2008 14th IEEE International Conference on Parallel and Distributed Systems

Mining Community Structures in Peer-to-Peer Environments

Ching-Hua Yu, Wen-Chih Peng,
National Chiao Tung University
Hsinchu, Taiwan, ROC
{tsinghua, wcpeng}@cs.nctu.edu.tw

Wang-Chien Lee
The Pennsylvania State University
State College, PA 16801, USA
wlee@cse.psu.edu

Abstract

Most social networks *exhibit* community structures*, in which nodes are tightly connected to each other within a community but only loosely connected to nodes in other communities. Researches on community mining have received a lot of attention; however, most of them are based on a centralized system model and thus not applicable to the distributed model of P2P networks. In this paper, we propose a distributed community mining algorithm, namely* ***A****synchronous* ***C****lustering and* ***M****erging scheme (ACM), for computing environments. Due to the dynamic and distributed nature of P2P networks, The ACM scheme employs an asynchronous strategy such that local clustering is executed without requiring an expensive global clustering to be performed in a synchronous fashion. Experimental results show that ACM is able to discover community structures with high quality while outperforming the existing approaches.*

Keywords:Distributed node clustering, connected graph, peer-to-peer networks.

1. Introduction

Peer-to-peer (P2P) networks and systems have been received well by internet users in the past few years [10] [6] [11]. Via P2P networks, people exchange messages and data, share resource and load, contribute software and hardware to each other. While various information have been flowing among peer nodes and their users, social interdependencies (e.g., common interests, friendships, trust, and so on) may have been established implicitly among peer nodes. Therefore, P2P networks can be naturally treated as social networks.

We consider a ***peer-to-peer social network*** to consist of peer nodes and logical links which represent the above-mentioned interdependent connections. Conveniently, a social network can be denoted as a ***graph*** $G(V, E)$, where V = $\{V_1, V_2, ..., V_n\}$ is a set of vertices and E = $\{E_1, E_2, ..., E_n\}$ is a set of edges, by mapping nodes to vertex and connections to edges, respectively. A ***cluster*** is a set of nodes denoted as C = $\{V_1, V_2, ..., V_n\}$, while a ***clustering*** refers to partition of a graph into a set of clusters denoted as Cl = $\{C_1, C_2, ..., C_k\}$. One important property of social network is ***community structure***, defined as a clustering of a graph, in which nodes within a cluster are connected tightly while nodes in different clusters are connected only loosely. ***Mining community structures*** can help discover rich information inherent in social networks, e.g., common attributes and characteristics of nodes within the same clusters, differences amongst clusters, interdependent strength between any two nodes, also prediction of potential interdependency between nodes, topics concerned by a group of peer nodes, etc.

This paper proposes a distributed scheme to discover the community structure, allowing nodes in a peer-to-peer network to form communities automatically. Although some researches have been conducted on community mining or clustering [7] [4] [3] [9], most of them are based on a centralized system model and thus not applicable to the distributed model of P2P networks. Generally, P2P networks have the following characteristics:

1. Peer nodes do not have a global view of the network and only connect with a few neighbors;
2. Due to interactions and join/leave of peers, P2P networks are highly dynamic;
3. Global flooding and synchronous mechanisms in P2P networks are inefficient;

To overcome challenges resulted from the above characteristics, we adopt an asynchronous strategy to implement a two-phase approach for mining community structures. Explicitly, in the first phase, a node will determine whether it should become a "***cluster originator***" to initiate a local clustering based on connecting conditions with its neighbors. Specifically, a spontaneous originator examination process is developed to obtain amendable originators. Thereafter, in the second phase, cluster originators will evaluate the overlapping conditions between clusters and determine whether the clusters should be merged or not.

Clusters may be recursively merged until a terminating criterion is satisfied. In short, we present a distributed scheme, namely *Asynchronous Clustering and Merging* (*ACM*), to discover the community structure. Note that ACM is executed in a pure peer-to-peer manner, without relying on any super-peer node or other global information, and thus very suitable for dynamic P2P environments.

G	the graph of the corresponding network
V	the set of vertices in the graph, corresponding to the nodes in the network
E	the set of edge in the graph
$W(V_i, V_j)$	the edge weight of $E(V_i, V_j)$
$Nbr(V_i)$	the set of the neighbors of V_i
$Cluster(V_i)$	the cluster which V_i is belong to
C_i	the ith cluster in a graph after clustering
Cl	the set of clusters in a graph after clustering, a clustering of the graph
$Degree(V_i)$	the number of edges connected to V_i
V_o	the initial cluster originator

Table 1. The descriptions of notations

2 Preliminaries

In the following, we first review a closely related work, discuss the quality measurement of clustering, and then formulate our problem. To facilitate the presentation of this paper, some notations are summarized in Table 1.

