Knowledge-Based Systems 163 (2019) 404–415

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

An incremental method to detect communities in dynamic evolving social networks

Zhongying Zhao [a,b,*], Chao Li [a], Xuejian Zhang [a], Francisco Chiclana [c,d], Enrique Herrera Viedma [d]

[a] College of Computer Science and Engineering, Shandong Province Key Laboratory of Wisdom Mine Information Technology, Shandong University of Science and Technology, Qingdao, China
[b] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[c] Institute of Artificial Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester, UK
[d] Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain

ARTICLE INFO

Article history:
Received 7 February 2018
Received in revised form 21 July 2018
Accepted 5 September 2018
Available online 11 September 2018

Keywords:
Social network mining
Dynamic evolving network
Online interaction
Community detection
Social network analysis

ABSTRACT

Detecting communities in dynamic evolving networks is of great interest. It has received tremendous attention from researchers. One promising solution is to update communities incrementally taking the historical information into consideration. However, most of the existing methods are only suitable for the case of one node adding or one edge adding. Factually, new data are always generated continuously with subgraphs joining simultaneously in dynamic evolving networks. To address the above problem, we present an incremental method to detect communities by handling subgraphs. We first make a comprehensive analysis and propose four types of incremental elements. Then we propose different updating strategies. Finally, we present the algorithms to detect communities incrementally in dynamic evolving networks. The experimental results on real-world data sets indicate that the proposed method is effective and has superior performance compared with several widely used methods.

1. Introduction

The fast development of information technology has accelerated the prevalence of social media, where people are able to share news, pictures, publish opinions and comments anytime anywhere. All this data, together with users' behaviors and social relations, have abundant potential values. As a result, social network analysis has become a hot research area and has received considerable attention [1–4].

Community detection aims to identify cohesive clusters or groups in real-world graphs such as social networks, web graphs and biological networks. It is a problem of considerable practical interest [5–7]. Identifying user communities with similar interests can foster sharing resources; detecting consumer communities from online shopping sites is beneficial to target potential customers; discovering student communities from online education systems can promote the collaborative learning efficiency of students. Thus, it has received a great deal of attention [8,9]. Numerous techniques have been developed for community detection, such as partition based methods, hierarchical clustering methods, density based methods, modularity optimizing methods and game theory based methods [10].

However, most existing work assume social networks to be static. That is, the network studied is often accumulated by aggregating all the nodes and edges appeared in a period of time. Such an example is shown in Fig. 1. In Fig. 1, there are two networks shown in (A) and (B), with different evolving processes. To make it clear, the new joining nodes at the next time slice are marked in yellow and the new edges are marked in orange. Let G^i, $(i = 1, 2, 3, 4, 5 \ldots)$ denote the graph at different time stamps. As shown in Fig. 1(A), two nodes labeled "2" and "4" joined up with two edges at time $t2$. In the next time slice ($t3$), no node joined the network but three edges (marked in orange). At time $t4$, a new node labeled "5" and a new edge joined the network. Finally, the network was updated with two nodes labeled "6" and "7" and three new edges. According to Fig. 1(B), we can see that three nodes (labeled "5", "6" and "7") and the connected 4 edges are new emerging elements at time $t2$. Then the node labeled "2" joined the network at time $t3$. In the next two time slices, only edges joined the network. The above example illustrates that the aggregated method can deliver the same network (e.g., G^5) with different evolving processes.

Therefore, existing methods based on static network easily lead to the losing of important changing status. By extending the

* Correspondence to: 579 Qianwangang Road, Huangdao District, Qingdao, 266590, PR China
E-mail addresses: zzysuin@163.com (Z. Zhao), lichao@sdust.edu.cn (C. Li), xuejian@163.com (X. Zhang), chiclana@dmu.ac.uk (F. Chiclana), viedma@decsai.ugr.es (E.H. Viedma).

https://doi.org/10.1016/j.knosys.2018.09.002
0950-7051/

static network analyzing techniques to detect communities from dynamic evolving social networks, we can broaden the potential applications. It is useful for predicting the development of common interest groups for marketing and business intelligence purposes [11].

In light of the benefit of community detection from dynamic networks, a growing number of research work has been devoted to this topic, such as the methods based on two-steps strategy [12–14], incremental clustering [15–17], evolutionary clustering [18,19], multi-agent perspective [20], and stochastic block model [21]. Among them, one of the most promising research directions is to identify communities incrementally from dynamic evolving networks. The main idea of this kind of work is to continuously update the communities by considering the results in the last time slice. Sun et al. [15] proposed an incremental density-based clustering algorithm IncOrder to detect communities in dynamic networks. It contains two stages. The online stage maintains the traversal sequence of a network and the offline stage extracts communities from the sequence. The method of iLCD [16] updates the existing community by adding a new node to it if the number of second neighbors is greater than expected values. A limitation is that it cannot add two new nodes and a link in-between them at the same time. Actually, it is common that several subgraphs (incremental elements) appear almost simultaneously, as new data are always generated continuously in dynamic evolving networks.

In view of the mentioned limitations of existing methods, we characterize the community detection in dynamic evolving networks with tools that enable to support the following applications. The method should be capable of handling subgraphs (including nodes and edges) addition. Moreover, the communities should be identified by taking the historical influence into account.

To answer the above questions, in this paper, we investigate: (1) how to model and analyze the incremental elements comprehensively, and (2) how to update communities incrementally taking into account historical information with no human intervention. In an attempt to address the above problems, we propose a new incremental method to identify communities in dynamic evolving social networks. Our research contributions are summarized as follows.

- provide a comprehensive analysis on the incremental elements and propose four different types;
- present the incremental updating strategies for the different types of incremental elements;
- propose an incremental algorithm to detect communities in dynamic evolving social network;
- evaluate the proposed algorithm extensively with real-world social networking data;

The remainder of the paper is organized as follows. We briefly review some related work in Section 2. The notations and problem statement are presented in Section 3. Section 4 describes our method with its framework, incremental elements analysis, updating strategies, and algorithm. Experimental results and evaluation are discussed in Section 5. Finally, we conclude the whole paper in Section 6.

2. Related works

Recently, a growing body of research work is being devoted to detect communities in dynamic temporal networks, which can be classified into three main types.

The first type is the well-known two-stage method. The main idea of this kind of work is to detect communities in each snapshot of a network stream, and then analyze community evolution in adjacent time slices by employing similarity metrics [21]. Some representative methods are found in [12,22,13], although they do not make use of the community information of previous epochs to infer hidden communities of the current epoch. The second type is the so called evolutionary clustering method, which produces local clusters for each time step by introducing temporal smoothness [18,19]. Wang et al. [23] proposed a model named Dynamic Bayesian Nonnegative Matrix Factorization (DBNMF), for automatic detection of overlapping communities in temporal networks. In the DBNMF model, the community structure at snapshot $t+1$ is influenced by the community structure of the snapshot t and independent of the previous snapshot networks. The determination of the number of communities is based on the automatic relevance determination. The overlapping community structure is obtained based on the probabilistic group membership of nodes based on the nonnegative matrix factorization (NMF). The last type of approaches is called incremental method, which continuously update the communities by translating the evolving network as stream data. Sun et al. [15] propose an incremental density-based clustering algorithm IncOrder to detect communities in dynamic networks. It contains two stages. The online stage maintains the traversal sequence of a network and the offline stage extracts communities from the sequence. The method of iLCD [16] updates the existing community by adding a new node to it if the number of second neighbors is greater than expected values. A limitation is that it cannot add two new nodes and a link in-between them at the same time. Actually, it is common that several subgraphs (incremental elements) appear almost simultaneously, as new data are always generated continuously in dynamic evolving networks.

Besides the above three main classifications, there are also some other interesting work to tackle the community detection in temporal networks. Bu et al. [20] proposed a novel autonomy-oriented computing-based method for community mining (AOCCM) from the multi-agent perspective. It utilizes reactive agents to pick the neighborhood node with the largest structural similarity as the candidate node, and thus determine whether it should be added into local community based on the modularity gain. They further improved AOCCM to a more efficient incremental version named AOCCM-i for mining communities from dynamic networks. The limitation of this work lies in that the selection of core nodes is based on the high degree. Jiao et al. [24] proposed a constrained common cluster based model to explore community structure hidden in the temporal or multiplex networks. The intrinsic assumption of the proposed model is that, there are common or coincident structures (called common cluster) in the snapshots of temporal networks or slices of multiplex networks. It also assumes that the number of common clusters is equal to the number of communities across the temporal or multiplex networks, and nodes are divided into different communities at each snapshot or slice. However, these assumptions may be too strong sometimes. As concluded by Jiao et al. [24], this method ignores the temporal order of snapshots and the difference of slices of multiplex networks.

