Local Community Mining on Distributed and Dynamic Networks From a Multiagent Perspective

Zhan Bu, Zhiang Wu, *Member, IEEE*, Jie Cao, and Yichuan Jiang, *Senior Member, IEEE*

***Abstract*—Distributed and dynamic networks are ubiquitous in many real-world applications. Due to the huge-scale, decentralized, and dynamic characteristics, the global topological view is either too hard to obtain or even not available. So, most existing community detection methods working on the global view fail to handle such decentralized and dynamic large networks. In this paper, we propose a novel autonomy-oriented computing-based method for community mining (AOCCM) from the multiagent perspective in the distributed environment. In particular, AOCCM 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. We further improve AOCCM to a more efficient incremental version named AOCCM-i for mining communities from dynamic networks. AOCCM and AOCCM-i can be easily expanded to detect both nonoverlapping and overlapping global community structures. Experimental results on real-life networks demonstrate that the proposed methods can reduce the computational cost by avoiding repeated structural similarity calculation and can still obtain the high-quality communities.**

***Index Terms*—Autonomy-oriented computing (AOC), distributed and dynamic networks, incremental computing, local community detection (LCD), multiagent.**

I. Introduction

REAL-LIFE networks, such as the transportation systems [1], the computer science networks [2], and the online friendship network systems (e.g., Twitter and Facebook) [3], [4], are composed of a large number of highly interconnected nodes/actors. And they often display a common topological feature-community structure. Discovering the latent communities therein is a useful way to infer some important functions.

In general, a community should be thought of a set of nodes that have more and/or better-connected edges between its members than between its members and the remainder of the network. The existing community definitions in the literature can be roughly divided into three categories, one is global-based [5]–[7], the other is based on the node-similarity [8]–[11], and the third is local-based [12]–[14].

1) The global-based definitions consider the graph as a whole, and they follow the assumption that a graph has community structure if it is different from a random graph, i.e., null model.
2) The node-similarity-based definitions are based on the assumption that communities are groups of nodes similar to each other. Traditional methods, including hierarchical [8], partitional [9], and spectral clustering [10], [11], are based on this kind of definition.
3) The local-based definitions compare the internal and external cohesion of a subgraph. The first recipe of this kind is Luccio Sami (LS)-set [12], which stems from social network analysis. The condition of LS-set is quite strict and can be relaxed into the so-called weak definition of community [13]. Some other local-based definitions can be found in [15]–[17].

Since existing global-based and the node-similarity-based community detection approaches require clear pictures of the entire graph structure, they are often described as global community detection (in short as GCD henceforth). In those GCD methods, the networks concerned are centralized (i.e., they are processed in a centralized manner and with a global control) instead of being distributed. However, in the real world, many applications involve distributed networks, in which resources and controls are often decentralized. As they are based on the structure-oriented view, the characteristics and effects of social actors are neglected.

To consider the effects of actor characteristics, the actor-structure crossing view has been recently introduced into community detection, especially applications of the local community detection (LCD) or autonomy-oriented computing (AOC) [18]–[20]. The networks concerned by LCD/AOC methods can be distributed, and each actor in the networks is modeled as an agent who acts autonomously to find

Manuscript received October 28, 2014; revised February 5, 2015; accepted March 31, 2015. Date of publication June 16, 2015; date of current version March 15, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant 71372188 and Grant 61472079, in part by the National Center for International Joint Research on E-Business Information Processing under Grant 2013B01035, in part by the National Key Technologies Research and Development Program of China under Grant 2013BAH16F01, in part by Industry Projects in Jiangsu S&T Pillar Program under Grant BE2014141, in part by the Priority Academic Program Development of Jiangsu Higher Education Institutions, and in part by the Key/Surface Project of Natural Science Research in Jiangsu Provincial Colleges and Universities under Grant 12KJA520001, Grant 14KJA520001, and Grant 14KJB520015. This paper was recommended by Associate Editor B. Chaib-draa. *(Corresponding author: Zhiang Wu.)*

Z. Bu, Z. Wu, and J. Cao are with the Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, Nanjing 210003, China (e-mail: zawuster@gmail.com).

Y. Jiang is with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China, and also with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2015.2419263

its local community. In [20], the community evolution has been introduced into the proposed incremental AOC-based method (AOC-i), in which the new community structure can be quickly derived based on the previous one and the incremental network update. Although existing LCD/AOC methods have already achieved significant success, further study is still needed on seeking a nice balance between the high efficiency of local search models and the high accuracy of detected communities. Therefore, we proposed a novel AOC method from the multiagent perspective for detecting community structures in the distributed environment, in which three key problems are carefully concerned.

1) How to model the real-life networks in the distributed environment?
2) How to effectively and accurately detect the local community starting from an arbitrary distributed node?
3) How to monitor the influence on a given local community and incrementally compute its current structure?

Targeting at effectively solving these above problems, in this paper, we propose a fully distributed system guided by AOC methodology for modeling decentralized networks. In this system, every node is assigned an autonomous agent for LCD. For the LCD task of each agent, a heuristic algorithm named AOCCM is presented, and then its incremental version called AOCCM-i is designed for handling community evolution. More specifically, our main contributions are summarized in threefold as follows.

1) We present a novel modularity gain criterion, based on which a heuristic algorithm named AOCCM is designed for the LCD task of each agent. The proposed method is able to start from an arbitrary node in a distributed network, and repeats two iterative steps (`Update` and `Join`) until the local community has reached its convergent status or the agent's clock time is over.
2) We expand AOCCM to a more efficient incremental method (AOCCM-i) for mining communities from dynamic and distributed networks. The process of AOCCM-i is an iterative process consisting of a series of discrete evolutionary cycles. In each cycle, the new objective can be incremental updated based on the previous results and the dynamic changes of the network.
3) Based on the local communities detected by AOCCM or AOCCM-i, we further propose two global versions for nonoverlapping and overlapping community detections. Thorough experiments on real-life networks demonstrate that the proposed methods can keep a nice balance between the high accuracy and short running time.

The remainder of this paper is organized as follows. Section II presents the related work about AOC and dynamic network mining. In Section III, we give a problem definition of distributed community mining and the basic ideas behind our method. Section IV introduces the AOC-based method for community mining. In Section V, we validate the proposed methods using some real-world networks, and examine its performances in detail. We further present an incremental AOC method for dynamic network mining in Section VI, and finally the conclusion of this paper is given in Section VII.

II. Related Work

Here, we discuss related work from two areas: 1) AOC and 2) dynamic network mining.

A. Autonomy-Oriented Computing

Early work in LCD can be adopted to AOC, which can be classified into two main categories, namely: 1) degree-based methods and 2) similarity-based methods.

Degree-based methods evaluate the local community quality by investigating nodesdegrees. Some naive solutions, such as l-shell search algorithm [21], discovery-then-examination approach [15], and outwardness-based method [16], only consider the number of edges inside and outside a local community. Clauset [22] defined local modularity by considering the boundary points of a subgraph, and proposes a greedy algorithm on optimizing this measure. Similarly, Luo *et al.* [17] presented another measurement as the ratio of the internal degree and external degree of a subgraph. Both measurements can achieve high recall but suffer from low precision due to including many outliers [15].

Similarity-based methods utilize similarities between nodes to help evaluate the local community quality. Local tightness expansion (LTE) algorithm [23] is a representative of similarity-based methods, using a well-designed metric for local community quality known as tightness. There are a few alternative similarity-based metrics such as vertex similarity probability model [24] and relation strength similarity [25] that can also help evaluate the local community quality, although they are not originally designed for LCD.

Some multiagent technologies have been introduced into community detection [26], [27], in which, each actor in the networks is modeled as an agent and acts autonomously to find its community. For example, Chen *et al.* [28] formulated the agents' utility by the combination of a gain function and a loss function and make agents select communities by a game-theoretic framework to achieve an equilibrium for interpreting a community structure. To consider in the distributed experiment, Yang *et al.* [20] utilized reactive agents to make distributed and incremental mining of communities based only on their local views and interactions.

Our new autonomy-oriented computing-based method for community mining (AOCCM) is also based on the multiagent perspective, in which, the local search model of each agent is also an extension of the similarity model. However, in comparison to the above approaches which calculate the quantitative metrics for every node in the neighbor sets, the structural similarity of each pair of nodes in AOCCM is calculated only once. By introducing the notion of modularity gain, which is seen as a quantified criterion to decide whether the candidate node can be added into the local community or not, the effectiveness of AOCCM is very high.

B. Dynamic Network Mining

Recently, finding communities in dynamic networks has gained more and more attention. A family of events on both communities and individuals have been introduced in [29] to characterize evolution of communities. An evolutionary

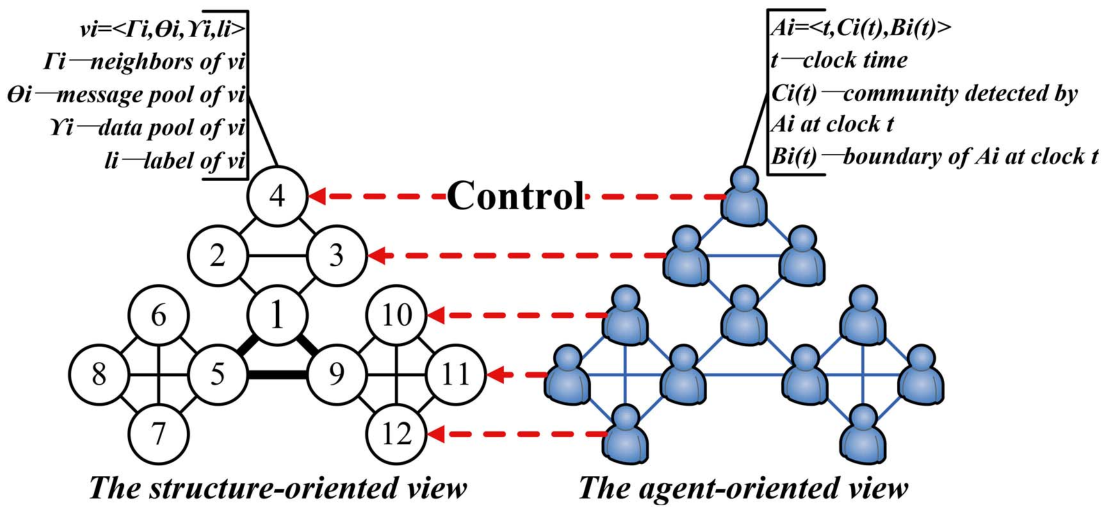


Fig. 1. Environment of the AOC system.

version of the spectral clustering algorithms has been firstly proposed by Chi *et al*. [30], in which the graph cut is used as a metric for measuring community structures and community evolutions. Their work has been further expanded by Lin *et al*. [31], in which, a graph-factorization clustering algorithm named FacetNet has been proposed to analyze dynamic networks. The above mentioned studies often adopted a two-step approach where first static analysis is applied to the snapshots of the social network at different time steps, and then community evolutions are introduced afterwards to interpret the change of communities over time. As they overlooked the old community structures as obtained in the previous snapshot, this strategy of recalculating is not efficient. In the framework of multiagent system (MAS), Yang *et al*. [20] introduced an AOC-i, in which the new community structure can be quickly derived based on the incremental change and the old community structure as obtained in the previous cycle. The proposed incremental computing method in this paper is also AOC-based. Compared with AOC-i, AOCCM-i does not require any user-defined parameters, and the network updating can be more arbitrarily, e.g., gradually inputting any number of new edges. We will also prove in the experiment that the average convergent speed of AOCCM-i is much faster than AOC-is.