2.1 Node Clustering

Mining P2P community structures is similar to clustering peer nodes in P2P networks. A connectivity-based node clustering algorithm (CDC) [5] has been proposed to partition a P2P network into several clusters such that nodes in the same cluster are similar and nodes belonging to different clusters are dissimilar.

The CDC scheme is basically a randomized method. With different originators selected, the clustering results may be different. Although there are some conditions to check for good originators, these conditions have some problems themselves. One example can be found in Fig. 1(a). Because V_1 has the highest $TwoHopProb$ in the network, with the restriction of vicinity check, V_1 could become the only originator in the CDC scheme, and results in a worse clustering as showed in Fig. 1(a). Besides, because the CDC scheme requires the whole network to synchronously perform a clustering, it is challenging to work in a pure peer-to-peer environment efficiently.

2.2 Clustering Quality

We use the following measure to evaluate the clustering quality of a community structure. Given a P2P network

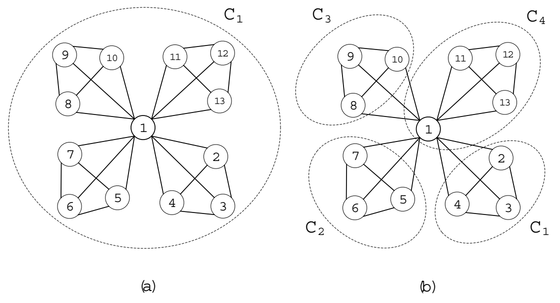


Figure 1. An example of community structures in P2P networks.

modelled as a graph G(V, E), and a community structure as a clustering Cl of G, the number of nodes neighboring V_i that are out of $Cluster(V_i)$ is expressed by $out(V_i)$, and the number of nodes in $Cluster(V_i)$ that are not neighbors of V_i is expressed by $in(V_i)$. Similar to [8] [5], the coverage measurement is defined as follows:

$$Cov(G, Cl, V_i) = 1 - \frac{out(V_i)+in(V_i)}{\|Nbr(V_i)\cup Cluster(V_i)\|}$$

The quantity falls in [0..1], where the value 1 can be achieved only if $Nbr(V_i)$ and $Cluster(V_i)$ are the same.

In light of the definition of the coverage measurement, the average coverage measurement for a graph G is formulated as follows:

$$avgCov(G, Cl) = \frac{\sum_{V_i \in V} Cov(G,Cl,V_i)}{\|V\|},$$

which is defined as the quality of clustering C_l in G.

We take the graph $G(V, E)$ in Fig.1 for example. In Fig.1(a), $Cl_1 = \{C_1\}$. Since none of $Nbr(V_i)$ outside C_1, $out(V_i) = 0, \forall V_i \in V$. Because each node of C_1 is connected to V_1, $in(V_1) = 0$. For each $V_i, i = 2, ..., 13$, there are 9 nodes leaving unconnected in C_1. As a result, $in(V_i) = 9$. Therefore, $Cov(G, Cl_1, V_1) = 1 - \frac{0+0}{13}$, and $Cov(V_i) = 1 - \frac{0+9}{13}, i = 2, ..., 13$. Hence $avgCov(G, Cl_1) = 0.361$. In Fig.1(b), we get $Cov(G, Cl_2, V_1) = \frac{4}{13}$, $Cov(G, Cl_2, V_i) = 1, i = 2, 3, 4$, and $Cov(G, Cl_2, V_j) = \frac{3}{4}$, $j = 5, ..., 13$. Hence $avgCov(G, Cl_2) = 0.774$.

It can be seen that $avgCov(G, Cl_2)$ is much higher than $avgCov(G, Cl_1)$. So Cl_2 is regarded as a better clustering.

2.3 Problem of Mining Community Structures

Given a P2P social network which is highly distributed and dynamic, we are attacking the challenging problem of discovering community structures in such a network. By denoting the social network as a graph $G(V, E)$, where the set of vertices V represents the set of nodes, and the set of edges E is the set of connections, a community structure is defined as a clustering Cl, where $avgCov(G, Cl)$ should be as high as possible. Therefore, the edges inside a cluster are dense, and edges between clusters are loose. By mapping each cluster to a community, the interactions inside a community are strong, and the interactions between communities are weak. Due to the dynamic and distributed characteristics of P2P environments, we aim at developing a new distributed asynchronous algorithm to discover community structures without assuming any global information.

