This paper was presented as part of the main technical program at IEEE INFOCOM 2011

Adaptive Algorithms for Detecting Community Structure in Dynamic Social Networks

Nam P. Nguyen, Thang N. Dinh, Ying Xuan, My T. Thai
Department of Computer and Information Science and Engineering, University of Florida
Email: {nanguyen, tdinh, yxuan, mythai}@cise.ufl.edu

***Abstract*—Social networks exhibit a very special property: community structure. Understanding the network community structure is of great advantages. It not only provides helpful information in developing more social-aware strategies for social network problems but also promises a wide range of applications enabled by mobile networking, such as routings in Mobile Ad Hoc Networks (MANETs) and worm containments in cellular networks. Unfortunately, understanding this structure is very challenging, especially in dynamic social networks where social activities and interactions are evolving rapidly. Can we quickly and efficiently identify the network community structure? Can we adaptively update the network structure based on previously known information instead of recomputing from scratch?**

In this paper, we present Quick Community Adaptation (QCA), an adaptive modularity-based method for identifying and tracing community structure of dynamic online social networks. Our approach has not only the power of quickly and efficiently updating network communities, through a series of changes, by only using the structures identified from previous network snapshots, but also the ability of tracing the evolution of community structure over time. To illustrate the effectiveness of our algorithm, we extensively test QCA on real-world dynamic social networks including ENRON email network, arXiv e-print citation network and Facebook network. Finally, we demonstrate the bright applicability of our algorithm via a realistic application on routing strategies in MANETs. The comparative results reveal that social-aware routing strategies employing QCA as a community detection core outperform current available methods.

I. Introduction

Many social networks exhibit the property of containing community structure [1][2], i.e., they naturally divide into groups of vertices with denser connections inside each group and fewer connections crossing groups, where vertices and connections represent network users and their social interactions, respectively. Members in each community of a social network usually share things in common such as interests in photography, movies, music or discussion topics and thus, they tend to interact more frequently with each other than with members outside of their community. Community detection in a network is the gathering of network vertices into groups in such a way that nodes in each group are densely connected inside and sparser outside.

It is noteworthy to differentiate between community detection and graph clustering. These two problems share the same objective of partitioning network nodes into groups; however, the number of clusters is predefined or given as part of the input in graph clustering whereas the number of communities is typically unknown in community detection. Detecting communities in a network provides us meaningful insights to its internal structure as well as its organization principles. Furthermore, knowing the structure of network communities could also provide us more helpful points of view to some uncovered parts of the network, thus helps in preventing potential networking diseases such as virus or worm propagation. Studies on community detection on static networks can be found in an excellent survey [3] as well as in the work of [4][5][6][7] and references therein.

Real-world social networks, however, are not always static. In fact, most of social networks in reality (such as Facebook, Bebo and Twitter) evolve and witness an expand in size and space as their users increase, thus lend themselves to the field of dynamic networks. A dynamic network is a special type of evolving complex networks in which changes are frequently introduced over time. In the sense of an online social network, such as Facebook, Twitter or Flickr, changes are usually introduced by users joining in or withdrawing from one or more groups or communities, by friends and friends connecting together, or by new people making friend with each other. Any of these events seems to have a little effect to a local structure of the network on one hand; the dynamics of the network over a long period of time, on the other hand, may lead to a significant transformation of the network community structure, thus raises a natural need of reidentification. However, the rapidly and unpredictably changing topology of a dynamic social network makes it an extremely complicated yet challenging problem.

Although one can possibly run any of the static community detection methods, which are widely available [4][5][6][8], to find the new community structure whenever the network is updated, he may encounter some disadvantages that cannot be neglected: (1) the long running time of a specific static method on large networks (2) the trap of local optima and (3) the almost same reaction to a small change to some local part of the network. A better, much efficient and less time consuming way to accomplish this expensive task is to adaptively update the network communities from the previous known structures, which helps to avoid the hassle of recomputing from scratch. This adaptive approach is the main focus of our study in this paper. In Figure 1, we briefly generalize the idea of dynamic network community structure adaptation.

Detecting community structure of a dynamic social network is of considerable uses. To give a sense of it, consider the routing problem in communication network where nodes and links present people and mobile communications, respectively.

