Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 112 (2017) 407–416

Procedia Computer Science

www.elsevier.com/locate/procedia

21st International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2017, 6-8 September 2017, Marseille, France

An evidential influence-based label propagation algorithm for distributed community detection in social networks

Mehdi Azaouzi*, Lotfi Ben Romdhane

MARS[☆] *Research Laboratory LR17ES05, Higher Institute of Computer Science and Telecom (ISITCom), University of Sousse, Tunisia*

Abstract

Community detection in social networks is a computationally challenging task that has attracted many researchers in the last decade. Most of approaches in the literature focus only on modeling structural properties, ignoring the social aspect in the relations between users. Additionally, they detect the communities, one after another, in a serial manner. However, the size of actual real-world social networks grows exponentially which makes such approaches inefficient. For this, several models tend to parallelize the community detection task. Unfortunately, social networks data often exhibits a high degree of dependency which renders the parallelization task more difficult. To overcome this difficulty, amongst the proposed distributed community detection methods, the label propagation algorithm (LPA) emerges as an effective detection method due to its time efficiency. Despite this advantage in computational time, the performance of LPA is affected by randomness in the algorithm. Indeed, LPA suffers from poor stability and occurrence of monster community. This paper introduces a new LPA algorithm for distributed community detection based on *evidence theory* which has shown a high efficiency in handling information. In our model, we will use the belief functions in the update of labels as well as in their propagation in order to improve the quality of the solutions computed by the standard LPA. The mass assignments and the plausibility, in our model, are computed based on *the social influence* for detecting the domain label of each node. Experimentation of our model on real-world and artificial LFR networks shows its efficiency compared to the state of the art algorithms.

Keywords: Social networks; Distributed community detection; Theory of belief functions; Influence measure; Label propagation

1. Introduction and Related Work

With the overwhelming explosion of social networks, a large number of techniques to understand the structures of networks are developed. One of the most used is community detection that is based on the hypothesis that the set of individuals in a community should have strong interactions between them and little interactions with the outside[1].

☆ Modeling of Automated Reasoning Systems
* Corresponding author. Tel.: +216-98-100-843.
E-mail address: mehdi.azaouzi@gmail.com

1877-0509 © 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of KES International
10.1016/j.procs.2017.08.045

Most of proposed methods in the literature detect communities in a serial manner as follows; i.e.; they detect only one community in each iteration following a given strategy such as optimization methods[1,2,4,5,6]; divisive clustering algorithms[3]; consensus strategies[7] and the relational concept analysis[8]. However, this *centralized* and global control is a significant bottleneck when used in very large scale networks because of high time and space complexities. In order to overcome this difficulty, many efforts have been taken to find various algorithms to efficiently and effectively address the community mining in large-scale networks. Recently, several methods based on *distributed* detection are proposed. These approaches could be roughly classified into two categories. The first and most studied one aims to distributed detection framework with shared-memory parallelism or "distributed computing"[9,10]. These parallelization strategies divide the social network into non-overlapping subnetworks that are treated in parallel. The main obstacle in this first category lies in the fact that the community mining algorithms often exhibit a high degree of data dependency. To address this issue, the second category – which is the focus of this paper – is presented. The rationale behind the models in this category is to detect several communities in a distributed manner using an *undistribtued* social graph. In this second category, a promising algorithm, called the *label propagation algorithm* (*LPA*), is proposed in[11]. The basic idea of LPA could summarized as follows. Initially, each node is assigned a unique label. Thereafter, labels are diffused through the network and each node decides of its label using a decision rule (generally the majority label). In addition to its simplicity and its distributed nature, LPA is able to detect communities in *linear time*. Thus, it can process extremely large networks. For this, several LPA-based algorithms were proposed in the literature– see for instance[12,13,14]. Unfortunately, the performance of LPA is affected by randomness in the algorithm. Indeed, LPA suffers from poor stability and occurrence of monster community. This instability is mainly caused by the order in which nodes are updated. In fact, in LPA, we have observed that the probabilistic system of updating label with the most frequent label among neighbors leads to the uncertain labeling. Explicitly, at step t, in the case of most frequent label among neighbors, one of them is chosen randomly. This probabilistic choice at t becomes certain in the next step $t + 1$; and thereby, in the rest of the diffusion process. Unfortunately, this leads to weak robustness.