3 Algorithm ACM: Asynchronous Clustering and Merging

In this section, we propose our scheme Asynchronous Clustering and Merging (ACM) to extract community structures in P2P networks. The ACM scheme consists of two phases: the clustering phase and the merging phase. In the clustering phase, each node should first check whether itself should be an originator or not. Such a procedure is called spontaneous originator examination (SOE). Once each peer performs SOE, the set of originators will be determined. Then, a Markov process starting from the originators will disseminate the reach probabilities among peers. Peers in the network accumulate the reach probabilities from the originators. Finally, local clustering is performed. In the clustering phase, we could have rough clustering results. In the merging phase, these rough clusters will be merged according to the merging criteria that will be described later.

3.1 The Local Clustering Phase

In the clustering phase, we select originators and then perform local clustering subsequently.

3.1.1 Determining Originators

A good cluster should have intensive interactions among peer nodes within the cluster, and the only information in each peer initially is the peer's local connectivity. Thus, a good originator should be those nodes with a larger number of edges. Thus, originators could be roughly selected according to the local connectivity of each peer node. Practically, each peer could decide whether it should be a candidate originator (referred to as *quasi-originator*) or not. Having strong mutual relations with many peers in a cluster is a necessary criterion to become originators. In a simple graph G, we let a node V to be a quasi-originator if the number of the node's edge reaches a predefined threshold, *QuasiSOEThreshold*. Notice that the threshold also restricts the minimum size of a cluster, where the size of a cluster is defined by the number of nodes included; no cluster with nodes less than *QuasiSOEThreshold*+1 could be formed. Therefore, if a group of nodes have enough relations, at least one of them would become a quasi-originator. The next step of a quasi-originator is to probe one-hop neighbors, so as to decide whether it could become an originator

Algorithm 1 SOE Check: Executed by V_o to check to be an Originator

```
1:  if The number of edges of $V_o \geq$ QuasiSOEThreshold then
2:      $V_o$ has enough connective coverage nearby
3:      $V_o$ becomes a Quasi-Originator. Send SOEcheckMsg to the neighbors
4:      Create an SOEcheckMsg
5:      SOEcheckMsg.OID = $V_o$
6:      SOEcheckMsg.list = $\{V_o, Nbr(V_o)\}$
7:      for each node $V_i \in Nbr(V_o)$ do
8:          Send SOEcheckMsg to $V_i$
9:      end for
10: end if
11: Wait for SOEreplyMsg
12: While $V_o$ receive SOEreplyMsg,
        $avgRatio = \frac{\sum_{\forall V_i \in Nbr(V_o)} SOEreplyMsg(V_i).Ratio}{|Nbr(V_o)|}$
13: if avgRatio $\geq$ SOEThreshold then
14:     $V_o$ becomes an Originator and initialize a local clustering process.
15: end if
```

or not. At the beginning, the quasi-originator sends its connected list (i.e. a list of its connected nodes) to the neighbor nodes. While receiving the message, the neighbor nodes compare the list with their own connected list and send back the overlapping ratio, where the overlapping ratio is defined as $\frac{|Nbr(V_o) \cap Nbr(V_i)|}{|Nbr(V_i)|}, \forall V_i \in Nbr(V_o)$. The quasi-originator averages the overlapping ratio of its neighbors. If the average ratio reaches a threshold, *SOEThreshold*, the quasi-originator becomes an originator and then performs a local clustering process. The average ratio here is an approximate coverage measurement of the cluster centered by the originator within one hop, to ensure the lower-bound coverage of the cluster. Thus, the threshold also limits the minimum coverage of cluster; i.e., clusters with low coverage could not be formed.

Msg.OID	the originator ID
Msg.list	the list of the originator's neighbors
Msg.Ratio	the overlapping ratio returned to the originator
avgRatio	the average of the returned overlapping ratios counted by the originator
Msg(V_i)	the message comes directly from V_i

Table 2. The notations in SOE algorithm.

Obviously SOE is deterministic and can work asynchronously without special supervising peer. Additionally, the algorithm dispatches the computation to each peer, fully utilizing the distributed, load sharing feature of P2P environment.