3. Notations and problem definition

This section formulates the research problem by introducing some definitions. For convenience, we list out the notations used in this paper at first in Table 1.

Definition 1 (*Network Stream (GS)*)**.** We define the graph stream (GS) as a sequence of networks at different time slices: $GS = \{G^1, G^2, \ldots, G^t, \ldots\}$, where

- G^t represents the snapshot of network at time t, and $G^t = (V(G^t), E(G^t))$;
- $V(G^t)$ denotes the set of vertices in G^t;
- $E(G^t)$ denotes the set of edges in G^t ;

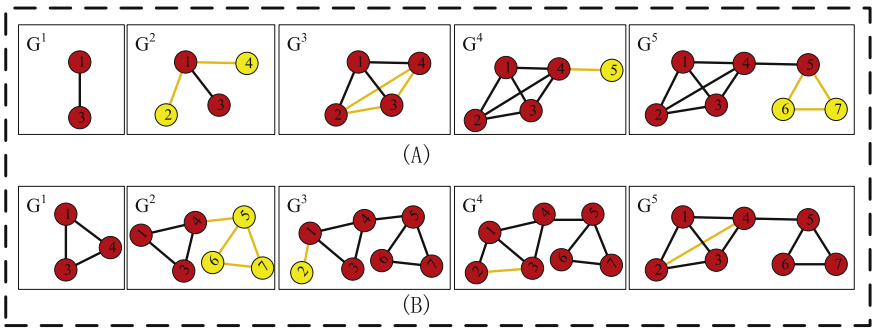


Fig. 1. Two networks that can be aggregated to the same state at time $t5$ but have different evolving processes.

Table 1
The notations used in this paper.

Notations	Descriptions
GS	The graph stream of the evolving network
G^t	The network at the time t
ΔG^{t+1}	The incremental changes from time t to $t+1$
CS	The community streams detected in evolving networks
CS^t	The communities detected from the network at the time of t
C_i^t	The ith community of G^t
$V(G^t)$	The set of vertices in G^t
$E(G^t)$	The set of edges in G^t
K^t	The number of communities at the time of t

Definition 2 (*Community Structure* (CS^t)). Let CS^t represent the set of community structures detected from G^t. That is, $CS^t = \{C_1^t, C_2^t, \ldots, C_i^t, \ldots, C_{K^t}^t\}$, where

- K^t represents the number of communities in G^t;
- C_i^t represents the ith community of G^t, $i = 1, 2, 3, \ldots, K$;
- $C_k^t \cap C_l^t$ is not necessary to be empty;

Definition 3 (*Community Stream* (CS)). We define the community stream as $CS = \{CS^1, CS^2, \ldots, CS^t, \ldots\}$, where CS^t denotes set of community structures detected from network G^t.

Given the community structure CS^t of G^t and the incremental changes ΔG^{t+1} from time t to $t+1$, the research problem is to find the community structure CS^{t+1} corresponding to G^{t+1}. Thus the research problem is formalized as follows:

$$f : (CS^t, \Delta G^{t+1}) \rightarrow CS^{t+1} \tag{1}$$

4. Proposed method

4.1. The framework to detect communities in dynamic networks

The framework to detect communities incrementally in evolving social networks is illustrated in Fig. 2. The communities should be first identified in the original social network at time t_0. As time passes, it is required to collect the incremental elements in the social network and make a formal analysis. Then the incremental updating strategies are designed based on the communities at the last state. Finally, the latest communities are derived, and the evolutionary paths are also updated accordingly.

4.2. The analysis of incremental elements

The incremental elements ΔG^{t+1} refer to all changes between time t and $t+1$, which could be composed of various subgraphs. Let $subG$ represent one subgraph of ΔG^{t+1}, then $V(subG)$ denotes the set of vertices in $subG$ and $E(subG)$ denotes the set of edges in $subG$. According to the relationship between $subG$ and the communities

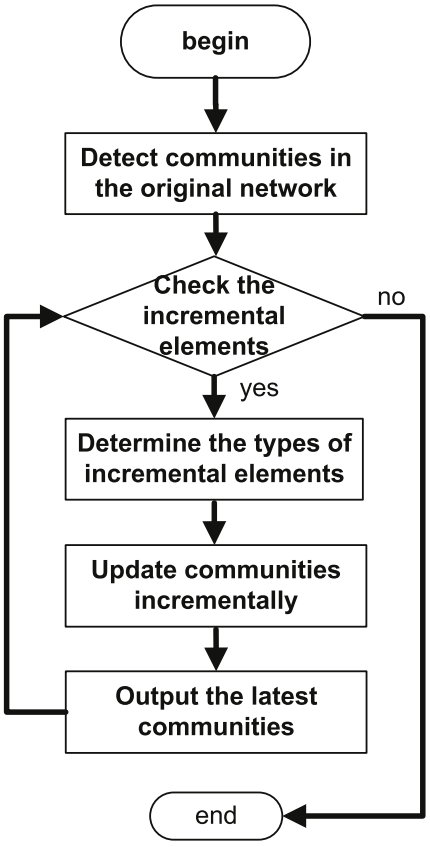


Fig. 2. The framework to detect communities incrementally in evolving social networks.

at previous time slice CS^t, we propose the following four types of $subG$.

(1) Complete Independent (CompInde)

The incremental subgraph $subG$ is considered to be complete independent iff none of its vertices is in a community of CS^t. That is, all the vertices of $subG$ are newly joined in the network at time $t+1$. The complete independent incremental subgraph type is defined as follows:

$$CompInde(subG) = 1 \quad iff$$
$$(\forall C_k^t)(C_k^t \in CS^t) \rightarrow (V(subG) \cap V(C_k^t) = \emptyset) \tag{2}$$

(2) Complete Contained(CompCont)

The incremental subgraph $subG$ is considered to be complete contained iff its vertices have appeared before in a certain community of CS^t. Hence, all those vertices in $subG$ can be considered as old nodes. The complete contained incremental subgraph type

is formally defined as follows:

$$CompCont(subG) = 1 \quad iff$$
$$(\forall v)v \in V(subG) \rightarrow (\exists C_k^t)v \in V(C_k^t) \tag{3}$$

(3) Mixed with New and Old Nodes (Mixed)

The incremental subgraph *subG* is considered to be mixed with new and old nodes iff it has old nodes and new nodes in *subG*. This means that in *subG* there are some (old) vertices belonging to one or several communities of CS^t and other new emerging vertices joining the network at time $t+1$. The formal definition of this type of incremental subgraph is given as follows:

$$Mixed(subG) = 1 \quad iff$$
$$(\exists C_k^t)(V(C_k^t) \cap V(subG) \neq \emptyset) \wedge$$
$$(\exists v)(v \in V(subG) \wedge v \notin V(CS^t)) \tag{4}$$

(4) Multiple Contained (MultiCont)

The incremental subgraph *subG* is considered to be multiple contained iff all its vertices are old ones distributed in several(more than one) communities of CS^t. The formal definition of this type of incremental subgraph is given as follows:

$$MultiCont(subG) = 1 \quad iff$$
$$(\forall v)v \in V(subG) \rightarrow (v \in V(CS^t))$$
$$\wedge(\exists C_k^t)(\exists C_l^t)(C_k^t \cap subG \neq \emptyset)$$
$$\wedge(C_l^t \cap subG \neq \emptyset) \tag{5}$$

It is important to emphasize that the incremental subgraph mentioned above can be a vertex, an edge, a connected component or even a complete subgraph. Fig. 3 describes some examples of the above four types of incremental subgraphs.