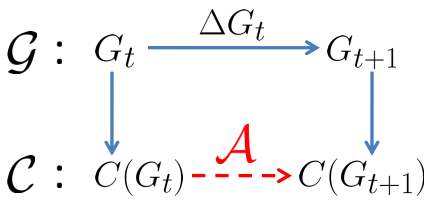


Fig. 1. The network evolves from time t to $t+1$ under the change ΔG_t. The adaptive algorithm $\mathcal{A}$ quickly finds the new community structure $\mathcal{C}(G_{t+1})$ based on the previous structure $\mathcal{C}(G_t)$ together with the changes ΔG_t.

Due to nodes mobility and unstable links properties of the network, designing an efficient routing scheme is extremely challenging. However, since people have a natural tendency to form groups of communication, there exist groups of nodes which are densely connected inside than outside in the underlying MANET as a reflection, and therefore, forms community structure in that MANET. An effective routing algorithm, as soon as it discovers the network community structure, can directly route or forward messages to nodes in the same (or to the related) community as the destination. By doing this way, we can avoid unnecessary messages forwarding through nodes in different communities, thus can lower down the number of duplicate messages as well as reduce the overhead information, which are essential in MANETs.

The contributions of this paper are:

- We propose QCA, a fast and adaptive algorithm for efficiently identifying the community structure of a dynamic social network. Our approach takes into account the previously discovered network structure and processes on network changes only, thus significantly reduces computational cost and processing time.
- We study the dynamics of a social network and prove theoretical results regarding its various behaviors over time, which are the basic features of our method.
- We extensively evaluate our algorithms on various real world dynamic social networks including Enron email network, ArXiv citation network and Facebook network. Experimental results show that our method not only achieves competitive modularities but also high quality community structures in a timely manner.
- As an application, we employ QCA as a community identification core in routing strategies in MANETs. Simulation results show that QCA outperform the current available methods and confirm the bright applicability of our proposed method in mobile computing.

The rest of this paper is organized as follow: Section II presents the preliminaries and problem definition. Section III gives a complete description of our algorithms and their theoretical analysis. Section IV shows experimental results of our approach on various real world datasets. In sections V, we present a practical application of our approach in MANETs. In Section VI, we discuss about the related work and finally conclude our work in Section VII.

II. Preliminaries

In this section, we present the notations, objective function as well as the dynamic graph model representing a social network that we will use throughout the paper.

(Notation) Let $G = (V, E)$ be an undirected unweighted graph with N nodes and M links representing a social network. Let $\mathcal{C} = \{C_1, C_2, .., C_k\}$ denote a collection of disjoint communities, where $C_i \in \mathcal{C}$ is a community of G. For each vertex u, denote by d_u, $C(u)$ and $NC(u)$ its degree, the community containing u and the set of its adjacent communities. Furthermore, for any $S \subseteq V$, let m_S, d_S and e_S^u be the number of links inside S, the total degree of vertices in S and the number of connections from u to S, respectively. The pairs of terms *community* and *module*; *node* and *vertex* as well as *edge* and *link* and are used interchangeably.

(Dynamic social network) Let $G^s = (V^s, E^s)$ be a time dependent network snapshot recorded at time s. Denote by ΔV^s and ΔE^s the sets of vertices and links to be introduced (or removed) at time s and let $\Delta G^s = (\Delta V^s, \Delta E^s)$ denote the change in term of the whole network. The next network snapshot G^{s+1} is the current one together with changes, i.e., $G^{s+1} = G^s \cup \Delta G^s$. A *dynamic network* $\mathcal{G}$ is a sequence of network snapshots evolving over time: $\mathcal{G} = (G^0, G^1, .., G^s)$.

(Objective function) In order to quantify the goodness of a network community structure, we take into account the most widely accepted measure called *modularity* $\mathcal{Q}$ [6], which is defined as: $\mathcal{Q} = \sum_{C \in \mathcal{C}} (\frac{m_C}{M} - \frac{d_C^2}{4M^2})$. Basically, $\mathcal{Q}$ is the fraction of all links within communities subtracts the expected value of the same quantity in a graph whose nodes have the same degrees but links are distributed randomly, and the higher modularity $\mathcal{Q}$, the better network community structure is. Therefore, our objective is to find a community assignment for each vertex in the network such that $\mathcal{Q}$ is maximized. Modularity, just like other quality measurements for community identifications, has some certain disadvantages such as its non-locality and scaling behavior [7] or resolution limit [9]. However, it is still very well considered due to its robustness and usefulness that closely agree with intuition on a wide range of real world networks.

