Overlapping Community Structure Detection and Evolution in Networks *

Nan Du, Bin Wu, Bai Wang
Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia
Beijing University of Posts and Telecommunications
{dunan,wubin,wangbai}@bupt.edu.cn

ABSTRACT

Recent researches have seen that rich interactions among entities in nature and society bring about complex networks with community structures which could represent interacting patterns in protein-protein networks, or circles of friends and families in social networks. Although the investigation of the community structure has motivated many kinds of algorithms, most of them only find separated communities. However, for the vast majority of real-world networks, few communities really disjoint with each other. As a matter of fact, they often overlap to some extent. Being contrary to earlier methods, in this paper, we propose a new algorithm *COCD*(**C**lique-based **O**verlapping **C**ommunity **D**etection) to efficiently mine overlapping communities in large-scale networks. *COCD* is only based on the network topology and does not require any priori knowledge about the original partition of the network. We find that the extent to which communities overlap with each other actually depends on the specific types of the links in the network.

Moreover, due to the frequent changes of the interactions among entities, the associated networks are able to develop and evolve over time. Consequently, uncovering the underlying micro mechanisms that govern the dynamics of the various communities is critical to have a better understanding of the macro network evolution as a whole. Therefore, based on *COCD*, we propose a novel method to track the evolution of communities in a series of time snapshots. We show that the community's lifetime and persistence depend not only on its internal structure, but also on the constitution of the network. Our studies on the real networks from different domains demonstrate that the existence and evolution of overlapping communities has meaningful and non-trivial statistics.

*This work is supported by the National Science Foundation of China under grant number 60402011, and the National Science and Technology Support Program of China under Grant No.2006BAH03B05.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Algorithms Theory Performance

Keywords

Complex Networks, Community Detection, Community Evolution

1. INTRODUCTION

In recent years, people have found that both of the physical systems in nature and the engineered artifacts in human society have essentially networked organizations which are large intricate webs of connections between the massive entities they are composed of[20][21]. Although these systems come from very different domains, they all have the ***community*** structure in common, that is, they have vertices in a group structure that vertices within the groups have higher density of edges while vertices among groups have lower density of edges[8][14]. For instance, the communities in World Wide Web correspond to topics of interest. In social networks, individuals belong to the same community tend to have similar preference. As a result, the ability to discover communities has important practical significance and can help us to achieve a better understanding of the network.

A wide range of successful algorithms[1][5][10][13] have been developed to discover the community structures. These methods assume that communities are separated, placing each vertex in only one cluster. However, as first mentioned in [16], most real networks are made of overlapping and nested community structures. Few communities totally disjoint with each other. For example, in social networks, each of us may participate in many social cycles according to our hobbies, educational background, working environment and family relationships; In collaboration networks, an author might work with researchers from different groups; In biological networks, a protein might interact with many groups of proteins, and in word association network, a single word may be involved in various clusters of words with different concepts and meanings. As a result, most existing algorithms can not detect them directly. Only a few methods can uncover the overlapping community structures, yet they are generally not efficient enough in real networks. Therefore, to address this problem, we propose a new algorithm *COCD*

which can efficiently mine overlapping communities in large-scale networks. To the best of our knowledge, *COCD* is the first method that can handle networks consisting of millions of nodes and edges. Experimental results in the networks of different domains indicate that the overlapping between communities is correlated with the meanings and types of the edges. In addition, networks generally evolve[3][15] and develop over time. Sine that communities constitute the whole network as certain function units, we are also interested in answering the questions like: How can a specific community persist and evolve over time? What is the difference of the communities' evolving patterns between social networks and non-social networks? Do the abrupt changes of the evolution indicate some unexpected events? So, we propose an algorithm to study the time dependence of overlapping communities and describe the underlying dynamics.

The main contributions of this paper concentrate on both of the static and dynamic properties of the overlapping communities. In terms of the static aspect, a novel algorithm *COCD* is proposed to mine the overlapping communities, which requires no user input parameters, such as the number of communities, the original community partition, or any other thresholds. With respect to the dynamic aspect, a new method is presented to track and monitor the evolving process of various overlapping communities in networks from different domains.

The rest of the paper is organized as follows: section 2 reviews the related work. Section 3 describes the overlapping community detection algorithm. Section 4 discusses the proposed method for monitoring the community evolution. Experimental results are presented in section 5; and we conclude the paper in section 6.

2. RELATED WORK

Here we review the related work from two areas: mining overlapping communities and tracking community evolution.

2.1 Overlapping Community Detection

Steve Gregory has proposed *CONGA*(Cluster-Overlap *Newman Girvan* Algorithm) by extending *GN* algorithm[8] based on the betweenness centrality value. *CONGA*[9] introduces a *splitting betweenness* to identify when to split vertices, what vertex to split and how to split it. Because vertex can be split into multiple copies, single vertex can thus appear in many clusters simultaneously. Since that *CONGA* is still based on the framework of *GN*, its time complexity is $O(m^3)$ in the worst case. *Pinney* and *Westhead*[17] have also proposed extending the *GN* algorithm with the ability to split vertices between clusters. It is based on both of the edge betweenness and vertex betweenness to decide whether to split a vertex or remove an edge, which requires an user input value to assess the vertex similarity. *Palla et al*[16] proposed a k-clique based method. He defined a community as the set of k-cliques that can all be reached from each other via a sequence of adjacent k-cliques; two k-cliques are adjacent if they share $k-1$ vertices. One problem of this method is that the required user input value k often has a significant impact on the discovered communities. Moreover, vertices that are not included in any k-clique will be ignored, so the set of all the detected communities usually can not cover all the vertices of the original graph. *Shihua Zhang et al.* find overlapping clusters by extending *GN* algorithm[22] with the fuzzy c-means clustering, yet it requires a user input variable to indicate an upper bound of the community's number, which is often hard to give in real networks. *Li et al.*[12] detect clusters by combining the network topology with the content of the vertices. It first finds the densely connected subgraphs of the network, and then add triangles and edges whose keywords are similar to those of the subgraphs.

To sum up, the *GN*-based extensions generally have the efficiency problem, and the modularity optimization strategy used by *GN* introduces a resolution limit[6] as well(we are unable to detect communities smaller than $\sqrt{2m}$), while the other existing methods often require more information from the users. To overcome these shortages, we design *COCD* by a local optimization strategy, which does not suffer from the resolution problem, and does not require any priori knowledge about the communities' number or other related thresholds to assess the community structures.