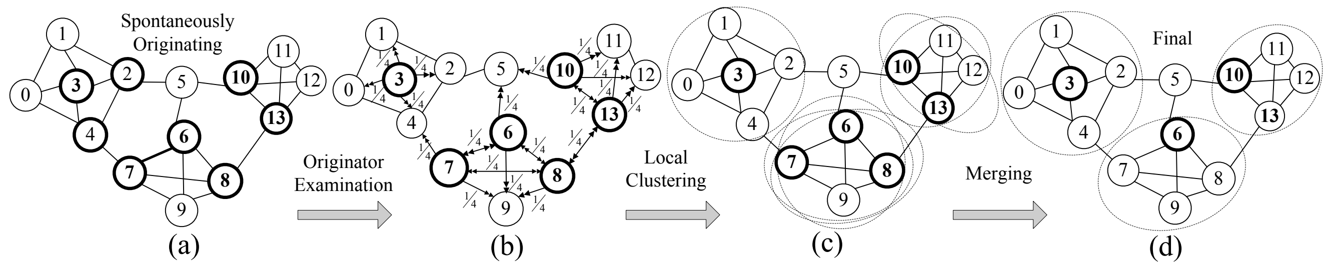


Figure 2. Illustration of SOE processes.

Algorithm 2 SOE Reply: Executed by V_i receiving *SOEcheckMsg*

```
1: {Calculate the overlapping ratio of the neighborhood between SOEcheckMsg.OID and $V_i$}
2: Create a new message SOEreplyMsg
3: $SOEreplyMsg.Ratio = \frac{|SOEcheckMsg.list \cap Nbr(V_i)|}{|Nbr(V_i)|}$
4: Send SOEreplyMsg back to SOEcheckMsg.OID
```

In Fig. 2, we take the same example in [5] to see how ACM processes. At First, node V_2, V_3, V_4, V_6, V_7, V_8, V_{10}, V_{13} which has greater degree number would become quasi-originators and proceed with one-hop probes (Fig. 2a). After the examination, only V_3, V_6, V_7, V_8, V_{10}, V_{13} have enough local coverage to become the cluster originators (Fig. 2b).

To illustrate how to compute the overlapping ratio, consider an example V_6 in Fig. 2a. Assume V_6 has become a quasi-originator. V_6 would send $Nbr(V_6) \cup V_6 = \{V_5, V_6, V_7, V_8, V_9\}$ to V_5, V_6, V_7, V_8, V_9. After a while, V_6 would receive the returned messages with value $Msg(V_5).Ratio = \frac{1}{3}$, $Msg(V_7).Ratio = \frac{3}{4}$, $Msg(V_8).Ratio = \frac{3}{4}$, $Msg(V_9).Ratio = 1$. Hence V_6 gets $avgRatio = 0.708$.

ClusterTable: maintain several sets of (OID, Weight)

OID	The ID of the originators which the node has received the cluster message from.
Weight	The message weight which the node accumulated from the originator OID.

Variable

MainCluster	The ID of the originator which the node accumulated the most message weight from.

Table 3. Data maintained by each peer node

MemberList: maintain the member list

MID	The ID of the cluster members.

GuestTable: maintain several sets of (GID, Weight)

GID	The ID of the cluster guests
Weight	The accumulated weight the guest GID obtained from this originator.

Table 4. Additional data of each originator

3.1.2 Local Clustering

With limited information, we allow more originators in P2P and then perform the local clustering procedure.

Once originators are determined, originators issue local clustering individually. Peers accumulate the reach probabilities of originators disseminated as the message weights. While the accumulated weight of some originator reaches a predefined minimum threshold, ClusterThreshold, the node would apply to join the cluster. Thus, each peer should maintain a Cluster Table (showed in Table 3) to record the accumulated weight from each originator. Furthermore, we define two levels for joining a cluster. The first level is that if the accumulated message weight from some originator over a threshold in a node, the node would apply to join the cluster to be a ”cluster guest.” A node could be many clusters’ guest. The second level is that among the clusters of which the node satisfies the condition to be a guest, the node chooses only one, from which the maximum weight the node accumulates, representing the most related one, to join as a ”cluster member.” It can be understood that a ”cluster member” is also a ”cluster guest”. In addition, while a node joins the cluster as a cluster guest, it would tell the originator additional information - the accumulated message weight which the node receives from the originator. Thus, each originator should maintain a guest table (showed in Table 4) that contains the ID list and the weight. We need to distinguish ”cluster member” from ”cluster guest” because there are no intersection between member sets of different clusters.