In this paper, we propose a new LPA-based model for distributed community detection in mega-scale social networks that addresses each of the issues raised above. The central idea of our model, named Evidential Influence LPA (EILPA), is the use of influential nodes as "leaders" in community detection. In fact, the information is much easier to be propagated from influential to common nodes, and the influential ones can attract others to their communities. EILPA is run mainly in two phases. In a first phase, we extract a set of most influential nodes that could form the centers of communities. Then, each center will be assigned with unique label and the remained nodes are not labeled. In a second phase, we will apply LPA, in each part of the network, to propagate the label center based on the influence measure. EILPA adopts the framework of belief functions based on the K-nearest neighbor rule. Hence, in EILPA, an influence piece of evidence quantities the impact (influence) of each center on the current node; and is used to initialize nodes labels. Moreover, at stage t, EILPA avoids the randomness of LPA, by adopting a new measure of clustering coefficient based on the neighborhood labels at stage $(t - 1)$. The rest of this paper is structured as follows. Section 2 introduces background material about the standard LPA and the theory of belief functions. Section 3 introduces our model EILPA; while Section 4 outlines experimental results. The last section offers concluding remarks and suggests future research directions.

2. Background

2.1. The LPA algorithm

The label propagation algorithm (LPA)[11] identifies network communities by the following procedure. Initially, LPA assign a unique label to every node in the network and arrange them in a random order. Then, LPA updates the label of each node by choosing the most frequent label within its neighbors. If more than one label have the same frequency among neighbors, the algorithm chooses one of them randomly. This process is repeated until each node in the network gets the most frequent label from its neighbors. Consequently, densely connected groups reach a common label quickly. When many such dense (consensus) groups are created throughout the network, they continue to expand outwards until it is impossible to do so. We denote by $G(V, E)$ an undirected network graph modeling the social network at hand, where V is the set of nodes and E is the set of edges; and by $N(v)$ the set of neighbors of node $v \in V$. The update rule of l_v, the label of node v, can be described as follows:

$$l_v = \arg\max \sum_{u \in N(v)} \delta(l_u, l) \tag{1}$$

where l_u is the label for node u and $\delta(l_u, l)$ is the Kronecker delta. If more than one labels satisfies the equation, one of them will be chosen at random. The distributed nature of LPA and its linear time-complexity makes it an efficient algorithm for community detection in mega-scale social networks. Unfortunately the efficiency of LPA is affected by its randomness since it suffers from poor stability and occurrence of monster community.

2.2. Theory of belief functions

In this section, we introduce some basic concepts of the theory of belief functions that were used in the proposed approach. The first ancestor of the evidence theory is proposed by Dempster[15] as the *Upper* and *Lower probabilities*. Thereafter, the mathematical theory of evidence is introduced by Shafer[16], often called *Shafer model*. The main goal of the Dempster-Shafer theory is the reasoning under uncertainty based on the explicit representation and combination of items evidence in order to achieve more precise, reliable and coherent information.

Let us assume that we are interested in the value of some variable ω taking values in a finite domain $\Omega = \{\omega_1, \cdots, \omega_n\}$, called the *frame of discernment*. Uncertain evidence about ω may be represented by the basic belief assignment (BBA), or a mass function, m, representing the agent belief on Ω. It is defined as:

$$\begin{aligned} &2^\Omega \rightarrow [0,1] \\ &A \;\; \mapsto m(A) \end{aligned} \tag{2}$$

where $2^\Omega = \{\emptyset, \{\omega_1\}, \{\omega_2\}, \{\omega_1, \omega_2\}, \cdots, \{\omega_1, \cdots, \omega_n\}\}$ is the power set (set of all subsets) of Ω. $m(A)$ is the mass value assigned to $A \subseteq \Omega$. The mass function m must respect:

$$\sum_{A \subseteq \Omega} m(A) = 1. \tag{3}$$

Each number $m(A)$ is interpreted as the probability that the evidence supports exactly the assertion $\omega \in A$ (and no more specific assertion), i.e., the probability of knowing that $\omega \in A$, and nothing more. In the case where we have $m^\Omega(A) > 0$ is called *focal element* of m^Ω. The difference between a BBA distribution and a probability distribution, is that BBA allows a subset of Ω to be a focal element when we have some doubt about the decision, while the probability theory forces the equiprobability in such a case.
To each normalized mass function m, we may associate *belief* and *plausibility* functions from 2^Ω to [0, 1] defined as follows:

$$Bel(A) = \sum_{\emptyset \neq B \subseteq A} m(B) \tag{4}$$

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B) \tag{5}$$

Each quantity $Bel(A)$ may be interpreted as the probability that the assertion $\omega \in A$ is implied by the evidence, while $Pl(A)$ is the probability that this assertion is not contradicted by the evidence. These two functions are linked by the relation $Pl(A) = 1 - Bel(\overline{A})$, for all $A \subseteq \Omega$. The function $pl : \Omega \rightarrow [0,1]$ that maps each element Ω of Ω to its plausibility $pl(\Omega) = Pl(\{\Omega\})$ is called the *contour function* associated to m.
A key idea in Dempster–Shafer theory is that beliefs are elaborated by aggregating independent items of evidence. Assume that we have two pieces of evidence represented by m_1 and m_2 on the same Ω. If one piece of evidence tells us that $\omega \in A$ and the other source tells us that $\omega \in B$ for some non-disjoint subsets A and B of Ω, and if both sources are reliable, then we know that $\omega \in A \cap B$. Under the independence assumption, the probabilities $m_1(A)$ and $m_2(B)$ should be multiplied. If, however, A and B are disjoint, we can conclude that the interpretations "$\omega \in A$" and "$\omega \in B$"

cannot hold jointly, and the probabilities must be conditioned to eliminate such pairs of interpretations[17]. This line of reasoning leads to the combination rule:

$$m_{1\oplus 2}(A) = \begin{cases} \frac{1}{1-k} \sum\limits_{B\cap C=A} m_1(B)m_2(C) & ; A \subseteq \Omega\backslash\{\emptyset\} \\ 0 & ; A = \emptyset \end{cases} \tag{6}$$

where

$$k = \sum_{B\cap C=\emptyset} m_1(B)m_2(C) \tag{7}$$

is the *degree of conflict* between m_1 and m_2. If $k = 1$, there is a logical contradiction between the two pieces of evidence and they cannot be combined. Dempster's rule is commutative, associative, and it admits the vacuous mass function as neutral element. Computation of the full combined mass function $m_1 \oplus m_2$ may be prohibitive in very large frames of discernment, the corresponding contour function can be computed in time proportional to the size of the frame, using the following property:

$$pl_1 \oplus pl_2 = \frac{pl_1 pl_2}{1-k} \tag{8}$$

where pl_1 and pl_2 are the contour functions of two mass functions m_1 and m_2, and the same symbol $\oplus$ is used for mass functions and contour functions.

After introduction of the basic idea of LPA as well as the basic theory of belief functions, now we are ready to introduce our proposal for distributed community detection in the next section.

3. Proposed Method

The basic idea of of our model EILPA for distributed community detection is to use the "influential nodes" as "leaders" for community detection. This choice is motivated by the fact that information is much easier to be propagated from influential to common nodes; and that the influential ones can attract others to their communities. EILPA is mainly composed of two phases : (i) a seeding phase in which an optimal set of leaders is computed; and (ii) an extension phase in which communities are extended from the leading nodes using a modified LPA. The main idea of this second step is to extend the LPA algorithm by harnessing the influence measure of each node and the effect of the most influential nodes on the computed communities. The extension process is guided by measures of influence on the k-nearest neighbor rule. Hereafter, we will detail each phase of our proposal.

3.1. Finding the core community

The main objective of this first phase is to compute an optimal set of seed nodes which will be the starting point for computing communities in the second phase. This set of seed nodes should *maximize* the influence spread in the social network at hand. Stated otherwise, given a piece of information placed on such nodes, it should spread over the maximal number of nodes of the network. For this, we have proposed a new model called SAIM[19] to deal with the influence maximization problem (IM). The key feature of SAIM is that it does not require the seed size k in IM, but rather computes an optimal one for the social network at hand. This is very important since the majority of existing models require the number of seeds k to be computed as input in addition to the social graph. In such approaches, the optimal k is computed experimentally through a trial-and-error approach. In our proposal SAIM, each node is quantified by an "influence power". Intuitively, this influence power of an individual (or node) is computed from users actions on the published statutes of such individual. The optimal set of influential nodes is then computed using a new concept called "influence BFS-tree"[19]. SAIM is composed of three major steps [1] :

[1] For more details about the theoretical background of SAIM, the interested readers are referred to [19].