2.2 Community Evolution

Aggarwal et al.[2] focused on an On-line Analytical Processing (OLAP) approach for providing on-line exploratory capabilities to users in performing change detection across communities of interest over different time horizons. *John Hopcroft et al.*[11] tracked evolving communities in the scientific citation network from *CiteSeer* by focusing on the emergence of new communities, which might represent new research areas, such as "Wireless Network", and "Quantum Computing". However, they did not describe other important evolving processes such as community merging and splitting. *Palla et al.*[15] provided a method to quantify the evolving paths of social groups by considering the common edges of the communities in two consecutive snapshots. The algorithm first built a joint graph consisting of all the edges and nodes of the two subsequent graphs, and then used *CPM*[16] to find the communities in this graph. Any two communities in the corresponding subsequent graphs that mostly matched with one of the discovered communities in the joint graph were regarded to be in the same evolving path. Nevertheless, this approach has two major deficiencies. One is that it may cost much to build the joint graph and find the communities when the networks are large. Furthermore, the matching scheme is too strict that given any community c, c often has no descender at the subsequent time step when it actually does have one. The other is that this approach can not distinguish between the split communities and the newly born ones. It assumes that each community must have one major descender. When a community splits into two, the mostly matched one will be the descender, and the other one is regarded as a "new" born community. However, this regarded "new" community probably has a close relationship with its successor, and could be very different from the real new born communities. *Jimeng Sun*'s *GraphScope*[19] was a parameter-free mining method over time-evolving graph streams. Essentially, *GraphScope* depended on the changing of the graph partition(separated communities) to reflect the macroscopic changes of the whole network. By contrast, our proposed community evolution method focuses on the microscopic evolving path of each specific community instead.

3. CLIQUE-BASED OVERLAPPING COMMUNITY DETECTION ALGORITHM

Instead of dividing a network into its most loosely connected parts, *COCD* identifies the communities based on the most densely connected parts, namely, the ***cliques***. We treat each group of highly overlapping maximal cliques as the clustering cores. Surrounding each core, we build up the communities in an gradually expanding way according to certain metrics until each vertex in the network belongs to at least one community.

3.1 Notations and Definitions

In this paper, we consider simple graphs only, i.e., the graphs without self-loops or multi-edges. Given graph G, $V(G)$ and $E(G)$ denote the sets of its vertices and edges respectively.

Definition 1. $S \subseteq V(G), \forall u, v \in S, u \neq v$, such that $(u, v) \in E$, then S is a clique in G. If any other S' is a clique and $S' \supseteq S$ iff $S' = S$, S is a maximal clique of G.

Definition 2. For a given vertex v, $N(v) = \{u|(v, u) \in E(G)\}$, we call $N(v)$ is the neighbor set of v. Given set $S \subseteq V(G), N|_S = \bigcup N(v_i) - S, v_i \in S, N|_S$ is the set of all neighbors of S. The subgraph induced on S is represented as G_S. Given subgraph G_i and G_j, $E_{bet}(G_i, G_j)$ denotes the set of edges between them.

Definition 3. Let $Com(G)$ be the set of all components in G. The giant component is denoted by C_G and $M(C_G)$ is the set of all the maximal cliques in C_G. We use $V_M \subseteq V(G)$ to represent the set of all vertices covered by $M(C_G)$.

Definition 4. Given vertex $v_i \in V_M$, C_i is the set of all maximal cliques containing v_i, and $C = \{C_i|C_i \subseteq M(C_G)\}$. $\forall C_i, C_j \in C, |C_i| \geq |C_j|$, G_i and G_j are the subgraphs induced on C_i and C_j respectively. We use a *Closeness Function* $isClose(G_i, G_j)$ which will be discussed and implemented in the next section to assess the link patterns between G_i and G_j. If $isClose(G_i, G_j)$ returns *true*, we say C_j is contained in C_i, denoted by $C_j < C_i$. If C_i is not contained by any other element in C, C_i is called the **core** of G and v_i is the **center** of C_i. The set of cores is denoted by $Core$.

Definition 5. Let P_0,P_1,...,P_{n-1} be the subgraph of G such that $V(P_0) \cup ... \cup V(P_{n-1}) = V(G)$. For any pair of P_i and P_j, if $|E(P_i)| > |E_{bet}(P_i, P_j)|$ and $|E(P_j)| > |E_{bet}(P_i, P_j)|$, P_i is defined as the ***community*** or ***cluster*** of G.

Definition 6. Given a set of communities $\mathcal{C}$, the ***overlap coefficient***[9] is defined as

$$overlap = \frac{\sum_{P_i \in \mathcal{C}} |P_i|}{|V(G)|}$$

Definition 7. Given a set of communities $\mathcal{C}$, the ***vertex average degree***[9] is defined as

$$vad = \frac{2 \sum_{P_i \in \mathcal{C}} |E(P_i)|}{\sum_{P_i \in \mathcal{C}} |P_i|}$$

3.2 Algorithm

COCD first enumerates all maximal cliques in the giant component C_G. Because a maximal clique is a complete subgraph, it is thus the densest community which can represent the closest relationship involving a single entity in the given network.

3.2.1 Core Formation

For any $v_i \in V(G)$, C_i is the set of all maximal cliques containing v_i. Every maximal clique in C_i corresponds to some kind of relationship involving v_i, in other words, C_i reflects the fact that individuals belong to different kinds of relationships simultaneously. Since that C_i covers all the densest communities in which v_i has participated, set C reflects the nature of the overlapping communities in networks. For any $v_i, v_j \in V_M$, if $isClose(G_i, G_j)$ returns true (G_i and G_j are the subgraphs induced on C_i and C_j respectively), which means all or most of v_j's relationships are covered by those of v_i, we say v_j depends on v_i and $C_j < C_i$. Otherwise, if $\forall C_i \in C$, $i \neq j$, $C_j \not< C_i$, then C_j becomes the core. Therefore, the larger that the size of C_i can be, the more likely that a core it would become. We rearrange all the elements of set C according to the descending order of their sizes and delete those elements whose sizes are smaller than 2, which means if C_i is a core, v_i must participate in at least two different relationships.