As illustrated in Fig. 2b and 2c, after Spontaneous Originator Examination, only V_3, V_6, V_7, V_8, V_{10}, V_{13} are qualified to be cluster originators and to issue the local clustering, disseminate the reach probability to execute Markov process. Due to the dense connectivity among $\{V_0, V_1, V_2, V_3, V_4\}$, they could accumulate reach probability from originator V_3 higher and faster than from the other originators. Finally $\{V_0, V_1, V_2, V_3, V_4\}$ form a local cluster. Similar processes happen in $\{V_6, V_7, V_8, V_9\}$ and $\{V_{10}, V_{11}, V_{12}, V_{13}\}$. In this case, V_3, V_6, and V_7 generate equivalent clusters, and V_{10} and V_{13} also do. The question how to accurately identify and merge equivalent or similar clusters in a

Algorithm 3 Join Process of Local Clustering: Executed by node V_i receiving Clustering Messge Msg

```
1:  Calculate the proceeding reach probability and pass it
    to neighbors as message weights. (Similar to CDC)
2:  Accumulate he received weight to ClusterTable.
3:  if ClusterTable.Weight(Msg.OID)
        $<$ ClusterThreshold then
4:      Remain the state.
5:  else
6:      Join Msg.OID as a guest.
7:      if ClusterTable.Weight(Msg.OID)
            $>$ ClusterTable(MainCluster) then
8:          Quit MainCluster (but still a cluster guest)
9:          Join Msg.OID as a cluster member.
10:         MainCluster = Cluster(Msg.OID)
11:     end if
12: end if
```

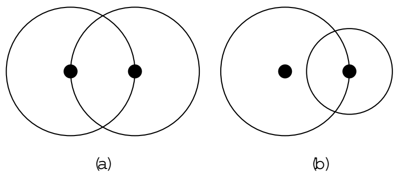


Figure 3. Resolvability of two local clusters

distributed environment is managed in the merging phase.

3.2 The Merging Phase

The merging examination depends on the definition of the overlapping. Fig. 3(a) is showed a sketch of resolvability. If two circles do not cover each other's center, we call them resolvable. We draw the circle for local clustering with the cluster originator as the center and the threshold of reach probability as the radius. We let the originator covered by the other originator, meaning in the scope of its local clustering and being a guest, to issue a merging check. Only one cluster covers the other is sufficient for the merging condition, as showed in Fig. 3(b) because two clusters with different sizes may often cause only one to cover the other; especially in the situation of contiguous merging, there may be some larger clusters and some smaller ones.

In order to perform the merging process, a cluster originator should maintain two pieces of data. One is MemberList which maintains the cluster members' ID when they join or leave. The other is GuestTable which maintains the cluster guests' ID and their accumulated cluster weight with respective to the cluster. The data structure is showed in Table 4. With the help of these additional data, we can define the overlapping in the merging phase now. The method uses a distributed inner product of vectors. Let us see a GuestTable as a vector based on the inner product space composed by mapping each node to one dimension. The nodes not in a GuestTable are thought as weight 0. Therefore, without knowing all nodes in the network, we can still perform vector computations on the basis of the whole network.

Algorithm 4 Merging Check: Executed by Originator V_o

```
1:  After finishing the local clustering
    {Calculate the scale of my own clustering}
2:  $ScaleOfCluster =$
        $\sum_{\forall V_i \in GuestTable} GuestTable.Weight(V_i)^2$
    {Check if $V_o$ is in the range of other clusterings}
3:  for each node $V_p$ in ClusterWeightTable do
4:      if $ClusterWeight(V_p) \geq$ ClusterThreshold then
5:          {$V_o$ is $V_p$'s guest. Execute merging check.}
6:          Create a MergeCheckMsg
7:          MergeCheckMsg.OID = $V_o$
8:          MergeCheckMsg.ClusterScale = ScaleOfCluster
9:          MergeCheckMsg.GuestTable = GuestTable
10:         Send MergeCheckMsg to $V_p$
11:     end if
12: end for
```

First, we can obtain the scale of a cluster by the inner product of the cluster's GuestTable as

$$ScaleOfCluster = \|GuestTable.Weight\|^2$$
$$= \sum_{V_i} GuestTable.Weight(V_i)^2$$

Second, we can obtain the overlapping scale by the inner product of the GuestTables of the two clusters, which is the projection from one cluster to the other and vice versa as

$$Projection(C_m, C_n) = \sum_{\forall V_i} (C_m.GuestTable.Weight(V_i)) \times (C_n.GuestTable.Weight(V_i))$$

Third, for a cluster C_i, we define its intersection ratio with another cluster C_j as

$$ProjectRatio(C_m, C_n) = \frac{Projection(C_m, C_n)}{ScaleOfCluster(C_m)},$$

which is the normalized intersection scale. We use $ProjectRatio$ as the overlapping measurement in the merging phase.

