

# Community Detection in Large-Scale Social Networks \*

Nan Du

Beijing Key Laboratory of  
Intelligent Telecommunications  
Software and Multimedia  
Beijing University of Posts and  
Telecommunications, China  
dunan@bupt.edu.cn

Bin Wu

Beijing Key Laboratory of  
Intelligent Telecommunications  
Software and Multimedia  
Beijing University of Posts and  
Telecommunications, China  
wubin@bupt.edu.cn

Xin Pei

Beijing Key Laboratory of  
Intelligent Telecommunications  
Software and Multimedia  
Beijing University of Posts and  
Telecommunications, China  
peixin@tseg.org

Bai Wang

Beijing Key Laboratory of  
Intelligent Telecommunications  
Software and Multimedia  
Beijing University of Posts and  
Telecommunications, China  
wangbai@bupt.edu.cn

Liutong Xu

Beijing Key Laboratory of  
Intelligent Telecommunications  
Software and Multimedia  
Beijing University of Posts and  
Telecommunications, China  
xliutong@bupt.edu.cn

## ABSTRACT

Recent years have seen that WWW is becoming a flourishing social media which enables individuals to easily share opinions, experiences and expertise at the push of a single button. With the pervasive usage of instant messaging systems and the fundamental shift in the ease of publishing content, social network researchers and graph theory researchers are now concerned with inferring community structures by analyzing the linkage patterns among individuals and web pages. Although the investigation of community structures has motivated many diverse algorithms, most of them are unsuitable for large-scale social networks because of the computational cost. Moreover, in addition to identify the possible community structures, how to define and explain the discovered communities is also significant in many practical scenarios.

In this paper, we present the algorithm *ComTector*(Community DeTector) which is more efficient for the community detection in large-scale social networks based on the nature of overlapping communities in the real world. This algorithm does not require any priori knowledge about the number or the original division of the communities. Because real networks are often large sparse graphs, its running time is thus  $O(C \times Tri^2)$ , where  $C$  is the number of the detected communities and  $Tri$  is the number of the triangles in the given network for the worst case. Then we propose a general naming method by combining the topological information

with the entity attributes to define the discovered communities. With respect to practical applications, *ComTector* is challenged with several real life networks including the Zachary Karate Club, American College Football, Scientific Collaboration, and Telecommunications Call networks. Experimental results show that this algorithm can extract meaningful communities that are agreed with both of the objective facts and our intuitions.

## Categories and Subject Descriptors

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

## General Terms

Algorithms Theory Performance

## Keywords

Social Network Analysis, Community Detection, Graph-based Data Mining

## 1. INTRODUCTION

In recent years, easy connections brought about by cheap devices, modular content, and shared computing resources are having a profound impact on our social structures. People now increasingly take their required information from one another rather than from institutional sources like corporations, media outlets, religions, and political bodies[3]. Perhaps the most outstanding success of such kind of communication is the World Wide Web(WWW). Powered by Web 2.0 applications, WWW becomes the most popular social media which covers all forms of sharing: from experiences, to photos, to recommendations. As a result, people are implicitly involved in many social networks which are formed by our friend lists in the instant messaging softwares, by the bloggers who comment on a certain topic in your blogspace, or by the users who write collaboratively in a wiki site.

Most of these networks are generally sparse in global yet dense in local. They have vertices in a group structure

\*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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Joint 9th WEBKDD and 1st SNA-KDD Workshop '07 ( WebKDD/SNA-KDD'07) August 12, 2007, San Jose, California, USA  
Copyright 2007 ACM 978-1-59593-848-0 ...\$5.00.

that the vertices within the groups have higher density of edges while vertices between groups have lower density of edges[21][24]. This kind of structure is called the *community* which is an important network property and can reveal many hidden features of the given network. Individuals belonging to the same community are probable to have properties in common. The communities in the blogspace often correspond to topics of interests. Monitoring the aggregate trends and opinions revealed by these communities provides valuable insight to a number of business applications, such as marketing intelligence and competitive intelligence. Hence, identifying the communities is a fundamental step not only for discovering what makes entities come together, but also for understanding the overall structural and functional properties of a large network[13][23].

A popular quantitative definition called *Network Modularity*, proposed by Girvan and Newman[9][15], is widely used as a quality metric for assessing the partitioning of a given network into communities. The search for the largest modularity value is a NP-hard problem due to the fact that the space of possible partitions grows faster than any power of the system size[8]. For this reason, many recent algorithms adopt various heuristic strategies to achieve the optimization of this metric. However, as mentioned in [16], most actual networks are made of highly overlapping cohesive subgroups of nodes simply because individuals often belong to numerous different kinds of relationships simultaneously. For example, each of us may participate in many social cycles according to our hobbies, educational background, working environment and family relationships. As a result, when the network is large and the overlapping is significant, most of the existing algorithms in general will have high computational cost due to their heuristic optimization strategies.

Therefore, the main contribution of this paper is first to propose an algorithm *ComTector* which is efficient for the community detection in large-scale social networks by using such overlapping nature of the communities in real world scenarios. Given a large sparse graph, the running time of our algorithm is  $O(C \times Tri^2)$ , where  $C$  is the number of the detected communities and  $Tri$  is the number of the triangles in the given network for the worst case. Then we present a method for describing and naming the discovered communities by combining the network topological information with the vertex natural attributes.

The paper is then structured as follows: in section 2, we mainly review some related work. Section 3 describes the community detection algorithm in great details. Section 4 discusses the naming method to define the discovered communities. The experimental results and analysis are presented in section 5; and we conclude the paper in section 6.

## 2. RELATED WORK

In social network analysis (SNA), a community is often regarded as some kind of cohesive sub-structures[21][24], such as the cliques[2][7], n-cliques, n-clans, n-plexes[25], as well as the quasi-cliques[1][17][26]. These dense sub-structures always impose extra restrictions on the community definition. For example, the definition of n-clique requires that the distance between any pair of vertices should be no more than  $n$ , while in a quasi-clique the proportion of the number of each vertex's neighbors to the number of all the vertices in the sub-structure is no less than a threshold value. At the

same time, these sub-structures are usually small in size, and people may get tremendous number of them, which actually hides the global organization of the given network. Compared with the defined cohesive sub-structures, hierarchical clustering[12] is another widely used technique which groups similar vertices into larger communities in SNA. Donetti and Munoz[6] have adopted this method by treating the Laplacian eigenvectors of the graph as a similarity measurement among vertices. The complexity is determined by the computation of all the eigenvectors, in  $O(n^3)$  time for sparse matrices. While it does not require us to specify the size or number of the communities beforehand, this method does not know when to stop the agglomerative process for the best division of the network.

