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

Zhan Bu, Zhiang Wu, *Member, IEEE*, Jie Cao, 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 method (AOCCM) from the multiagent perspective for detecting community structures 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 non-overlapping 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*—Distributed and Dynamic Networks; Local Community Detection; Multiagent; Autonomy-Oriented Computing; Incremental Computing**

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], [6], [7], the other is based on the node-similarity [8], [9], [10], [11], and the third is local-based [12], [13], [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 the assumption that communities are groups of nodes similar

.ouo d

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], [19], [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 its local community. In the work of Yang et al. [20], the community evolution has been introduced into the proposed AOC-i method, 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 autonomy-oriented computing method from the multiagent perspective for detecting community structures in the distributed environment, in which three key problems are carefully concerned:

- How to model the real-life networks in the distributed environment?
- How to effectively and accurately detect the local community starting from an arbitrary distributed node?
- 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

Zhan Bu, Zhiang Wu (corresponding author) and Jie Cao are with Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, Nanjing, China. E-mail: buzhan@nuaa.edu.cn, zawuster@gmail.com, caojie690929@163.com.

Yichuan Jiang is with the School of Computer Science and Engineering, Southeast University, Nanjing, China. E-mail: yjiang@seu.edu.cn.

AOC methodology for modeling decentralized networks. In this system, every node is assigned an autonomous agent for local community detection. 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 three-fold 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 non-overlapping 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 the paper is organized as follows: Section II presents the related work about autonomy-oriented computing 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 conclude this paper in Section VII.

II. RELATED WORK

Here we discuss related work from two areas: autonomy-oriented computing, and dynamic network mining.

A. Autonomy-oriented computing

Early work in LCD can be adopted to autonomy-oriented computing, 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] defines local modularity by considering the boundary points of a sub-graph, and proposes a greedy algorithm on optimizing this measure. Similarly, Luo et al. [17] present another measurement as the ratio of the internal degree and external degree of a sub-graph. 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. 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 VSP [24] and RSS [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 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 method (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 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 re-calculating is not efficient. In the framework of multiagent system, Yang et al. [20] introduced an incremental AOC-based method (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 our work is also AOC-based. Compared

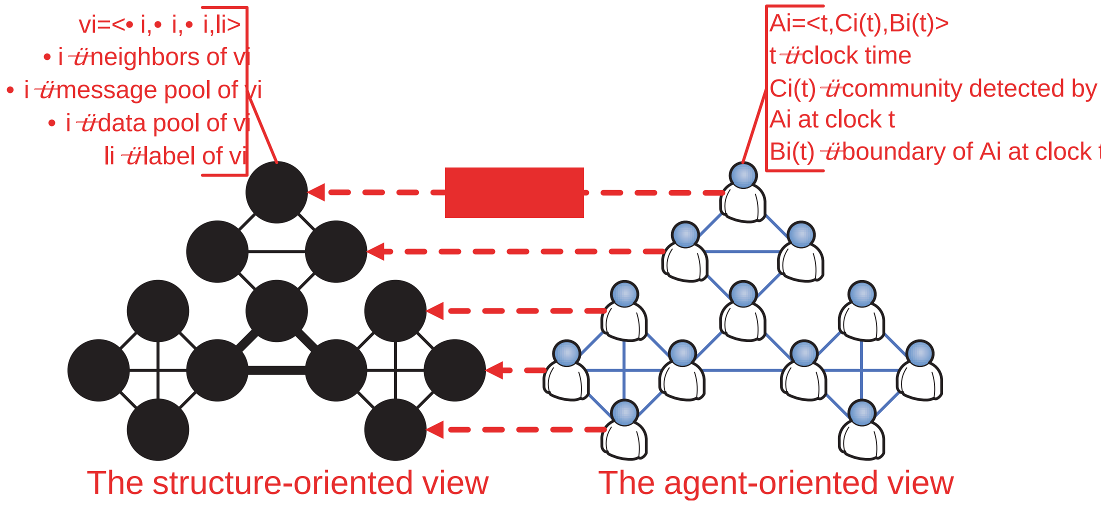


Fig. 1. The environment of the AOC system.