The merging processes as follows. During local clustering, the event that one originator reaches the condition to be the other's cluster guest would detonate the merging check between them in the next phase. After local clustering, the originator sends the GuestTable and the cluster-self's scale to the other cluster for merging check. The cluster receiving the merging check message calculates the ProjectRatio to determine whether the coverage between two clusters reaches the MergeThreshold. We take the originator with the highest ScaleOfCluster as the major cluster head and also retain the original clustering information in each other originator, as a minor cluster head, thus form a hierarchical structure. However the exact information deployment depends upon the requirement of practical P2P applications.

Algorithm 5 Merging Reply: Executed by Originator V_p receiving MergeCheckMsg

```
1:  {Calculate the inner product between GuestTable and
    MergeCheckMsg.GuestTable as follows}
2:  IntersectWeight = 0
3:  for each node $V_i$ in GuestTable do
4:      for each node $V_j$ in the MergeCheckMsg.GuestTable
        do
5:          if $V_i = V_j$ then
6:              $Projection = Projection +$
                  GuestTable.Weight($V_i$) $\times$
                $MergeCheckMsg.GuestTable.Weight(V_j)$
7:          end if
8:      end for
9:  end for
10: {Calculate the Projective Ratio between GuestTable
    and MergeCheckMsg.GuestTable as follows}
11: $ProjectRatio = \frac{Projection}{ScaleOfCluster}$
12: if $ProjectRatio \geq MergeThreshold$ then
13:     $Cluster(V_p)$ and Cluster(MergeCheckMsg.OID)
        should be the same cluster
14:     if MergeCheckMsg.ClusterScale > ScaleOfCluster
        then
15:         Join MergeCheckMsg.OID with the whole cluster
16:     else
17:         Ask MergeCheckMsg.OID with
            Cluster(MergeCheckMsg.OID) to join my cluster
18:     end if
19: end if
```

As illustrated in Fig. 2c and 2d, after local clustering, the clusters leaded by V_6, V_7, and V_8 would cover each other. Then, the merging process would calculate the $ProjectRatio$, and there are finally three clusters leaded by V_3, V_6, and V_{10} derived. Node V_5 would remain an outlier or be a member of the cluster leaded by V_{10} according to the threshold defined in the local clustering algorithm, but each situation is good in this graph.

4 PERFORMANCE EVALUATION

In this section, we implement a simulator and conduct experiments to evaluate the performance of ACM. The simulation model is presented in Section 4.1 and Section 4.2 presents the experimental results.

4.1 Simulation Model

The simulation implemented consists of two parts: the generation of P2P network topology and the implementation of clustering schemes. For the network topology generation, the ground truth of clustering results should be determined. We first set the number of nodes in the network and the size of clusters. Same as in [1] [2], we utilize Zipf-distribution for some parameters used for the network topology generation. Table 5 shows the notations and the settings for these parameters. Specifically, Zipf-of-Cluster-Size determines the variation in size of clusters. The network topology is composed by clusters of largely varied sizes when Zipf-of-Cluster-Size is set to a high value. Besides, for each peer, Zipf-of-node-degree determines the edge connectivity among other peers. After generating the distribution, the network topology is further modified by loss edge ratio which stands for the loss information, and irregular edge ratio for the misleading information.

We use the average coverage (*avgCov*) introduced in Section 2 to evaluate the quality of communities. An inflation parameter in the MCL scheme can control the clustering granularity. The algorithm would lead to fewer and larger clusters with a lower value of the parameter, and vice versa. So we set the value from 1.5 to 5 to generate the various clustering results and select the best one from them. In the CDC scheme, two parameters, *Vicinity* and *TwoHopThreshold*, can control the number of originators, and hence also effect the clustering granularity. We test and select the best values of these parameters in the CDC scheme for each network topology. In addition, as the originator selection in the CDC scheme is random, we perform the CDC scheme with different random seeds 10 times for 10 various networks which are generated by the identical topology parameters. Then average the results of these 100 runs in each experiment.

	Parameter	Value, default
N	Number of nodes in the network	8 – 4096, 512
M	Size of clusters	4 – 64, 32
α_1	Zipf-of-Cluster-Size	0 – 2, 0.2
α_2	Zipf-of-Node-Degree	0 – 2, 0.5
γ_1	Loss edge ratio	0% – 100%, 10%
γ_2	Irregular edge ratio	0% – 100%, 20%

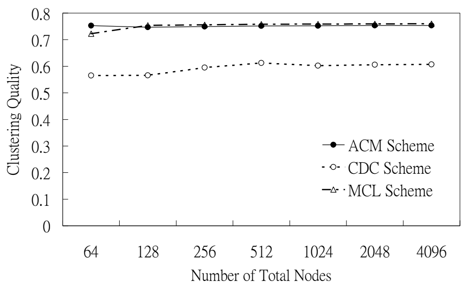
Table 5. Parameters of network topology generation.