III. Problem Definition and Preliminaries

As the nodes and the connectivity information in a distributed network are located at different positions. Community mining in the distributed network is aimed to find all the underlying communities only based on the local views and the cooperations among the nodes. One possible solution to solve this problem is to provide the clear picture of the network to an administrative agent, on which a GCD algorithm is applied to discover the underlying communities. However, this strategy will have several inherent limitations when confronting large-scale or highly-dynamic networks. To address this issues, the methodology of AOC is presented in this paper, which is derived from [18] and [19]. Generally speaking, an AOC system can be viewed a MAS, in which agents interact with each other according to some predefined transfer protocols to detect the local community.

A. Environment of AOC System

The presented AOC system is defined as a distributed network in which nodes and the connectivity information are decentralized, as shown in Fig. 1. Let $\mathcal{G} = (V, E)$ be a given weighted distributed network, where V is the set of nodes ($|V| = n$), E is the set of edges ($E = \{e_{ij}\}$) that connect the nodes in V, and w_{ij} is the weight on edge e_{ij}. In the AOC system, each node is defined as a tuple $<\Gamma_i, \Theta_i, \Upsilon_i, l_i>$, where the components denote the identifiers of its adjacent neighbors (Γ_i), the message pool on the node (Θ_i), the data pool on the node (Υ_i), and the community label of the node (l_i), respectively. For each node v_i of the distributed network, there is one autonomous agent A_i residing on it, which is characterized by three characteristics, including `clock`, `community`, and `boundary`, denoted as $<t, \mathcal{C}_i(t), \mathcal{B}_i(t)>$. The attribute t denotes the clock maintained by agent A_i, who will adjust its clock by $t \leftarrow t + 1$ after each iteration. When its clock reaches the final time T, A_i will become inactive. $\mathcal{C}_i(t)$ denotes the local community detected by agent A_i at time t. Initially, each agent only includes its appurtenant node in the community, e.g., $\mathcal{C}_i(0) = \{v_i\}$. Then, it starts to enlarge its local community based on the information it gathers from its environment, until its clock reaches the final time T or the community cannot be changed any more. In other words, we have $\mathbb{C}_i = \mathcal{C}_i(T)$. The `boundary` of each agent is closely associated with its local community, which can be defined as $\mathcal{B}_i(t) = \{v_j | v_j \notin \mathcal{C}_i(t), v_k \in \mathcal{C}_i(t), <v_j, v_k, w_{jk}> \in E\}$. At the beginning, $\mathcal{B}_i(0)$ is set to be the adjacent neighbors (Γ_i) of node v_i. During the community updating of agent A_i, nodes belonging to its community will be added from the boundary one by one. In contrast, those nodes not belonging to the community will be gradually excluded out of its boundary. The above mentioned notations are summarized in Table I.

Remark: The identifier of node v_i, e.g., i, is also used to denote the address of the location, in which the node situates. After knowing the identifier of v_i, A_i can send messages to it. In the AOC system, the message pool on each node (Θ_i) stores the messages from others agents, and the data pool Υ_i stores the structural similarity between node v_i and its adjacent nodes.

TABLE I
NOTATIONS OF THE AOC SYSTEM

Symbol	Description
Γ_i	the identifiers of adjacent neighbors of v_i
Θ_i	the message pool on v_i, which stores the messages from others agents
Υ_i	the data pool on v_i, which the structural similarity between v_i and its adjacent nodes
l_i	the community label of v_i
t	the clock maintained by agent A_i
$\mathcal{C}_i(t)$	the local community detected by agent A_i at time t
$\mathcal{B}_i(t)$	the boundary area of agent A_i at time t

Initially, both Θ_i and Υ_i are empty. Without loss of generality, let us consider agent A_i. Before conducting local community mining, A_i sends messages to the neighbors of node v_i ($\Theta_{j,v_j\in\Gamma_i} \leftarrow A_i$.message); then $A_{j,v_j\in\Gamma_i}$ return $\Gamma_{j,v_j\in\Gamma_i}$ to calculate the structural similarities between v_i and its neighbors; finally, A_i stores all the structure similarity values in the data pool of node v_i ($\Upsilon_i \leftarrow \{s_{ij}|v_j \in \Gamma_i\}$). Therefore, the structural similarity of each pair of nodes in AOC system is calculated only once.

B. Objective of AOC System

The objective of the AOC system can be achieved if the all agents autonomously achieve their respective objectives

$$\mathbb{C} = \{\mathbb{C}_1, \ldots, \mathbb{C}_n\} \tag{1}$$

where $\mathbb{C}_i$ is the stable local community of agent A_i, and $\cup_{1\leq i\leq n}\mathbb{C}_i = V$. The objective of agent A_i is to find a local community structure starting from its appurtenant node v_i, only based on its local view [e.g., $\mathcal{C}_i(t) \cup \mathcal{B}_i(t)$] and interactions with other agents.

Generally, a community is measured by a specific property of the nodes within it. For this task, different community measurements have been proposed [13], [16], [17], [22] in recent years. In this paper, we adopt a structural similarity measure from the cosine similarity function [23] to effectively denotes the local connectivity density of any two adjacent nodes in a distributed network. Here, we first formalize some notions of the local community.

Definition 1 (Structural Similarity): Given a distributed network $\mathcal{G} = (V, E)$, the structural similarity between two adjacent nodes v_i and v_j is defined as

$$s_{ij} = \frac{\sum_{v_k\in\Gamma_i\cap\Gamma_j} w_{ik}w_{jk}}{\sum_{v_k\in\Gamma_i} w_{ik}^2 \sum_{v_k\in\Gamma_j} w_{jk}^2}. \tag{2}$$

When we consider an unweighted network, the weight w_{ij} of any edge can be set to 1 and the equation above can be transformed to

$$s_{ij} = \frac{|\Gamma_i \cap \Gamma_j|}{|\Gamma_i||\Gamma_j|} \tag{3}$$

which corresponds to the edge-clustering coefficient introduced in [13].

Definition 2 (Internal Similarity and External Similarity): By employing the structural similarity, we introduce internal and external similarities of the community $\mathcal{C}_i(t)$ as follows:

$$S_{\text{in}}(\mathcal{C}_i(t)) = \sum_{v_j,v_k\in\mathcal{C}_i(t),<v_j,v_k,w_{jk}>\in E} s_{jk} \tag{4}$$

$$S_{\text{out}}(\mathcal{C}_i(t)) = \sum_{v_j\in\mathcal{C}_i(t),v_k\in\mathcal{C}_i^c(t),<v_j,v_k,w_{jk}>\in E} s_{jk} \tag{5}$$

where $\mathcal{C}_i^c(t)$ is the complement of $\mathcal{C}_i(t)$.

The criterion, agent A_i used to find the local community containing the appurtenant node v_i, is derived from [32], which finds a community with a large number of edges within itself and a small number of edges to the rest of the network.

Definition 3 (Local Modularity): The local modularity of the community $\mathcal{C}_i(t)$, denoted as $W(\mathcal{C}_i(t))$, is given as follows:

$$W(\mathcal{C}_i(t)) = \frac{I(\mathcal{C}_i(t))}{|\mathcal{C}_i(t)|^2} - \frac{O(\mathcal{C}_i(t))}{|\mathcal{C}_i(t)||\mathcal{C}_i^c(t)|} \tag{6}$$

where $I(\mathcal{C}_i(t)) = \sum_{v_i,v_j\in\mathcal{C}_i(t)} A_{ij}$, $O(\mathcal{C}_i(t)) = \sum_{v_i\in\mathcal{C}_i(t),v_j\in\mathcal{C}_i^c(t)} A_{ij}$, $A = [A_{ij}]$ is an $n \times n$ adjacency matrix of the distributed network $\mathcal{G}$.

Based on the definition of local modularity, we have the following theorem.

Theorem 1: The local modularity value of the community $\mathcal{C}_i(t)$ will increase when $\mathcal{C}_i(t)$ has high-intracluster density and low intercluster density.

Proof: The term $I(\mathcal{C}_i(t))$ is twice the number of the edges within $\mathcal{C}_i(t)$, and $O(\mathcal{C}_i(t))$ represents the number of edges between $\mathcal{C}_i(t)$ and the rest of the network. Each term is normalized by the total number of possible edges in each case. Note that, we normalize the first term by $|\mathcal{C}_i(t)|^2$ rather than $|\mathcal{C}_i(t)|(|\mathcal{C}_i(t)| - 1)$ in order to conveniently derive the modularity gain discussed below, but in practice this makes little difference. Subject to this small difference, the local modularity can be described as the intracluster density minus the intercluster density. Thus, the proof completes. ■

Based on Definition 2 and Theorem 1, we have the following corollary.

Corollary 1: The local modularity value of the community $\mathcal{C}_i(t)$ will increase when $\mathcal{C}_i(t)$ has high-internal similarity and low external similarity.

Proof: A high value of $S_{\text{in}}(\mathcal{C}_i(t))$ reveals a large number of common neighbors of any adjacent node pair in $\mathcal{C}_i(t)$, resulting in a high value of intracluster density. While, a low value of $S_{\text{out}}(\mathcal{C}_i(t))$ reveals a small number of common neighbors of any adjacent node pair between $\mathcal{C}_i(t)$ and $\mathcal{C}_i^c(t)$, resulting in a low value of intracluster density. ■

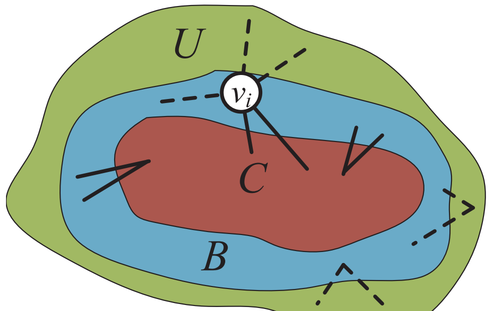


Fig. 2. $\hat{W}$ variant when a node v_j joins $\mathcal{C}_i(t)$.

In Definition 2, as the second term will be made negligible by the large $|\mathcal{C}_i^c(t)|$, a very small community can give a high value of $W(\mathcal{C}_i(t))$. We further make an adjustment in the spirit of the ratio cut and maximize the following criterion:

$$\hat{W}(\mathcal{C}_i(t)) = |\mathcal{C}_i(t)||\mathcal{C}_i^c(t)|\left(\frac{I(\mathcal{C}_i(t))}{|\mathcal{C}_i(t)|^2} - \frac{O(\mathcal{C}_i(t))}{|\mathcal{C}_i(t)||\mathcal{C}_i^c(t)|}\right) \quad (7)$$

where the factor $|\mathcal{C}_i(t)||\mathcal{C}_i^c(t)|$ penalizes very small and very large communities and produces more balanced solutions.