Girvan and Newman have introduced a divisive approach[9][10] which includes the removal of the edges depending on their betweenness values. By iteratively cutting the edge with the greatest betweenness value, it uses the *Network Modularity*  $Q$  to get an optimized division of the network with  $O(m^3)$  time complexity[14]. Radicchi has proposed a similar methodology with GN[19] by using the edge-clustering coefficient as the new metric. Its time complexity is  $O(m^2)$  which is less than that of GN. To improve the computation efficiency, Clauset, Newman and Moore have also proposed a fast clustering algorithm[4] with  $O(n \log n)$  time complexity on sparse graph which uses a greedy strategy to get a maximal  $\Delta Q$  by merging pairs of nodes iteratively until it becomes negative. Pascal Pons and Matthieu Latapy[18] have designed another clustering algorithm based on the random walk method to measure the similarity between vertices. It also uses *Network Modularity*  $Q$  to determine when to stop the agglomerative process and has  $O(n^2 \log n)$  time complexity.

Other interesting algorithms include Jordi Duch and Alex Arenas's extremal optimization method proposed in[8] with  $O(n^2 \log n)$  time complexity, Aaron Clauset's method for finding local community structures in[5], the agent-based algorithm proposed by Ismail Gunes and Haluk Bingol in[11], as well as the approach based on the information theoretic framework in [20].

All these current algorithms are successful approaches for community detection with different backgrounds and applicable scopes. However, the actual social networks are large sparse graphs with significant overlapping among groups of vertices[16]. As a consequence, the betweenness based divisive algorithms will have very low computation efficiency while the fast agglomerative method[4] in general can not give a satisfactory division due to its local optimization strategy. Therefore, we follow a different track by presenting an algorithm which can generate a higher network modularity than the fast algorithm while perform more efficiently than the GN algorithm.

## 3. COMMUNITY DETECTION

In most social networks, triangles counts are usually high than they are in nonsocial networks, so our approach for community detection is based on the enumeration of all maximal cliques. Each group of the overlapping maximal cliques is regarded as a certain clustering kernel. We carry out an agglomerative process to assign the rest vertices to their closest kernels according to a proposed distance measure. In the end, the obtained fractional communities will be properly merged so as to prevent the network from being

divided into too small pieces.

### 3.1 Problem Formulation

Community detection in networks aims to find groups of vertices within which connections are dense, but between which connections are sparser. 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 set of all neighbors 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  $S$ .

*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.* Let  $P_0, P_1, \dots, P_{n-1}$  be the subgraph of  $G$  such that  $\forall P_i, P_j, V(P_i) \cap V(P_j) = \emptyset$ , and  $V(P_0) \cup \dots \cup V(P_{n-1}) = V(G)$ . For any pair of  $P_i$  and  $P_j$ , if  $|E(P_i)| > |N|_{P_i \cap P_j}|$ ,  $P_i$  is defined as a community of  $G$ .

*Definition 5.* Given vertex  $v_i \in V_M$ , define  $C_i = \{S | S \in M(C_G), v_i \in S\}$  to be the set of all maximal cliques containing  $v_i$ , and  $C$  the set of all  $C_i$ 's.  $\forall C_i, C_j \in C$ , if  $\frac{|C_i \cap C_j|}{|C_j|} \geq f$  which is a threshold to describe the extent to which  $C_i$  overlaps with  $C_j$ , we call  $C_j$  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 **kernel** of  $G$  and  $v_i$  is the **center** of  $C_i$ .

*Definition 6.* Let  $K$  be the set of all kernels in  $G$ .  $V_K = \{v_i | v_i \in k_j, k_j \in K\}$  is the set of all vertices covered by  $K$  and  $I_K = \bigcup (k_i \cap k_j), k_i, k_j \in K, i \neq j$  is the union of all the vertices that any pair of elements in  $K$  has in common.

### 3.2 Algorithm

*ComTector* first enumerates all maximal cliques in the giant component  $C_G$ . Because a maximal clique is a complete sub-graph, it is thus the densest community which can represent the closest relationship involving a single entity in the given network.

#### 3.2.1 Kernel Generation

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 one 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 statistics of the overlapping communities in networks. For any  $v_i, v_j \in V_M$ , if  $\frac{|C_i \cap C_j|}{|C_j|} \geq f$  ( $f$  is an empirical value), 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 kernel. Therefore, the larger that the size of  $C_i$  can be, the more likely that a kernel it would become. We rearrange all the elements of set  $C$  according to the descending order of

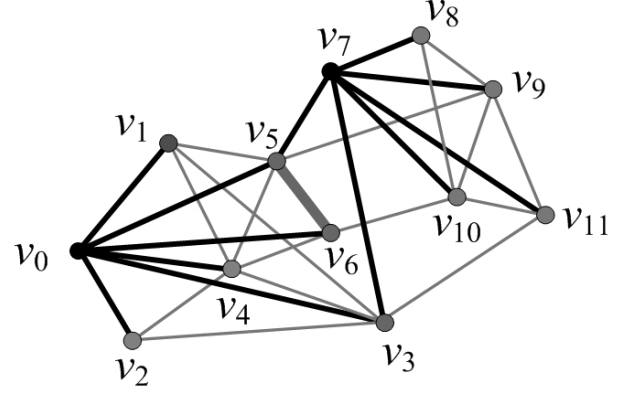


Figure 1: Overlapping Communities

their sizes and delete those elements whose sizes are smaller than 2, which means if  $C_i$  is a kernel,  $v_i$  must participate in at least two different relationships.

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.  $K$  is the set of all kernels.  $C_{i_0}$  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  $C_{i_0}$ . If  $C_{i_0}$  is not empty, it is put in  $K$ . Again, the element with the largest size is chosen from the rest elements of  $C$ , such as  $C_{i_n}$ . Remove it from  $C$ , remove all the elements contained by  $C_{i_n}$ , and delete each maximal clique that includes the centers of the left elements in  $C$  from  $C_{i_n}$ . If there is any maximal clique that contains the centers of the elements in  $K$ , it will also be deleted from  $C_{i_n}$  to get rid of unnecessary duplications. If  $C_{i_n}$  is not empty, it is put in  $K$ . The process 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.  $C_0 = \{\{v_0, v_1, v_4, v_5\}, \{v_0, v_1, v_3, v_4\}, \{v_0, v_2, v_3, v_4\}, \{v_0, v_4, v_5, v_6\}\}$  with  $v_0$  being as the center.  $C_1 = \{\{v_0, v_1, v_4, v_5\}, \{v_0, v_1, v_3, v_4\}\}$  whose center is  $v_1$ . Apparently  $C_1 < C_0$ ,  $C_1$  is not a kernel. 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  $C_7$ . Therefore,  $C_0$  and  $C_7$  are two different kernels respectively. The overall process is depicted in algorithm 1. Each element of  $K$  corresponds to the kernel of a possible community in  $G$ . In fact, the process to generate set  $K$  is similar to that of the classic k-means algorithm for finding the clustering center. People may argue that another very intuitive method to search for the kernels might depend on the degree of each vertex. All the vertices are sorted by the descending order of the vertex's degree, and the set of each vertex together with their neighbors is regarded as the element of set  $C$  for generating the kernels. Even though this method seems to be simple and straightforward, doing so can not help us to find the communities.