TABLE I
NOTATIONS OF THE AOC SYSTEM.

Symbol	Description
$_i$	the identifiers of adjacent neighbors of v_i
$_i$	the message pool of v_i, which stores the messages from others agents
$_i$	the data pool of 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
$C_i(t)$	the local community detected by agent A_i at time t
$B_i(t)$	the boundary area of agent A_i at time t

The criterion agent A_i uses 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 $C_i(t)$, denoted as $W(C_i(t))$, is given as follows:

$$W(C_i(t)) = \frac{I(C_i(t))}{|C_i(t)|^2} - \frac{O(C_i(t))}{|C_i(t)||C_i^c(t)|}, \qquad (6)$$

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

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

Theorem 1: The local modularity value of the community $C_i(t)$ will increase when $C_i(t)$ has high intra-cluster density and low inter-cluster density.

PROOF. The term $I(C_i(t))$ is twice the number of the edges within $C_i(t)$, and $O(C_i(t))$ represents the number of edges between $C_i(t)$ 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 $|C_i(t)|^2$ rather than $|C_i(t)|(|C_i(t)| - 1)$ 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 intra-cluster density minus the inter-cluster 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 $C_i(t)$ will increase when $C_i(t)$ has high internal similarity and low external similarity.

PROOF. A high value of $S_{in}(C_i(t))$ reveals a large number of common neighbors of any adjacent node pair in $C_i(t)$, resulting in a high value of intra-cluster density. While, a low value of $S_{out}(C_i(t))$ reveals a small number of common neighbors of any adjacent node pair between $C_i(t)$ and $C_i^c(t)$, resulting in a low value of intra-cluster density.

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

$$\hat{W}(C_i(t)) = |C_i(t)||C_i^c(t)|\left(\frac{I(C_i(t))}{|C_i(t)|^2} - \frac{O(C_i(t))}{|C_i(t)||C_i^c(t)|}\right); \qquad (7)$$

where the factor $|C_i(t)||C_i^c(t)|$ 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 $B_i(t)$, as shown in Fig. 2. It distinguishes three types of links: those internal to the community $C_i(t)$ (L), between $C_i(t)$ and the node v_j (L_{in}), between $C_i(t)$ and others nodes in $B_i(t)$ (L_{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 $L_{in} = a_1 L = a_2 k_j$, $L_{out} = b_1 L$, with $b_1 \geq 0$, $a_1 \geq \frac{1}{L}$, $a_2 \geq \frac{1}{k_j}$ (since any v_j in $B_i(t)$ at least

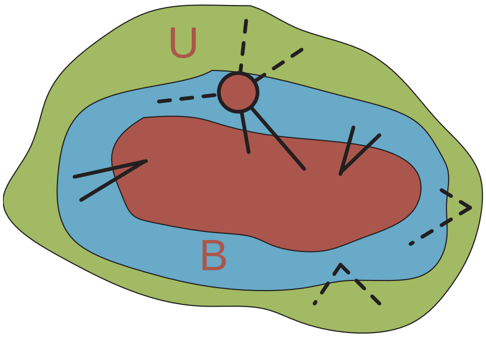


Fig. 2. The $\hat{W}$ variant when a node v_j joins $C_i(t)$.

has one neighbor in $C_i(t)$). So, the value of $\hat{W}$ for the current community can be written as:

$$\hat{W}(C_i(t)) = \frac{n-|C_i(t)|}{|C_i(t)|}2L - (a_1+b_1)L. \tag{8}$$

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

$$\hat{W}(C_i(t)\cup v_j) = \frac{n-|C_i(t)|-1}{|C_i(t)|+1}2L(1+a_1) - (b_1L+k_j-a_2k_j). \tag{9}$$

So we define the modularity gain in the following.