Algorithm 1 $isClose(G_i, G_j)$

```
{G_i and G_j are the subgraphs induced on C_i and C_j respectively}
G_L ⇐ G_i or G_j with the larger set of vertices
G_S ⇐ G_i or G_j with the smaller set of vertices
v_L ⇐ center of G_L, v_S ⇐ center of G_S
if V(G_S) ⊆ V(G_L) then
    return true
else
    I_LS ⇐ V(G_L) ∩ V(G_S)
    α ⇐ |V(G_L)| / |V(G_S)|
    inf_LS = α|E_bet(G_{V(G_L)−{v_S}}, G_{V(G_S)−I_LS})|
    inf_SL = |E_bet(G_{V(G_S)−{v_L}}, G_{V(G_L)−I_LS})|
    if (⌊inf_LS⌋ ≥ |E(G_{V(G_S)−I_LS})|) and
    (⌊inf_SL⌋ ≥ |E(G_{V(G_L)−I_LS})|) then
        return true
    else
        return false
    end if
end if
```

Given C_i and C_j, G_i and G_j are the induced subgraphs respectively. $\mathcal{I}_{ij}$ denotes the common vertices shared by $V(G_i)$ and $V(G_j)$. The basic idea of the *closeness function* $isClose(G_i, G_j)$ depends on the link pattern among G_i and G_j to quantify the influence that they put on each other. Being as the centers of G_i and G_j, v_i and v_j actually have connections with all the other vertices, so we remove v_j from $V(G_i)$, and v_i from $V(G_j)$. Then, the influence that G_i puts on G_j is formulated as

$$inf_{ij} = \alpha |E_{bet}(G_{(V(G_i)-\{v_j\})}, G_{(V(G_j)-\mathcal{I}_{ij})})|$$

where

$$\alpha = \begin{cases} \frac{|V(G_i)|}{|V(G_j)|} & |V(G_i)| > |V(G_j)| \\ 1 & \text{otherwise} \end{cases}$$

α is a weighted parameter which also considers the influence of the graph size. As a matter of fact, we believe that if $|V(G_i)| \gg |V(G_j)|$, the vertices of G_i usually could have many connections with those of G_j, which will put more "gravitation" on G_j. If both $\lfloor inf_{ij} \rfloor$ and $\lfloor inf_{ji} \rfloor$ are greater

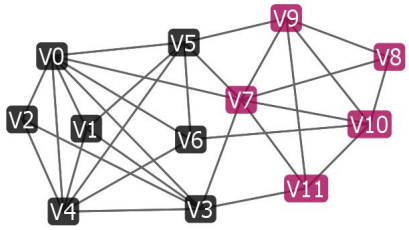

Figure 1: Core Formation

than or equal to the number of the inner edges of the subgraphs induced on $V(G_j)-\mathcal{I}_{ij}$ and $V(G_i)-\mathcal{I}_{ij}$, then C_i and C_j would be merged together. If $|C_i| > |C_j|$, C_j is thus regarded to be contained by C_i. Otherwise, they are independent of each other. The implementation of $isClose(G_i, G_j)$ is formulated in Algorithm 1.

Let C_{i_0} be the element of C whose size is the largest, C_{i_1} be the element of C whose size ranks second... C_{i_n} be the element of C whose size ranks n and etc. $Core$ is the set of all cores. C_{i_0} is first picked up and those elements contained by C_{i_0} are removed from C. In the next step, each maximal clique that includes the centers of the left elements in C will be deleted from C_{i_0} in order to erase duplications. If C_{i_0} is not empty, it is put in $Core$. Again, we will pick up the next element with the largest size from the rest elements of C. The process shown in Algorithm 2 continues iteratively until C becomes empty.

To make things more concrete, an illustrated example is given as follows on the network shown in Figure 1. $C_0 = \{\{v_0, v_1, v_4, v_5\}, \{v_0, v_1, v_3, v_4\}, \{v_0, v_2, v_3, v_4\}, \{v_0, v_4, v_5, v_6\}, \{v_0, v_5, v_7\}\}$ with center v_0. $C_1 = \{\{v_0, v_1, v_4, v_5\}, \{v_0,\ v_1,\ v_3,\ v_4\}\}$ with center v_1. Apparently $C_1 < C_0$, C_1 is not a core. Similarly,C_2, C_3, C_4, C_5 are also contained by C_0, and C_8, C_9, C_{10}, C_{11} are contained by C_7. For C_0 and C_7, they share a common maximal clique $\{v_0, v_5, v_7\}$. The common vertices between $V(G_0)$ and $V(G_7)$ are v_0, v_5, and v_7 respectively. v_7 is then removed from $V(G_0)$ and v_0 is removed from $V(G_7)$. As a result, $\lfloor inf_{07} \rfloor = \frac{7}{5} \times 3 = 4$. $\lfloor inf_{70} \rfloor = 6$. As a result, because $\lfloor inf_{07} \rfloor = 4 < |E(G_{(V(G_7)-\mathcal{I}_{07})})| = 5$, and $\lfloor inf_{70} \rfloor = |E(G_{(V(G_0)-\mathcal{I}_{07})})| = 6$, C_0 and C_7 are independent of each other.

3.2.2 Clustering

Once all the cores have been detected, we carry out a clustering process to associate the left vertices to their closest cores. For each subgraph G_i induced on $C_i \in Core$, we gradually expand G_i by adding the vertices in set $N|_{V(G_i)}$. Since that any vertex v_i might have connections with more than one core, v_i can thus be assigned to multiple cores simultaneously. The clustering continues until all the vertices of V_M are covered.

Let P be the set of the expanded clusters. Given $P_i, P_j \in P$, we use the same process as the *Core Formation* to compare the closeness between P_i and P_j. If the *Closeness Function* $isClose(G_{P_i}, G_{P_j})$ returns *true*, P_i and P_j is merged together. Otherwise, for each shared vertex $v_s \in P_i \cap P_j$, v_s will be assigned to the one that it can have more connections with, which means it may be more influential. If v_s has the same number of connections with P_i and P_j, v_s is thus kept in both of them, so the final clusters may overlap with each other. The whole procedure is described in Algorithm 3.

Algorithm 2 $CoreFormation(C)$

```
Core ⇐ ∅
sort C by the descending order of |C_i|, C_i ∈ C
{center stores the centers of the filtered out kernels}
center ⇐ ∅
for C_i ∈ C do
    contained ⇐ C_j, j ≠ i, C_j < C_i
    independent ⇐ k, k ≠ i, C_k ⊄ C_i
    delete C_i from C
    C ⇐ C − contained
    for s ∈ C_i do
        if s ∩ (independent ∪ center) ≠ ∅ then
            delete s from C_i
        end if
    end for
    if C_i ≠ ∅ then
        put C_i to Core
    end if
    put v_i to center
end for
return Core
```