Suppose at clock t, A_i explores the adjacent nodes in the boundary area $\mathcal{B}_i(t)$, as shown in Fig. 2. It distinguishes three types of links: those internal to the community $\mathcal{C}_i(t)(L)$, between $\mathcal{C}_i(t)$ and the node $v_j(L_{\text{in}})$, between $\mathcal{C}_i(t)$ and others nodes in $\mathcal{B}_i(t)(L_{\text{out}})$. To simplify the calculations, we express the number of external links in terms of L and k_j (the degree of node v_j), so $L_{\text{in}} = a_1L = a_2k_j$, $L_{\text{out}} = b_1L$, with $b_1 \geq 0$, $a_1 \geq 1/L$, $a_2 \geq 1/k_j$ [since any v_j in $\mathcal{B}_i(t)$ at least has one neighbor in $\mathcal{C}_i(t)$]. So, the value of $\hat{W}$ for the current community can be written as

$$\hat{W}(\mathcal{C}_i(t)) = \frac{n - |\mathcal{C}_i(t)|}{|\mathcal{C}_i(t)|}2L - (a_1 + b_1)L. \quad (8)$$

Then, the variant $\hat{W}$ of the community $\mathcal{C}_i(t) \cup v_j$ becomes

$$\hat{W}\big(\mathcal{C}_i(t) \cup v_j\big) = \frac{n - |\mathcal{C}_i(t)| - 1}{|\mathcal{C}_i(t)| + 1}2L(1 + a_1) - \big(b_1L + k_j - a_2k_j\big). \quad (9)$$

So, we define the modularity gain in the following.

Definition 4 (Modularity Gain): The modularity gain for the community $\mathcal{C}_i(t)$ adopting a neighbor node v_j can be denoted as

$$\begin{aligned}\Delta\hat{W}_{\mathcal{C}_i(t)}(v_j) &= \hat{W}\big(\mathcal{C}_i(t) \cup v_j\big) - \hat{W}(\mathcal{C}_i(t))\\ &= \frac{n - |\mathcal{C}_i(t)| - 1}{|\mathcal{C}_i(t)| + 1}2L(1 + a_1) - \big(b_1L + k_j - a_2k_j\big)\\ &\quad - \left(\frac{n - |\mathcal{C}_i(t)|}{|\mathcal{C}_i(t)|}2L - (a_1 + b_1)L\right)\\ &= 2n\frac{a_2k_j|\mathcal{C}_i(t)| - L}{|\mathcal{C}_i(t)|(|\mathcal{C}_i(t)| + 1)} - k_j. \end{aligned} \quad (10)$$

It means that if a small node in terms of degree links many nodes in community $\mathcal{C}_i(t)$, adopting it may increase the local modularity of $\mathcal{C}_i(t)$. Therefore, $\Delta\hat{W}_{\mathcal{C}_i(t)}(v_j)$ can be utilized as a criterion for A_i to determine whether the candidate node v_j should be included in the community $\mathcal{C}_i(t+1)$ or not.

Algorithm 1 Life-Cycle of Agent A_i (AOCCM(A_i))

```
1: /*Initialization phase*/
2: $t \leftarrow 0$;
3: $\mathcal{C}_i(0) \leftarrow \{v_i\}$;
4: $\mathcal{B}_i(0) \leftarrow \{v_j | v_j \in \Gamma_i\}$;
5: /*Active phase*/
6: while $t < T$ do
7:     $v_j^* = \arg\ max_{v_j \in \mathcal{B}_i(t)} \sum_{v_j \in \mathcal{C}_i(t)} s_{ij}$;
8:     if $\Delta\hat{W}_{\mathcal{C}_i(t)}(v_j^*) > 0$ then
9:         $\mathcal{B}_i(t+1) \leftarrow \mathcal{B}_i(t) \cup \{v_k | v_k \in \Gamma_{j^*}, v_k \notin \mathcal{C}_i(t)\} - \{v_j^*\}$;
10:        $\mathcal{C}_i(t+1) \leftarrow \mathcal{C}_i(t) \cup \{v_j^*\}$;
11:    else
12:        $\mathcal{B}_i(t+1) \leftarrow \mathcal{B}_i(t) - \{v_j^*\}$;
13:    end if
14:    $t \leftarrow t + 1$;
15:    if $\mathcal{B}_i(t) = \emptyset$ then
16:        break;
17:    end if
18: end while
19: /*Inactive phase*/
20: $\mathbb{C}_i = \mathcal{C}_i(t)$;
21: $l_i \leftarrow \arg\ max_{v_j \in \mathbb{C}_i} k_j$;
```

IV. AOC-Based Method for Community Mining

In this section, we propose an AOC-based method for community mining (in short as AOCCM henceforth). First, we introduce the basic idea of AOCCM and then present algorithmic details including the complexity analysis. Second, we introduce how to use AOCCM to detect the global nonoverlapping and overlapping community structures.

In the AOC system, each agent, e.g., A_i, starts from its appurtenant node v_i to find the densely connected local community. A_i works with two iterative steps: 1) `Update` step and 2) `Join` step. First, the appurtenant node v_i is added into the local community, e.g., $\mathcal{C}_i(0) = \{v_i\}$. In the `Update` step, A_i refreshes the boundary area $\mathcal{B}_i(t)$, and calculate the structural similarities between nodes in the community $\mathcal{C}_i(t)$ and their neighbor nodes in $\mathcal{B}_i(t)$. In the `Join` step, A_i tries to absorb a node in $\mathcal{B}_i(t)$, e.g., v_j^*, having highest structural similarity with nodes in $\mathcal{C}_i(t)$ into the local community. If $\Delta\hat{W}_{\mathcal{C}_i(t)}(v_j^*) > 0$, then the node v_j^* will be inserted into $\mathcal{C}_i(t+1)$. Otherwise, it will be removed from $\mathcal{B}_i(t+1)$ and other nodes will be considered in the descending order of the structural similarity. The two procedures above will be repeated by A_i in turn until its clock reaches the final time T or its boundary is empty. Then, the whole community $\mathbb{C}_i$ is discovered. A_i further selects the node with maximum degree in $\mathbb{C}_i$ as the core node, the identifier of which can be seen as the label of detected community. The life-cycle of agent A_i on node v_i is given in the following.

Remark: Unlike existing methods [16], [17], [22], which calculate the quantitative metrics for each node in $\mathcal{B}$ and select the node who produces the greatest increment of the metric to join $\mathcal{C}$, each agent A_i in the AOC system picks the neighbor node with the largest structure similarity as the candidate node v_j^* and calculate $\Delta\hat{W}_{\mathcal{C}_i(t)}(v_j^*)$ to determine whether it should be added into $\mathcal{C}_i(t+1)$ or not. The structural similarity reflects the local connectivity density of the network. The larger the similarity between a node inside $\mathcal{C}_i(t)$ and a node outside it, the more common neighbors the two nodes share, and the more probability they are at the same community. So the execution of AOCCM on each agent is accelerated and the accuracy remains high.

A. Complexity Analysis

The running time of AOCCM on agent A_i is mainly consumed in line 7 of Algorithm 1, which is selecting the neighbor node with the largest structure similarity. Agent A_i can implement it using a binary Fibonacci heap H_i [23], which takes two steps.

1) `Extract` (extract the maximum element from H_i). As each `Extract` operation of H_i takes $O(\log n_i')$ time and the body of the while loop is executed n' times, the total time for all `Extract` steps is $O(n' \log n_i')$, where n_i' is the number of nodes inferred (nodes in $\mathbb{C}_i \cup \mathbb{B}_i$).
2) `Update` [for each node in current $\mathcal{B}_i(t)$, A_i updates its sum of structure similarities with nodes in $\mathcal{C}_i(t)$]. First, the sum of structure similarities with nodes in $\mathcal{C}_i(t)$ for each node $v_j \in \mathcal{B}_i(t)$ should be computed, which can be completed in $O(k_i')$ time, where k_i' is the mean degree of inferred nodes. For nodes which are not in H_i, A_i inserts them into H_i in $O(1)$ time; otherwise, it takes $O(1)$ time to make an increase-key operation. As the above steps are executed $O(m_i')$ times, where m_i' is the number of edges in $\mathbb{C}_i \cup \mathbb{B}_i$. Therefore, the total time of the `Update` steps is $O(m_i'k_i')$. Adding all together, the total time complexity is $O(m_i'k_i' + n_i' \log n_i')$ for AOCCM on agent A_i.

B. Nonoverlapping Community Detection

Nonoverlapping community detection aims to find a good K-way partition $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_K\}$, where $\mathcal{P}_k$ is the kth community, in which $l_i = l_j\ \forall v_i, v_j \in \mathcal{P}_k$, and $\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_K \subseteq V$, $\mathcal{P}_k \cap \mathcal{P}_{k'} = \emptyset\ \forall\ k \neq k'$. K is automatically determined by results of each AOCCM(A_i). Our assumption is that similar adjacent agents will return analogous community structures, in which the core nodes are almost unanimous. Therefore, if A_i detects the same community label, their appurtenant nodes are likely to be in the same community. The process of AOCCM expansion algorithm for nonoverlapping (in short as AOCCMnO henceforth) is given as in Algorithm 2, where $L = \{l_i | i = 1, \ldots, n\}$ is the label list of nodes in the distributed network. AOCCMnO could be completed in $O(m^*k^* + n^* \log n^*)$ time, where n^*, m^* are the number of nodes and edges in the largest $\mathbb{C}_i \cup \mathbb{B}_i$, and k^* is the mean degree of inferred nodes in it.

C. Overlapping Community Detection

While, for an overlapping partition, overlapping communities can be represented as a membership matrix $\mathbf{U} = [u_{i,k}]$,

Algorithm 2 AOCCMnO($\mathcal{G}$)
```
1: for $i = 1$; $i <= n$; $i++$ do
2:     $[\mathbb{C}_i, l_i] \leftarrow$ AOCCM($A_i$); //Parallel Computing
3: end for
4: $\mathcal{L} = unique(L)$; // $\mathcal{L} = \{l_k' | 1 \le k \le K\}$
5: $K = Length(\mathcal{L})$;
6: for $k = 1$; $k <= K$; $k++$ do
7:     $\mathcal{P}_k = \{v_i | \forall l_i = l_k'\}$;
8: end for
```

Algorithm 3 AOCCMO($\mathcal{G}$)
```
1: for $s = 1$; $s <= n$; $s++$ do
2:     $[\mathbb{C}_i, l_i] \leftarrow$ AOCCM($A_i$); //Parallel Computing
3: end for
4: $\mathcal{L} = unique(L)$; // $\mathcal{L} = \{l_k' | 1 \le k \le K\}$
5: $K = Length(\mathcal{L})$;
6: for $i = 1$; $i <= n$; $i++$ do
7:     for $k = 1$; $i <= K$; $k++$ do
8:         $u_{i,k} = \frac{\sum_{j=1,\cdots,n \wedge l_j = l_k'} \delta(v_i, \mathbb{C}_j)}{\sum_{j=1,\cdots,n} \delta(v_i, \mathbb{C}_j)}$;
9:     end for
10: end for
```