1. The computation of influence power (IP) for each node: based on personalized PageRank model, we propose new quantifiable measure based on the interaction behaviors that aims at computing the influence power for each user and in which actions are not treated equally.
2. In second phase, a set of significant nodes is generated as candidate seeds. Intuitively, these significant nodes are those with high local influence.
3. Finally, the "influence zone" of each node is modeled through a concept called influence BFS-tree upon which an optimal set of seeds is computed.

As a result to this first phase; and in which our model SAIM[19] is executed on the input graph; we obtain an optimal set of influential nodes (INF) allowing the maximum spreading of information in the social network. Moreover, each node of the network is labeled with its influence power (IP). Now having this optimal set INF, we will use it to expand in a distributed way, the communities in the network as described subsequently.

3.2. The extension phase

The extension of communities from a set of seed nodes is based on LPA. The rationale behind this choice is that LPA have proved to be efficient in handling large-scale social networks in linear time. However, in order to overcome the drawbacks of LPA due to its randomness, we will develop a spreading process taking into consideration the influence power of each node. Our idea is inspired by[20] where the authors attempted to make use of the K-nearest neighbor method based on the evidence theory (EK-NN) to solve the data clustering problem. The EK-NN method is K-nearest neighbor (K-NN) data classification method based on the Dempster–Shafer theory of belief functions. In this paper, we seek to adopt this method for community detection in social networks. The idea of our model is that the most influential nodes in the social network can be responsible on the label update of other nodes. The power influence of each most influential node is critical to determine the rule of update label and the community mining. Moreover, since a node may be influenced by many other nodes, then it is natural to take into consideration in that influence the distance (in terms of graph shortest paths) between such nodes.

From an abstract point view, the propagation of labels is similar to informatiopn spreading in social networks. For this, the EK-NN rule can be easily applied in community mining. Surprisingly, few methods based on EK-NN rule for community mining in social network are proposed. However, before going into its details of our extension phase, let us introduce the following definitions.

Definition 3.1. *(Direct neighbor). In $G = \langle V, E\rangle$, vertex v is a direct neighbor of vertex u if v and u are connected by an edge. This relationship is represented by the edge $(u, v) \in E$*

Definition 3.2. *(Edge influence). In $G = \langle V, E\rangle$, where each vertex v is labeled with its IP value, we define the influence measure of u to v as follows:*

$$im(u, v) = \frac{IP(u)}{IP(v)} \tag{9}$$

The edge-influence $im(u, v)$ measures how much u is able to influence its direct neighbor v. Hence, if the influence power of node u ($IP(u)$) is greater than the influence of v, then the information is much easier to be propagated from u to v; and vice versa. It is clear from equation (9) that, in general, $im(u, v) \neq im(v, u)$. After having defined the concept of edge-influence, now we will introduce the concept of "influence-path" as follows.

Definition 3.3. *(Influence Path). In $G = \langle V, E\rangle$, the influence path between two nodes u and v of graph G, noted as $Inf(u, v)$, is the sum of edge-influence being in the shortest path (noted $\Pi(u, v)$) which leads u towards v. It is given by:*

$$Inf(u, v) = \sum_{(e_i, e_j)\in\Pi(u,v)} im(e_i, e_j) \tag{10}$$

In the evidence-theoretic K-nearest neighbor classification (EK-NN) rule[20], each neighbor of a sample to be classified is treated as an item of evidence that supports certain hypotheses regarding the class label of this sample. The strength of this evidence decreases with the distance to the test sample. Evidence from the K nearest neighbors is pooled using Dempster's combination rule (Equation 6) to make the final decision.

Following the same basic principle, we consider the influence power of each node the social network as well as the set of most influential nodes INF computed in the first phase of our model. Since, we assume that each influential node is responsible for forming its own community, we denote by $c = \|INF\|$ the number of the communities to be mined. Hence, we are able to classify each node in the network in one of the c communities based on the influence received from its members. In fact, a given node u is at a distance d_j from the community center C_{v_j} and the influence relationship between them is a piece of evidence. The influence relationship is represented by the influence of the path relating them. The influence of a path is measured by the influence of all nodes on that path.