Algorithm 3 $Clustering(Core, Center)$

```
{Center is the set of the center nodes of each core}
for C_i ∈ Core do
    ∀v_i ∈ V(G_{C_i}) is marked as covered
end for
while not all vertices in V_M are covered do
    {extend is the set of the vertices to be added}
    for C_i ∈ Core do
        ∀v_i ∈ N|_{(V(G_{C_i})−Center)} is added to extend
        v_i is added to V(G_{C_i})
    end for
    ∀v_i ∈ extend is marked as covered
end while
sort Core according to the size of V(G_{C_i}), C_i ∈ Core
for C_i ∈ Core do
    ∀C_j < C_i, V(G_{C_j}) is merged to V(G_{C_i})
    for C_K ⊄ C_i, V(G_{C_k}) ∩ V(G_{C_i}) ≠ ∅ do
        common ⇐ V(G_{C_k}) ∩ V(G_{C_i})
        if ∀v_i ∈ common, |N(v_i) ∩ V(G_{C_i})| > |N(v_i) ∩
        V(G_{C_k})| then
            v_i is removed from V(G_{C_k})
        else
            if |N(v_i) ∩ V(G_{C_i})| < |N(v_i) ∩ V(G_{C_k})| then
                v_i is removed from V(G_{C_i})
            end if
        end if
    end for
end for
```

3.2.3 Complexity and Efficiency

We use an efficient algorithm $Peamc$[7] which is optimized particularly on large sparse graphs where $|V(G)| \approx |E(G)|$, to find the maximal cliques. It costs $O(\Delta \times M_C \times Tri^2)$ in the worst case on a single processor, where Δ is the maximal degree of G, M_C is the size of the maximum clique and Tri is the number of all triangles in G. For the core formation, we need to traverse all the elements of C whose size is larger than 2, which will cost $O(\max(|V(G_{C_i})|) \times |C|^2)$, $C_i \in C$. For the clustering process, it costs $O(\Delta \times |Core| \times |V(G)| \times I)$ to associate the left vertices to the cores, where I is is the average times to repeat until all vertices are covered. On small-world networks, $I \approx 6$. It also costs $O(\max(|V(G_{C_k})|) \times |Core|^2)$, $C_k \in Core$ to adjust the common vertices. Because real networks are often sparse graphs[20][21], we have $|V(G)| \approx |E(G)|$, $|C| < |V(G)|$, $|Core| \approx |C|$, $|V(G)| < Tri^2 \ll |V(G)|^2$, and $\Delta \times M_C < \max(|V(G_{C_k})|)$, $C_k \in Core$. Let C_M denote the maximum size of the discovered communities. $COCD$ will cost $O(C_M \times Tri^2)$ in the worst case.

4. CORE-BASED COMMUNITY EVOLUTION

The rich interactions and the frequent activity changes among entities make the networks gradually evolve and develop over time. People's knowledge of the micro-mechanisms governing the underlying community dynamics is limited, yet is essential to have a better understanding of the macro network evolution as a whole. By taking the self-similarity[18] and scale-free properties into account, we assume that most sub-graphs in networks often have central entities in the same way as the whole network often has several hub nodes. These central entities usually put important impacts on the overall formation and development of the given community. For example, in scientific collaboration networks, the central entity of a research team could be a common advisor who originally started the research area, while in telecommunication call networks, the central person in a group of frequently contacted people may be the chief officer in a department. Consequently, the stability of the cores often determines the persistence of the communities, in other words, although the constitution of a community may change over time, its core nodes may still remain the same. Therefore, in this section, we provide an algorithm based on $COCD$ to investigate the time dependence of the overlapping communities.

4.1 Notations and Definitions

Given a community or cluster $P_i \in \mathcal{C}$, the vertices $v_i \in P_i$ is sorted by the descending order of certain "cental" metrics which evaluate its "importance" or "influence", such as the *degree*, *betweenness* and other specific weights represented as $w(v_i)$. We want to find the vertices whose weights are the most distinguishable from the others', so we have the community ***core*** definition as follows.

Definition 1. For the vertices v_0, v_1,..., $v_{|P_i|-1}$ with $w(v_0) \geq w(v_1)$, ..., $\geq w(v_{|P_i|-1})$, if $\exists v_k \in P_i$ such that it is the first time when $w(v_k) - w(v_{k+1}) > (\sum_{i=k+1}^{|P_i|-2}(w(v_i) - w(v_{i+1})))/(|P_i| - k - 1)$, v_0, v_1, ..., v_k are called the ***cores*** of P_i denoted by $Core(P_i)$.

Definition 2. Given a community at time t, $\mathcal{C}_t^i$, if there exits no other community $\mathcal{C}_s^k$, $s < t$ such that $Core(\mathcal{C}_s^k) \cap Core(\mathcal{C}_t^i) \neq \emptyset$, $\mathcal{C}_t^i$ is called the ***newborn*** community. Contrarily, if there exits no other community $\mathcal{C}_r^j$, $r > t$ such that $Core(\mathcal{C}_r^j) \cap Core(\mathcal{C}_t^i) \neq \emptyset$, $\mathcal{C}_t^i$ is called the ***dieout*** community. In addition, for $\mathcal{C}_t^i$, if there exits no community at time $t+1$, $\mathcal{C}_{t+1}^k$, such that $Core(\mathcal{C}_{t+1}^k) \cap Core(\mathcal{C}_t^i) \neq \emptyset$, yet there does exit a community $\mathcal{C}_r^j$, $r > t+1$, such that $Core(\mathcal{C}_r^j) \cap Core(\mathcal{C}_t^i) \neq \emptyset$, we say $\mathcal{C}_t^i$ is ***reborn*** at time r.

Definition 3. Given the set of communities at time t, $\mathcal{C}_t$, $\forall \mathcal{C}_t^i \in \mathcal{C}_t$ if there exist more than one community $\mathcal{C}_{t+1}^j, \mathcal{C}_{t+1}^k \in \mathcal{C}_{t+1}$ at time $t+1$ such that $Core(\mathcal{C}_t^i) \cap Core(\mathcal{C}_{t+1}^j) \neq \emptyset$ and $Core(\mathcal{C}_t^i) \cap Core(\mathcal{C}_{t+1}^k) \neq \emptyset$, we say community $\mathcal{C}_t^i$ ***splits*** into community $\mathcal{C}_{t+1}^j$ and $\mathcal{C}_{t+1}^k$. $\mathcal{C}_t^i$ is the ***successor*** of $\mathcal{C}_{t+1}^j$ and $\mathcal{C}_{t+1}^k$, while $\mathcal{C}_{t+1}^j$ and $\mathcal{C}_{t+1}^k$ are the ***descenders*** of $\mathcal{C}_t^i$. For $\mathcal{C}_{t+1}^j$, $\mathcal{C}_{t+1}^k$, the one who has more cores overlapping with $\mathcal{C}_t^i$ is called the ***main descender*** of $\mathcal{C}_t^i$.