$i = 1, \ldots, n,\ k = 1, \ldots, K$, where $0 \le u_{i,k} \le 1$ denotes the ratio of membership that node v_i belongs to $\mathcal{P}_k$. If node i belongs to only one community, $u_{i,k} = 1$, and it clearly follows that $\sum_{k=1}^{K} u_{i,k} = 1$ for all $1 \le i \le n$. With the detected communities of AOCCM(A_i), $u_{i,k}$ can be calculated as follows:

$$u_{i,k} = \frac{\sum_{j=1,\ldots,n \wedge l_j = l_k'} \delta\left(v_i, \mathbb{C}_j\right)}{\sum_{j=1,\ldots,n} \delta\left(v_i, \mathbb{C}_j\right)} \tag{11}$$

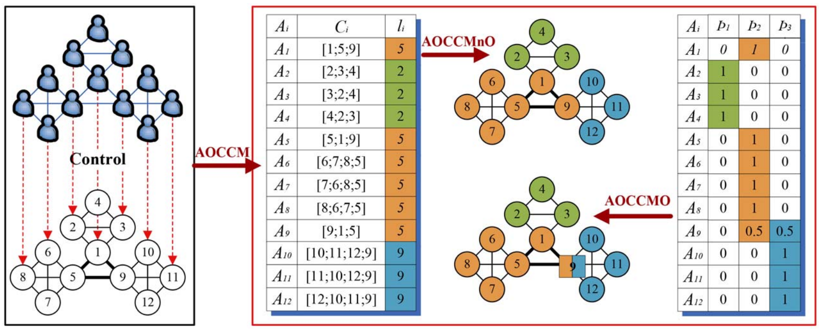
$$\delta(v_i, \mathbb{C}_j) = \begin{cases} 1 & \text{if } v_i \in \mathbb{C}_j \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

The process of AOCCM expansion algorithm for overlapping (in short as AOCCMO henceforth) is given as follows. The running time of AOCCMO is mainly consumed in lines 6–9 of Algorithm 3, which is calculating the membership that node i belongs to $\mathcal{P}_k$. The total time of those steps is $O(nK)$. Adding the local community extraction steps, the total time complexity is $O(m^*k^* + n^* \log n^* + nK)$ for AOCCMO.

In the following example, we show how to detect communities hidden in the distributed network (see Fig. 3) using the AOCCM algorithm. The original distributed network includes 12 nodes and 20 edges, the weight on the edges between nodes v_1, v_5, and v_9 is 3, and 1 on other edges. We will focus on a single agent and observe its local community at different time step during its entire life-cycle. In this case, we choose agent A_1, and the final time T is set to be 12.

First, A_1 initializes itself. The time of its clock is set to 0, and its local community and boundary area at time 0 are set to $\mathcal{C}_1(0) = \{v_1\}$, $\mathcal{B}_1(0) = \{v_2, v_3, v_5, v_9\}$, respectively.

Then, A_1 starts its active phase. After calculating the structure similarities in step 7, it selects node v_5 from $\mathcal{B}_1(0)$, which has the highest structural similarity with the nodes in $\mathcal{C}_1(0)$. In step 8, $\Delta\hat{W}_{\mathcal{C}_1(0)}(v_5)$ is calculated to determine whether node v_5 should be added into $\mathcal{C}_1(1)$ or not.



A_i	C_i	l_i
A_1	[1;5;9]	5
A_2	[2;3;4]	2
A_3	[3;2;4]	2
A_4	[4;2;3]	2
A_5	[5;1;9]	5
A_6	[6;7;8;5]	5
A_7	[7;6;8;5]	5
A_8	[8;6;7;5]	5
A_9	[9;1;5]	5
A_{10}	[10;11;12;9]	9
A_{11}	[11;10;12;9]	9
A_{12}	[12;10;11;9]	9

A_i	p_1	p_2	p_3
A_1	0	1	0
A_2	1	0	0
A_3	1	0	0
A_4	1	0	0
A_5	0	1	0
A_6	0	1	0
A_7	0	1	0
A_8	0	1	0
A_9	0	0.5	0.5
A_{10}	0	0	1
A_{11}	0	0	1
A_{12}	0	0	1

Fig. 3. Illustrative example for AOCCM, AOCCMnO, and AOCCMO.

As $\Delta \hat{W}_{\mathcal{C}_1(0)}(v_5) > 0$, node v_5 is added into $\mathcal{C}_1(1)$, and $\mathcal{B}_1(1)$ is refreshed to be $\{v_2, v_3, v_6, v_7, v_8, v_9\}$. Next, in step 14, A_1 updates its clock, the time of its clock is 1. In step 15, A_1 checks whether it has reached a convergent status by observing its boundary area. As $\mathcal{B}_1(1)$ is not empty and the current time is less than T, A_1 goes to step 7 and starts a new iteration. After finishing the second iteration, the local community and boundary area of A_1 and time 2 becomes: $\mathcal{C}_1(2) = \{v_1, v_5, v_9\}$, $\mathcal{B}_1(2) = \{v_2, v_3, v_6, v_7, v_8, v_{10}, v_{11}, v_{12}\}$, respectively. As $\mathcal{B}_1(2)$ is not empty and the current time is still less than T. A_1 goes to step 7 and starts the third iteration, after which the local community of A_1 at time 3 keeps unchanged while its boundary area $\mathcal{B}_1(3)$ is shrank to be $\{v_3, v_6, v_7, v_8, v_{10}, v_{11}, v_{12}\}$. Once again, A_1 has not reached its convergent status and continues the iterative process until its clock time reaches 10. In this case, the boundary area $\mathcal{B}_1(10)$ is empty, so it quits the cycle of community updating.

Finally, A_1 records its convergent local community as $\mathbb{C}_1 = \mathcal{C}_1(10) = \{v_1, v_5, v_9\}$, and further selects the identifier of the node with maximum degree (e.g., v_5) as the label of detected community. The entire life-cycle of agent A_1 is completed and A_1 becomes inactive.

Similarly, we can observe the activities of other agents. When we acquire the attributive tags of all 12 nodes, the global nonoverlapping community structure has been detected. Finally, all local communities are further assembled to be a membership matrix $\mathbf{U} = [u_{i,k}]$, which promulgates the global overlapping community structure.

V. Evaluation of the AOC System

Four real-world undirected networks: 1) `Karate`; 2) `National Collegiate Athletic Association (NCAA)`; 3) `Facebook`; and 4) `pretty good privacy (PGP)` are used to validate the AOC system. Although these networks are given as centralized representations, for the purpose of testing our distributed method, here we treat them as distributed networks, i.e., we consider that their nodes and links are distributed (e.g., over different sources, or geographical locations). Some characteristics of these networks are shown in Table II, where $|V|$ and $|E|$ indicate the numbers of nodes and edges respectively in the network, and $\overline{k}$ indicates the average degree. `Karate` is a well known social network that describes the friendship relations between members of a karate club. `NCAA` is a representation of the schedule of American Division I college football games. Vertices in the network represent teams, which are divided into eleven communities (or conferences) and five independent teams. Edges represent regular season games between the two teams they connect. `Facebook` has been anonymized by replacing the Facebook-internal ids for each user with a new value. Each edge tells whether two users have the same political affiliations. `PGP` is a large scale social network, where each node represents a peer and each tie points out that one peer trusts the other.

TABLE II
Real-World Networks for Experiments

Network	$\|V\|$	$\|E\|$	$\overline{k}$
`Karate`	34	78	4.59
`NCAA`	115	616	10.71
`Facebook`	4,039	88,234	43.69
`PGP`	10,680	24,340	4.56

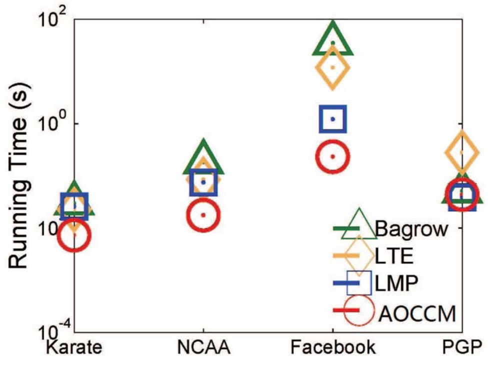


Fig. 4. Comparison on efficiency.

A. Performance of AOCCM

1) Effectiveness: To test the effectiveness of our approach, the results of AOCCM are compared with the ground truth communities of each network. To be special, let $\mathbb{T}_i$ be the ground truth community including the node v_i, we can compare $\mathbb{T}_i$ and $\mathbb{C}_i$ in the framework of precision, recall, and f-measure (PRF) to assess our results. A higher value of precision (P) indicates fewer wrong classifications, while a higher value of recall (R) indicates less false negatives. It is common to use the harmonic mean of both measurements, called F-measure, which weighs precision and recall equally