4.3. Strategies to update communities

The edge weight plays a very important role in detecting communities. Bu et al. [8] proposed a very interesting method named GLEAM to identify communities based on game optimization. In their work, they employed cosine similarity to weight each edge including intra-edge and inter-edge. However, they did not consider temporal dynamics of networks. In evolving networks, the time decaying strategy for edge weights should be designed to determine the influence of historical relationships. In this section, we define the updating rule of edge weights as follows:

$$w_{ij}^{t+1} = \alpha \cdot w_{ij}^t + w_{ij}^{\Delta G} \tag{6}$$

where,

- w_{ij}^{t+1} denotes the weight at time $t+1$, with time decaying updating rule used;
- $w_{ij}^{\Delta G}$ denotes the weight of e_{ij} that appears in ΔG. If e_{ij} does not emerge, then $w_{ij}^{\Delta G} = 0$;
- α is a parameter ($0 \leq \alpha \leq 1$), which denotes the extent of historical influence.

When $\alpha = 0$, the influence of historical relationships is ignored; while when $\alpha = 1$, the influence of historical relationships does not decay with time. A threshold parameter θ is often set to avoid infinite updating. If $w_{ij}^{t+1} \leq \theta$, let $w_{ij}^{t+1} = 0$.

(1) The updating strategy for "Complete Independent" (ComInde) subgraph:

Let *subG* be an incremental subgraph of type ComInde. This means that all the nodes of *subG* are new emerging ones at time $t+1$. Thus, $V(subG) \cap V(G^t) = \emptyset$. In this case, we consider *subG* to be a new community at time $(t+1)$. This corresponds to the birth of a new community.

(2) The updating strategy for "Complete Contained" (CompCont) subgraph:

Let *subG* be an incremental subgraph of type CompCont. This means that all the nodes of *subG* are old ones and have appeared at previous time t. Thus, $(\exists C_k^t)(C_k^t \in CS^t) \cap (C_k^t \cap V(subG) \neq \emptyset)$.

In this case, the time decaying strategy is first adopted to update the weight of each edge according to Eq. (6). Then we should determine which edges should be removed. Inspired by the work in [20], we aim to maximize the local modularity of current community by removing some edges. Thus, we first calculate the local modularity of C_k^t to get $Q(C_k^t)$. Then the edges in C_k^t are ranked in ascending order according to their weights. By selecting the edge e with minimum weight, we calculate ΔQ and determine whether the edge e is kept or not according to the following expressions (7) and (8):

$$\triangle Q = Q(G(C_k^t) - e) - Q(G(C_k^t)) \tag{7}$$

$$\begin{cases} \text{if } \triangle Q > 0, & \text{remove } e \text{ from } C_k^t \\ \text{else } \triangle Q \leq 0, & \text{keep } e \text{ in } C_k^t. \end{cases} \tag{8}$$

(3) The updating strategy for the subgraph "Mixed with New and Old Nodes" (Mixed):

We define the subordinating strength of one vertex v to a community C as follows:

$$S_v^C = \frac{\sum_{u \in N(v) \cap V(C)} w_{vu}}{\sum_{u \in N(v)} w_{vu}} \tag{9}$$

where,

- $N(v)$ denotes the neighbors of v;
- w_{vu} denotes the weight of e_{vu}.

Let *subG* be an incremental subgraph of Mixed type. This means that some nodes in *subG* belong to C_k^t, and the rest are new emerging nodes. Thus $V(subG) = V_{old} \cup V_{new}$, and $V_{old} \subseteq V(C_k^t)$. The updating strategies for the mixed type is described in Algorithm 1. Obviously, the time complexity of the Algorithm 2 is $O(L)$, where $L = max\{|V(C_j^t)|, |V(subG)|\}$ and $L \ll n$.

Algorithm 1 Update communities for the Mixed type.

Input: $V_{old}, V_{new}, C_k^t, subG$
Output: the communities updated at $t+1$

```
updateCS^{t+1} ← ∅;
for each v ∈ V_old do// update old nodes first
    calculate S_v^{C_k^t}, S_v^{subG} according to Equ. (9);
    if S_v^{C_k^t} ≤ S_v^{subG} then
        copy v to subG;
    else
        remove v from subG and add it to C_k^t;
    end if
end for
for each v ∈ V_new do // update new nodes next
    calculate S_v^{C_k^t}, S_v^{subG} according to Equ. (9);
    if S_v^{C_k^t} ≤ S_v^{subG} then
        copy v to subG;
    else
        remove v from subG and add it to C_k^t;
    end if
end for
updateCS^{t+1} ← updateCS^{t+1} ∪ {V(subG)} ∪ {V(C_k^t)};
return updateCS^{t+1};
```

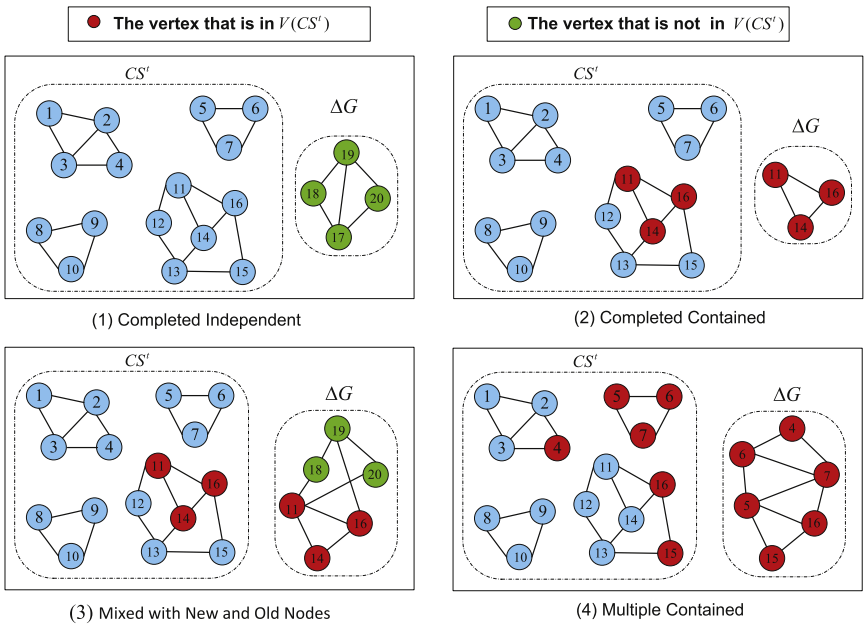


Fig. 3. The four different types of incremental elements.

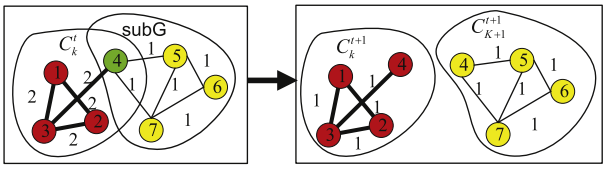


Fig. 4. Community updating for the type of Mixed: the birth of new community.

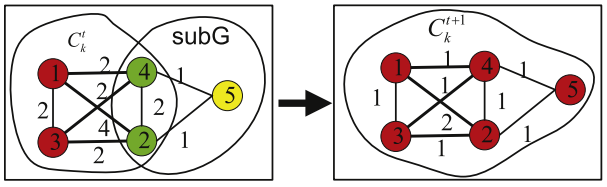


Fig. 5. Community updating for the type of Mixed: the enlarging of the previous community.

Figs. 4 and 5 illustrate two examples. In Fig. 4, we make a copy on v_4 and let it belong to both C_k^t and *subG* because $S_{v_4}^{C_k^t} == S_{v_4}^{subG}$. The new nodes v_5, v_6 and v_7 are kept in *subG*. Thus, the above updating strategy results in a new community. Furthermore, this strategy allows for the overlapping of communities. In Fig. 5, the old nodes v_4 and v_2 are first processed. By calculating the subordinating strength, we can determine that both of them are kept in C_k^t and removed from *subG*. According to Algorithm 1, we can find that v_5 should be removed from *subG* and added to C_k^t. By doing that, we get an enlarged C_k^{t+1}.

(4) The updating strategy for the "Multiple Contained" (MultiCont) subgraph:

Let *subG* be an incremental subgraph of type MultiCont. Assuming $V(subG) = V_k \cup V_l$, where V_k represents a set of nodes belonging to the community C_k^t, and V_l represents a set of nodes belonging to the community C_l^t. The updating strategy for the MultiCont type is described in Algorithm 2. Obviously, the time complexity of the Algorithm 2 is $O(K \times L)$, where K denotes the number of communities at t and it is nearly a constant; $L = max\{|V(C_j^t)|, |V(subG)|\}$ and $L \ll n$.

Algorithm 2 Update communities for the MultiCont type.