Definition 4 (Modularity Gain): The modularity gain for the community $C_i(t)$ adopting a neighbor node v_j can be denoted as:

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

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

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 the algorithmic details including the complexity analysis. Second, we introduce how to use AOCCM to detect the global non-overlapping 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: *Update* step and *Join* step. First, the appurtenant node v_i is added into the local community, e.g., $C_i(0)=\{v_i\}$. In the *Update* step, A_i refreshes the the boundary area $B_i(t)$, and calculate the structural similarities between nodes in the community $C_i(t)$ and their neighbor nodes in $B_i(t)$. In the *Join* step, A_i tries to absorb a node in $B_i(t)$, e.g., v_j, having highest structural similarity with nodes in $C_i(t)$ into the local community. If $\Delta\hat{W}_{C_i(t)}(v_j) > 0$, then the node v_j will be inserted into $C_i(t+1)$. Otherwise, it will be removed from $B_i(t+1)$ 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 C_i is discovered. A_i further selects the node with maximum degree in 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.

Algorithm 1 The life-cycle of agent A_i (AOCCM (A_i))

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

Remark. Unlike existing methods [16], [22], [17], which calculate the quantitative metrics for each node in B and select the node who produces the greatest increment of the metric to join 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 calculates $\Delta\hat{W}_{C_i(t)}(v_j)$ to determine whether it should be added into $C_i(t+1)$ or not. The structural similarity reflects the local connectivity density of the network. The larger the similarity between a node inside $C_i(t)$ 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.

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

$O(n^0 \log n_i^0)$, where n_i^0 is the number of nodes inferred (nodes in $C_i \cup B_i$). 2) Update (for each node in current $B_i(t)$, A_i updates its sum of structure similarities with nodes in $C_i(t)$). First, the sum of structure similarities with nodes in $C_i(t)$ for each node $v_j \in B_i(t)$ should be computed, which can be completed in $O(k_i^0)$ time, where k_i^0 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 $O(m_i^0)$ times, where m_i^0 is the number of edges in $C_i \cup B_i$. Therefore, the total time of the Update steps is $O(m_i^0 k_i^0)$. Adding all together, the total time complexity is $O(m_i^0 k_i^0 + n_i^0 \log n_i^0)$ for AOCCM on agent A_i.

Non-overlapping Community Detection. Non-overlapping community detection aims to find a good K-way partition $P = \{P_1, \cdots, P_K\}$, where P_k is the k-th community, in which $l_i = l_j \ \forall v_i, v_j \in P_k$, and $P_1 \cup \cdots \cup P_K = V$, $P_k \cap P_{k'} = \emptyset \ \forall k \neq k'$. 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 the same community label, their appurtenant nodes are likely to be in the same community. The process of AOCCM expansion algorithm for non-overlapping (in short as AOCCMnO henceforth) is given as follows, where $L = \{l_i | i = 1, \cdots, n\}$ is the label list of nodes in the distributed network. AOCCMnO could be completed

)1.9565521(i)8.23564]TJ /R29 9.96264 Tf 2.cT*[(w)[5.03984963071(e)-1.6-455961]TJ /R20 9. /R12 6.97385 Tf 6 3.6 Td [(0)
946516(r)-4.2603(e)-495.526(a)-1.603(e)-495.526(a)-1.cT*[(w)[5.0