TABLE III
ACCURACY COMPARISON ON REAL-WORLD NETWORKS

Community Comm.		AOCCM			LWP			ELC			LTE		
	size	P	R	F1	P	R	F1	P	R	F1	P	R	F1
`Karate`-A	16	1.00	0.58	**0.73**	0.94	0.49	0.64	0.93	0.49	0.64	1.00	0.49	0.66
`Karate`-B	18	0.97	0.47	0.63	0.97	0.44	0.61	0.89	0.48	0.63	1.00	0.57	**0.73**
`NCAA`-AC	9	1.00	1.00	**1.00**	0.70	0.48	0.57	0.68	0.56	0.61	1.00	1.00	**1.00**
`NCAA`-BE	8	1.00	1.00	**1.00**	0.48	0.47	0.48	0.51	0.67	0.58	0.80	1.00	0.89
`NCAA`-Ten	11	1.00	1.00	**1.00**	0.33	0.26	0.29	0.17	0.21	0.19	1.00	1.00	**1.00**
`NCAA`-SE	12	1.00	1.00	**1.00**	0.81	0.55	0.65	0.83	0.85	0.84	1.00	1.00	**1.00**
`NCAA`-PT	10	0.91	0.82	**0.86**	0.68	0.58	0.62	0.68	0.73	0.70	0.91	0.82	**0.86**
`NCAA`-Others	5	0.12	0.24	0.16	0.21	0.40	**0.27**	0.14	0.52	0.22	0.19	0.32	0.24
`NCAA`-MA	13	1.00	0.50	0.67	0.78	0.48	0.60	0.81	0.78	**0.79**	0.86	0.50	0.64
`NCAA`-MV	8	1.00	1.00	**1.00**	0.76	0.70	0.73	0.67	0.70	0.69	1.00	1.00	**1.00**
`NCAA`-WA	10	1.00	1.00	**1.00**	0.65	0.45	0.53	0.67	0.60	0.63	1.00	1.00	**1.00**
`NCAA`-Twelve	12	1.00	1.00	**1.00**	0.67	0.40	0.52	0.61	0.56	0.35	1.00	1.00	**1.00**
`NCAA`-SB	7	0.64	0.51	**0.57**	0.49	0.61	0.54	0.23	0.69	0.35	0.64	0.51	0.56
`NCAA`-USA	10	0.74	0.66	**0.70**	0.41	0.32	0.36	0.25	0.23	0.24	0.74	0.66	**0.70**
`Facebook`-1	341	1.00	0.16	0.28	0.99	0.05	0.10	0.88	**0.40**	0.55	1.00	0.15	0.26
`Facebook`-2	66	0.88	0.48	0.61	0.42	0.14	0.21	0.16	0.96	0.27	0.94	0.57	**0.71**
`Facebook`-3	308	0.94	0.26	0.41	0.92	0.07	0.13	0.41	0.15	0.22	0.97	0.18	**0.30**
`Facebook`-4	25	0.96	1.00	0.98	1.00	0.36	0.53	0.97	0.59	0.74	1.00	1.00	**1.00**
`Facebook`-5	206	1.00	0.33	**0.50**	1.00	0.09	0.17	0.97	0.31	0.47	1.00	0.33	0.49
`Facebook`-6	62	0.94	0.42	0.58	0.90	0.19	0.31	0.56	0.32	0.41	0.99	0.44	**0.61**
`Facebook`-7	408	0.94	0.58	0.71	0.38	0.04	0.07	0.21	0.17	0.18	0.96	0.60	**0.74**
`Facebook`-8	483	0.94	0.19	**0.31**	0.81	0.05	0.09	0.16	0.13	0.14	0.97	0.16	0.27
`Facebook`-9	442	0.98	0.30	**0.45**	0.97	0.07	0.14	0.97	0.20	0.33	1.00	0.24	0.38
`Facebook`-10	73	0.94	0.92	0.93	0.53	0.19	0.28	0.06	0.12	0.08	1.00	1.00	**1.00**
`Facebook`-11	237	0.99	0.87	**0.92**	0.26	0.07	0.10	0.15	0.03	0.04	1.00	0.82	0.90
`Facebook`-12	226	0.98	0.68	**0.80**	0.96	0.13	0.23	0.10	0.21	0.14	0.99	0.46	0.63
`Facebook`-13	554	0.98	0.18	0.31	0.96	0.06	0.10	0.63	0.37	**0.46**	0.99	0.18	0.21
`Facebook`-14	548	1.00	0.11	0.20	0.99	0.03	0.07	0.98	0.24	**0.39**	1.00	0.08	0.12
`Facebook`-15	60	1.00	0.33	**0.50**	0.98	0.13	0.23	0.99	0.33	**0.50**	0.98	0.14	0.24
`PGP`-1	395	0.95	0.16	**0.28**	0.86	0.16	0.27	0.82	0.12	0.21	0.96	0.12	0.21
`PGP`-2	303	0.93	0.22	**0.36**	0.92	0.22	**0.36**	0.73	0.19	0.30	0.93	0.19	0.32
`PGP`-3	974	0.94	0.13	0.24	0.74	0.13	0.22	0.84	0.17	**0.29**	0.94	0.18	0.31
`PGP`-4	379	0.99	0.12	0.21	0.78	0.12	0.20	0.90	0.17	**0.29**	0.99	0.13	0.21
`PGP`-5	1457	0.93	0.11	**0.20**	0.88	0.11	**0.20**	0.76	0.08	0.15	0.93	0.09	0.17
`PGP`-6	798	0.98	0.06	0.11	0.94	0.06	0.11	1.00	0.08	**0.16**	0.98	0.08	0.15
`PGP`-7	1289	0.96	0.14	0.24	0.80	0.14	0.23	0.76	0.13	0.22	0.96	0.17	**0.29**
`PGP`-8	513	0.97	0.17	**0.29**	0.87	0.17	0.28	0.87	0.11	0.20	0.97	0.11	0.20
`PGP`-9	417	0.93	0.17	0.28	0.93	0.17	0.28	0.92	0.27	**0.41**	0.92	0.13	0.22
`PGP`-10	1091	0.93	0.22	0.36	0.86	0.22	0.35	0.83	0.19	0.31	0.93	0.31	**0.47**

important. They are calculated as follows:

$$P(v_i) = \frac{|\mathbb{C}_i \cap \mathbb{T}_i|}{|\mathbb{C}_i|} \tag{13}$$

$$R(v_i) = \frac{|\mathbb{C}_i \cap \mathbb{T}_i|}{|\mathbb{T}_i|} \tag{14}$$

$$F1(v_i) = \frac{2P(v_i)R(v_i)}{P(v_i) + R(v_i)}. \tag{15}$$

Since the last two networks (`Facebook` and `PGP`) have no ground truth, we apply FUC [7] to identify communities with the global structure information of these two networks, and utilize its detection results as the ground truth for the AOCCM and LCD methods. This is based on the intuition that an LCD method is acceptable if it can achieve an approximate result as a GCD approach does, because LCD methods usually perform faster than GCD approaches. In addition, since the global community quality metrics such as the well-known modularity metric [5] are not suitable to evaluate the quality of the detected local community, we use each node in a community as a seed and report algorithms' average precision, recall, and F1-measure on this community. We compare AOCCM with classical degree-based LCD algorithms Luo-Wang-Promislow [17], ELC [16], and a similarity-based algorithm LTE [23]. The comparison results are presented

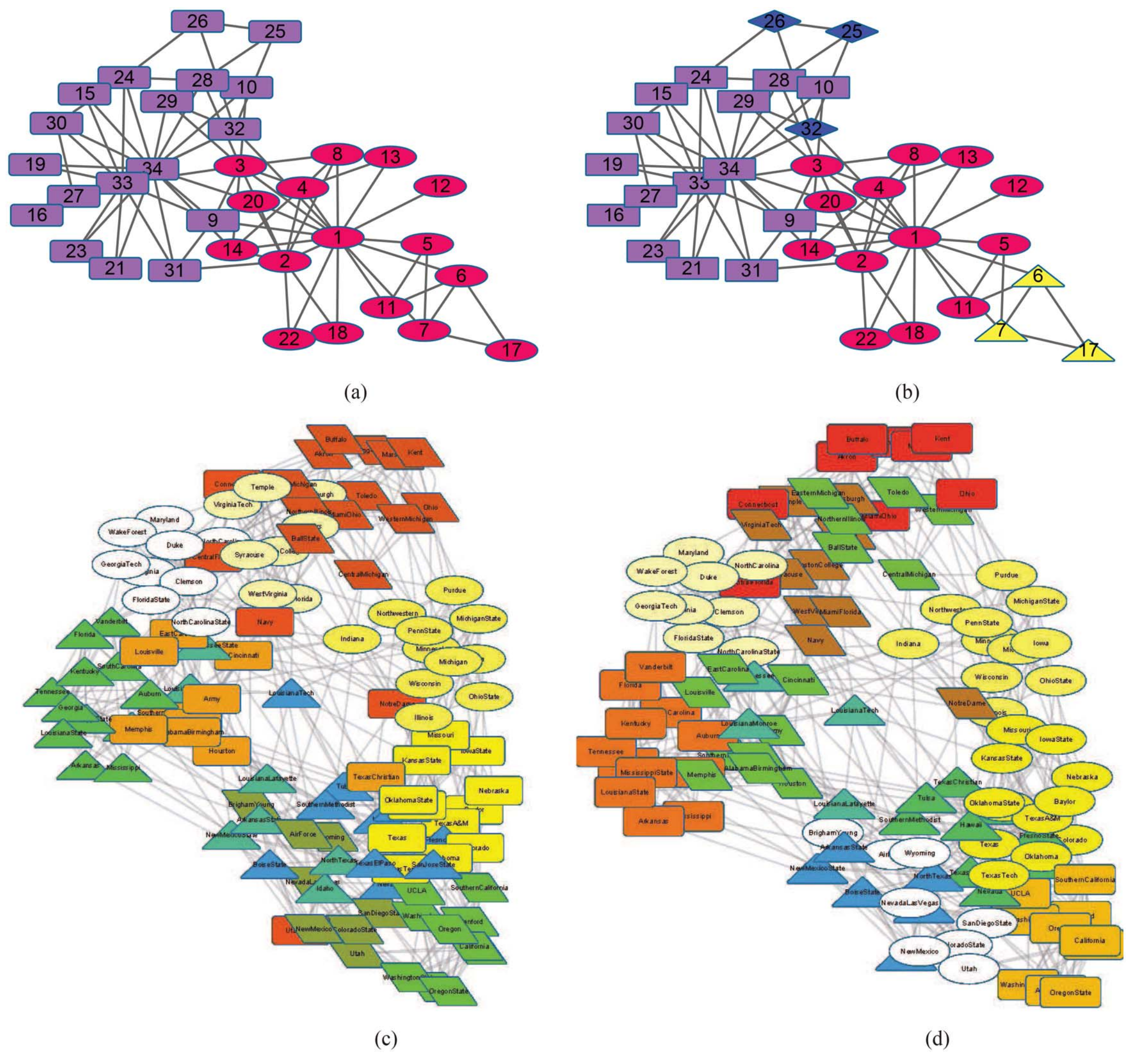


Fig. 5. AOCCMnO on small social networks. (a) `Karate`: ground truth. (b) `Karate`: AOCCMnO results. (c) `NCAA`: ground truth. (d) `NCAA`: AOCCMnO results.

in Table III. Note that FUC totally detects 85 communities on `PGP`, from which ten large communities in terms of size are selected to compare the accuracy. In Table III, we can observe that: 1) the recall values for all methods are overall worse than precision values, this is because LCD methods are based on the greedy search, which will trend to find a local optimal solution; 2) AOCCM almost achieves the high precision for all datasets, which demonstrate the superiority of its local search model over the other methods; and 3) AOCCM usually outperforms LMR and ELC, and have a slight advantage over LTE, even though the later has been proven by extensive experiments to be one of the most accurate algorithms among previous LCD methods in [23].

2) Efficiency: We further compare the efficiency of AOCCM, LMR, ELC, and LTE, Fig. 4 shows the average running time of LCD methods starting from each node in the four test graphs. Apparently, the execution of AOCCM is more accelerated. Both AOCCM and LTE are similarity-based algorithms, their difference lies at the definition of local modularity. Compared with AOCCM, the calculation of modularity gain in LTE is more complex, which will consume extra time. LMR and ELC are degree-based LCD algorithms, which need calculate the quantitative metrics for each node in $\mathcal{B}$. The metric calculations are somewhat duplicate, which can not be simplified. Especially, the stopping criteria for ELC is to jude whether the current community is a "p-strong community," which will cost more time in every search step.

B. Performance of AOCCMnO

Here, we first apply AOCCMnO to the two small social networks with ground truth: 1) `Karate` and 2) `NCAA`. The purpose is to gain a direct understanding of nonoverlapping community detection by network visualization. Then, we further compare AOCCMnO with classical GCD methods, such as FNM [5], FUC [7], METIS [33], and Cluto [34].