Input: $C_1^t, C_2^t, ..., C_K^t, subG$
Output: the communities updated at $t+1$

```
updateCS^{t+1} ← ∅;
for k = 1; k ≤ K; k++ do
    V_k ← V(C_k^t) ⋂ subG;
    for each v ∈ V_k do
        calculate S_v^{C_k^t}, S_v^{subG} according to Equ. (9);
    end for
end for
for each v in ⋃ V_k do
    if max(S_v^{C_k^t}) ≤ S_v^{subG} then
        copy v to subG;
    else
        remove v from subG and add it to the community with maximum strength;
    end if
end for
updateCS^{t+1} ← updateCS^{t+1} ∪ {V(subG)} ∪ {V(C_k^t)};
return updateCS^{t+1};
```

Figs. 6 and 7 demonstrate two examples. In Fig. 6, the nodes *a* and *b* are copied to *subG*. The nodes *c* and *d* are also copied to *subG*. Thus, a new community is generated. In Fig. 7, the nodes 2, 3 and 5 are removed from *subG* and added to C_k^{t+1} and C_l^{t+1}, respectively.

Besides the above four incremental relevant types, there are still some communities which are not concerned with the incremental elements. As to these communities, the time decaying strategy is first adopted, which results in the disappearance of some edges. If the previous community is still kept connected, the updating strategies for "Completed Contained" is adopted to determine which edge should be removed. This also allows us to update the communities. Else, if the disappearance of some edges results in two or more isolated connected components, then we consider that the previous community is divided into two or more communities.

4.4. The algorithm to detect communities incrementally in dynamic evolving networks

It is necessary to mention that the algorithm is used outside of the network. In our work, we did not design any mechanism for finding the changes. The changes are detected by checking the time stamps. For instance, one person sends an email to another, this social event will takes the time stamp in nature with the email service. In the microblogging platform, the time when a user uploads/replies/forwards some postings could be recorded automatically. Thus, when we design and implement the crawling program, we can get the timestamp of the users (nodes) and social relationship (edge) easily. With the help of timestamp, we can get the changes (the new coming nodes and edges). By comparing the changes (nodes and edges) with the previous network, we can determine the old/new emerging nodes/edges. The algorithm to identify the types of incremental elements is described in Algorithm 3. The time complexity of the Algorithm 3 is $O(K \times L)$, where K denotes the number of communities at t and it is nearly a constant; $L = max\{|V(C_j^t)|, |V(subG)|\}$ and $L \ll n$.

With the types of those incremental elements at hand, we can update communities incrementally. The algorithm to detect communities incrementally in time-evolving network is presented in Algorithm 4. The running time of the step 2 is $O(n)$. The most time consuming parts of the Algorithm 4 are from step 4 to step 20. The time complexity of the above parts is $O(S \times K \times L)$, where S denotes the number of incremental subgraphs; K denotes the number of communities at t and it is nearly a constant; $L = max\{|V(C_j^t)|, |V(subG)|\}$ and $L \ll n$. The running time of step 21 to step 25 is $O(E)$, where $E = max\{|E(C_j^t)|, |E(subG)|\}$. Thus, the time complexity of Algorithm 4 is $O(n + S \times K \times L + E)$. That is, the proposed incremental method has a nearly linear complexity, which guarantees high efficiency.

Algorithm 3 Identifying the types of incremental elements.

Input: CS^t, $subG_i$
Output: $Type(subG_i)$, $RelComLab$

$V_{old(i)} \leftarrow \emptyset$, $RelComLab \leftarrow \emptyset$;
for each community $C_j^t \in CS^t$ **do**
  calculate $V_{old(ij)}$ by $V_{old(ij)} \leftarrow V(subG_i) \bigcap V(C_j^t)$;
  if $V_{old(ij)} \subseteq V(C_j^t)$ **then** //case2: completed contained
    $Type(subG_i) = 2$;
    $RelComLab(subG_i) \leftarrow RelComLab(subG_i) \bigcup \{j\}$;
  end if
  if $V_{old(ij)} != \emptyset$ **then**
    $V_{old(i)} \leftarrow V_{old(i)} \bigcup \{V_{old(ij)}\}$;
    $RelComLab(subG_i) \leftarrow RelComLab(subG_i) \bigcup \{j\}$;
  end if
end for
if $V_{old(i)} == \emptyset$ **then** //case1 completed independent
  $Type(subG_i) = 1$;
end if
if $|V_{old(i)}| == 1$ **then** //case3: mixed with new and old nodes
  $Type(subG_i) = 3$;
end if
if $|V_{old(i)}| > 1$ **then** //case4: multi-contained
  $Type(subG_i) = 4$;
end if
return $(i, Type(subG_i), RelComLab(subG_i))$;

As a summary, the proposed incremental method has the following advantages. The number of communities is not required to be preset. The time decaying strategy on edge weights can help us adjust the historical influence. It also enables the deletion of a node, which helps us to identify the shrinking, death and division of an old community. The overlapping communities are also allowed which enables the birth of a new community and the enlarging or merging of existing communities. Therefore, the method presented in this paper is very flexible and can overcome the shortcomings of existing work.

Algorithm 4 Detecting communities in the evolving network incrementally.

Input: CS^t, ΔG^{t+1}, α, θ
Output: CS^{t+1}

$Set\_subG^t \leftarrow \emptyset$, $upLabel \leftarrow \emptyset$;//initialization
obtain all the connected components from ΔG^{t+1} to get Set_subG^t;
update the weight of each edge according to Equ(8);
for each $subG_i \in Set\_subG^t$ **do**
  determine the types of $subG_i$ by invoking Algorithm 3;
  $upLabel(subG_i) \leftarrow upLabel \bigcup RelComLab(subG_i)$;
  if $Type(subG_i) == 1$ **then**//new
    add $subG_i$ to CS^{t+1};
  end if
  if $Type(subG_i) == 2$ **then**//contained
    calculate ΔQ and update it according to Equ(7) and Equ (8);
    add the above results to CS^{t+1};
  end if
  if $Type(subG_i) == 3$ **then**//Mixed
    update communities by invoking Algorithm 1;
  end if
  if $Type(subG_i) == 4$ **then**//MultiCont
    update communities by invoking Algorithm 2
  end if
end for
for $j \notin upLabel$ **do** //not concerned with incremental elements
  calculate ΔQ and update it according to Equ. (7) and Equ. (8);
  add the above results to CS^{t+1};
end for
return CS^{t+1};

5. Experiments and evaluation

5.1. Experimental data

We evaluated the proposed method on two real-world networking data sets: Enron email data set and DBLP coauthor data set.

(1) Enron Email network: Enron data set is an email corpus prepared by CALO Project and it is available at http://www.cs.cmu.edu/~enron/. It consists of 275,332 emails sent by 150 Enron employees from December 1999 to March 2002. We firstly extracted this data into database to structure it. Then an email network is constructed. In the network, a node represents a staff with an email address, while edges represent the email communication relationships. The weight attached to an edge represents the number of emails sent between one user to another.

(2) DBLP coauthor network: DBLP provides open bibliographic information on major computer science journals and proceedings. In this paper, we takes DBLP as the data source and crawled all the coauthor relationships of articles published in the most important international conferences on data mining (IJCAI, AAAI, KDD, CIKM, ICDE, ICDM, PAKDD). It contains 30,489 authors and 20,931 papers, published from 1994 to 2015. The coauthor network is constructed, in which a node represents an author, while edge represent the collaborative relationships between authors. The weight attached to each edge indicates the number of collaborations of two authors.

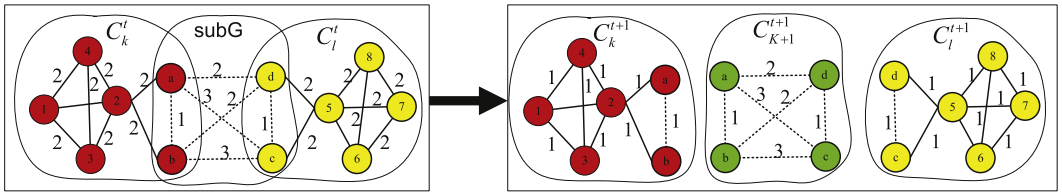


Fig. 6. Community updating for the type of MultiCont: the birth of a new community.