Definition 4. Given more than one community at time t, $\mathcal{C}_t^i$, $\mathcal{C}_t^j$, if there exists a community at time $t+1$, $\mathcal{C}_{t+1}^k$ such that $Core(\mathcal{C}_t^i) \cap Core(\mathcal{C}_{t+1}^k) \neq \emptyset$ and $Core(\mathcal{C}_t^j) \cap Core(\mathcal{C}_{t+1}^k) \neq \emptyset$, we say $\mathcal{C}_{t+1}^k$ is ***merged*** from $\mathcal{C}_t^i$ and $\mathcal{C}_t^j$. $\mathcal{C}_{t+1}^k$ is called the ***descender*** of $\mathcal{C}_t^i$ and $\mathcal{C}_t^j$, while $\mathcal{C}_t^i$ and $\mathcal{C}_t^j$ are the ***successors*** of $\mathcal{C}_{t+1}^k$.

Definition 5. Given a community at time t, $\mathcal{C}_t^i$, and the only one ***descender*** at time $t+1$, $\mathcal{C}_{t+1}^j$, if $|\mathcal{C}_{t+1}^j| > |\mathcal{C}_t^i|$, we say $\mathcal{C}_t^i$ ***grows*** to $\mathcal{C}_{t+1}^j$;Otherwise, we say $\mathcal{C}_t^i$ ***shrinks*** to $\mathcal{C}_{t+1}^j$.

Definition 6. Given a graph stream G_t, the ***community evolution problem*** is formulated to build family graphs from the starting communities $\mathcal{C}_{t_0}^0$, $\mathcal{C}_{t_0}^1$, ..., in G_{t_0}, and investigate the statistics of their underlying activities such as *birth/death*, *growth/shrinking*, and *splitting/merging* over time on a large scale.

4.2 Tracking Overlapping Communities

To study the temporal evolution process of the communities in detail, we will track and build a family graph for each community by using the core-based successor/descender matching between consecutive time steps. Thus, we are going to first describe the algorithm for the core's discovery. Given community $\mathcal{C}_t^i$, $Core(\mathcal{C}_t^i)$ are the hub vertices whose weights are far more than others'. According to *Definition 1*, let $\Delta(w) = w(v_i) - w(v_{i+1})$. We need to find the cut-off point that can result in the $\Delta(w)$ that is larger than the average level. Given vertex $v_i \in \mathcal{C}_t^i$, $Avg_{forward}$ is the average value of $\Delta(w)$ from v_{i+1} to $v_{|\mathcal{C}_t^i|-2}$, and $Avg_{backward}$ is the one from v_0 to v_{i-1}. Algorithm 4 describes the whole procedure.

For community $\mathcal{C}_t^i$, based on the detected $Core(\mathcal{C}_t^i)$, we can build a family tree rooted with $\mathcal{C}_t^i$ as follows: the direct children of $\mathcal{C}_t^i$ are its splitting descenders $\mathcal{C}_{t+1}^j$, $\mathcal{C}_{t+1}^k$, ... If $\mathcal{C}_t^i$ does not have any descender at time step $t+1$, we further search for its possible reborn moment. If $\mathcal{C}_t^i$ is reborn at time t_r, then the matching community $\mathcal{C}_{t_r}^m$ is regarded as $\mathcal{C}_t^i$'s descender, and we continue to build the family tree recursively until the end time step. Together with other communities that may merge with each node in the family tree, we obtain an evolving graph of $\mathcal{C}_t^i$ in the end.

5. EXPERIMENTAL RESULTS

Algorithm 4 $Core(\mathcal{C}_t^i)$

1: $\{v_0, v_1, ..., \in \mathcal{C}_t^i, w(v_0) \geq w(v_1) \geq, ..., w(v_{|\mathcal{C}_t^i|-1})\}$
2: $Core \Leftarrow \emptyset$
3: **for** $\forall v_j \in \mathcal{C}_t^i$ **do**
4: $\quad Avg_{forward} = \frac{(\sum_{k=j+1}^{|\mathcal{C}_t^i|-2}(w(v_k)-w(v_{k+1})))}{(|\mathcal{C}_t^i|-j-1)}$
5: $\quad Avg_{backward} = \frac{(\sum_{k=0}^{j-1}(w(v_k)-w(v_{k+1})))}{j}$
6: $\quad$ **if** $(w(v_j) - w(v_{j+1}) > Avg_{forward})$ and $(w(v_j) - w(v_{j+1}) > Avg_{backward})$ **then**
7: $\quad\quad$ add $v_0, v_1, ..., v_j$ to $Core$
8: $\quad\quad$ break
9: $\quad$ **end if**
10: **end for**
11: **if** $Core = \emptyset$ **then**
12: $\quad$ **return** $\mathcal{C}_t^i$
13: **end if**

Table 1: Datasets Summary

Graph	Vertices $\|V\|$	Edges $\|E\|$
KARATE CLUB[23]	34	78
DOLPHIN SOCIAL NETWORK[23]	62	159
NETSCIENCE COLLABORATION[23]	379	914
COLLEGE FOOTBALL NETWORK[23]	115	616
BLOG NETWORK[25]	3,982	6,803
CELLPHONE NETWORK	385,407	488,808
E-PRINT COLLABORATION[31]	39,577	175,693
ENRON EMAIL NETWORK[29]	1,506	3,288
DEVICE NETWORK[26]	51	169
NORTH AMERICAN POWER GRID[23]	4,941	6,594
WORD ASSOCIATION NETWORK[28]	10,225	80,330
ANT CLASS NETWORK[30]	1,319	4,784
TOMCAT CLASS NETWORK[30]	1,352	6,020
INTERNET[32]	786	18,922
PROTEIN INTERACTION NETWORK[27]	1,846	4,406
E-PRINT CITATION NETWORK[31]	34,546	421,534

In this section, we will present the experimental results and analysis on both overlapping community detection and evolution in several real, large networks from various domains. Table 1 describes these graph datasets.

Specifically, DEVICE NETWORK consists of the students in an experimental network in MIT whose Bluetooth devices such as *cellphone*, *laptop computers*, and *PDA*s periodically scan for each other from Jul, 2004 to April, 2005 on a weekly basis. Two people with connection time more than 1 hour are regarded to be acquainted with each other. We also analyze the class structures including *extend*, *compose*, *reference* and *implement* of the famous *Apache Ant* 1.x and *Tomcat* 5.5.x [30] to build the software networks.

COCD is evaluated and compared with the existing algorithms: *CONGA*, *Pinney* and *Westhead*'s algorithm($P\&W$), and *CFinder*. We also analyze the occurring frequency of overlapping between communities, and demonstrate the difference between overlapping and non-overlapping communities. Finally, we will show the different statistic characteristics of the temporal evolution of the community structures in the networks from both social and physical systems.

