1) (2000) 24–43.
[21] M. Juenger, P. Mutzel (Eds.), Graph Drawing Software, Springer, 2004. http://dblp.uni-trier.de/db/books/collections/Juenger2004.html.
[22] C. Klymko, D. Gleich, T.G. Kolda, Using triangles to improve community detection in directed networks., CoRR abs/1404.5874 (2014). http://dblp.uni-trier.de/db/journals/corr/corr1404.html#KlymkoGK14.
[23] S.G. Kobourov, Force-directed Drawing Algorithms., in: R. Tamassia (Ed.), Handbook of Graph Drawing and Visualization, Chapman and Hall/CRC, 2013, pp. 383–408. http://dblp.uni-trier.de/db/reference/crc/gd2013.html#Kobourov13.
[24] A. Lancichinetti, S. Fortunato, Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities, Phys. Rev. E 80 (1) (2009a) 016118, doi:10.1103/PhysRevE.80.016118. http://pre.aps.org/abstract/PRE/v80/i1/e016118.
[25] A. Lancichinetti, S. Fortunato, Community detection algorithms: a comparative analysis, Phys. Rev. E 80 (2009b) 056117, doi:10.1103/PhysRevE.80.056117.
[26] A. Lancichinetti, S. Fortunato, Limits of modularity maximization in community detection, CoRR abs/1107.1155 (2011). http://dblp.uni-trier.de/db/journals/corr/corr1107.html#abs-1107-1155.
[27] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (4) (2008) 046110, doi:10.1103/PhysRevE.78.046110.
[28] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J.J. van Wijk, J.-D. Fekete, D.W. Fellner, Visual analysis of large graphs: state-of-the-art and future research challenges., Comput. Graph. Forum 30 (6) (2011) 1719–1749. http://dblp.uni-trier.de/db/journals/cgf/cgf30.html#LandesbergerKSKWFF11.
[29] M. Latapy, Theory and practice of triangle problems in very large (sparse (power-law)) graphs, CoRR abs/cs/0609116 (2006). http://dblp.uni-trier.de/db/journals/corr/corr0609.html#abs-cs-0609116.
[30] B. Lee, C. Plaisant, C.S. Parr, J.-D. Fekete, N. Henry, Task taxonomy for graph visualization., in: E. Bertini, C. Plaisant, G. Santucci (Eds.), BELIV, ACM Press, 2006, pp. 1–5. http://dblp.uni-trier.de/db/conf/avi/beliv2006.html#LeePPFH06.
[31] J. Leskovec, Stanford network analysis project, 2011 http://snap.stanford.edu/index.html.
[32] J. Leskovec, K.J. Lang, M.W. Mahoney, Empirical comparison of algorithms for network community detection, CoRR abs/1004.3539 (2010). http://dblp.uni-trier.de/db/journals/corr/corr1004.html#abs-1004-3539.
[33] J. Leskovec, A. Rajaraman, J.D. Ullman, Mining of Massive Datasets, 2nd ed., Cambridge University Press, 2014. http://www.mmds.org/.
[34] J. Lu, X. Yu, G. Chen, W. Yu, Complex Systems and Networks: Dynamics, Controls and Applications, Understanding Complex Systems, Springer Berlin Heidelberg, 2015.
[35] NetworkX developer team, Networkx, 2014 https://networkx.github.io/.
[36] M. Newman, Modularity and community structure in networks, Proc. Nat. Acad. Sci. 103 (23) (2006) 8577–8582.
[37] M.E.J. Newman, Analysis of weighted networks, Phys. Rev. E 70 (2004).
[38] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (026113) (2004).
[39] N.P. Nguyen, T.N. Dinh, Y. Xuan, M.T. Thai, Adaptive algorithms for detecting community structure in dynamic social networks., in: INFOCOM, IEEE, 2011, pp. 2282–2290. http://dblp.uni-trier.de/db/conf/infocom/infocom2011.html#NguyenDXT11.