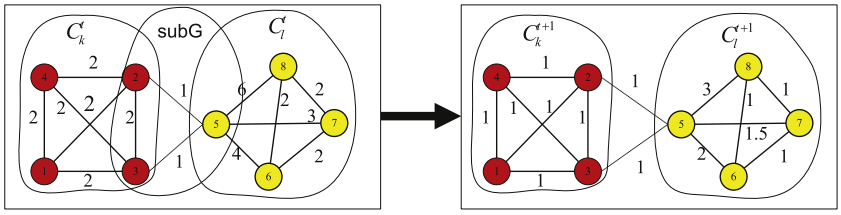


Fig. 7. Community updating for the type of MultiCont: the strengthening of previous communities.

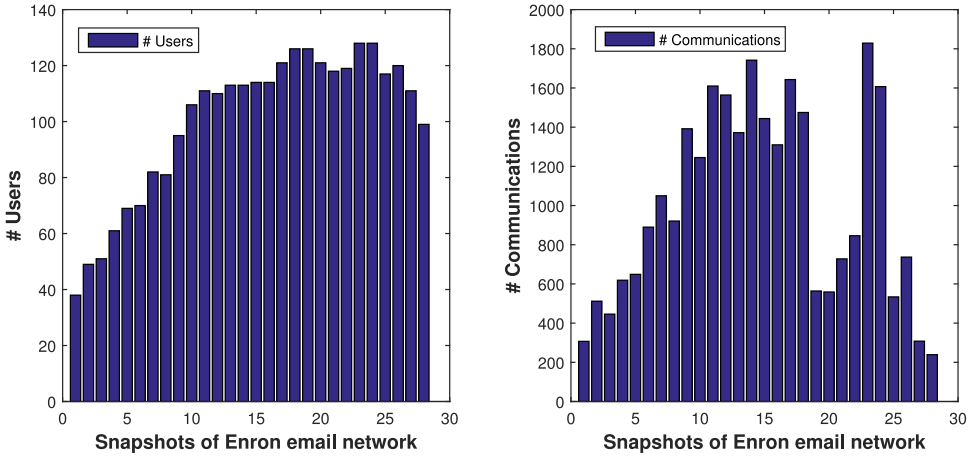


Fig. 8. The number of users and communications in each snapshot of Enron email network.

5.2. Performance metric

In this experiment, we use the common performance metric of modularity to evaluate the proposed algorithm. The metric of modularity is based on the principle that a good community is strongly connected internally and isolated from the rest of the network. Without loss of the generality, we adopt the definition of modularity proposed in [25], which is suitable for weighted network. Suppose we have a division for the network, the modularity Q is defined as follows.

$$Q = \sum_{k=1}^{K} \left[\frac{Cut(V_k, V_k)}{Cut(V, V)} - \left(\frac{Cut(V_k, V)}{Cut(V, V)} \right)^2 \right] \tag{10}$$

where

- V is the set of the nodes in the network;
- V_k is the set of the nodes in the community C_k;
- $Cut(V_i, V_j) = \sum_{p \in V_i, q \in V_j} w_{pq}$, it denotes the sum of the weights of edges between community C_i and C_j;

5.3. Data processing

The Enron email data is organized into folders. It contains all the emails from about 150 users, mostly senior management of Enron. To avoid repeated statistics, we designed a program to read all the "sent" directories of each user, and extracted all the communicating relations from the head of each email file. When processing these data, we found that there were a large number of emails sent and ccd to others as well. Thus, the email addresses beginning with "From:, to:, cc:" are all taken into account. The email network was constructed by month, which begins in December 1999. The number of users and the number of communications in each snapshot are shown in Fig. 8.

As to the DBLP co-author data, we first parsed the xml format meta data. Then we extracted all the information about "authors" and "year" for each publication. The coauthoring relationship was constructed between "A" and "B" if they have collaborated on one paper. By doing that, an undirected weighted network in the form of edge list was obtained by year. The number of authors and the number of coauthoring relationships in each snapshot were shown in Fig. 9.

5.4. Visualization of the identified communities

We implemented the incremental algorithm proposed in this paper, and identified those communities hidden in networking data. In the visualizing results, different communities were drawn with different coloring blocks, with vertices belonging to one community were painted with the same color.

Fig. 10 illustrates the communities detected from four adjacent email networks. The vertex represented a person using the corresponding email address. It was labeled with person's abbreviated

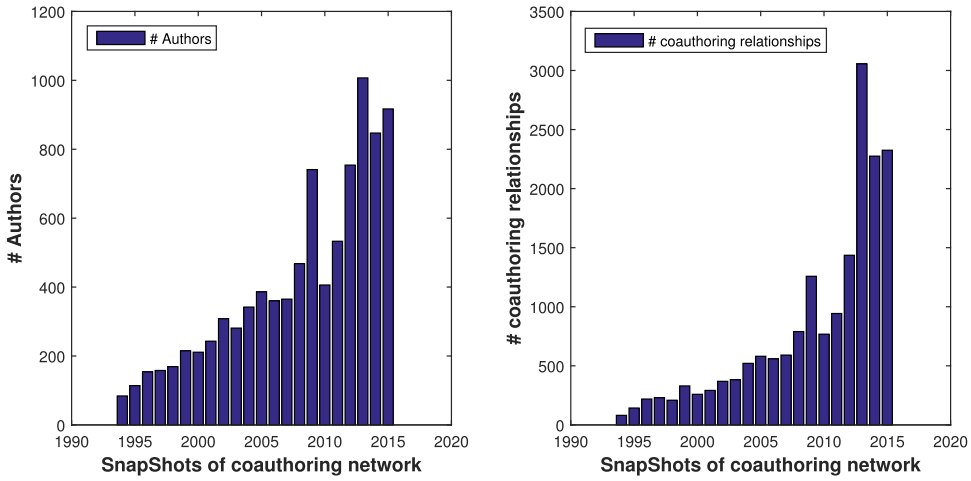


Fig. 9. The number of authors coauthoring relationships in each snapshot of DBLP coauthoring network.