5.1 Evaluation of Overlapping Communities

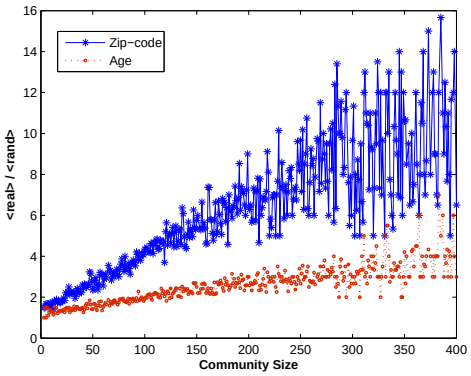

Figure 2: Communities of CELLPHONE NETWORK

We evaluate *COCD*, *CONGA*, $P\&W$ as well as *CFinder*[24] in the networks of Table 1. We use *Peamc*[7] to enumerate the maximal cliques on large sparse graphs ($|V(G)| \approx |E(G)|$). On the dense E-PRINT CITATION NETWORK, we use an improved BK algorithm[4] to find the maximal cliques. All experiments are done on a single PC (3.0GHz processor with 2Gbytes of main memory on Linux AS3 OS). The execution time of *COCD* includes both of the ***clique finding time*** and ***the community detection time***. To measure how well each algorithm can perform from the networks whose community structures are already known, we introduce the following two values: ***Recall***: the fraction of vertex pairs belonging to the same community that are also in the same cluster; ***Precision***: the fraction of vertex pairs in the same cluster that also belong to the same community. Table 2 presents the experimental results on the KARATE CLUB, DOLPHIN SOCIAL NETWORK, and COLLEGE FOOTBALL NETWORK. We see that on KARATE CLUB, ***COCD*** has both higher *Recall* and *Precision*. Just like GN algorithm, it finds almost perfect with only one misclassified vertex. For *CONGA*, although it has a more higher Vad, suggesting that the communities it discovered are more denser, *CONGA* does not perform very accurately; On DOLPHIN SOCIAL NETWORK, the observer *Lusseau* has used GN to find 2 principle communities and then further refined these 2 communities to 4 sub-communities. ***COCD*** can find the 4 communities directly with high *recall* and *precision* values; For COLLEGE FOOTBALL NETWORK, ***COCD*** is the only one that can perfectly find the 12 communities with the lower $Overlap$ and higher Vad values. In terms of ***CFinder***($k = 3$), although it covers all the vertices with a high $Overlap = 2.13$ and $Recall = 1.00$, it only correctly classifies a small portion($Precision = 0.11$) of the vertices. With respect to ***CFinder***($k = 5$), it performs better by obtaining 15 communities, however, it does not cover all the vertices of the network for $Overlap = 0.97$.

In addition, ***COCD*** is tested on other networks given in Table 3. We use ~ for name abbreviation. ***CONGA*** costs 30,182s to run on the BLOG NETWORK. Because both ***CONGA*** and ***P&W*** are the extensions of the classic GN algorithm, they actually cost too much time(more than 5h on average) to run on the networks in Table 3. For ***CFinder***, it generally can not cover all the vertices of the network to give very satisfactory results. We further evaluate the homogeneity of ***COCD***'s discovered communities on the CELLPHONE

Table 2: Results on Networks with Known Community Structures

Graph	Algorithms	Recall	Precision	Overlap	Vad	Time(s)	Communities$\|\mathcal{C}\|$
KARATE CLUB	**COCD**	**1.00**	**0.95**	**1.05**	**4.00**	**0.001**	**2**
	CONGA	0.80	0.55	1.03	4.45	0.002	2
	P&W	0.44	0.73	1.73	3.02	0.063	4
	$CFinder(k=3)$	0.62	0.53	1.00	3.94	0.063	3
DOLPHIN SOCIAL NETWORK	**COCD**	**0.99**	**0.88**	**1.06**	**4.06**	**0.001**	**4**
	CONGA	0.95	0.90	1.03	4.91	0.011	2
	P&W	0.49	0.65	1.50	3.52	0.010	15
	$CFinder(k=3)$	0.46	0.58	0.85	4.65	0.01	4
	$CFinder(k=5)$	0.05	1.00	0.18	4.36	0.003	2
COLLEGE FOOTBALL NETWORK	**COCD**	**0.91**	**0.82**	**1.04**	**7.23**	**0.014**	**12**
	CONGA	0.93	0.32	1.75	5.87	7.80	15
	P&W	0.72	0.52	3.11	3.66	0.02	63
	$CFinder(k=3)$	1.00	0.11	2.13	7.83	0.8	4
	$CFinder(k=5)$	0.68	0.95	0.97	6.28	0.8	15

Table 3: Results on Networks with Unknown Community Structures

Graph	Overlap	Vad	Time(s)	$\|\mathcal{C}\|$
BLOG~	1.11	2.94	0.56	196
CELLPHONE~	1.67	2.05	678	6027
COLLABORATION~	1.20	5.07	2127	3629
POWER GRID~	1.18	2.23	0.25	230
WORD~	1.43	4.01	65	790
ANT 1.7~	1.07	6.72	10	22
TOMCAT 5.5.9~	1.28	7.33	9	43
INTERNET	1.45	3.92	850	69
PROTEIN~	1.11	2.21	0.09	55
CITATION~	1.44	15.43	5819	1048

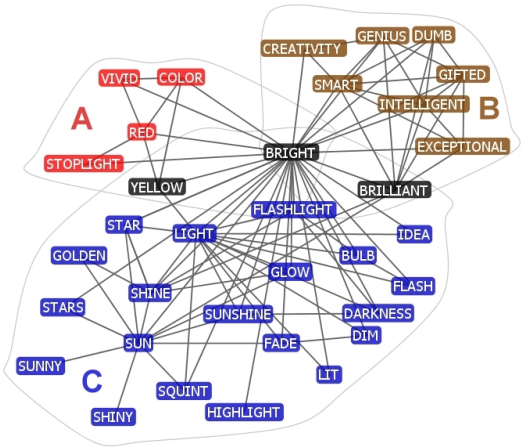

Figure 3: "Bright" Community

NETWORK by using the additional information: *zip-code* and *users' age*. According to Figure 2, the blue star symbols correspond to the average size of the largest subset of members with the same zip-code, $< real >$, in the discovered cellphone call communities divided by the same quantity found in random sets, $< rand >$. Similarly, the red circle symbols show the average size of the largest subset of members with the same age, $< real >$, in the discovered cellphone call communities divided by the same quantity found in random sets, $< rand >$. We can see that the $< real > / < rand >$ ratio is significantly larger than 1 for both the *zip-code* and *age*, indicating that the communities discovered by ***COCD*** tend to contain individuals living in the same area and having a comparable age, a homogeneity that supports the validity and effectiveness of the discovered communities.