Given these basic ideas and principles, now we will introduce our model for label propagation. Let $\Omega = \{C_1, C_2, \cdots, C_c\}$ be our frame of discernment where C_i models the community formed by the node center v_i. We should remark that in our proposal, we assume that any given node belongs to only one community at a time; i.e., we do not allow overlaps between communities ($C_i \cap C_j = \emptyset \ \forall i, j$). Since network nodes are labeled by their influence power and networks links are labeled by the edge-influence which is asymmetric, then our social graph could be modeled in this phase by a *directed graph* $G = (V, E, W)$ where V is the set of nodes, E is the set of links (i.e. $(u, v) \in E$ is the link having u as a source and v as a destination), and W is the set of influence power of each node; i.e., $w_v \in W$ is the influence power associated to v (measured in Section 3.1).

Based on the influence path measure and the EK-NN rule, suppose that the set of neighbors of node u is N_u, and let $v \in N_u$ be a node influenced by the center C_q. Then the mass function induced by v which supports the assertion that u also belongs to C_q is given by:

$$m_{uv}(\{C_q\}) = \alpha exp\left(-\gamma \frac{1 - Inf(u, v)}{Inf(u, v)}\right) \tag{11}$$

$$m_{uv}(\{\Omega\}) = 1 - \alpha exp\left(-\gamma \frac{1 - Inf(u, v)}{Inf(u, v)}\right) \tag{12}$$

where α and γ are constants. Parameter α is a weight factor can be heuristically set as 0.95 according to[20]. The coefficient γ can be fixed as follows[20]:

$$\gamma = 1/median\left(\left\{(\frac{1 - Inf(u, v)}{Inf(u, v)})^2, u \in \{1, .., n\}, v \in \Gamma_u\right\}\right) \tag{13}$$

Next, we apply Dempster's combination rule (Equation 6) to combine all neighbors' mass functions. Let n the number of elements in N_u, then the credal membership of node u is determined by the fused mass m_u as follows:

$$m_u = m_1 \oplus m_2 \oplus \ldots \oplus m_n \tag{14}$$

The difference between this kind of membership and fuzzy membership is that there is a mass assigned to the ignorant set Ω in BBA m. It is used to describe the probability that the node is an outlier of the graph. The domain label l_u of node u can be defined as follows.

$$l_u = \arg\max_{C_q}\{m(\{C_q\}), C_q \in \Omega\} \tag{15}$$

Since the focal elements of the BBAs here are the singletons and set Ω, Eq 15 is equal to:

$$l_u = \arg\max_{C_q}\{pl(\{C_q\}), C_q \in \Omega\} \tag{16}$$

where pl is the contour function associative with m.

Now, we will introduce our new label updating rule based on the affected labels in the previous step. At this stage, we should remind that among the shortcomings of standard LPA is the randomness of selecting multiple most frequent labels which is at the origin of the instability of the algorithm. Another issue in LPA is that only the labels of direct neighbors are considered, and "neighbors-of-neighbors" labels are ignored although they may contain valuable information. Hence, in our update rule, we prone to have a LPA that emphasizes neighbor points and takes into consideration faraway points. For this, when the number of the most frequent labels is more than one, we seek to check the status of neighbor labels. Naturally, in order to reach stability, it is better to label node v in such case with the neighbors' labels that do not change or rarely change. Stated otherwise, instead of a node being labeled with one of neighbors' labels randomly, it will be labeled with neighbors' labels with great probability that it remains unchangeable soon. For this, we proposed a new concept, named *stable neighborhood*, in which we proposed a new measure named *label clustering coefficient*, in which we will measure the clustering coefficient[18] based on the label vertex of each nodes at step $(t-1)$. Our label clustering coefficient, named *LCC*, is based on the label of the neighbors of a vertex i instead of the connectivity of the neighbors of a vertex i. Let $i \in V$ an vertex in a graph G and $e(i)$ is the set of vertices adjacent to the vertex i and $L_{e(i)}$ the set of label of $e(i)$. The *label clustering coefficient of node i* at step t, noted $LCC(i)$, is defined as the ratio between the number of edges connecting pairs of vertices adjacent to the vertex having the same label at step $(t-1)$ and the total number of possible connections among the neighbors of i. In a formal manner, *LCC* is defined as:

$$\mathcal{L}CC(i) = \frac{\sum\limits_{j,k \in N_i \wedge l_i \equiv l_j \equiv l_k} a(j,k)}{d(i)(d(i)-1)} \tag{17}$$