`Karate` is split into two parties following a disagreement between an instructor (node 1) and an administrator (node 34), which serves as the ground truth about the communities in Fig. 5(a). We employ AOCCMnO to extract nonoverlapping communities from the network. The result is shown in Fig. 5(b), which supplements the division of the club with more information. More interestingly, AOCCMnO actually tends to partition this network into four rather than two

TABLE IV
MODULARITY AND RUNNING TIME COMPARISON BY AOCCMNO, FNM [5], FUC [7], METIS [33], AND CLUTO [34]

Network	AOCCMnO	FNM	FUC	METIS	Cluto
`Karate`	0.38/0.03s	0.38/0.05s	**0.42**/0.03s	0.24/**0.01**s	0.36/0.02s
`NCAA`	0.58/0.20s	0.57/0.20s	**0.60**/0.06s	0.08/**0.01**s	0.60/0.03s
`Facebook`	0.73/2.68s	0.78/8.45m	**0.84**/6.29s	0.79/**0.53s**	0.82/4.24s
`PGP`	0.67/**0.44**s	0.85/179.42m	**0.88**/22.50s	0.83/1.76s	0.72/11.90s

communities, as indicated by the nodes in four colors/shapes in Fig. 5(b). This implies that there exits a latent subparty (including nodes 6, 7, and 11) inside the party led by node 1, and a latent subparty (including nodes 25, 26, 32) inside the party led by node 34.

The ground truth of `NCAA` labels nodes with their actual conferences, corresponding twelve different colors/shapes in Fig. 5(c). As shown in Fig. 5(d), AOCCMnO generally well captures the "sharp-cut" teams in conferences "AC," "BE," "Ten," "SE," "MV," "WA," and "Twelve," respectively, although there yet exists some teams assigned mistakenly. Note that nearly all the "orangered rectangle" in Fig. 5(c) are totally detected mistakenly by AOCCMnO. This is indeed reasonable since those nodes have very few internal connections, actually, they represent five independent teams (Utah State, Navy, Notre Dame, Connecticut, and Central Florida) in NCAA.

1) Modularity and Running Time Comparison: The global nonoverlapping community structure can be evaluated by some predefined quantitative criterions, in which, the modularity of [5] is one of most popular quality functions. Modularity can then be written as follows:

$$Q = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\chi\left(l_i, l_j\right) \tag{16}$$

where the χ-function yields one if nodes v_i and v_j are in the same community ($l_i = l_j$), zero otherwise.

In order to verity the effectiveness of AOCCMnO, we compare it with classical GCD methods, such as FNM [5], FUC [7], METIS [33], and Cluto [34]. For each method/network, Table IV displays the modularity that is achieved and the running time. The modularity obtained by AOCCMnO are slightly lower than FUCs, but it outperforms nearly all the other methods. In terms of running time, METIS has a great advantage due to its powerful parallel processing modules. However, it performs poorly on graphs with obscure community structure, e.g., `Karate` and `NCAA`. AOCCMnO, on the contrary, keeps a nice balance between high modularity and short running time.

C. Performance of AOCCMO

To evaluate the performance of AOCCMO, we also employ the PRF framework. Let $\hat{\mathbb{C}}_k$ be the kth overlapping community, which obeys $\hat{\mathbb{C}}_1 \cup \cdots \cup \hat{\mathbb{C}}_K \subseteq V$. In the following, we introduce a membership threshold α, $0 < \alpha \leq 1$, to control the scale at which we want to observe the overlapping communities in a network.

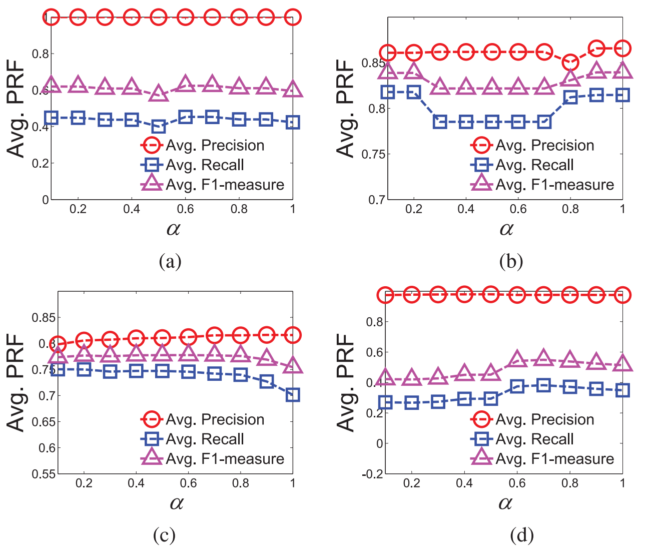


Fig. 6. Accuracy for different α on the four test networks. (a) `Karate`. (b) `NCAA`. (c) `Facebook`. (d) `PGP`.

Definition 5 (α-Overlapping Community): The kth α-overlapping community, denoted by $\hat{\mathbb{C}}_k(\alpha)$, is defined as

$$\hat{\mathbb{C}}_k(\alpha) = \{v_i | u_{i,k} \geq \alpha\}. \tag{17}$$

Therefore, we can use each node in a overlapping community as a seed and report AOCCMOs average precision, recall, and F1-measure. The precision ($\hat{P}(\alpha)$), recall ($\hat{R}(\alpha)$), and F1-measure ($\hat{F}1(\alpha)$) of the detected α-overlapping community structure are defined as follows:

$$\hat{P}(\alpha) = \frac{\sum_{k=1,\ldots,K}\sum_{v_i \in \hat{\mathbb{C}}_k(\alpha)} \frac{\left|\hat{\mathbb{C}}_k(\alpha) \cap \mathbb{T}_i\right|}{\left|\hat{\mathbb{C}}_k(\alpha)\right|}}{\sum_{k=1,\ldots,K}\sum_{v_i \in \hat{\mathbb{C}}_k(\alpha)} 1} \tag{18}$$

$$\hat{R}(\alpha) = \frac{\sum_{k=1,\ldots,K}\sum_{v_i \in \hat{\mathbb{C}}_k(\alpha)} \frac{\left|\hat{\mathbb{C}}_k(\alpha) \cap \mathbb{T}_i\right|}{|\mathbb{T}_i|}}{\sum_{k=1,\ldots,K}\sum_{v_i \in \hat{\mathbb{C}}_k(\alpha)} 1} \tag{19}$$

$$\hat{F}1(\alpha) = \frac{2\hat{P}(\alpha)\hat{R}(\alpha)}{\hat{P}(\alpha) + \hat{R}(\alpha)}. \tag{20}$$

Fig. 6 shows the accuracy in the function of α for the four test graphs, from which we can observe that: 1) the recall values for AOCCMO have a significant improvement in all scales, compared with previous AOCCM algorithms; 2) the values of α in the range [0.6, 0.8] are optimal, in the sense that overlapping communities extracted by AOCCMO in this region have a high F1-measure; and 3) AOCCMO performs better in dense networks rather than in sparse networks.

VI. Incremental AOC-Based Method for Mining Dynamic Networks

In real world, an AOC system could be updated periodically depending on new local updates. We can use $\mathbb{G} = \{\mathbb{G}^1, \mathbb{G}^2, \ldots, \mathbb{G}^T\}$ to denote a collection of snapshot graphs for a given dynamic network over T discrete time steps. Let $\mathbb{C}^l = \{\mathbb{C}_1^l, \ldots, \mathbb{C}_{n^l}^l\}$ be the archived objective of the AOC system at time l, where n^l is the total number of agents. The problem of incremental community detection can be simplified to accurately and efficiently compute $\mathbb{C}^{l+1}$ when the network is updated from $\mathbb{G}^l$ to $\mathbb{G}^{l+1}$.

One immediate approach to solve the above problem is to directly apply the AOCCM algorithm on each agent in the updated network as discussed in Section IV. Obviously, the strategy of recalculating is not efficient as it overlooks the old community structure in the previous snapshot. To address this issues, we try to find an incremental function $\digamma^*$, which can figure out the new community structure based on the previous archived objective and the incremental update

$$\mathbb{C}^l = \digamma^*\left(\mathbb{C}^{l-1}, \Delta\mathbb{G}^l\right) \tag{21}$$

where $\Delta\mathbb{G}^l = (\Delta V^l, \Delta E^l) = \mathbb{G}^l - \mathbb{G}^{l-1}$ denotes the incremental update of the network $\mathbb{G}$ at time l.

A. Incremental AOC-Based Method

In the incremental AOC-based method (in short as AOCCM-i henceforth), the network to be mined is dynamically changing, that will trigger the agents to detect the new community structure. We can understand the AOCCM-i algorithm as an iterative process consisting of a series of discrete evolutionary cycles. In the lth evolutionary cycle, the new objective of agent A_i can be quickly derived based on its previous local community ($\mathbb{C}_i^{l-1}$) and the incremental update of the network ($\Delta\mathbb{G}^l$). The life-cycle of agent A_i in the lth evolutionary cycle is given in Algorithm 4.

Remark: In Algorithm 4, the initial local community of agent A_i is set to be $\mathbb{C}_i^{l-1}$, which is the obtained objective in the previous cycle. In the AOC system, once the network is updated, agent A_i can quickly monitor the changes in its local environment by the cooperation among agents. If the incremental update, $\Delta\mathbb{G}^l$, happens to agent A_i, we have $\mathcal{B}_i^l(0) \neq \emptyset$, which will be initialized as $\{v_j | v_j \notin \mathbb{C}_i^{l-1}, v_k \in \mathbb{C}_i^{l-1}, < v_j, v_k, w_{jk} > \in \Delta E^l\}$. Otherwise, $\mathcal{B}_i^l(0) = \emptyset$, agent A_i will directly turn to the inactive phase. Except for the initialization phase, the life-cycle of agent A_i in AOCCM-i is almost the same as that in the AOCCM algorithm, as shown in Algorithm 1. Like AOCCM, AOCCM-i can be also expanded to detect the global nonoverlapping and overlapping community structures of the dynamic distributed networks.

Fig. 7 shows an example of AOCCM-i, which focuses on agent A_8 from its $l-1$th evolutionary cycle to the lth one. In the previous evolutionary cycle, A_8 detects a local community including nodes $\{v_8, v_6, v_7, v_5\}$; then the network is updated by adding into three new edges $\{e_{15}, e_{19}, e_{59}\}$. A_8 monitors the changes, and initializes its local community and boundary area to be $\mathcal{C}_8^l(0) = \{v_8, v_6, v_7, v_5\}$ and $\mathcal{B}_8^l(0) = \{v_1, v_9\}$, respectively. Next, A_8 starts its active phase and continues the iterative searching process until its clock time reaches T or it has reached its convergent status. Finally, the local community detected by A_8 at the lth evolutionary cycle is $\{v_8, v_6, v_7, v_5, v_1, v_9\}$. In Fig. 8, we further show how to use AOCCM-i to mine the dynamic NCAA. In each update of NCAA, 100 new edges are added. The entire process contains six evolutionary circles. Fig. 8(a)–(f) presents the snapshots of NCAA after each evolutionary cycle, respectively. The final NCAA from the entire process is shown in Fig. 8(f), from which we can see that the detected community structure is almost the same as that found by the AOCCMnO.