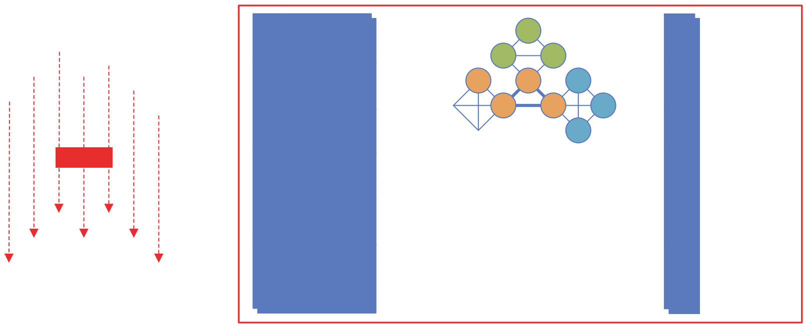
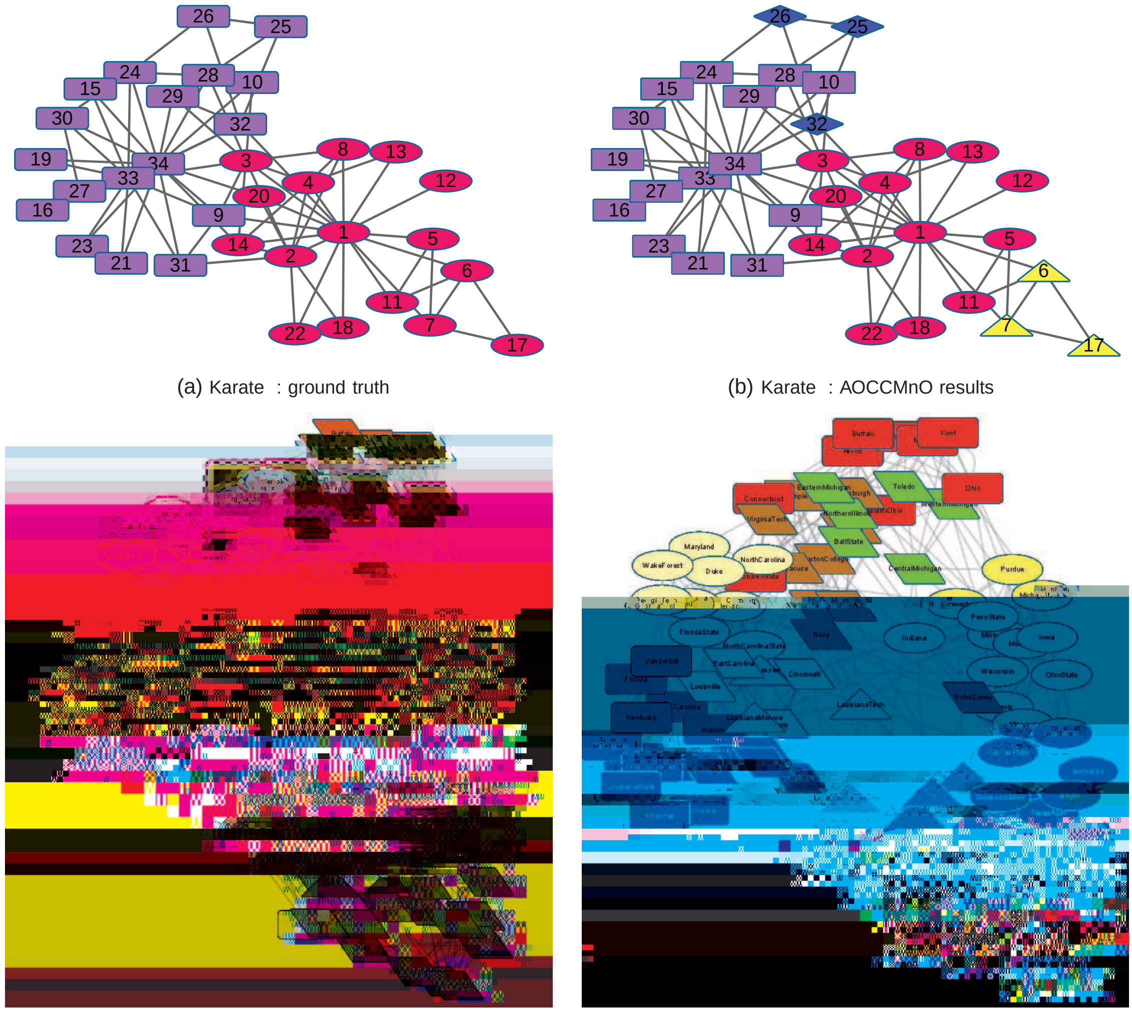


TABLE III
ACCURACY COMPARISON ON REAL-WORLD NETWORKS.