4.2 Experimental Results

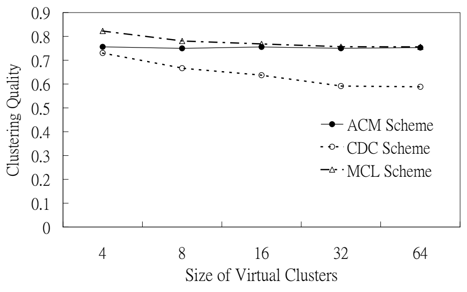
4.2.1 Scalability

We first investigate the scalability by varying the number of peers. The parameters are set to default values as showed in Table 5. Note that the default Zipf-of-Cluster-Size is set to a low value, with which the generated network topologies should yield a cluster structure with roughly the same cluster sizes.

Fig. 4(a) shows the stability of all three schemes under this setup. Since the network topology is very similar except for the number of nodes, the quality of clustering results are good with the number of nodes varied. We observe that ACM performs as good as the centralized MCL scheme and outperforms CDC scheme, although we have optimized the configurable parameters for cluster size in the CDC scheme. It indicates that the CDC scheme could not obtain good originator sets every time, and get lower quality. The result reflects the problem of originator determination discussed in Section 2. With a good design of ACM in selecting originators, local clustering and merging, ACM performs better than CDC.



(a)



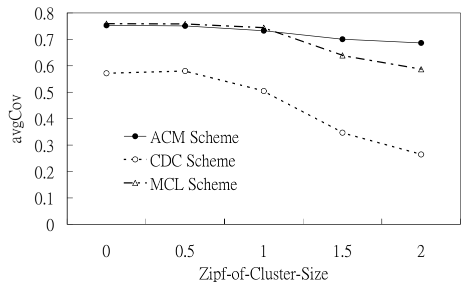
(b)

Figure 4. (a)Effect of Network Size and (b)Effect of Cluster Size

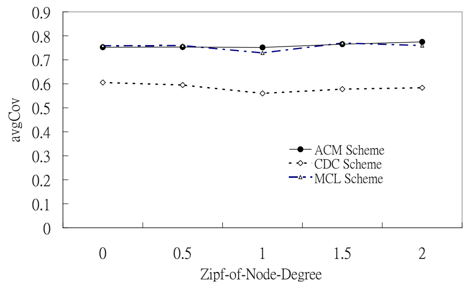
4.2.2 The Size of Clusters

Next, we investigate the impact of the size of clusters generated in the network topology. Without global information, clustering for large size of clusters could become challenging in distributed environments. The size of cluster varies from 4 to 64 while other parameters are set to default values. As seen in Fig. 4(b), the clustering quality of CDC decreases with the size of clusters. The reason is that with a larger cluster size, the number of clusters will be smaller. Therefore, the selection of originator has a great impact on the cluster results. In addition, the originator number in the CDC scheme is controlled with a random waiting method through routing, which would get harder while the cluster size gets larger. The experiment indicates that CDC suffers from the selection of originators.

On the other hands, the ACM scheme has almost constant clustering quality while the size of clusters increases as well as the centralized MCL scheme, which does not have problems of routing and originator selection. The experiment validates the excellent design of the ACM scheme for the distributed environment.



(a)

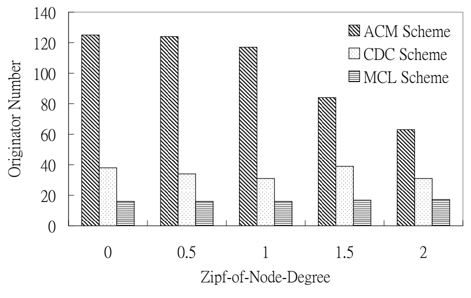


(b)

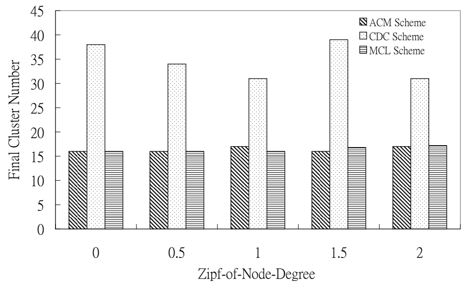
Figure 5. (a)Effect of variation in Cluster-Size and (b)Effect of Variation in Node Degree

4.2.3 Variation in Cluster Size

In P2P applications, communities may have varied sizes. Thus, in this experiment, we further examine the impact of the variation in cluster size. As showed in Fig. 5(a), the ACM scheme performs well while the clustering quality of the CDC deteriorates as the variation in cluster sizes increases. The result indicates that the CDC scheme is designed only for uniform cluster size. As cluster sizes varies, CDC cannot derive a good cluster quality. In Fig. 5(a), the ACM scheme even performs better than the centralized MCL scheme as the variation in cluster sizes increases. Notice that the MCL scheme also manages the whole network topology with the same clustering granularity. The experiment demonstrates the adaptability of the ACM scheme for varied network topology.