The reason is that the vertices contained in communities do not hold large degree necessarily. The fact that vertex  $v_i$  has a large degree only indicates that as a single entity  $v_i$  has many connections with others, yet it does not mean  $v_i$  is

---

**Algorithm 1** FilterOutKernels( $C, f$ )

---

```

1:  $K \leftarrow \emptyset$ 
2: sort  $C$  by the descending order of  $|C_i|, C_i \in C$ 
3:  $\{core\}$  stores the centers of the filtered out kernels}
4:  $core \leftarrow \emptyset$ 
5: for  $C_i \in C$  do
6:    $contained \leftarrow C_j, j \neq i, C_j < C_i$ 
7:    $independent \leftarrow k, k \neq i, C_k \not\subset C_i$ 
8:   delete  $C_i$  from  $C$ 
9:    $C \leftarrow C - contained$ 
10:  for  $s \in C_i$  do
11:    if  $s \cap (independent \cup core) \neq \emptyset$  then
12:      delete  $s$  from  $C_i$ 
13:    end if
14:  end for
15:  if  $C_i \neq \emptyset$  then
16:     $K \leftarrow C_i$ 
17:  end if
18:   $core \leftarrow v_i$ 
19: end for
20: return  $K$ 

```

---

involved in a large community. In our experiments, we have found that approximate 40 percent of the top 10 elements in set  $C$  has their centers' degrees also ranked top 10 on average. Most vertices in the communities of average size do not have large degree. Let  $v_k$  be the center of the element in  $C$  with the smallest size and  $v_d$  be the vertex with the maximum degree. We have found that the proportion of the number of vertex  $v$  such that  $|N(v_k)| \leq |N(v)| \leq |N(v_d)|$  to  $|V(G)|$  is 75% on average, which is far more than  $|C|$  and thus leads to a low efficiency for generating the kernels. Therefore, whether an individual would participate in a community also depends on how closely its neighboring vertices are connected with each other. This is another important motivation for us to use the overlapping maximal cliques to find the possible kernels.

The discovered communities will form a partition of the given network, which requires every pair of elements in  $K$  should not have any vertex in common. As a result, pairwise intersection among elements of  $K$  will be performed and all the common vertices will be put in a set  $I_K$ . For each  $v_i \in I_K$ , we use the *Freeman Relative Centrality* [21] to evaluate its positions in the correspondent kernels.

$$C_{RD} = \frac{|N(v_i)|}{n-1} \quad (1)$$

For a given sub-graph  $S_G$ , the larger  $C_{RD}(v_i)$  can be, the more important vertex  $v_i$  would become in  $S_G$ . This metric characterizes the distance between a given vertex and the characterizing kernels. The maximum  $C_{RD}$  value in  $S_G$  is denoted by  $C_{RDmax}$ . Every vertex  $v_i$  in  $I_K$  is then assigned to its closest kernel based on the value of  $C_{RD}(v_i)$ . Exceptionally, if  $v_i$  has the same maximum relative degree in two different kernels  $k_m$  and  $k_n$ ,  $v_i$  will be assigned to the one whose value  $|C_{RDmax} - C_{RD}(v_i)|$  is less than the other. Finally, even if  $k_m$  and  $k_n$  have the same  $|C_{RDmax} - C_{RD}(v_i)|$  value, which is a rather rare case,  $v_i$  is just assigned to one of them randomly. The whole procedure is given in algorithm 2. In figure 1,  $C_0$  and  $C_7$  share  $v_5$ . Since that the distances of  $v_5$  to  $C_0$  and  $C_7$  are 0.6 and 0.3 accordingly,  $v_5$  is thus assigned to  $C_0$  and removed from  $C_7$ .

---

**Algorithm 2** DeDuplication( $K$ )

---

```

1:  $I_K \leftarrow \emptyset$ 
2: for  $k_i \in K$  do
3:   for  $k_j \in K, i < j$  do
4:      $I_K \leftarrow I_K \cup (k_i \cap k_j)$ 
5:   end for
6: end for
7: for  $v \in I_K$  do
8:   remove  $v$  from all the kernels except for the one having
     the maximum distance
9: end for

```

---

### 3.2.2 Kernel-based Clustering

Once all the kernels does not have any vertex in common, each of them is regarded as a clustering center, and every vertex in  $V(C_G) - V_K$  will also be assigned to their closest kernels. We adopt a marking strategy to differentiate new vertices from the old ones. All vertices in  $V_K$  are first marked as old. Next, every new vertex in the set  $\bigcup N(k_i) - V_K, k_i \in K$  will be put in a tentative set  $V_E$ . In the third step, all vertices in  $V_E$  are assigned to their closest kernels and will be marketed as old. As a result, every kernel is now expanded. Again every new vertex in set  $N|_{V_E}$  is added to a tentative set  $V_E'$ . Then the vertices in  $V_E'$  are also assigned to their closest kernels and are marked as old. This process is repeated iteratively until the kernels can not be expanded any more, which is given in Algorithm 3.