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													



(a) *Karate* : ground truth

(b) *Karate* : AOCCMnO results

(c) *NCAA* ground truth

(d) *NCAA* AOCCMnO results

Fig. 5. AOCCMnO on small social networks.

in 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 *Karate* and *NCAA*. The purpose is to gain a direct understanding of non-overlapping 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].

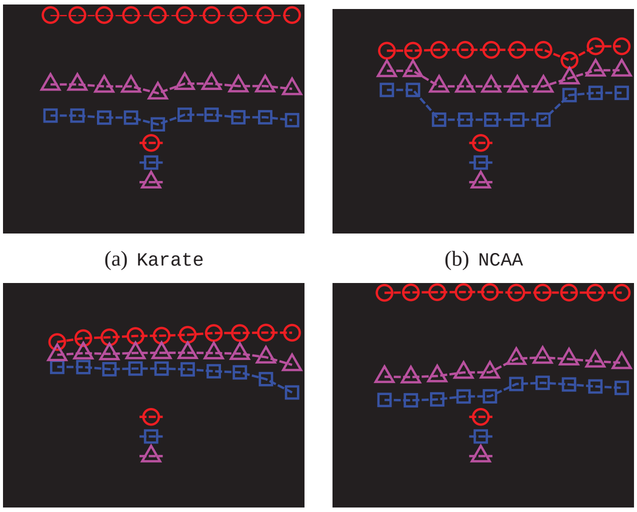
Karate is split into two parties following a disagreement between an instructor (node 1) and an administrator (node 34), which serves as the ground truth about the communities in Fig. 5(a). We employ AOCCMnO to extract non-overlapping communities from the network. The result is shown in Fig. 5(b), which supplements the division of the club with more information. More interestingly, AOCCMnO actually tends to partition this network into four rather than two communities, as indicated by the nodes in four colors/shapes in Fig. 5(b). This implies that there exits a latent sub-party (including nodes 6, 7, 11) inside the party led by node 1, and a latent sub-party (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.

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

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



(a) Karate (b) NCAA (c) Facebook (d) PGP

Fig. 6. The accuracy for different α on the four test networks.

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

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 FUC's, 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 perform poor 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 $\hat{C}_k$ be the k-th overlapping community, which obeys $\hat{C}_1 \cup \cdots \cup \hat{C}_K \subseteq V$. 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.

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

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

Therefore, we can use each node in a overlapping community as a seed and report AOCCMO's 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:

$$\hat{P}(\alpha) = \frac{\sum_{k=1,\cdots,K}\sum_{v_i \in \hat{C}_k(\alpha)} \frac{|\hat{C}_k(\alpha)\cap T_i|}{|\hat{C}_k(\alpha)|}}{\sum_{k=1,\cdots,K}\sum_{v_i \in \hat{C}_k(\alpha)} 1}, \quad (18)$$

$$\hat{R}(\alpha) = \frac{\sum_{k=1,\cdots,K}\sum_{v_i \in \hat{C}_k(\alpha)} \frac{|\hat{C}_k(\alpha)\cap T_i|}{|T_i|}}{\sum_{k=1,\cdots,K}\sum_{v_i \in \hat{C}_k(\alpha)} 1}, \quad (19)$$