Problem Definition: Given a dynamic social network $\mathcal{G} = (G^0, G^1, .., G^s)$ where G^0 is the original network and G^1, G^2,.., G^s are the network snapshots obtained through ΔG^1, ΔG^2,.., ΔG^s, we need to devise an adaptive algorithm to efficiently detect and identify the network community structure at any time point utilizing the information from the previous snapshots as well as tracing the evolution of the network community structure.

III. Method description

Let us first discuss about how changes to the evolving network topology affect the structure of its communities. Assume that $G = (V, E)$ is the current network and $\mathcal{C} = \{C_1, C_2, .., C_k\}$ is its corresponding community structure. We use the term *intra-community links* to denote edges whose two endpoints belong to the same community and the term *inter-community links* to denote those with endpoints connecting

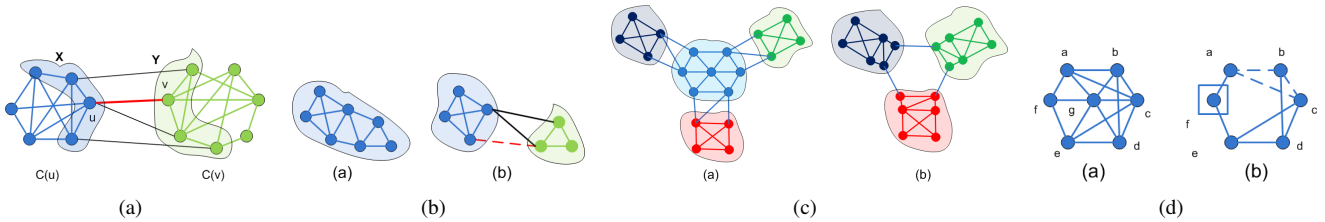


Fig. 2. $2(a)$: When an edge (u, v) joining $C(u)$ and $C(v)$ is introduced. Tests on membership changing are performed on sets X and Y $2(b)$: a) The original community b) After an edge (in dotted line) is removed, the community is broken into two smaller communities $2(c)$: a) The original network with four communities b) After the highest degree node is removed, the leftover nodes join in different modules, forming a new network with three communities $2(d)$: a) The original community b) When the central node g is removed, a 3-clique is placed at a to discover b, c, d and e. f assigned singleton afterwards

different communities. For each community C of G, the number of connections linking C with other communities are much fewer than the number of connections within C itself, i.e., nodes in C are densely connected inside than outside. Intuitively, adding intra-community links inside or removing inter-community links between communities of G will strengthen those communities and make the structure of G more clear. Vice versa, removing intra-community links and inserting inter-community links will loosen the structure of G. However, when two communities have less distraction caused by each other, adding or removing links makes them more attractive to each other and thus, leaves a possibility that they will be combined to form a new community. The community updating process, as a result, is extremely challenging since any insignificant change in the network topology can possibly lead to an unexpected transformation of its community structure. We will discuss in detail possible behaviors of a dynamic network community structure in Section III-A.

In order to reflect changes introduced to a social network, its underlying graph is constantly updated by either inserting or removing a node or a set of nodes, or by either introducing or deleting an edge or a set of edges. In fact, the introduction or removal of a set of nodes (or edges) can be decomposed as a sequence of node (or edge) insertions (or removals), in which a single node (or a single edge) is introduced (or removed) at a time. This observation helps us to treat network changes as a collection of *simple events* where a simple event can be one of *newNode, removeNode, newEdge, removeEdge* whose details are as follow:

- *newNode* $(V+u)$: A new node u with its associated edges are introduced. u could come with no or more than one new edge(s).
- *removeNode* $(V-u)$: A node u and its adjacent edges are removed from the network.
- *newEdge* $(E+e)$: A new edge e connecting two existing nodes is introduced.
- *removeEdge* $(E-e)$: An existing edge e in the network is removed.

A. Algorithms

Our approach first requires an initial community structure $\mathcal{C}_0$, which we refer to as the basic structure, in order to process further. Since the input model is restricted as an undirected unweighted network, this initial community structure can be obtained by performing any of the available static community detection methods [4][5][8]. To obtain a good basic structure, we choose the method proposed by Blondel *et al* in [5] which produces a network community structure of high modularity in a reasonable amount of time [3].