---

**Algorithm 3** AssignVertex( $K$ )

---

```

1: for  $v_i \in V_K$  do
2:    $v_i$  is marked as old
3: end for
4:  $V_E \leftarrow$  vertices not marked as old in  $\bigcup N(k_i) - V_K$ 
5: while  $V_E \neq \emptyset$  do
6:   for  $v_i \in V_E$  do
7:     assign  $v_i$  to its closest kernel  $k_i$ 
8:      $v_i$  is marked as old
9:   end for
10:   $V_E' \leftarrow \emptyset, V_E' \leftarrow$  vertices not marked as old in  $N|_{V_E}$ 
11:   $V_E \leftarrow \emptyset, V_E \leftarrow V_E'$ 
12: end while

```

---

### 3.2.3 Modularity Optimization

Once the clustering process is finished, all the obtained sub-structures constitute the original division of the network. We adopt the *Network Modularity*  $Q$  to evaluate such division. Given the  $p \times p$  symmetric matrix  $\mathbf{e}$  whose element  $e_{ij}$  is the fraction of all edges in the network that link vertices in community  $i$  to vertices in community  $j$ , the row sums  $a_i = \sum_{j=0} e_{ij}$  represent the fraction of edges that connect to vertices in community  $i$ .  $Q$  is thus defined as

$$\sum_{i=0} (e_{ii} - a_i^2) \quad (2)$$

This quantity measures the fraction of within-community edges minus the expected value of the same quantity in a network with the same community divisions but random connections between the vertices. If the number of within-community edges is no better than random, we will get  $Q$

= 0. Values approaching  $Q = 1$ , which is the maximum, indicate strong community structure.

In our experiments, we have found that there exists a number of fractional communities which are derived from the corresponding tiny kernels compared with others. The hidden reason that causes to generate such fractional kernels is that for a specific  $C_i$ ,  $v_i$  may not become the "real" center of  $C_i$ . In other words, although each maximal clique in  $C_i$  contains  $v_i$ , it may also include many centers of other kernels, which means  $v_i$  is not the expected core figure and is just a normal individual participating in the social cycles of other dominating figures. Consequently, even though  $C_i$  would become a kernel, its size would be tiny, because the maximal cliques involving the centers of other kernels are all deleted from  $C_i$ . As a result, the communities resulted from these tiny kernels partition the network into too small pieces.

To address this problem, we propose two approaches to adjust the original division. In terms of the first one, we borrow the basic idea from Newman's fast algorithm to perform a local greedy optimization. We iteratively search for the changes  $\Delta Q$  resulted from the amalgamation of each pair of communities, choose the largest of them, and perform the corresponding amalgamation until  $\Delta Q$  becomes negative. The modularity value of the original division is  $Q_0$ . Suppose we first merge community  $i$  with  $j$  and the new community is denoted as  $(ij)$ . We can have

$$\Delta Q = \begin{cases} a_{ij} - a_{(ij)}^2 + a_i^2 + a_j^2 & i, j \text{ is connected} \\ 0 & \text{otherwise} \end{cases}$$

$$a_i = \frac{E_i}{m} \quad E_i \text{ is the number of edges in community } i \quad (3)$$

Here,  $a_i$  is the fraction of edges that connect vertices in community  $i$  and  $a_{ij}$  is the fraction of edges that connect vertices in community  $i$  to vertices in community  $j$ .  $m$  is the number of edges in graph  $G$ . Once the initial values of  $\Delta Q$  and  $a_i$  are obtained, we use the amalgamation process of the fast algorithm to increment  $Q_0$  by the largest  $\Delta Q$  until it becomes negative, which is shown in algorithm 4.

---

#### Algorithm 4 AdjustDivision(K)

---

- 1: calculate  $\Delta Q$  from pairs of connected communities
  - 2: **while** maximal  $\Delta Q > 0$  **do**
  - 3:   select the maximal  $\Delta Q$
  - 4:   join the pair of communities with the maximal  $\Delta Q$
  - 5:   update the  $\Delta Q$  matrix
  - 6: **end while**
- 

With respect to the second method, the fractional communities of the original division whose sizes are under the average level will be directly merged with the rests. In our experiments, we have found that the final modularity value obtained by this straightforward method is often close to that of the former with even less computational costs. Finally, the whole procedure of *ComTector* is given in algorithm 5.

#### 3.2.4 Performance Analysis

From the priori discussion, the enumeration of all maximal cliques in the giant component by using *Peamc*[7] will cost  $O(\Delta \times M_C \times Tri^2)$  in the worst case on a single processor, where  $\Delta$  is the maximal degree of  $G$ ,  $M_C$  is the size of the

---

#### Algorithm 5 ComTector(G)

---

- 1: Read Graph  $G$
  - 2:  $Com(G) \leftarrow$  all components of  $G$
  - 3:  $M(C_G) \leftarrow$  all maximal cliques in  $C_G$
  - 4: FilterOutKernels( $C, f$ )
  - 5: DeDuplication( $K$ )
  - 6: AssignVertex( $K$ )
  - 7: AdjustDivision( $K$ )
  - 8: return  $K \cup (Com(G) - C_G)$
- 

maximum clique and  $Tri$  is the number of all triangles in  $G$ . To find the kernel set  $K$ , we need to traverse all the elements of  $C$  whose size is larger than 2, which will cost  $O(M_C \times |C|^2)$ . The parameter  $f$  to identify whether one element of  $C$  is contained by another influences the number of kernels. Based on our experiments, we suggest it should be larger than 0.3. Since that  $V_M - V_K \approx V(G)$ , assigning the rest vertices in  $V_M - V_K$  will cost  $O(|K| \times |V(G)| \times I)$ , where  $I$  is the average times for the process to repeat until  $K$  is empty. In sparse graphs, we have  $|V(G)| \approx |E(G)|$ ,  $|C| < |V(G)|$ ,  $|K| \approx |C|$ ,  $|V(G)| < Tri^2 \ll |V(G)|^2$ , and  $\Delta \times M_C < |K|$ . Let  $C$  denote the number of the communities in the original division. The adjustment phase using modularity optimization will cost  $O(C \times \log C)$ . Because  $C \approx |K|$  and  $I$  has the value of 6 according to the small-world property, the overall cost will be  $O(C \times Tri^2)$  in the worst case.

## 4. COMMUNITY NAMING MECHANISM

Once all possible communities have been detected successfully; one may further ask the question like what the discovered communities could be? Or since a community covers some kind of common relationship shared among the involved entities, could we find out what the relationship is indeed. Here, we propose a general method to describe a specific community by combining the topology information with the natural attributes of the contained entities. The process is much like making a profile for the given community. Suppose we are given two data sets: one is the entity relational data which is modeled by graph  $G = \{V, E\}$  with  $V$  being the entity (vertices) set and  $E$  the relation(edges) set respectively. The other is the entity attribute data  $R = \{a_0, a_1, \dots, a_{n-1}, r_c\}$ , where  $a_0, a_1, \dots, a_{n-1}$  are the natural attributes and  $r_c$  is the community index number of each discovered community by the previous process. Based on  $G$  and  $R$ , our task is to find some key attribute values to characterize the given community.