[40] T. Opsahl, P. Panzarasa, Clustering in weighted networks, Soc. Netw. 31 (2) (2009) 155–163.
[41] B. Otjacques, F. Feltz, Representation of graphs on a matrix layout., in: IV, IEEE Computer Society, 2005, pp. 339–344. http://dblp.uni-trier.de/db/conf/iv/iv2005.html#OtjacquesF05.

[42] P. Pons, M. Latapy, Computing Communities in Large Networks Using Random Walks, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 284–293, 10.1007/11569596_31.
[43] H.C. Purchase, Metrics for graph drawing aesthetics, J. Visual Lang. Comp. 13 (5) (2002) 501–516, doi:10.1006/S1045 -926X(02)00016-2.
[44] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, D. Parisi, Defining and identifying communities in networks, PNAS 101 (9) (2004) 2658–2663.
[45] U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 76 (3) (2007).
[46] W. Rand, Objective criteria for the evaluation of clustering methods, J. Am. Stat. Assoc. 66 (336) (1971) 846–850.
[47] K. Reda, C. Tantipathananandh, A. Johnson, J. Leigh, T. Berger-Wolf, Visualizing the evolution of community structures in dynamic social networks, in: Proceedings of the 13th Eurographics / IEEE - VGTC Conference on Visualization, in: EuroVis'11, The Eurographs Association & John Wiley & Sons, Ltd., Chichester, UK, 2011, pp. 1061–1070, doi:10.1111/j.1467-8659.2011.01955.x.
[48] J. Reichardt, S. Bornholdt, Statistical mechanics of community detection, Arxiv preprint cond-mat/0603718(2006).
[49] P. Ronhovde, Z. Nussinov, Multiresolution community detection for megascale networks by information-based replica correlations, Phys. Rev. E 80 (2009) 016109, doi:10.1103/PhysRevE.80.016109.
[50] M. Rosvall, D. Axelsson, T.C. Bergstrom, The map equation, Eur. Phys. J. Spec. Topics 178 (1) (2009) 13–23, doi:10.1140/epjst/e2010-01179-1.
[51] N. Schlitter, T. Falkowski, J. Lssig, Dengraph-ho: density-based hierarchical community detection for explorative visual network analysis, in: Springer (Ed.), Research and Development in Intelligent Systems XXVIII Incorporating Applications and Innovations in Intelligent Systems XIX Proceedings of AI-2011, the Thirty-first SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Springer, London, 2011, pp. 283–296.
[52] B. Shneiderman, The eyes have it: a task by data type taxonomy for information visualizations, IEEE Vis. Lang.UMCP-CSD CS-TR-3665) (1996) 336–343.
[53] M.A. Smith, B. Shneiderman, N. Milic-Frayling, E.M. Rodrigues, V. Barash, C. Dunne, T. Capone, A. Perer, E. Gleave, Analyzing (social media) networks with nodexl., in: J.M. Carroll (Ed.), C&T, ACM, 2009, pp. 255–264. http://dblp.uni-trier.de/db/conf/candt/candt2009.html#SmithSMRBDCPG09.
[54] C. Staudt, A. Sazonovs, H. Meyerhenke, Networkit: an interactive tool suite for high-performance network analysis, CoRR abs/1403.3005 (2014). http://arxiv.org/abs/1403.3005.
[55] R. Tamassia, Handbook of graph drawing and visualization, 2014 http://www.amazon.de/Handbook-Visualization-Discrete-Mathematics-Applications/dp/1584884126/ref=sr_1_1?ie=UTF8&qid=1445359603&sr=8-1&keywords=graph+drawing+handbook.
[56] S. Wasserman, K. Faust, Social Network Analysis: Methods and applications, 506, Cambridge University Press, 1994.
[57] J. Yang, J.J. McAuley, J. Leskovec, Community detection in networks with node attributes., CoRR abs/1401.7267 (2014). http://dblp.uni-trier.de/db/journals/corr/corr1401.html#YangML14.