where N_i is neighbors of node i, $d(i)$ refers to the degree of vertex i and l_i, l_j and l_k are the labels for nodes i, j and k respectively. For example, let us consider the sample network snapshot in Fig. 1 where the central node "?" has six neighbors and each node labeled at step $(t-1)$. Nodes 1, 2 and 3 has the label 'B'; whereas nodes 3, 4, and 5 with the label 'A'. Notice that we have two most frequent labels that are 'A' and 'B' each having a frequency of three. Hence, according to the original LPA algorithm, the central node "?" will be randomly labeled either by 'A' or 'B'. Notice that vertices 4, 5 and 6 have label clustering coefficients of 1, 0.66 and 1; which are higher than the label clustering coefficients of vertices 4, 5 and 6 which are equal to 0, 0 and 0.33, respectively. So, the probability of the node labels of 1, 2 and 3 remain the same and thereby remain stable at step t is very small. On the opposite, the probability of the labels of nodes 4, 5 and 6 being stable at step t is very high. Intuitively, the central node "?" is more likely to be labeled with 'A'.

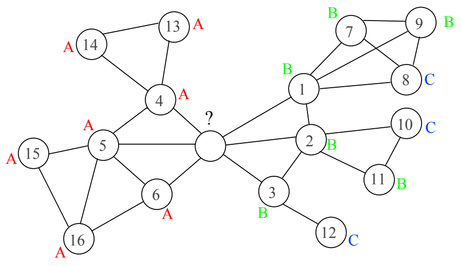

Fig. 1. An example of a label clustering coefficient.

Based on above analysis, we add the label clustering coefficient $\mathcal{L}CC$ of node v's neighbor to LPA. Thus, the label propagation updating rule in Eq. 1 is rewritten into:

$$l_v = \arg\max \sum_{u \in N(v)} \mathcal{L}CC_{max}(l_u)\delta(l_u, l) \tag{18}$$

Our algorithm EILPA for label propagation is outlined in Algorithm 1. Initially, a set of influence paths is computed all the nodes of the graph and the optimal seed set *INF* representing node centers and already computed in the first phase of our model. Thereafter, node centers are assigned unique labels. Subsequently, the rest of the nodes will be initialized with the labels of node centers having the highest plausibility. Thereafter, nodes labels are propagated using equation (18).It is natural that this propagation scheme is more robust than the random update in standard LPA since it takes into account the "neighbors-of-neighbors" information as explained previously. This will be verified experimentally in the next section.

Algorithm 1: EILPA algorithm

```
Data: A graph $G = \langle V, E \rangle$, where each vertex is labeled by its influence power; A set of influential nodes $INF$
Result: Communities $C$ (i.e. node labels)
1  begin
2      Compute the Influence Path between each node and each center in $INF$;
3      Initialize each node center $\in INF$ with a unique label and the rest of the nodes remain empty;
4      Arrange the nodes in $V$ in a random order;
5      Initialize each node with the label center having the highest plausibility using Eq. (16);
6      repeat
7          foreach $v \in V$ do
8              update $v's$ label to that carried by the largest sum in Eq. (18)
9          end
10     until (every node has a label that has the largest sum by Eq. (18)) ;
11 end
```

4. Experimental Results

The main purpose of this section is to analyze the behavior of EILPA experimentally[2]. For this, we conducted extensive simulation on synthetic and real-world networks. We compared EILPA with three algorithms: (1) the original LPA[11]; (2) ELP[21] based on evidential label propagation; and (3) LPAbp[22] based on combining maximum belonging coefficient and edge probability. As evaluation criteria, we considered: (1) the extended normalized mutual information (NMI) which measures the discrepancy between the computed and the exact partitions of the graph; (2) the modularity that measures the fraction of intra-community edges minus its expected value in a null model; and (3) the running time.

4.1. Tests on synthetic networks

For synthetic networks, we adopted the LFR benchmark[23], for which the exact partition of the network is known, to generate random social graphs along with their random social actions. Using LFR generator, we generated four graphs with complex structures. The graph complexity here is quantified by several properties including the graph size (number of nodes n), the node's average degree (k), the node's maximum degree ($maxk$), the minimum communities sizes ($minC$), the maximum communities sizes ($maxC$), and the ratio of external edges of a node with respect to its total degree (μ) (the higher the mixing parameter μ of a network is, the more difficult it is to reveal the community structure). In our simulation, we set $\mu = 0.45$ which simply means that the community boundaries are very hard to compute. The generated artificial social networks are described in Table 1.

Table 1. Characteristics of the complex graphs.