Algorithm 4 Life-Cycle of A_i in the lth Evolutionary Cycle

```
1: /*Initialization phase*/
2: $t \leftarrow 0$;
3: $\mathcal{C}_i^l(0) \leftarrow \mathbb{C}_i^{l-1}$;
4: $\mathcal{B}_i^l(0) \leftarrow \{v_j | v_j \notin \mathbb{C}_i^{l-1}, v_k \in \mathbb{C}_i^{l-1}, < v_j, v_k, w_{jk} > \in \Delta E^l\}$;
5: if $\mathcal{B}_i^l(0) = \emptyset$ then
6:     go to Step 22;
7: end if
8: /*Active phase*/
9: while $t < T$ do
10:     $v_j^* = \arg\ max_{v_j \in \mathcal{B}_i^l(t)} \sum_{v_j \in \mathcal{C}_i^l(t)} s_{ij}$;
11:     if $\Delta\hat{W}_{\mathcal{C}_i^l(t)}(v_j^*) > 0$ then
12:         $\mathcal{B}_i^l(t+1) \leftarrow \mathcal{B}_i^l(t) \cup \{v_k | v_k \in \Gamma_{j^*}, v_k \notin \mathcal{C}_i^l(t)\} - \{v_j^*\}$;
13:         $\mathcal{C}_i^l(t+1) \leftarrow \mathcal{C}_i^l(t) \cup \{v_j^*\}$;
14:     else
15:         $\mathcal{B}_i^l(t+1) \leftarrow \mathcal{B}_i^l(t) - \{v_j^*\}$;
16:     end if
17:     $t \leftarrow t+1$;
18:     if $\mathcal{B}_i^l(t) = \emptyset$ then
19:         break;
20:     end if
21: end while
22: /*Inactive phase*/
23: $\mathbb{C}_i^l = \mathcal{C}_i(t)$;
24: $l_i^l \leftarrow \arg\ max_{v_j \in \mathbb{C}_i^l} k_j$;
```

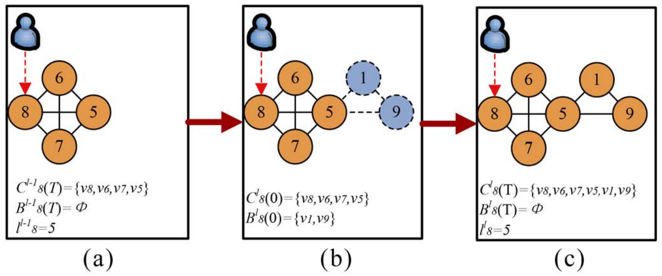


Fig. 7. Illustrative example for AOCCM-i. (a) $l-1$th evolutionary cycle. (b) Network updating. (c) lth evolutionary cycle.

B. Performance Analysis of the AOCCM-i Method

We first take the similar method as in [20] to analyze the computational complexity of AOCCM-i. The time steps

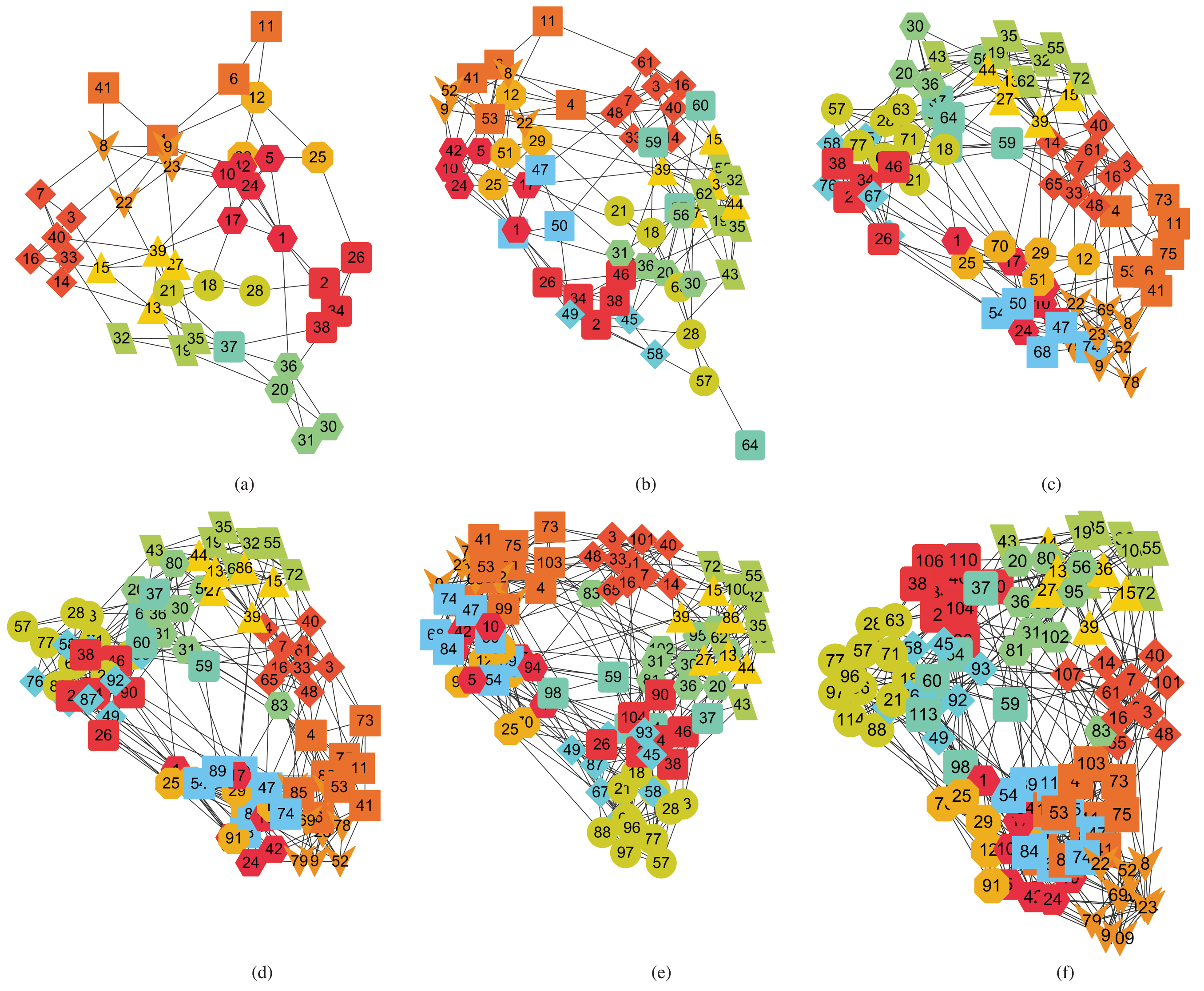


Fig. 8. Process of mining the dynamic NCAA. (a) First evolutionary cycle: $n^1 = 42$ and $m^1 = 100$. (b) Second evolutionary cycle: $n^2 = 64$ and $m^2 = 200$. (c) Third evolutionary cycle: $n^3 = 79$ and $m^3 = 300$. (d) Fourth evolutionary cycle: $n^4 = 93$ and $m^4 = 400$. (e) Fifth evolutionary cycle: $n^5 = 104$ and $m^5 = 500$. (f) Sixth evolutionary cycle: $n^6 = 115$ and $m^6 = 616$.

required by A_i to finish local community mining in the lth evolutionary cycle is defined as t_i^l. Without considering the parallel computing, the total time steps required by all agents in the lth evolutionary cycle can be calculated as $\tau^l = \sum_{i=1}^{n^l} t_i^l$.

Thus, we can approximately evaluate the efficiency of the AOCCM-i method using τ^l. Smaller value of τ^l indicates that less time is required in the lth evolutionary cycle. In the experiments, for each network, we use three different strategies to mine its community structure. The first one is based on the recalculating strategy, that is, directly applying AOCCMnO on the updated network for nonoverlapping community detection. The second one is based on (21), in which, the new community structure will be detected by AOCCM-i based on the previous objective and the incremental update of the network. The last one is AOC-i proposed by Yang *et al.* [20]. Note that all networks used here are considered as dynamic ones, which grow gradually by adding a fixed number (μ) of edges in each evolutionary cycle. As shown in Fig. 9, AOCCM-i outperforms AOC-i, and the τ value of AOCCM-i is smaller than AOCCMs by nearly two magnitude. Our method works efficiently and requires much less time to deal with the new network after each update.

In each evolutionary cycle of AOCCM-i, a fixed number (μ) of edges are added into the network, which can be seen as the updating speed of a dynamic network. Fig. 10 further presents the relationship between the average τ value and the updating speed μ, from which, we can see that Avg. τ grows with the increase of μ. This implies that if the average convergent speed of agents could be matched with the updating speed of the dynamic network, AOCCM-i will perform well without any delay; otherwise, it may result in some delay.

We finally compare AOCCM-i and AOC-i on the effectiveness. For each algorithm/network, Fig. 11(a)–(d) displays the modularity that is achieved in each evolutionary cycle. As shown in Fig. 11, the Q values of AOCCM-i are larger than AOC-is on the first three networks, while AOCCM-i is slightly inferior to AOC-i on PGP. This might due to the strict definition of the modularity gain, which decides whether the

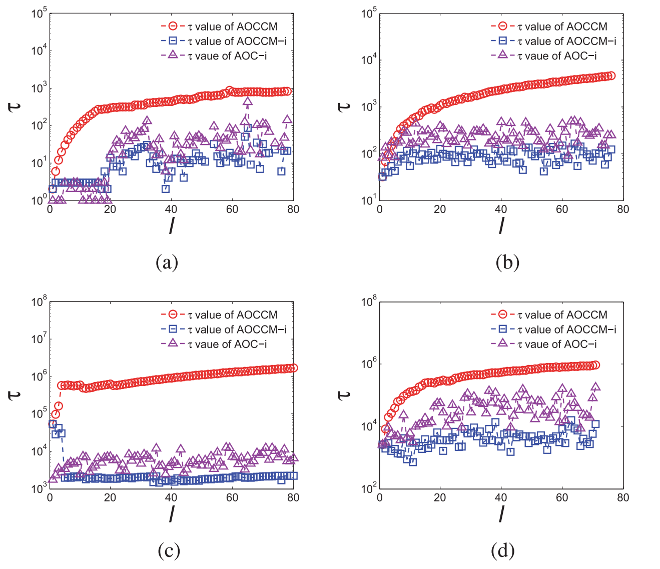


Fig. 9. Efficiency comparisons between AOCCM, AOCCM-i, and AOC-i on the four test networks. (a) `Karate`: $\mu = 1$. (b) `NCAA`: $\mu = 8$. (c) `Facebook`: $\mu = 1100$. (d) `PGP`: $\mu = 350$.

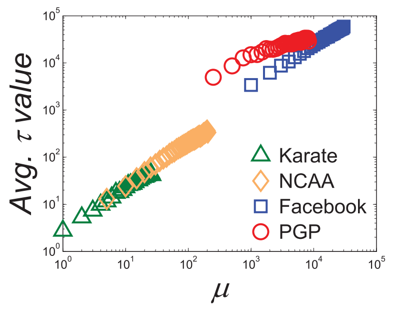


Fig. 10. Average τ value with different μ.