(a)



(b)

Figure 6. (a)Number of Initial Originators and (b)Number of Final Clusters

Since SOE depends on the node degree and local coverage, the more variation in node degree, the less nodes could be qualified as originators. As can be seen in Fig. 6(a) that the originator number decreases as Zipf-of-Node-Degree increases. Most P2P interactions have Zipf-distribution. Even if there are many originators in a uniformly distributed network, and hence many duplicate clusters, the ACM scheme still reduces the number of clusters after merging phase to gain better avgCov than the CDC scheme as the result in Fig. 5(b).

In Fig. 6(b), the ACM scheme and the MCL scheme has similar high clustering quality. Their similar numbers of clusters stand for good clusterings. Although we have adjusted the parameters in the CDC scheme to optimize the cluster number, it still suffers from the random factor and routing process, thus does not always derive good clustering quality.

5 Conclusion

In this paper, we developed a new scheme ACM – Asynchronous Node Clustering and Merging to discover the community structure of social networks in the peer-to-peer environment. Note that the scheme does not require any specific supervised node. By exploiting the Spontaneous Originator Examination mechanism (SOE), community structures are formed automatically. Due to the adopted asynchronous mechanisms, the clustering processes do not need to be performed throughout the whole network synchronously, but executed by peer nodes as needed. Experiment results show the quality of community structures obtained by the ACM scheme better than other schemes under various topology settings. The experiments also indicate that ACM forms good community structures in networks with high variation in cluster size feature. Due to the asynchronous feature, the ACM scheme is well performed in distributed, dynamic environments. New strategies and mechanisms for dynamic maintenance will be studied in the future. Moreover, we plan to conduct a thorough performance evaluation to study the ACM scheme.

6 Acknowledgements

This work was supported in part by the National Science Council, Project No. NSC 95-2211-E-009-61-MY3 and NSC 97-2623-7-036-001-D, Taiwan, Republic of China.

References

[1] M. Li, W. Lee, and A. Sivasubramaniam. ”Semantic Small World: An Overlay Network for Peer-to-Peer Search”. In *12th ICNP*, pages 228– 238, Oct. 2004.

[2] A. Löser, S. Staab, and C. Tempich. ”Semantic Social Overlay Networks”. *IEEE JSAC*, 19(10):5–14, Jan. 2005.

[3] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. ”Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society”. *Nature*, 435(7043):814–818, Jun. 2005.

[4] P. Pons and M. Latapy. ”Computing Communities in Large Networks Using Random Walks”. In *20th ISCIS*, pages 284–293, 2005.

[5] L. Ramaswamy, B. Gedik, and L. Liu. ”A Distributed Approach to Node Clustering in Decentralized Peer-to-Peer Networks”. *IEEE TPDS*, 16(9):896–915, Sep. 2005.

[6] M.-T. Sun, C.-T. King, W.-H. Sun, and C.-P. Chang. ”Attribute-Based Overlay Network for Non-DHT Structured Peer-to-Peer Lookup”. In *36th ICPP*, page 62, 2007.

[7] S. van Dongen. *A New Cluster Algorithm for Graphs*. Centrum voor Wiskunde en Informatica (CWI), INS-R9814, ISSN 1386-3681, 1998.

[8] S. van Dongen. ”Performance Criteria for Graph Clustering and Markov Cluster Experiments”. *technical report, Natl Research Inst. for Math. and Computer Science in the Netherlands, Amsterdam*, 2000.

[9] B. Yang, W. K. Cheung, and J. Liu. ”Community Mining from Signed Social Networks”. *IEEE TKDE*, 19(10):1333 – 1348, Oct. 2007.

[10] X. Yang, Y. Zhu, and Y. Hu. ”A Large-Scale and Decentralized Infrastructure for Content-Based Publish/Subscribe Services”. In *36th ICPP*, page 61, 2007.

[11] J. Zhang and J. Wu. ”XYZ: A Scalable, Partially Centralized Lookup Service for Large-Scale Peer-to-Peer Systems”. In *12th ICPADS*, 2006.