Technically, there exist two challenges for the naming task itself. On one hand, data gathering and synthesis of  $G$  and  $R$  in real applications are rather complicated. Sometimes there would be even no enough attributes for the entities. On the other hand, the classic approach to give a profile for a group of entities is known as *rule induction*. However, we are often unable to use this method directly after the communities are discovered even if the entities have enough attributes because of the losing of the important topological information about the internal structure. Therefore, our naming approach consists of two steps. In the first step, we are going to find the central entities of each community. In the second step, several naming mechanisms are proposed according to the type and number of the attribute values hold by the central entities.

## 4.1 Central Entity Resolution

By taking the self-similarity[22] and scale-free properties into account, we assume that most sub-graphs extracted from the social networks often have central entities in the same way like the whole social 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 collaboration network, the central entity of a research team could be a common advisor who originally started the research area, while in telecommunication call network, the central person in a group of frequently contacted people may be the chief officer in a department. Therefore, the characteristics of the core figures greatly influence the properties of the community.

Many methods in literature have been proposed to quantify the centrality of an entity based on the topology of a given graph, such as *degree*, *relative centrality*, *betweenness* and *PageRank*. These approaches can bring us an unique value  $U(v)$  for each vertex  $v$ . Generally speaking, the greater the value is, the more important the vertex will be in the community. How to find out the central vertices according to these values is what we actually need to do. If the value of some vertex is significantly greater than those of the other vertices, or, if there exists a large gap between two neighboring vertices, such as  $v$  and  $w$  with  $U(v) > U(w)$ , after we rank all of them according to the centrality measurement in the descending order, the vertices, whose value is more greater than that of  $v$ , are the central entities of the community. Algorithm 6 describes this procedure. In figure 2a, the

---

### Algorithm 6 CentralEntityResolution( $C, p$ )

---

```

1:  $\{C$  is the given community and  $p$  is a threshold value $\}$ 
2: Calculate the centrality of each vertex  $v_i$  in  $C$ 
3:  $v_0, v_1, \dots, v_{n-1}$  is arranged by the descending order of
   their centrality  $c_0, c_1, \dots, c_{n-1}$ 
4:  $i \leftarrow 0$ 
5: while  $i < n - 1$  do
6:   if  $\frac{c_i - c_{i-1}}{c_i - c_{n-1}} > p$  then
7:     return  $\{c_k | 0 \leq k \leq i\}$ 
8:   else
9:      $i \leftarrow i + 1$ 
10:  end if
11: end while

```

---

relative centrality values of vertices "PeiDa Ye" and "Ming Zhang" are 0.86 and 0.57, which ranks the first and the second respectively. The minimum relative centrality value of this community is 0.03. Since the threshold  $p$  actually determines the distance between two neighboring values and the third ranking relative centrality value is only 0.2 which is far less than the priori two, the vertices "PeiDa Ye" and "Ming Zhang" are thus the central entities which could be used to define this community. By contrast, in figure 2b, almost all vertices have similar relative centrality values, so we use them all together to describe the community.

## 4.2 Naming Approach

Based on the data sets  $G$  and  $R$ , we propose the following approaches to name a specific community. In terms of the first one, set  $R$  only has one element, namely, there exists only one attribute that the central entities could have. In such case, the union set of the central entities' attribute val-

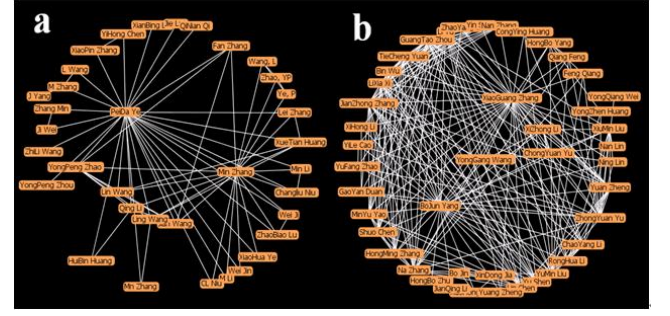


Figure 2: Central Entity Resolution  $p = 0.6$

ues is used to define the given community. With respect to the second one, there exist as many attributes as possible to reflect various aspects of an entity. We use an attribute selection technique to find the most informative characteristics of the central entities. Here, an attribute can be used to name the community if its value of the central entity is not significantly different from that of the non-central entities. A representative value of this attribute among the central entities is calculated according to its type (discrete attribute such as sex and address or continuous attribute such as age and incoming). Otherwise, if the values of an attribute are absent among all central entities, it can not be used to name the community. This process is depicted in algorithm 7. The representative value of attribute  $a$  hold by the central entities is denoted as  $C_a$ , the frequency of  $C_a$  among all the entities as  $F_{C_a}$ , the average value of  $a$  as  $A_a$ . To sum up, the above two naming mechanisms should be ap-

---

### Algorithm 7 NamingCommunity( $R, Center, p_1, p_2$ )

---

```

1:  $\{R$  is the attribute set,  $Center$  is the community set,
    $p_1, p_2$  are two threshold values $\}$ 
2: if  $|Center| > 1$  then
3:   for each attribute  $a \in R$  do
4:     if  $a$  is discrete then
5:        $C_a \leftarrow$  the most frequent value of  $a$  among the
         central entities
6:     else
7:        $C_a \leftarrow$  average value of  $a$  over the central entities
8:     end if
9:   end for
10: else
11:   for each attribute  $a \in R$  do
12:      $C_a \leftarrow$  value of  $a$ 
13:   end for
14: end if
15: for each attribute  $a \in R$  do
16:   if  $a$  is discrete and  $F_{C_a} > p_1$  then
17:      $a$  is selected as the key attribute
18:     return  $C_a$ 
19:   else
20:     if  $\frac{C_a - A_a}{A_a} < p_2$  then
21:        $a$  is selected as the key attribute
22:       return  $C_a$ 
23:     end if
24:   end if
25: end for

```

---

plied to different practical scenarios. If the entity only has



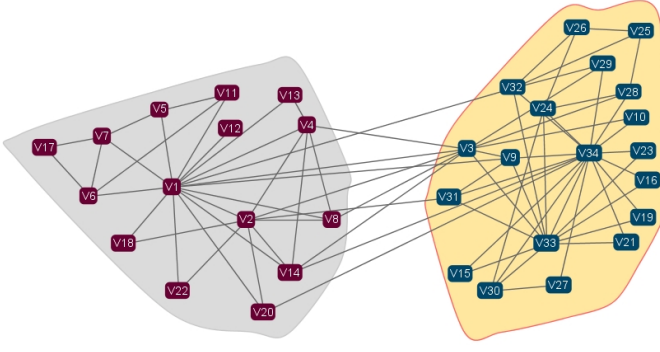


Figure 3: Zachary Karate Club

one attribute, the value of this attribute hold by the central entity is used to define the community. If a community has a few central entities, each of which has several attributes, we could use the frequent attribute values to name this community. Suppose that there are  $n$  vertices with  $m$  attributes each, and  $c$  central entities in the given community. The time complexity to calculate the centrality, which depends on the algorithm of the measurement, is  $O(T)$ , so the whole naming procedure will cost  $O(T + cm + nm)$ .