Datasets	*n*	*k*	*maxk*	*minC*	*maxC*
Graph 1	10,000	20	50	10	50
Graph 2	10,000	20	50	10	100
Graph 3	100,000	40	100	50	100
Graph 4	100,000	40	100	100	200

Table 2 and 3 summarizes simulation results for LPA, ELP, LPAbp and our model EILPA. It is clear that ELP and our model EILPA outperform LPABb and LPA for both criteria *NMI* and modularity. Regarding the latter criterion in Table , our model EILPA is better than ELP for the majority of simulated social graphs. This shows clearly that

[2] Our proposed algorithm is implemented in Java and ran on a PC (Intel Core i3 Quad CPU 1.4 GHz with 6.0 GB of memory).

our proposal is able to compute communities whose "boundaries" are clear. The reason for this behavior might be explained by the effectiveness of the clustering coefficient measure in the update rule of EILPA. By comparing the results in Table 2, we can see that our algorithm EILPA has the best partitioning with respect to NMI when we consider graphs with large communities. This means that it computes partitions that are very close to the exact real ones.

Table 2. Normalized Mutual Information (NMI).

Datasets	LPAbp	LPA	ELP	EILPA
NMI(*Graph 1*)	0.621	0.842	**0.879**	0.863
NMI(*Graph 2*)	0.523	0.801	0.812	**0.832**
NMI(*Graph 3*)	0.825	0.926	**0.981**	0.974
NMI(*Graph 4*)	0.794	0.839	0.872	**0.881**

Table 3. Modularity.

Datasets	LPAbp	LPA	ELP	EILPA
Modularity(*Graph 1*)	0.784	0.676	**0.692**	0.671
Modularity(*Graph 2*)	0.648	0.593	0.775	**0.787**
Modularity(*Graph 3*)	0.742	0.741	0.791	**0.794**
Modularity(*Graph 4*)	0.752	0.674	0.812	**0.844**

4.2. Tests on real-world networks

We considered two widely used real datasets embedding the social actions: the public dataset Tencent Weibo [3] and the Higgs Tweet available in SNAP [4]. Tencent Weibo is a sampled snapshot numbered in millions of users provided with rich information and Higgs Tweet is extracted from Twitter between 1st and 7th July 2012 on a specific topic. Due to either the massive size of these bases and the memory inefficiency, we extracted a network of 20000 nodes from each dataset. Table 4 reports the modularity measure; whereas Table 5 reports the number of iterations for each model to converge. It is clear that LPAbp and ELP outperform LPA and EILPA regarding the modularity measure. However EILPA still do better than LPA. This may be explained by the fact that in these networks there are users with few connections but still receive many social actions which makes them "leaders" but without having strong connectivity. In fact, LPAbp and ELP are based on the purely structural measures. For this, these algorithms overcome EILPA. The average number of iterations (over 10 runs) of all algorithms are very similar; however EILPA takes less iterations than the rest of the models– see Table 5.

Table 4. Modularity measure

Datasets	LPAbp	LPA	ELP	EILPA
Modularity(*Tencent Weibo*)	0.821	0.716	0.894	0.766
Modularity(*Higgs Tweet*)	0.854	0.713	0.885	0.782

Table 5. Number of iterations (averaged over 10 runs)

Datasets	LPAbp	LPA	ELP	EILPA
Tencent Weibo	9.2	12.3	9.7	7.8
Higgs Tweet	10.9	11.5	9.8	7.1

[3] Kddcup. http://www.kddcup2012.org/c/kddcup2012-track1

[4] Snap. https://snap.stanford.edu/data/higgs-twitter.html

As a summary of these runs, we can say that our proposal EILPA has a good performance in mining community structure in social networks compared to the state of the art models in the literature based on label propagation.

5. Conclusion

In this paper, we have proposed EILPA, a model for distributed community detection in social networks based on label propagation. EILPA is main composed of two steps. In a first step, we compute an optimal set of "leader"; i.e., a set of influential nodes identified as community centers. In a second step, each community is extended in a distributed way, from each center using a modified label propagation process. In fact, in order to avoid randomness in the standard LPA, we have developed a new label propagation rule based on the "influence power" of each node as well as the evidential K-nearest neighbor rule. Doing it this way, we ensure more stability in the algorithm. Experiments on synthetic and real-world social networks show the effectiveness of our proposal.