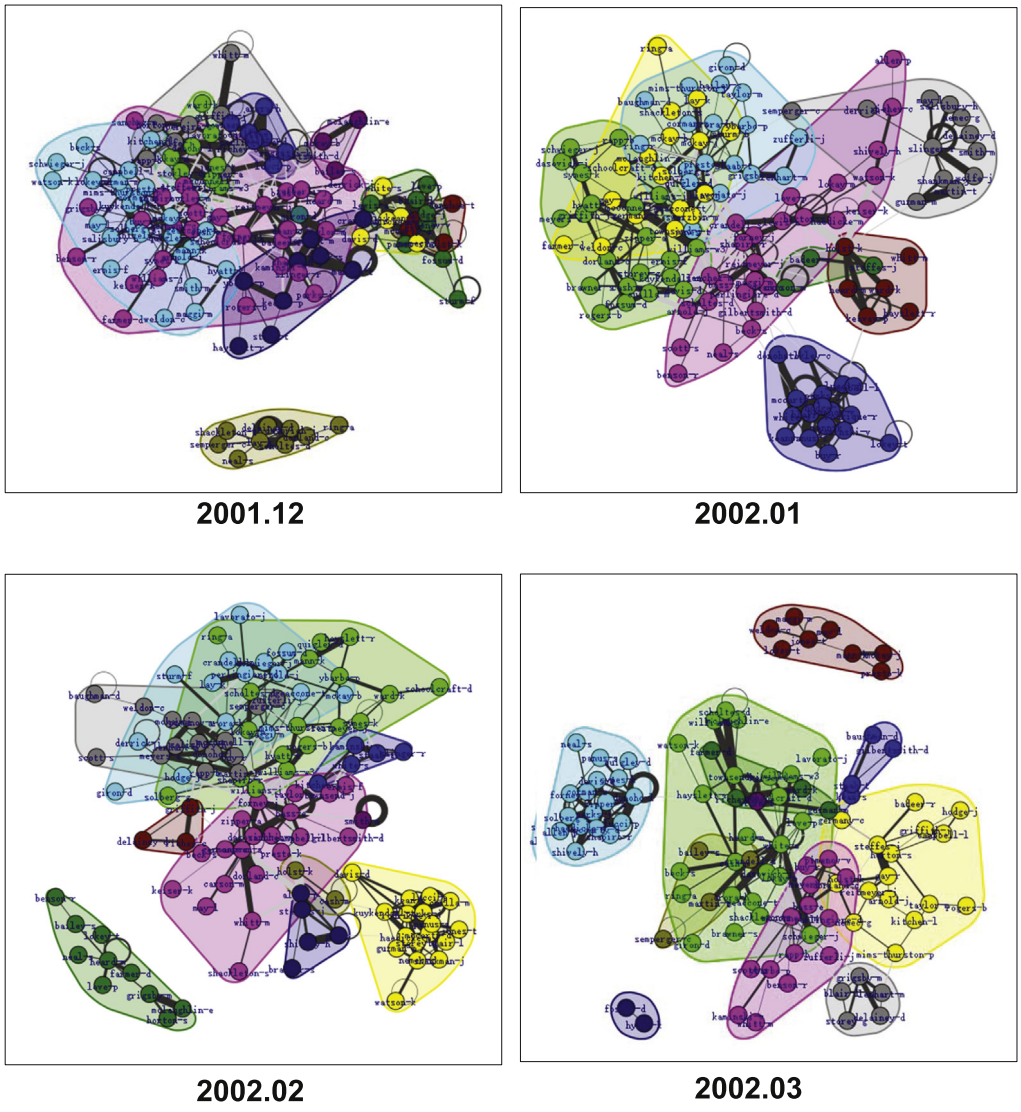


Fig. 10. The communities detected from 4 adjacent Enron email networks.

name (same as the folder name of each person). The edge between a pair of vertices represented the interacting relationship of both persons. It is worth emphasizing that each edge was attached with a weight to represent the number of communications by email

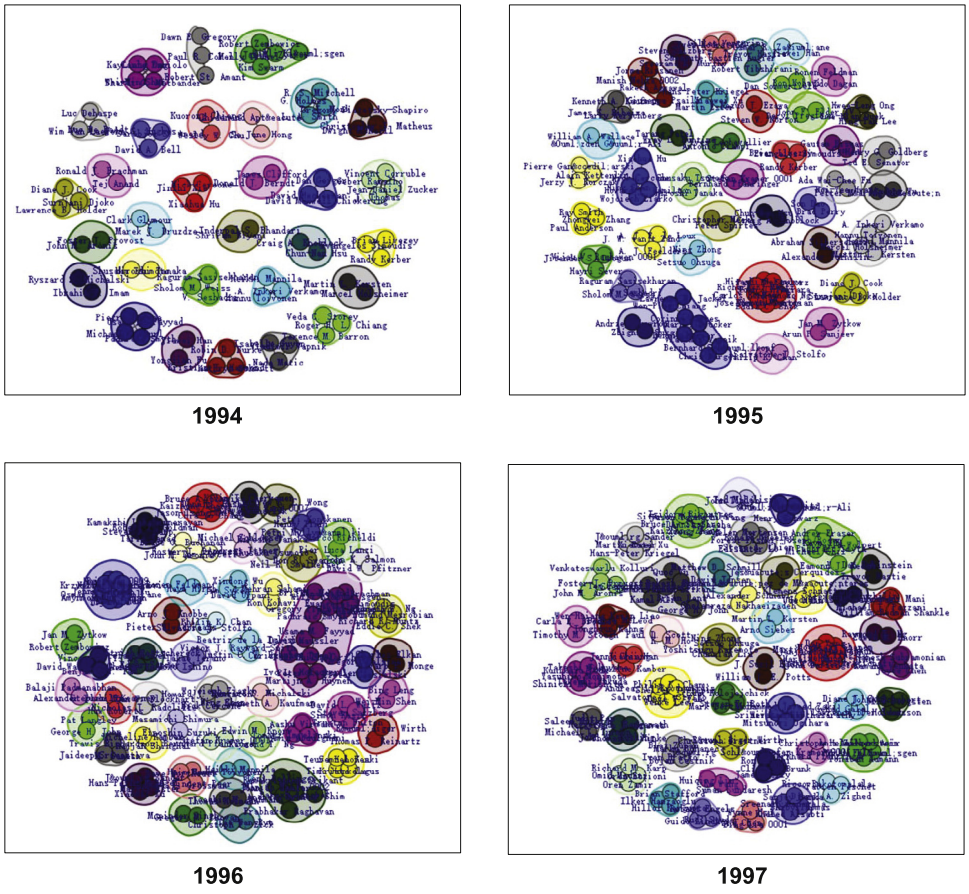


Fig. 11. The communities detected from 4 adjacent coauthoring networks.

between the connected two persons. Here, to make the results more clearly, we adopted lines with different thickness to differentiate the tightness of the relationship. The thicker the line was, the closer both persons in terms of relationship (communication) were. Different communities were drawn within different coloring blocks, in which the vertices belonging to one community were painted with the same color. From Fig. 10, we found that there were 12, 8, 11, 11 communities detected respectively from December of 2001 to March of 2002.

Fig. 11 illustrates some resulting communities detected from four adjacent coauthoring networks. The vertex represented an author. It was labeled with his or her name. The edge between a pair of vertices represented the coauthoring relationship. There was a weight value attached to each edge to denote the strength of both authors collaboration. As before, the thicker the edge is, the more the authors collaborated with each other. From Fig. 11 we found that the entire screen was crowded with vertices. That is a result of the large number of authors. Meanwhile, a variety of small communities tended to be detected from the coauthoring networks. That was also consistent with the fact that people belonging to the same academic team are more likely to collaborate.

The proposed incremental method is not only capable of detecting communities efficiently, but also enable us to observe the evolving paths. To make it more clearly, we selected some examples to show the advantages of our method in detail.

Figs. 12 and 13 show the identified communities from the email network at different snapshots. Fig. 12 illustrates the results from snapshot in Dec. 1999 and Jan. 2000. Seven and six communities were identified, respectively. It is noticed that when moving from Dec. 2019 to Jan. 2000, four communities enlarged; two communities merged into a bigger one in the next time slice; one community died and another new community was born. Fig. 13 illustrates the communities detected from the snapshot of Jan. 2000 and Feb. 2000. During that period, there were two communities which kept their members and size. One big community was divided into two small ones. One community is enlarged with new members "holst-k", "grgsby-m" and "farmer-d" joining in. The community which emerged in January with two colleagues interact with each other only once, became dead because no members continue to communicate in the next month. One community with "taylor-m", "shackleton-s" and "jones-t" shrank, because some of its members did not contact with each other any more.

5.5. Performance comparison

The proposed incremental algorithm (Algorithm 4) contains two parameters α and θ, where α controls the importance of the last community and implies the influential degree of historical information; θ is the weighting threshold that controls whether an edge continue into the next time period. In this paper, both of them were set to be tuning parameters and were selected empirically.

We first investigate the effect of parameter α by adjusting it from 0 to 1 with 0.25 as intervals. The performance on modularity is calculated by averaging the values in different time slices. We also analyze how the parameter θ affects the performance by fixing α. Considering that the average weight of the Enron Email network is 14.32, we investigate the effect of θ by adjusting it from 0 to 12 with 0.5 as intervals. In the DBLP coauthoring network, the average weight is 1.86. Hence, we tested the effect of θ from 0 to 2 with 0.25 as intervals. The experimental results are shown in Figs. 14(a) and 14(b).

From Fig. 14(a), for the Enron Email data sets, we can see that the best performance is achieved for $\alpha = 0.50$ and $\theta = 3.5$.

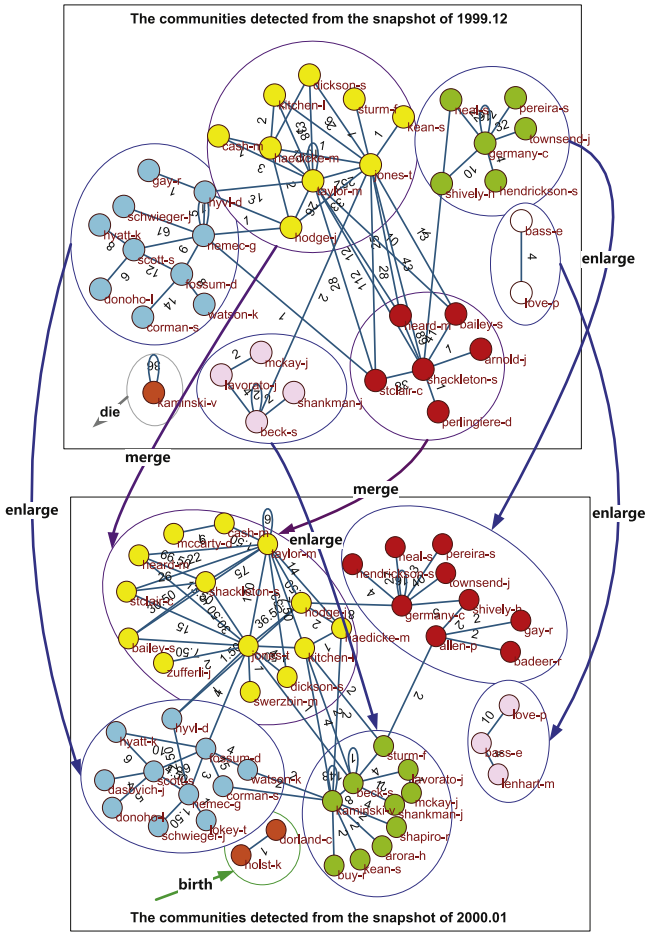


Fig. 12. Communities detected from Enron Email networks (Dec. 1999–Jan. 2000).