## 5. EXPERIMENTAL RESULTS

In this section, we present a number of applications to which our *ComTector* is applied. The algorithm is first tested on the Zachary Karate Club[27], American College Football[27], and Scientific Collaboration[28] networks. In each case we find that our algorithm reliably detects the known structures. Based on the experimental results, we have a detailed discussion about the optimization of parameter  $f$ .

In the end, *ComTector* is further tested on the large Telecommunication Call networks (T.C.) to illustrate the global structural properties of the large-scale social networks. All experiments are done on a single PC (3.0GHz processor with 2Gbytes of main memory on Linux AS3 OS). We use a very efficient parallel algorithm *Peamc* to enumerate all maximal cliques. In the telecommunication call networks, *Peamc* runs on the DAWN Cluster (3.2GHz Processor with 2Gbytes of main memory on each node, Linux AS3 OS) by using 20 processors, while for the other networks, *Peamc* just runs on a single processor.

### 5.1 Zachary Karate Club

Zachary Karate Club is one of the classic studies in social network analysis. Over the course of two years in the early 1970s, Wayne Zachary observed social interactions between the members of a karate club at an American university. He built network of connections with 34 vertices and 78 edges among members of the club based on their social interactions. By chance, a dispute arose during the course of his study between the club's administrator and the karate teacher. As a result, the club splits into two smaller communities with the administrator and the teacher being as the central persons accordingly. Figure 3 shows the detected two communities by *ComTector* which are exactly matched with the result of Zachary's study.

### 5.2 American College Football

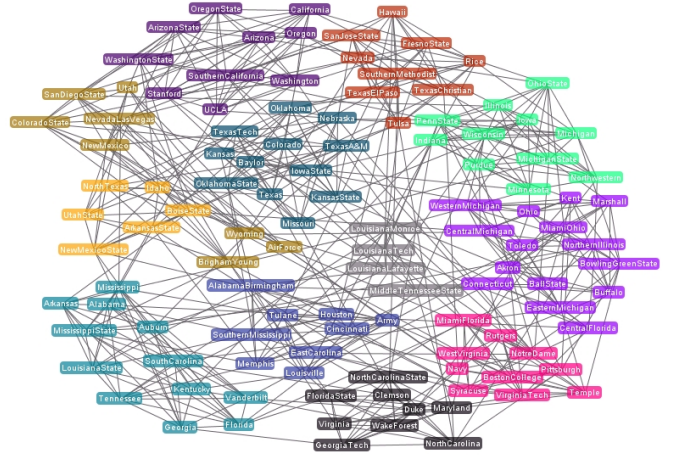


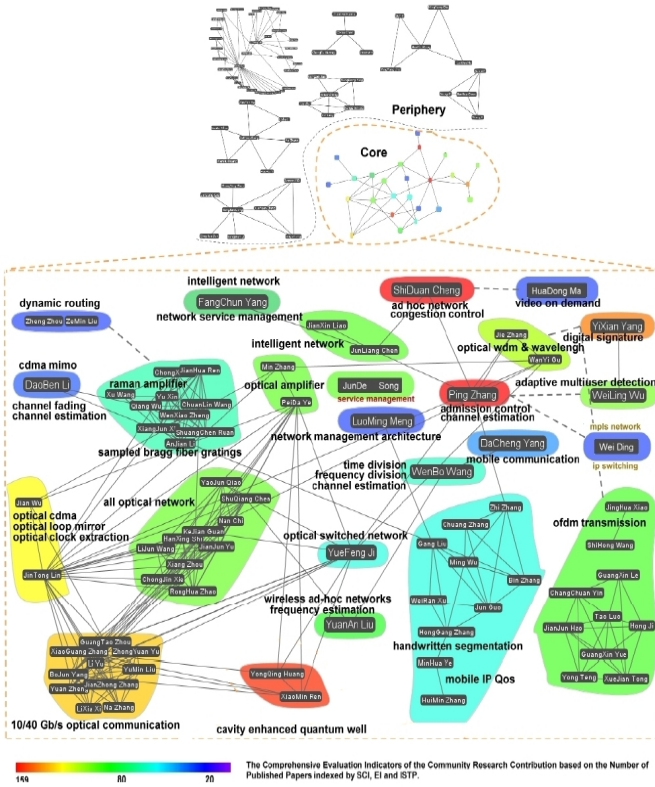
Figure 4: American College Football

As another test of *ComTector*, we turn to the network of American College Football. This network represents the schedule of Division I games for the 2000 season. It consists of 115 vertices and 616 edges which are the representations of football teams and regular season games among them respectively. During the 2000 season, all of the 115 teams are divided into 12 conferences containing around 8 to 12 teams each. Games are more frequent between members of the same conference than between members of different conferences. Apparently, each conference can be considered as one community of the network.

Applying *ComTector* to this network, we have found that it identifies the conference structure with a success percentage of 93.8% shown in Figure 4. Almost all teams are correctly grouped with the other teams in their conferences. The few cases where the algorithm fail actually correspond to the scheduling of the game. For example, *LouisianaMonroe* team, *MiddleTennesseeState* team and *LouisianaLafayette* team in *Sun Belt* are misclassified with *LouisianaTech* in *Western Athletic* as a single group. *Independents* conference is broken into two groups with its team *UtahState* being misclassified into *Western Athletic*. In all other respects however it performs remarkably well.