References

1. Fortunato S. Community detection in graphs. *Physics Report* 2009. p. 75-174.
2. Santos CP, Nascimento MCV. Growing Neural Gas as a memory mechanism of a heuristic to solve a community detection problem in networks. *In Proc. 20th Int. Conf. Knowl. Based Intell. Inf. Eng. Syst. (KES 2016)*, York, UK; 2016. p. 485-494.
3. Girvan M, Newman MEJ. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America* 2002;**99**(12):7821-7826.
4. Zardi H, Ben Romdhane L. An o(n2) algorithm for detecting communities of unbalanced sizes in large scale social networks. *Knowl Based Syst* 2013;**37**:19-36.
5. Rhouma D, Ben Romdhane L. An efficient algorithm for community mining with overlap in social networks. *Expert Syst Appl* 2014;**41**(9):4309-4321.
6. Ben Romdhane L, Chaabani Y, Zardi H. A robust ant colony optimization-based algorithm for community mining in large scale oriented social graphs. *Expert Syst Appl* 2013;**40**(14):5709-5718.
7. Mathias S, Rosset V, Nascimento MCV. Community Detection by Consensus Genetic-based Algorithm for Directed Networks. *In Proc. 20th Int. Conf. Knowl. Based Intell. Inf. Eng. Syst. (KES 2016)*, York, UK; 2016. p. 90-99.
8. Guesmi S, Trabelsi C, Latiri C. CoMRing: A framework for Community detection based on Multi-Relational querying exploration. *In Proc. 20th Int. Conf. Knowl. Based Intell. Inf. Eng. Syst. (KES 2016)*, York, UK; 2016. p. 627-636.
9. Gregori LLE, Mainardi S. Parallel k-clique community detection on large-scale networks. *IEEE Trans on Parallel and Distributed Systems* 2013;**24**(8):1651-1660.
10. Staudt C, Meyerhenke H. Engineering high performance community detection heuristics for massive graphs. *Proceedings of the International Conference on Parallel Processing*, Washington, DC, USA; 2013. p. 180-189.
11. Raghavan UN, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E* 2007;**76**:036106.
12. Leung IXY, Hui P, Lio P, Crowcroft J. Towards real-time community detection in large networks. *Physical Review E* 2009;**79**:066107.
13. Zhang Q, Qiu Q, Guo W, Guo K, Xiong N. A social community detection algorithm based on parallel grey label propagation. *Computer Networks* 2016;**107**(Part 1):133-143.
14. Lin Z, Zheng X, Xin N, Chen D. Ck-lpa: Efficient community detection algorithm based on label propagation with community kernel. *Phys. A: Stat. Mech. Appl.* 2014;**416**:386-399.
15. Dempster AP. Upper and lower probabilities induced by a multivalued mapping. *Ann Math Stat* 1967;**38**:325-339.
16. Shafer G. A Mathematical Theory of Evidence. *Princeton University Press* 1976.
17. Denoeux T, Kanjanatarakul O, Sriboonchitta S. EK-NNclus: A clustering procedure based on the evidential K-nearest neighbor rule. *Knowl Based Syst* 2015;**88**:57-69.
18. Nascimento, M.C.V. Community detection in networks via a spectral heuristic based on the clustering coefficient. *Discr Appl Math* 2014;**176**:89-99.
19. Azaouzi M, Ben Romdhane L. An Efficient Two-Phase Model for Computing Influential Nodes in Social Networks using Social Actions. *J Comput Sci Technol.* 2017;**32**(4):-. In Press
20. Denœux T. A k-nearest neighbor classification rule based on Dempster–Shafer Theory. *IEEE Transactions on Systems, Man and Cybernetics* 1995;**25**(5):804-813.
21. Zhou K, Martin A, Pan Q, Liu, Z. Evidential Label Propagation Algorithm for Graphs. *Proceedings of the 19th International Conference on Information Fusion*, Heidelberg, Germany; 2016. p. 1316-1323.
22. Zhang X, Li Y, Jiang S, Xie B, Li X, Zhang Q, Lu M. Efficient Community Detection Based on Label Propagation with Belonging Coefficient and Edge Probability. *Proceedings of the 5th National Conference Social Media Processing*, Nanchang, China; 2016. p. 54-72.
23. Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs to test community detection algorithms. *Physical Review E* 2008;**78**(10):046110. `http://sites.google.com/site/santofortunato/inthepress2`