Figure 3 shows parts of the communities in the vicinity of the word *BRIGHT*. The common vertices are drawn with *Black* color. *BRIGHT* is the common vertex of all the three communities, *YELLOW* is shared by community A and C, and *BRILLIANT* is hold by community B and C. It is easy to see that each community corresponds to one specific meaning or concept of the word *BRIGHT*. However, if we use the non-overlapping algorithm, such as *GN*, *BRIGHT* can only be associated with one of these three communities, which could result in some loss of information.

Moreover, in Table 3, we can observe an interesting phenomena that the *Overlap* values of CELLPHONE NETWORK, WORD ASSOCIATION NETWORK, INTERNET and E-PRINT CITATION NETWORK are much higher than those of the other networks. One reason we believe is that it results from the different types of the links in the network. For CELLPHONE NETWORK, people can contact with each other due to various reasons, so the link can represent the family relationship, student relationship, colleague relationship and so on. The link in this network can have multiple meanings simultaneously, so an individual tends to belong to multiple communities. Similarly, in WORD ASSOCIATION NETWORK shown in Figure 3, it is common for a word to have many meanings, each of which may corresponds to one community, so the links belonging to these different communities can have various meanings. The INTERNET takes the same situation with CELLPHONE NETWORK simply for that each computer may logically participate in many domains. For the E-PRINT CITATION NETWORK, one article may be cited by other articles coming from different fields because of the interdisciplinary collaborations. Each community may represent one possible research filed, so each paper is probably involved in multiple communities. In terms of the other networks of Table 3, the links' types are relatively simple. Vertices in these networks often inter-

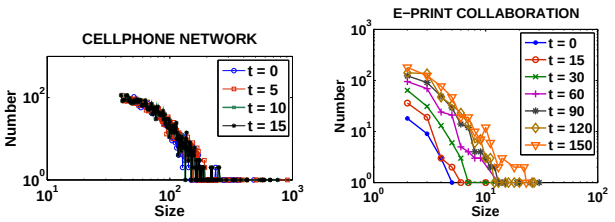

Figure 4: Community Size Distribution

act with each other for only one simple relationship, so their *Overlap* values are much lower.

5.2 Community Evolution

The time evolving graphs we consider are `CELLPHONE NETWORK` for 15 weeks, `E-PRINT COLLABORATION` for 168 weeks, `DEVICE NETWORK` for 39 weeks, which are social networks; `INTERNET`[32] for 29 snapshots, `ANT CLASS NETWORK` from version 1.0 to 1.7, and `TOMCAT CLASS NETWORK` from version 5.5.0 to 5.5.25, which are non-social networks. In addition, `CELLPHONE NETWORK` and `E-PRINT COLLABORATION` have rather different local structures. A phone call record of the `CELLPHONE NETWORK` captures the communication patterns between two people, whereas the publication of the `E-PRINT COLLABORATION` associates all authors of a given paper a fully connected clique as a one-mode projection between authors and papers. These fundamental differences of the above networks suggest that any common features of the community evolution in our experiments could reflect potentially general laws and properties, rather than depending on the details of the network representation.

Figure 4 describes the community size distribution at different time steps. They all resemble to a power-law with a high exponent. We have further studied the relationships among the lifetime, the stable rate, and the community size. Given community $\mathcal{C}_t^i$, the evolving trace consisting of its subsequent ***main descenders*** of the family tree is regarded as $\mathcal{C}_t^i$'s backbone trace denoted as $\tau = \{ \mathcal{C}_t^i, \mathcal{C}_{t+1}^i, ..., \mathcal{C}_{t+n}^i \}$. We use the auto-correlation coefficient to quantify the relative overlap between $\mathcal{C}_t^i$ and $\mathcal{C}_{t+1}^i$ as $A(\mathcal{C}_t^i, \mathcal{C}_{t+1}^i) = \frac{|\mathcal{C}_t^i \cap \mathcal{C}_{t+1}^i|}{|\mathcal{C}_t^i \cup \mathcal{C}_{t+1}^i|}$ Consequently, given community $\mathcal{C}_t^i$ and its evolving backbone τ, we adopt the average value of $A(\mathcal{C}_t^i)$ as $< A(\mathcal{C}_t^i) >= (\sum_{t=0}^{|\tau|-1} A(\mathcal{C}_t^i, \mathcal{C}_{t+1}^i))/|\tau|$ to represent its evolving stable rate.

Figure 5 shows the experimental results on the relationships between the community age, the stable rate, and the corresponding community size in various networks. From Figure 5(a) to Figure 5(f), the ***Average Age*** of communities with a given size is calculated as the average value of its lifetime over the set of available communities and the time steps. We see that larger communities tend to live longer than the smaller communities on average. For Figure 5(c), the community with size 30 marked by the red line appears in the summer vacation in Aug, 2004. At this moment, more students may get involved in the experiment network to form this large community for a relatively short lifetime. For the non-social networks `INTERNET`, `ANT 1.X CLASS NETWORK`, and `TOMCAT 5.5.X CLASS NETWORK`, they often have an extremely large community whose size is far more larger than others'. For example, in `TOMCAT 5.5.X CLASS NETWORK` this community corresponds to the classes within the pack-