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

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; 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 $G = \{G^1, G^2, \cdots, G^T\}$ to denote a collection of snapshot graphs for a given dynamic network over T discrete time steps. Let $C^l = \{C^l_1, \cdots, C^l_{n^l}\}$ 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 C^{l+1} when the network is updated from G^l to 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 re-calculating 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 z^*, which can figure out the new community structure based on the previous archived objective and the incremental update:

$$C^l = z^*(C^{l-1}, G^l), \quad (21)$$

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

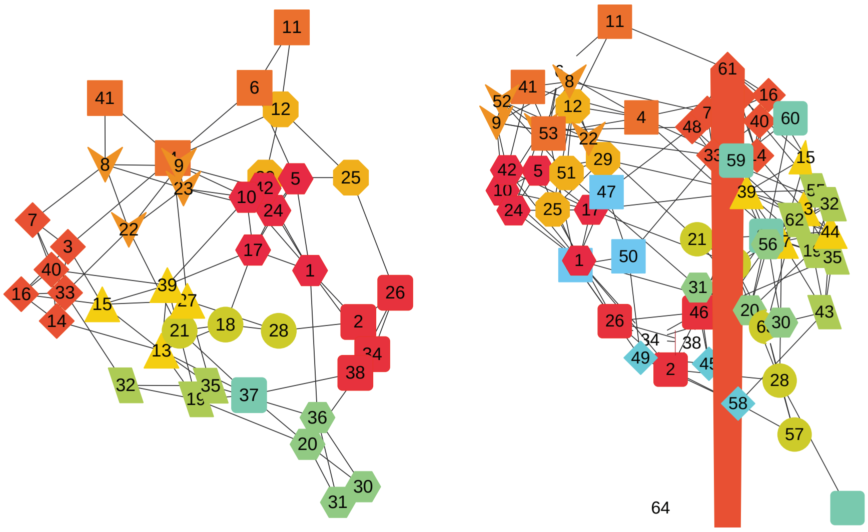
A. Incremental AOC-based method

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 l-th evolutionary cycle, the new objective of agent A_i can be quickly derived based on its previous local community (C_i^{l-1}) and the incremental update of the network (G^l). The life-cycle of agent A_i in the l-th evolutionary cycle is given in Algorithm 4:

Algorithm 4 The life-cycle of A_i in the l-th evolutionary cycle

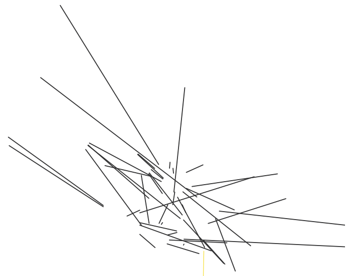
1: `/*Initialization phase*/`
2: $t \; 0$;
3: $C_i^l(0) \; C_i^{l-1}$;
4: $B_i^l(0) \; f \; v_j j v_j \; 62 C_i^{l-1}, v_k \; 2 \; C_i^{l-1}, < v_j, v_k, w_{jk} > 2 \; E^l g$;
5: **if** $B_i^l(0) = \;;$ **then**
6: go to Step 22;
7: **end if**
8: `/*Active phase*/`
9: **while** $t < T$ **do**
10: $v_j^* = \arg max_{v_j \in \mathcal{B}_i^l(t)} \sum_{v_j \in \mathcal{C}_i^l(t)} s_{ij}$;
11: **if** $4 \hat{W}_{\mathcal{C}_i^l(t)}(v_j^*) > 0$ **then**
12: $B_i^l(t+1) \; B_i^l(t) [f \; v_k j v_k \; 2 \; {}_j, v_k \; 2\!\!\!/ \; C_i^l(t) g \; f \; v_j^* g$;
13: $C_i^l(t+1) \; C_i^l(t) [f \; v_j^* g$;
14: **else**
15: $B_i^l(t+1) \; B_i^l(t) \; f \; v_j^* g$;
16: **end if**
17: $t \; t+1$;
18: **if** $B_i^l(t) = \;;$ **then**
19: break;
20: **end if**

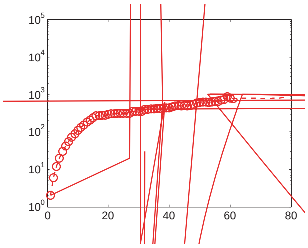
[(1)3.14361(R20 9.96264 Tf 25.d [([33043(d)-347.391(i)0.965521(f)-4.2603]TJ /R8 7.9731 Td82965521(o)-355.203(S)1.93104(t)0.964296(e)-1.66393(p)-355.