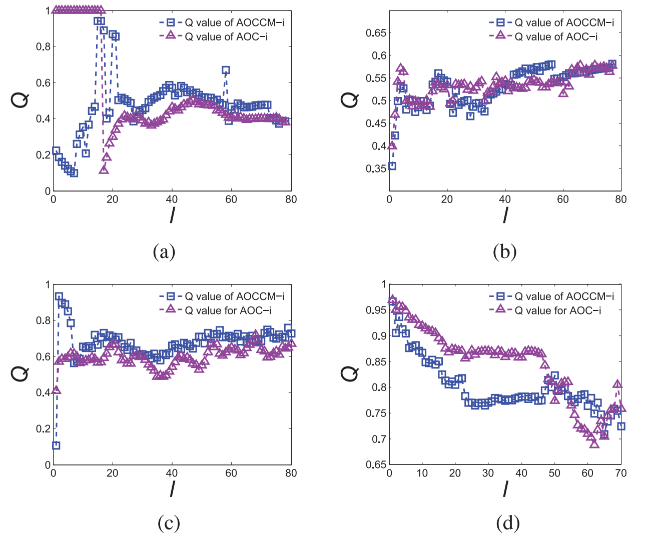


Fig. 11. Efficiency comparisons between AOCCM, AOCCM-i, and AOC-i on the four test networks. (a) `Karate`: $\mu = 1$. (b) `NCAA`: $\mu = 8$. (c) `Facebook`: $\mu = 1100$. (d) `PGP`: $\mu = 350$.

candidate node should be added into the local community. As `PGP` is a large sparse network compared with other networks, the local search model by AOCCM-i might quickly reach the convergent status. One possible solution is to relax the selection criteria, e.g., redefine the modularity gain as $\Delta\hat{W}_{\mathcal{C}_i(t)}(v_j) = \hat{W}(\mathcal{C}_i(t) \cup v_j)/\hat{W}(\mathcal{C}_i(t))$, and then use a threshold to control the candidate node selection. We will investigate this in our further work.

VII. Conclusion

Real-life networks are distributed and composed of a set of social actors and their interaction relations. Multiagent technologies have already achieved significant success in past years, especially for modeling and analyzing autonomous and distributed multientity systems. The questions then arise of how to connect real-life networks and MASs and how to use multiagent technologies to model and analyze the community structures of real-life networks. This paper attempts to answer this problem by surveying real-life networks from a multiagent perspective.

In this paper, we have proposed an AOC-based method for community mining (AOCCM), in which, each actor in the distributed networks is associated with an agent who spontaneously interacts with the local environment to find the local community. The identifier of core node in the local community can be seen as its community tag in the global view. Our method can be easily expanded to find the global nonoverlapping and overlapping community structures. Experimental results on real-life networks have demonstrated the advantage of AOCCM over previous LCD methods by either effectiveness or efficiency. Furthermore, we have described how AOCCM can be expanded to a more efficient incremental AOC-based method (AOCCM-i) for mining communities from dynamic and distributed networks. With the experiment, we found that if the average convergent speed in each evolutionary cycle is matched with the updating speed of a dynamic network, the performance of the AOCCM-i algorithm might be good without any delay.

One limitation of this paper is the core node selection: in AOCCM, large node in terms of degree is selected as the core node, the identifier of which is used as its community label in the global version. However, this assumption may be too strong, for example, in social networks, a person with wide social relations may act as an intermediary between different communities. In our future work, we will investigate the community mapping technology, which automatically generates community label according to the detected local community.

Another interesting topic for the future work is to consider the dynamic change in the AOC system. In this paper, the interaction relations between old nodes are fixed during community mining. However, in reality the interaction between any two actors can arbitrarily generate and disappear, even the weight of interaction can dynamically change with the time. Such a dynamic situation may bring about new problems to our current incremental computing model. Therefore, it is essential to devise feasible approaches to deal with the emergent problems in the dynamic AOC system.

References

[1] R. Guimerà, S. Mossa, A. Turtschi, and L. N. Amaral, “The worldwide air transportation network: Anomalous centrality, community structure, and cities’ global roles,” *Proc. Nat. Acad. Sci.*, vol. 102, no. 22, pp. 7794–7799, May 2005.

[2] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *Nature*, vol. 466, no. 7307, pp. 761–764, 2010.

[3] B. Yang, D. Liu, and J. Liu, "Discovering communities from social networks: Methodologies and applications," in *Handbook of Social Network Technologies and Applications*. New York, NY, USA: Springer, 2010, pp. 331–346.

[4] Z. Lu, X. Sun, Y. Wen, G. Cao, and T. La Porta, "Algorithms and applications for community detection in weighted networks," *IEEE Trans. Parallel Distrib. Syst.*, doi: 10.1109/TPDS.2014.2370031, 2015.

[5] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, no. 2, 2004, Art. ID 026113.

[6] Z. Bu, C. Zhang, Z. Xia, and J. Wang, "A fast parallel modularity optimization algorithm (FPMQA) for community detection in online social network," *Knowl. Based Syst.*, vol. 50, pp. 246–259, Sep. 2013.

[7] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Statist. Mech. Theory Exp.*, vol. 2008, no. 10, 2008, Art. ID P10008.

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer, 2009.

[9] A. Hlaoui and S. Wang, "A direct approach to graph clustering," in *Proc. Neural Netw. Comput. Intell. Conf.*, Grindelwald, Switzerland, 2004, pp. 158–163.

[10] B. Yang and J. Liu, "Discovering global network communities based on local centralities," *ACM Trans. Web*, vol. 2, no. 1, 2008, Art. ID 9.

[11] L. Yang, X. Cao, D. Jin, X. Wang, and D. Meng, "A unified semi-supervised community detection framework using latent space graph regularization," *IEEE Trans. Cybern.*, doi: 10.1109/TCYB.2014.2377154, 2015.

[12] F. Luccio and M. Sami, "On the decomposition of networks in minimally interconnected subnetworks," *IEEE Trans. Circuit Theory*, vol. 16, no. 2, pp. 184–188, May 1969.

[13] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proc. Nat. Acad. Sci. USA*, vol. 101, no. 9, pp. 2658–2663, Mar. 2004.

[14] T. Zhang and B. Wu, "A method for local community detection by finding core nodes," in *Proc. IEEE Comput. Soc. Int. Conf. Adv. Soc. Netw. Anal. Min. (ASONAM)*. Istanbul, Turkey, 2012, pp. 1171–1176.

[15] J. Chen, O. Zaïane, and R. Goebel, "Local community identification in social networks," in *Proc. IEEE Int. Conf. Adv. Soc. Netw. Anal. Min. (ASONAM)*, Athens, Greece, 2009, pp. 237–242.

[16] J. P. Bagrow, "Evaluating local community methods in networks," *J. Statist. Mech. Theory Exp.*, vol. 2008, no. 5, 2008, Art. ID P05001.

[17] F. Luo, J. Z. Wang, and E. Promislow, "Exploring local community structures in large networks," *Web Intell. Agent Syst.*, vol. 6, no. 4, pp. 387–400, 2008.

[18] J. Liu, X. Jin, and K. C. Tsui, "Autonomy-oriented computing (AOC): Formulating computational systems with autonomous components," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 35, no. 6, pp. 879–902, Nov. 2005.

[19] J. Liu, "Autonomy-oriented computing (AOC): The nature and implications of a paradigm for self-organized computing," in *Proc. 4th Int. Conf. Nat. Comput. (ICNC)*, vol. 1. Jinan, China, 2008, pp. 3–11.

[20] B. Yang, J. Liu, and D. Liu, "An autonomy-oriented computing approach to community mining in distributed and dynamic networks," *Auton. Agents Multi-Agent Syst.*, vol. 20, no. 2, pp. 123–157, 2010.

[21] J. P. Bagrow and E. M. Bollt, "Local method for detecting communities," *Phys. Rev. E*, vol. 72, no. 4, 2005, Art. ID 046108.

[22] A. Clauset, "Finding local community structure in networks," *Phys. Rev. E*, vol. 72, no. 2, 2005, Art. ID 026132.

[23] J. Huang, H. Sun, Y. Liu, Q. Song, and T. Weninger, "Towards online multiresolution community detection in large-scale networks," *PLoS ONE*, vol. 6, no. 8, 2011, Art. ID e23829.

[24] K. Li and Y. Pang, "A vertex similarity probability model for finding network community structure," in *Advances in Knowledge Discovery and Data Mining*. Berlin, Germany: Springer, 2012, pp. 456–467.

[25] H.-H. Chen, L. Gou, X. L. Zhang, and C. L. Giles, "Discovering missing links in networks using vertex similarity measures," in *Proc. 27th Annu. ACM Symp. Appl. Comput.*, Trento, Italy, 2012, pp. 138–143.

[26] Y. Jiang and J. Jiang, "Understanding social networks from a multiagent perspective," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 10, pp. 2743–2759, Oct. 2014.

[27] W. Wang and Y. Jiang, "Community-aware task allocation for social networked multiagent systems," *IEEE Trans. Cybern.*, vol. 44, no. 9, pp. 1529–1543, Sep. 2014.

[28] W. Chen, Z. Liu, X. Sun, and Y. Wang, "Community detection in social networks through community formation games," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, Barcelona, Spain, 2011, pp. 2576–2581.

[29] S. Asur, S. Parthasarathy, and D. Ucar, "An event-based framework for characterizing the evolutionary behavior of interaction graphs," *ACM Trans. Knowl. Disc. Data*, vol. 3, no. 4, 2009, Art. ID 16.

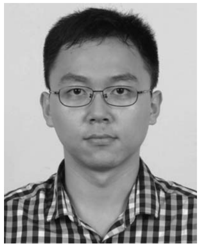
[30] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng, "Evolutionary spectral clustering by incorporating temporal smoothness," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, San Jose, CA, USA, 2007, pp. 153–162.

[31] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, "Facetnet: A framework for analyzing communities and their evolutions in dynamic networks," in *Proc. 17th Int. Conf. World Wide Web*, Beijing, China, 2008, pp. 685–694.

[32] Y. Zhao, E. Levina, and J. Zhu, "Community extraction for social networks," *Proc. Nat. Acad. Sci.*, vol. 108, no. 18, pp. 7321–7326, 2011.

[33] G. Karypis, "Multi-constraint mesh partitioning for contact/impact computations," in *Proc. ACM/IEEE Conf. Supercomput.*, Phoenix, AZ, USA, 2003, p. 56.

[34] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, Aug. 1999.



Zhan Bu received the Ph.D. degree in computer science from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2014.

He is currently a Lecturer with the Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, Nanjing. His current research interests include social network analysis, complex network, and data mining.

Dr. Bu is a member of China Computer Federation (CCF)

Zhiang Wu (M'12) received the Ph.D. degree in computer science from Southeast University, Nanjing, China, in 2009.

He is currently an Associate Professor with the Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, Nanjing. His current research interests include distributed computing, social network analysis, and data mining.

Dr. Wu is a member of ACM and CCF.

Jie Cao received the Ph.D. degree from Southeast University, Nanjing, China, in 2002.

He is currently a Professor and the Dean of the School of Information Engineering, Nanjing University of Finance and Economics, Nanjing. His current research interests include cloud computing, business intelligence, and data mining.

Prof. Cao was a recipient of the Young and Mid-Aged Expert with Outstanding Contribution in Jiangsu Province and was selected in the Program for New Century Excellent Talents in University.

Yichuan Jiang (SM'13) received the Ph.D. degree from Fudan University, Shanghai, China, in 2002.

He is currently a Full Professor and the Director of the Distributed Intelligence and Social Computing Laboratory, School of Computer Science and Engineering, Southeast University, Nanjing, China. His current research interests include multiagent systems, social networks, social computing, and complex distributed systems.

Prof. Jiang is a member of ACM.