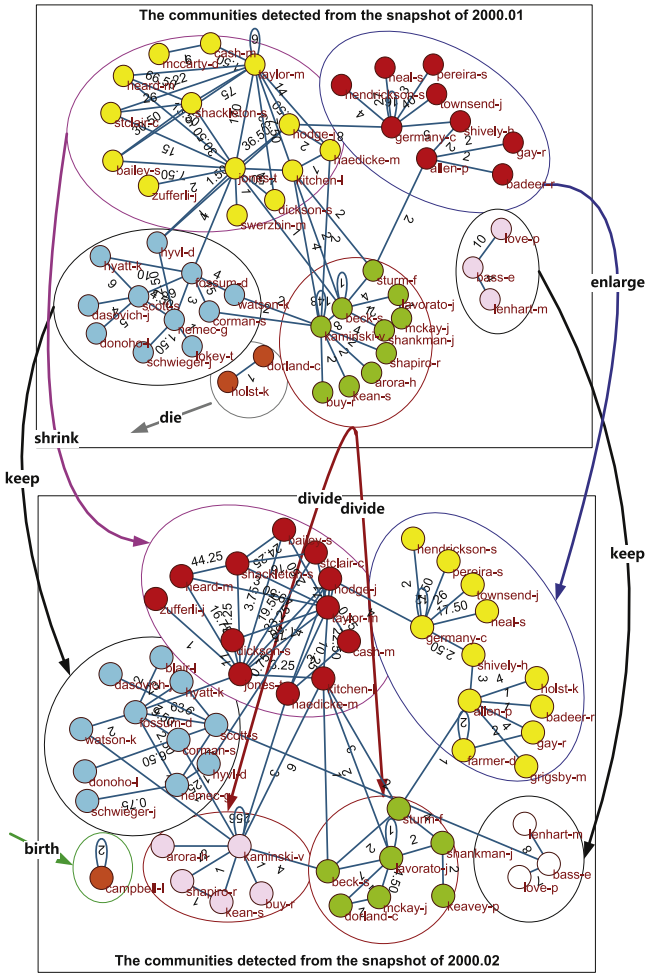


Fig. 13. Communities detected from Enron Email networks (Jan. 2000–Feb. 2000).

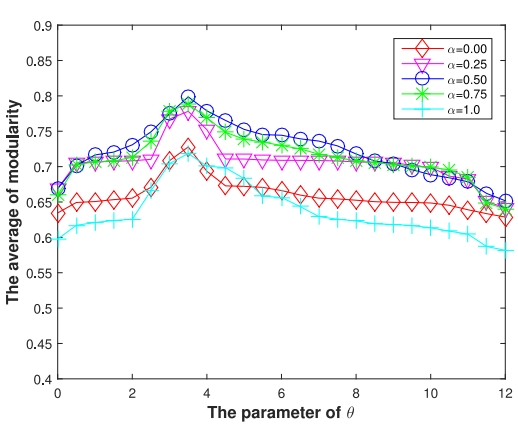
In order to interpret this phenomenon, we conduct a statistical analysis on the Enron Email data sets. We found that the number of communications in each different slices is 14.32 on average. Let us assume that the value $\alpha = 0.5$ is used in the time decaying process. That is, the number of communications will be changed to 7.16 and 3.58 if this person did not contact others within the next two time slices. When it comes to the third time slice, the relationship (edge) will be removed from the network because of $\theta = 3.5$. This setting is consistent with the real word situations. That means, if a person has been out of touch with his/her colleagues for three months, we can consider that he/she has deserted this job.

According to Fig. 14(b), for the DBLP data sets, we can see that the best performance is achieved for $\alpha = 0.75$ and $\theta = 0.75$. In order to interpret this phenomenon, we also conduct a statistical analysis on the DBLP data sets. We found that the number of coauthoring relationships in each different slice is 1.86 on average. When α is set to be 0.75, the coauthoring weight of the person on average will be changed to 1.39, 1.04 and 0.78 in the next three time slices. If this author still did not appear, then the coauthoring relationship will be removed as it is less than the threshold value ($\theta = 0.75$). As a summary, the above settings will not keep the author in the network if he/she has not coauthored with his friends for more than three years.

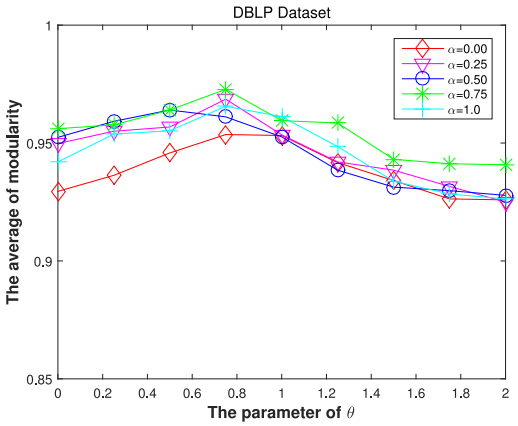
With the above parameter settings, we compared our proposed approach with four methods named IncOrder [15], ARTISON [17], FacetNet [26] and LabelRankT [27]. For the algorithm of LabelRankT, the conditional updating parameter q was set to be 0.6 as suggested in [27]. The Enron email network covered 28 months (from December 1999 to March 2002), which resulted in 28 modularity Q values. Similarly, as DBLP coauthor network concerned 22 years from 1994 to 2015, this resulted in 22 modularity Q values. The comparison results are shown in Fig. 15.

According to Fig. 15, we noticed that the proposed incremental algorithm outperformed the others. It achieved the highest Q value on the two real world networking data. As a whole, it can be seen that all algorithms performed better on DBLP coauthoring network than Enron email network. The value of the modularity was larger than 0.85. This is because most subgraphs in DBLP coauthoring network were isolated subgraphs and each of them is as dense as a completed graph. In the case of DBLP coauthoring network, considering modularity on average, our algorithm outperformed ARTISON by 2.41%, IncOrder by 3.65%, LabelRankT by 5.56%, and FacetNet by 7.75%. In the case of the Enron email network, our algorithm significantly improved the performance of the four compared algorithms on average by 3.96%, 16.73%, 21.65%, and 26.30%, respectively.

We also made a deep analysis on our algorithm. For one thing, the proposed incremental method preferred forming a larger community while guaranteeing the overlapping situation. For another, the incremental updating strategy also kept the internal connections as closer as possible. This means in each time step, the communities updated with our method could keep the inner structure denser. Thus, it was capable of reaching a better performance.

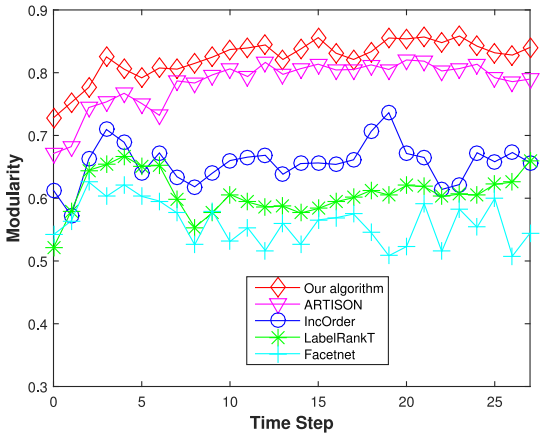


(a) The effect of different parameter settings on Enron Email data sets.