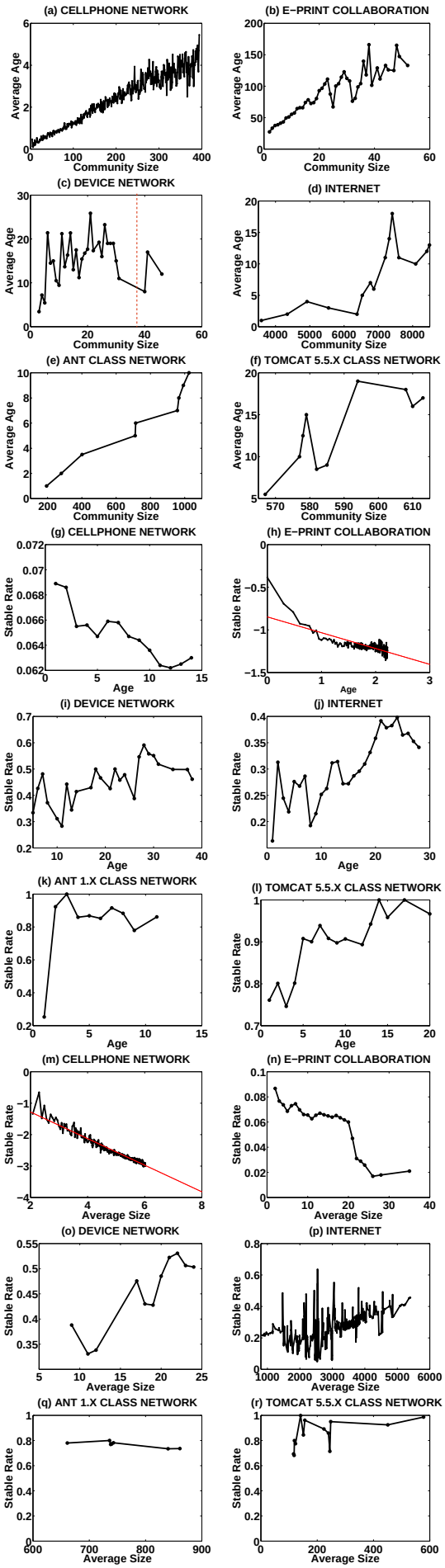

Figure 5: Relationships between Lifetime, Stable Rate, and Community Size

age ***catalina*** which is the core component of ***TOMCAT***. As a result, Figure 5(d), Figure 5(e) and Figure 5(f) mainly focus on such large communities, and they also indicate that the larger the community can be, the longer it would live.

Given community $\mathcal{C}_t^i$ and its evolving backbone τ, Figure 5(g) – 5(l) demonstrate the relationship between the average lifetime of τ and its stable rate. Figure 5(g) and 5(h) show that communities living longer have lower stable rate, whereas Figure 5(i) – 5(l) give the opposite situation that communities living longer have higher stable rate. This is because for the `CELLPHONE NETWORK` and the `E-PRINT COLLABORATION` network, they are the "open" systems that as vertices are joining in or leaving off the networks, communities are more easy to survive and develop by changing their constitutions over time. However, for the `DEVICE NETWORK`, `INTERNET`, `ANT 1.X CLASS NETWORK` and `TOMCAT 5.5.X CLASS NETWORK`, they are "close" systems whose constitutions are relatively stable. As a result, communities with high stable rate are more easy to survive and live longer than others. Moreover, we have a similar situation in Figure 5(m) – 5(r) which describe the relationship between the average community size of τ and its average stable rate. Large communities in the "open" systems `CELLPHONE NETWORK` and the `E-PRINT COLLABORATION` tend to change its members to live longer, yet they often try to keep stable in the more "close" systems `INTERNET`, `ANT 1.X CLASS NETWORK` and `TOMCAT 5.5.X CLASS NETWORK` to develop and evolve.

6. CONCLUSIONS

In this paper, we have proposed a new method *COCD* for efficient overlapping community identification in large-scale networks. We have demonstrated the effectiveness and efficiency of *COCD* over a number of real networks coming from disparate domains whose structures are otherwise difficult to understand. Experimental results show that *COCD* can extract meaningful communities that are agreed with both of the objective facts and our intuitions. We also show that the overlapping between communities depends on the meaning and types that the edges can actually take in the given networks. Moreover, based on *COCD*, we further present a *Core*-based method to investigate the temporal dependence of overlapping communities. We have found that if the network is an open system, large communities can live longer by consistently changing their members; However, if the network is a relatively close system, they tend to survive and develop by keeping stable. For the future work, we will continue our research by focusing on modeling the evolution and prediction of the community structures as well as extensions to find communities in bipartite networks.

7. REFERENCES

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *VLDB*, pages 81–92, 2003.

[2] C. C. Aggarwal and P. S. Yu. Online Analysis of Community Evolution in Data Streams. In *SDM*, pages 81–92, 2005.

[3] L. Backstrom, D. Huttenlocher, and J. Kleinberg. Community detection in complex networks using agents. In *The 12th ACM SIGKDD*, pages 44–54.

[4] C. Bron and J. Kerbosch. Finding all cliques of an undirected graph. *Communications of the ACM*, 16:575–577, 1973.

[5] D. Cai, Z. Shao, X. He, X. Yan, and J. Han. Community mining from multi-relational networks. In *PKDD*, pages 445–452, 2005.

[6] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *PNAS*, 104(1):36–41, 2006.

[7] N. Du, B. Wu, and B. Wang. A parallel algorithm for enumerating all maximal cliques in complex networks. In *ICDM Mining Complex Datd Workshop*, pages 320–324, December 2006.

[8] M. Girvan and M. Newman. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, June 2002.

[9] S. Gregory. An algorithm to find overlapping community structure in networks. In *The PKDD*, pages 91–102, 2007.

[10] K. Hildrum and P. S. Yu. Focused community discovery. In *ICDM*, pages 641–644, 2005.

[11] J. Hopcroft, O. Khan, B. Kulis, and B. Selman. Tracking evolving communities in large linked networks. *PNAS*, 101:5249–5253, April 2004.

[12] X. Li, B. Liu, and P. S. Yu. Discovering overlapping communities of named entities. In *The PKDD*, pages 593–600, 2006.

[13] M. Newman. Detecting community structure in networks. *The European Physical Journal B*, 38(2):321–330, May 2004.

[14] M. Newman. Modularity and community structure in networks. *PNAS*, 103(23):8577–8582, June 2006.

[15] G. Palla, A.-L. Barabasi, and T. Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, April 2007.

[16] G. Palla, I. Dernyi, and I. Farkas. Uncovering the overlapping community structure of complex network in nature and society. *Nature*, 435(7043):814–818, June 2005.

[17] J. Pinney and D. Westhead. *Betweenness-based decomposition methods for social and biological networks*. Leeds University Press.

[18] C. Song, M. Havlin and H. Makse. A Self-Similarity of Complex Networks. *Nature*, 433(7024):392–395, 2005.

[19] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *The 13th ACM SIGKDD*, pages 687–696, New York, NY, USA, 2007. ACM.

[20] D. Watts. *Small Worlds:The Dynamics of Networks between Order and Randomness*. Princeton University Press, Princeton, 1999.

[21] D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, June 1998.

[22] S. Zhang, R. S. Wang, and X. S. Zhang. Identification of overlapping community structure in complex networks using fuzzy c-means clustering. *PHYSICA A*, 374(1).

[23] http://www-personal.umich.edu/~mejn/netdata/

[24] http://www.cfinder.org/

[25] http://www.cs.bris.ac.uk/~steve/networks/index.html

[26] http://reality.media.mit.edu/download.php

[27] http://www.nd.edu/ networks/resources.htm

[28] http://w3.usf.edu/FreeAssociation/

[29] http://www.cs.cmu.edu/ enron/
[30] http://www.apache.org/
[31] www.cs.cornell.edu/projects/kddcup/datasets.html
[32] http://sk-aslinks.caida.org/data/