(a) 1-st evolutionary cycle: $n^1 = 42$, $m^1 = 100$

(b) 2-nd evolutionary cycle: $n^2 = 64$, $m^2 = 200$




10^5
10^4
10^3
10^2
10^1
10^0
0
20
40
60
80

[4] Z. Lu, X. Sun, Y. Wen, G. Cao, and T. La Porta, "Algorithms and applications for community detection in weighted networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, 2014.

[5] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

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

[7] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.

[8] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani, *The elements of statistical learning*. Springer, 2009, vol. 2, no. 1.

[9] A. Hlaoui and S. Wang, "A direct approach to graph clustering." in *Neural Networks and Computational Intelligence*, 2004, pp. 158–163.

[10] B. Yang and J. Liu, "Discovering global network communities based on local centralities," *ACM Transactions on the Web (TWEB)*, vol. 2, no. 1, p. 9, 2008.

[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 Transactions on Cybernetics*, vol. PP, no. 99, 2015.

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

[13] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 9, pp. 2658–2663, 2004.

[14] T. Zhang and B. Wu, "A method for local community detection by finding core nodes," in *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*. IEEE Computer Society, 2012, pp. 1171–1176.

[15] J. Chen, O. Zaïane, and R. Goebel, "Local community identification in social networks," in *Social Network Analysis and Mining, 2009. ASONAM'09. International Conference on Advances in*. IEEE, 2009, pp. 237–242.

[16] J. P. Bagrow, "Evaluating local community methods in networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 05, p. P05001, 2008.

[17] F. Luo, J. Z. Wang, and E. Promislow, "Exploring local community structures in large networks," *Web Intelligence and Agent Systems*, 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," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 35, no. 6, pp. 879–902, 2005.

[19] J. Liu, "Autonomy-oriented computing (aoc): The nature and implications of a paradigm for self-organized computing," in *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, vol. 1. IEEE, 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," *Autonomous Agents and Multi-Agent Systems*, vol. 20, no. 2, pp. 123–157, 2010.

[21] J. P. Bagrow and E. M. Bollt, "Local method for detecting communities," *Physical Review E*, vol. 72, no. 4, p. 046108, 2005.

[22] A. Clauset, "Finding local community structure in networks," *Physical review E*, vol. 72, no. 2, p. 026132, 2005.

[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, p. e23829, 2011.

[24] K. Li and Y. Pang, "A vertex similarity probability model for finding network community structure," in *Advances in Knowledge Discovery and Data Mining*. 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 *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. ACM, 2012, pp. 138–143.

[26] Y. Jiang and J. Jiang, "Understanding social networks from a multiagent perspective," *Transactions on Parallel and Distributed Systems*, vol. 25, no. 10, pp. 2743–2759, 2014.

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

[28] W. Chen, Z. Liu, X. Sun, and Y. Wang, "Community detection in social networks through community formation games," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 3, 2011, p. 2576.

[29] S. Asur, S. Parthasarathy, and D. Ucar, "An event-based framework for characterizing the evolutionary behavior of interaction graphs," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 3, no. 4, p. 16, 2009.

[30] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng, "Evolutionary spectral clustering by incorporating temporal smoothness," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 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 *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 685–694.

[32] Y. Zhao, E. Levina, and J. Zhu, "Community extraction for social networks," *Proceedings of the National Academy of Sciences*, vol. 108, no. 18, pp. 7321–7326, 2011.

In6(e)7.36(t)6.9841(i)6.9841(o)36(e).95793]TJ -123.72 -9.14285(.)-352.365(T)-626[(T)59.1229(r)12.5928(a)3.1425/
607(e)7.37006(n)3.14285(c)7.36394(e)-308.801(o)3.14285(n)-267.877(W)95.2349(o)3.14285ueno37006(c)(c)7.36394185(P)13.9l 01(o)3.907.36 0 Td [(t)6.98716((n)3.14285