### 5.3 Scientific Collaboration Network

The data of the collaboration network is obtained according to the 1990 published papers from the year 1998 to 2005 indexed by SCI, EI and ISTP in Beijing University of Posts and Telecommunications(BUPT). Each author corresponds to a vertex of the network and there is an edge between two vertices if the two correspondent authors have collaborated in a paper. A great deal of work has gone into disambiguation of similar names, so co-authorship relationships are relatively free of name resolution problems. We split the title of each published paper of a given author into some key phrases by getting rid of the unnecessary prepositions, articles as well as several nouns and adjectives that frequently appear with very general meaning, such as *system*, *algorithm*, *new* and etc. These key phrases are then used as the value of the research area attribute hold by each author. By using our naming approach, we present a map of scholarship



**Figure 5: All Discovered Communities and the Descriptions**

in Figure 5. Each community in the **Periphery** area is an independent small component of the network, and the **Core** area corresponds to the giant component with each vertex being the representation of the corresponding community. The edge between two adjacent vertices indicates that two communities have some overlapping parts. By zooming onto the vertices of the **Core** area, we come to the detailed description of each specific named community. In this magnified picture, the color of each community is the same as that of the vertex in the **Core** area. The vertices in every community are the central persons being as the representatives of the research group. The solid lines among these vertices show that the central persons of the given communities have collaborated together, while the dashed lines mean that the rest persons other than the central ones of the correspondent communities have once collaborated with each other. The color of each community from red to deep blue indicates the research contribution by the descending order according to the number of the published papers indexed by SCI, EI and ISTP.

Another scientific collaboration network is the network of coauthorships between scientists posting preprints on the Condensed Matter E-Print Archive between Jan 1, 1995 and March 31, 2005. It consists of 39577 vertices and 175693 edges. Table 1 gives the numerical results on both of the two networks. In fact, for the GN algorithm, the computation of the maximum edge betweenness value makes it inefficient on large-scale networks. With respected to Newman's fast algorithm, it uses a local optimization policy by searching for the maximal increment of the *Network Modularity*  $Q$ . However,

<b>Table 1: Scientific Collaboration Networks</b> BUPT (1667 vertices, 4487 edges)			
Algorithm	Result	$Q$	Time
GN	79	0.85	403s
Newman Fast	85	0.43	2.4s
ComTector	81	0.83	1s
E-Print Archive (39577 vertices, 175693 edges)			
GN	n/a	n/a	>24h
Newman Fast	1363	0.31	3.7h
ComTector	1056	0.65	2.2h (25s clique time)

<b>Table 2: Erdős Collaboration Network</b>			
Erdős 97 (5482 vertices, 8972 edges)			
Algorithm	Result	$Q$	Time
GN	n/a	n/a	>5h
Newman Fast	68	0.43	29s
ComTector	49	0.69	23s (1s clique time)
Erdős 98 (5816 vertices, 9505 edges)			
GN	n/a	n/a	>5h
Newman Fast	69	0.34	35s
ComTector	38	0.69	26s (1s clique time)
Erdős 99 (6094 vertices, 9939 edges)			
GN	n/a	n/a	>5h
Newman Fast	66	0.35	40s
ComTector	38	0.69	27s (1s clique time)

the changes of  $Q$  is not a monotonous process. Although fast algorithm can get a maximal value of  $Q$ , this value may be too far from the maximum one. Compared with fast algorithm, which starts its clustering process from each single vertex, the process of *ComTector* that generates all the possible kernels by using the overlapping of the maximal cliques directly locates the densest part of each community, which can quickly bring about a more faster convergence. Consequently, in large sparse graphs, *ComTector* usually performs better than the fast algorithm while becomes more efficient than the GN algorithm.

Paul Erdős was one of the most prolific mathematicians in the history, with more than 1500 papers to his name. He is also known as a promoter of collaboration and as a mathematician with the largest number of different co-authors, which was a motivation for the introduction of the Erdős number collaboration network[29]. Patrick Ion (Mathematical Reviews) and Jerry Grossman (Oakland University) collected the related data (Grossman and Ion, 1995, Grossman, 1996) which are updated annually through 1997 to 1999. Experimental results are given in table 2.

## 5.4 Telecommunication Call Network

In the algorithm, the possible values of parameter  $f$  affect the ultimate outcome of the partition. We adopt Newman's  $Q$  modularity to evaluate the strength of the detected community structures.  $f$  determines the kernels' number in the given network, which in turn has an influence on the  $Q$  value. Figure 6 shows this kind of relation in the American College Football, and the Scientific Collaboration network respectively. If  $f$  is too large, it will cut the network into smaller pieces. For each community  $i$ ,  $e_{ii}$  tends to be small and  $a_i$  is relatively large, which further causes  $Q$  to decrease. As a result, in Figure 6, we see when  $f \in (0.3, 0.5)$ ,  $Q$  often



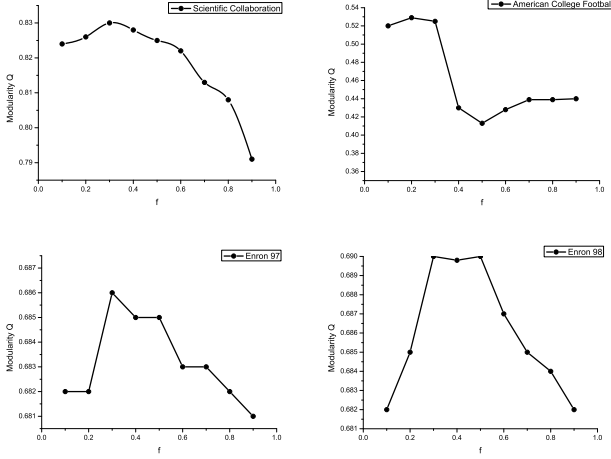


Figure 6: Network Modularity  $Q$  and  $f$

reaches its maximum value on average.

According to the priori analysis, *ComTector* is further tested on two large telecommunication call networks, which are built upon the datasets in a city and in a province within the period of one month from a Telecom Operator in China. The first one consists of 512024 vertices and 1021861 edges, and the second one includes 845750 vertices and 1544834 edges. We regard each subscriber as a single vertex and two vertices will share an edge if the subscribers have once contacted with each other by their mobile phones. Because each individual can have connections with others for various reasons and relationships, the telecommunication call networks often have significant overlapping among the detected communities. We have detected 28033 and 2171 communities in these two networks with 0.60 and 0.64  $Q$  value respectively within the period of 2h, while neither GN nor Newman Fast can generate satisfactory results in an acceptable time frame.

Looking at the large communities in the networks, we have found that they often tend to consists of people who have close consumption levels, similar ages or live in the same areas. Figure 7 gives the description results on the telecommunication network. In our studies,  $p_1 = 0.6$  and  $p_2 = 0.4$ , which are the empirical values and the relative degree is used as the centrality measurement. In Community 70, we can see that all subscribers share the same home address (HOME\_ZIP\_CODE) and their consumption levels are also similar according to their average bill fee (BILL\_FEE\_Mean). Meanwhile, the subscribers in community 61 correspond to a typical portion of the client market for their close consumption levels and the similar age 32, and Community 64 is a young social cycle apparently. To some extent, these obtained community structures and the correspondent common factors are useful clues for the Telecom Operator to adjust their client market policies.