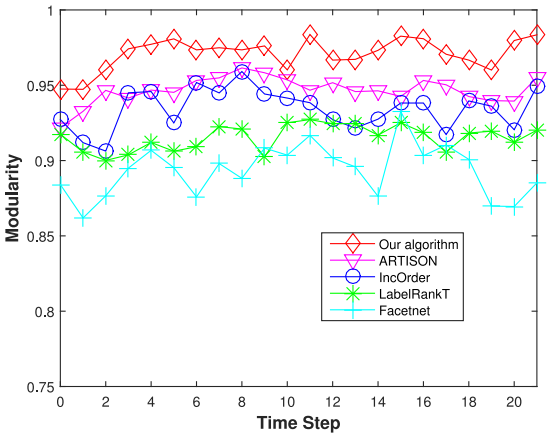


(b) The effect of different parameter settings on DBLP data sets.

Fig. 14. The analysis of parameter sensitivity on two kinds of data sets.



(a) Comparison of modularity on Enron Email communicating network.



(b) Comparison of modularity on DBLP coauthoring network.

Fig. 15. Comparison of modularity over time on two real world networks.

6. Conclusion

In this paper, an incremental method to detect communities in evolving networks is proposed. The main idea is to detect communities at the initial state, then to collect and analyze the incremental dynamic changes, and finally to update communities incrementally. The proposed method does not require the number of communities to be preset. The presented time decaying strategy on edge weights can help us adjust the degree of historical influence. Moreover, all evolving paths of communities can also be derived directly. Thus the incremental method presented in this paper is flexible and efficient. Its effectiveness has also been proved with extensive experiments on real world networking data sets.

In our future work, we plan to design some mechanisms for each node to monitor its changes based on multi-agent systems. By doing that, the cooperation, coordination, message exchanging and distributed processing will be taken into consideration.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant No. 61303167, 61702306, U1435215), the Ministry of Education in China Foundation for Humanities and Social Sciences (Grant No. 17YJCZH262, 18YJAZH136), the Natural Science Foundation of Shandong Province (Grant No. ZR2018BF013, ZR2017BF026), the Innovative Research Foundation of Qingdao (Grant No. 18-2-2-41-jch), the Key R&D Plan of Shandong Province (Grant No. 2016ZDJS02A11), the Scientific Research Foundation of SDUST for Innovative Team (Grant No. 2015TDJH102), the Key Project of Industrial Transformation and Upgrading of China (Grant No. TC170A5SW), and FEDER funds of Spain in the Project (Grant No. TIN2016-75850-R).

References

[1] W. Wu, J. Zhao, C. Zhang, F. Meng, Z. Zhang, Y. Zhang, Q. Sun, Improving performance of tensor-based context-aware recommenders using bias tensor factorization with context feature auto-encoding, Knowl.-Based Syst. 128 (C) (2017) 71–77.

[2] Z. Zhao, C. Li, Y. Zhang, J.Z. Huang, J. Luo, S. Feng, J. Fan, Identifying and analyzing popular phrases multi-dimensionally in social media data, Int. J. Data Warehouse. Min. 11 (3) (2015) 98–112.

[3] Z. Zhao, Y. Zhang, C. Li, L. Ning, J. Fan, A system to manage and mine microblogging data, J. Intell. Fuzzy Syst. 33 (1) (2017) 315–325.

[4] Z. Zhao, W. Liu, Y. Qian, L. Nie, Y. Yin, Y. Zhang, Identifying advisor-advisee relationships from co-author networks via a novel deep model, Inform. Sci. 466 (2018) 258–269.

[5] Z. Zhao, S. Feng, Q. Wang, J.Z. Huang, G.J. Williams, J. Fan, Topic oriented community detection through social objects and link analysis in social networks, Knowl.-Based Syst. 26 (2012) 164–173.
[6] J. Wu, F. Chiclana, H. Fujita, E. Herrera-Viedma, A visual interaction consensus model for social network group decision making with trust propagation, Knowl.-Based Syst. 122 (2017) 39–50.
[7] J. Wu, F. Chiclana, E. Herrera-Viedma, Trust based consensus model for social network in an incomplete linguistic information context, Appl. Soft Comput. 35 (2015) 827–839.
[8] Z. Bu, J. Cao, H.J. Li, G. Gao, H. Tao, Gleam: a graph clustering framework based on potential game optimization for large-scale social networks, Knowl. Inf. Syst. 55 (3) (2018) 741–770.
[9] Z. Bu, C. Zhang, Z. Xia, J. Wang, A fast parallel modularity optimization algorithm (fpmqa) for community detection in online social network, Knowl.-Based Syst. 50 (3) (2013) 246–259.
[10] Z. Zhao, S. Zheng, C. Li, J. Sun, L. Chang, F. Chiclana, A comparative study on community detection methods in complex networks, J. Intell. Fuzzy Syst. 35 (1) (2018) 1077–1086.
[11] Y. Dong, Z. Ding, F. Chiclana, E.H. Viedma, Dynamics of public opinions in an online and offline social network, IEEE Trans. Big Data (2018) http://dx.doi.org/10.1109/TBDATA.2017.2676810.
[12] G. Palla, A.L. Barabasi, T. Vicsek, Quantifying social group evolution, Nature 446 (7136) (2007) 664–667.
[13] S. Asur, S. Parthasarathy, D. Ucar, An event-based framework for characterizing the evolutionary behavior of interaction graphs, ACM Trans. Knowl. Discov. Data 3 (4) (2009) 1–36.
[14] C. Tantipathananandh, T.Y. Berger-Wolf, Finding communities in dynamic social networks, in: IEEE International Conference on Data Mining, 2011, pp. 1236–1241.
[15] H. Sun, J. Huang, X. Zhang, J. Liu, D. Wang, H. Liu, J. Zou, Q. Song, Incorder: Incremental density-based community detection in dynamic networks, Knowl.-Based Syst. 72 (2014) 1–12.
[16] R. Cazabet, F. Amblard, C. Hanachi, Detection of overlapping communities in dynamical social networks, in: IEEE Second International Conference on Social Computing, 2010, pp. 309–314.
[17] H.S. Cheraghchi, A. Zakerolhosseini, Toward a novel art inspired incremental community mining algorithm in dynamic social network, Appl. Intell. 46 (2) (2017) 409–426.
[18] X. Ma, D. Dong, Evolutionary nonnegative matrix factorization algorithms for community detection in dynamic networks, IEEE Trans. Knowl. Data Eng. 29 (5) (2017) 1045–1058.
[19] W. Zhang, R. Li, D. Feng, A. Chernikov, N. Chrisochoides, C. Osgood, S. Ji, Evolutionary soft co-clustering: formulations, algorithms, and applications, Data Min. Knowl. Discov. 29 (3) (2015) 765–791.
[20] Z. Bu, Z. Wu, J. Cao, Y. Jiang, Local community mining on distributed and dynamic networks from a multiagent perspective, IEEE Trans. Cybern. 46 (4) (2016) 986–999.
[21] X. Tang, C.C. Yang, Detecting social media hidden communities using dynamic stochastic blockmodel with temporal dirichlet process, ACM Trans. Intell. Syst. Technol. 5 (2) (2014) 1–21.
[22] T. Falkowski, J. Bartelheimer, M. Spiliopoulou, Mining and visualizing the evolution of subgroups in social networks, in: IEEE/WIC/ACM International Conference on Web Intelligence, 2007, pp. 52–58.
[23] W. Wang, P. Jiao, D. He, D. Jin, L. Pan, B. Gabrys, Autonomous overlapping community detection in temporal networks, Knowl.-Based Syst. 110 (C) (2016) 121–134.
[24] P. Jiao, W. Wang, D. Jin, Constrained common cluster based model for community detection in temporal and multiplex networks, Neurocomputing 275 (2018) 768–780.
[25] A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E Stat. Nonlinear Soft Matter Phys. 70 (6 Pt 2) (2004) 066111.
[26] Y.R. Lin, Y. Chi, S. Zhu, H. Sundaram, B.L. Tseng, Analyzing communities and their evolutions in dynamic social networks, ACM Trans. Knowl. Discov. Data 3 (2) (2009) 1–31.
[27] J. Xie, M. Chen, B.K. Szymanski, Labelrankt: incremental community detection in dynamic networks via label propagation, in: The Workshop on Dynamic Networks Management and Mining, 2013, pp. 25–32.