## 6. CONCLUSIONS

In this paper, we have followed a different track by proposing a new method *ComTector* for the community discovery in large-scale social networks. Directly based on the overlapping nature of communities in our real world, *ComTector* can extract meaningful results on networks whose commu-

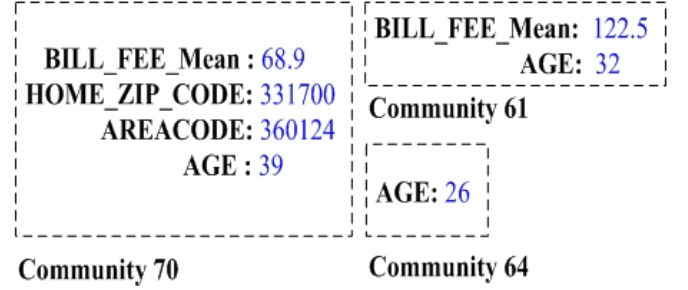


Figure 7: Some defined communities in the telecommunication call network

nity structures are known before. Our method consists of three critical steps. For the first step, we adopt a significantly efficient algorithm to enumerate all maximal cliques in the given network. The overlap of several maximal cliques corresponds to various relationships and connections each individual may participate in directly. The gathering of maximal cliques forms the kernels of every potential community. For the second step, we use an agglomerative technique to iteratively add the left vertices to their closest kernels based on the relative degree matrix. For the third step, the originally obtained clustering results will be adjusted by merging pairs of fractional communities to achieve a better Network Modularity. The finally obtained community structures together with other components constitute the ultimate partition of the network.

We have demonstrated the efficiency and utility of *ComTector* over a number of practical examples. Experimental results on true life networks show that *ComTector* can extract meaningful communities that are agreed with both of the objective facts and our intuitions. Further, we apply *ComTector* to analyze networks whose structures are otherwise difficult to understand, such as scientific collaboration and telecommunication call network. Moreover, we also propose a general naming method to describe and explain the discovered communities by combining the topological information with the entity attributes. Going through the detailed descriptive information concerning each community to the high level abstracted map of their organization, we are able to have a better understanding about the global properties of the whole network.

For the future work, we will continue our research by focusing on the evolution of the community as well as the network backbone by using time series analysis. We will also search for more refined theoretical models to describe the relationship between the knowledge diffusion mechanism and the community membership in social networks to have a deeper understanding of the network dynamics from both of the micro and macro perspectives.

## 7. ACKNOWLEDGMENTS

We thank Yan Xiong, Qi Ye, Chaoqun Xu, and Yerong Wu for their generous help for gathering the collaboration data, and Yi Wang, Xiangang Zhao, for the valuable discussions.

## 8. REFERENCES

- [1] J. Abello, M. Resende, and S. Sudarsky. Massive quasi-clique detection. In *5th Latin American*

- Symposium on Theoretical Informatics*, pages 598–612, 2002.
- [2] C. Bron and J. Kerbosch. Finding all cliques of an undirected graph. *Communications of the ACM*, 16:575–577, 1973.
  - [3] O. V. Burton. *Computing in the Social Sciences and Humanities*. University of Illinois Press, 2006.
  - [4] A. Clauset, M. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(066111), 2004.
  - [5] A. Clauset, M. Newman, and C. Moore. Finding local community structure in networks. *Physical Review E*, 72(026132), 2004.
  - [6] L. Donetti and M. Miguel. Detecting network communities: a new systematic and efficient algorithm. *Journal of Statistical Mechanics*, pages 100–102, 2004.
  - [7] N. Du, B. Wu, and B. Wang. A parallel algorithm for enumerating all maximal cliques in complex networks. In *The 6th ICDM2006 Mining Complex Data Workshop*, pages 320–324. IEEE CS, December 2006.
  - [8] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 72(027104), August 2005.
  - [9] M. Girvan and M. Newman. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, June 2002.
  - [10] M. Girvan and M. Newman. Finding and evaluating community structure in networks. *Physical Review E*, 69(026113), 2004.
  - [11] I. Gunes and H. Bingol. Community detection in complex networks using agents. In *AAMAS2007*, 2007.
  - [12] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*, 2nd ed. Morgan Kaufmann Publishers, 2006.
  - [13] R. Milo and S. Itzkovitz. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.
  - [14] M. Newman. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter*, 38:321–330, 2004.
  - [15] M. Newman. Modularity and community structure in networks. *PNAS*, 103(23):8577–8582, June 2006.
  - [16] G. Palla, I. Dernyi, and I. Farkas. Uncovering the overlapping community structure of complex network in nature and society. *Nature*, 435:814–818, June 2005.
  - [17] J. Pei, D. Jiang, and A. Zhang. On mining cross-graph quasi-cliques. In *The 12th ACM SIGKDD*, pages 228–237, 2006.
  - [18] P. Pons and M. Latapy. Computing communities in large networks using random walks. In *ISCIS2005*, pages 284–293, 2005.
  - [19] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *PNAS*, 101(9):2658–2663, March 2004.
  - [20] M. Rosvall and C. T. Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *PNAS*, 104(18):7327–7331, May 2007.
  - [21] J. Scott. *Social Network Analysis: A Handbook*. Sage Publications, London, 2002.
  - [22] C. Song, M. Havlin, and H. Makse. A self-similarity of complex networks. *Nature*, 433(7024):392–395, 2005.
  - [23] A. Vázquez, R. Dobrin, S. Sergi, J. Eckmann, Z. Oltvai, and A. Barabási. The topological relationship between the large-scale attributes and local interaction patterns of complex networks. *PNAS*, 101(52):17940–17945, 2004.
  - [24] S. Wasserman and K. Faust. *Social Network Analysis*. Cambridge University Press, Cambridge, 1994.
  - [25] B. Wu and X. Pei. A parallel algorithm for enumerating all the maximal k-plexes. In *PAKDD07 Workshops*, May 2007.
  - [26] Z. Zeng, J. Wang, and G. Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *The 12th ACM SIGKDD*, pages 797–802, 2006.
  - [27] <http://www-personal.umich.edu/~mejn/>
  - [28] <http://vlado.fmf.uni-lj.si/pub/networks/pajek/data/gphs.htm>
  - [29] V. Batagelj and A. Mrvar. Some Analyses of Erdos Collaboration Graph. <http://vlado.fmf.uni-lj.si/pub/networks/doc/erdos/erdos.pdf>