1) New node: Let us consider the first case when a new node u and its associated connections are introduced. Note that u may come with no adjacent edge or with many of them connecting one or more communities. If u has no adjacent edge, we create a new community containing only u and leave the other communities as well as the overall modularity $\mathcal{Q}$ intact. The interesting case happens, as it always does, when u comes with edges connecting one ore more existing communities. In this latter situation, we need to determine which community u should join in in order to maximize the gained modularity. There are several local methods introduced for this task, for instance the algorithms of [4][8]. Our method is inspired by a physical approach proposed in [10], in which each node is influenced by two forces: F_{in}^{C} (to keep u stays inside community C) and F_{out}^{C} (the force a community C makes in order to bring u to C) defined as follow: $F_{in}^{C}(u) = e_{C}^{u} - \frac{d_u(d_C - d_u)}{2M}$ and $F_{out}^{S}(u) = \max_{S \in NC(u)} \left\{ e_{S}^{u} - \frac{d_u d_{outS}}{2M} \right\}$ where d_{outS} is of opposite meaning of d_S.

Taking into account the above two forces, a node u can actively determines its best community membership by computing those forces and either lets itself join in the community having the highest $F_{out}(u)$ (if $F_{out}(u) > F_{in}^{C(u)}(u)$) or stays put in the current community otherwise. By Theorem 1, we bridge the connection between those forces and the objective function, i.e., joining the new node in the community with highest outer force will maximize the local gained modularity. This is the central idea for handling the first case when a new node and its adjacent links are introduced. The detailed process is presented in Alg. 1.

***Theorem** 1: Suppose C is the community that gives maximum $F_{out}^{C}(u)$ when a new node u with degree p is introduced to G, then joining u in C gives the maximal modularity contribution.*

2) New edge: In case that a new edge $e = (u, v)$ connecting two existing vertices u, v is introduced, we divide it further into two smaller cases: e is an intra-community link (totally inside a community C) or an inter-community link (connects two communities $C(u)$ and $C(v)$). If e is inside a community

Algorithm 1 New_Node

Input: New node u with associated links; Current structure $\mathcal{C}_t$.
Output: An updated structure $\mathcal{C}_{t+1}$

Create a new community of only u;
for $v \in N(u)$ **do**
  Let v determine its best community;
end for
for $C \in NC(u)$ **do**
  Find $F^C_{out}(u)$;
end for
Let $C_u \leftarrow \arg\max_C \{F^C_{out}(u)\}$;
Update $\mathcal{C}_{t+1} : \mathcal{C}_{t+1} \leftarrow (\mathcal{C}_t \backslash C_u) \cup (C_u \cup u)$;

C, its presence will strengthen the inner structure of C according to Lemma 1. Furthermore, by Theorem 2, we know that adding e should not split the current community C into smaller modules. Therefore, we leave the current network structure intact in this case.

The interesting situation happens when e is a link connecting communities $C(u)$ and $C(v)$ since the presence of e could possibly make u (or v) leave its current modules and join in the new community. Additionally, if u (or v) decides to change its membership status, it can eventually advertise its new community to all its neighbors and some of them might want to change their memberships as a consequence. By Lemma 2, we show that if u (or v) should ever change its cluster assignment then $C(v)$ (or $C(u)$) is the best new community for it. But how can we efficiently and quickly decide whether u (or v) should change its membership or not in order to form a better community structure when e is added? To this end, we provide a criterion to test for membership changing of u and v in Theorem 3. Here, if both $\Delta q_{u,C,D}$ and $\Delta q_{v,C,D}$ fail to satisfy the criteria, we can safely preserve the current network community structure and keep going (Corollary 1). Otherwise, we move u (or v) to its new community and consequently let its neighbors determine their best modules to join in, using local search and swapping to maximize gained modularity. Figure $2(a)$ describes the procedure for this latter case. The detailed algorithm is described in Alg. 2.

***Lemma** 1: For any $C \in \mathcal{C}$, if $d_C \leq M - 1$ then adding an edge within C will increase its modularity contribution.*

***Theorem** 2: If C is a community in the current snapshot of G, then adding any intra-community link to C will not split it into smaller modules.*

***Lemma** 2: When a new edge (u, v) connecting communities $C(u)$ and $C(v)$ is introduced, $C(v)$ (or $C(u)$) is the best candidate for u (or v) if it should ever change its membership.*

***Theorem** 3: Assume that a new edge (u, v) is added to G. Let $C \equiv C(u)$ and $D \equiv C(v)$. If $\Delta q_{u,C,D} \equiv 4(M+1)(e^u_D + 1 - e^u_C) + e^u_C(2d_D - 2d_u - e^u_C) - 2(d_u+1)(d_u+1+d_D-d_C) > 0$ then joining u to D will increase the overall modularity.*

***Corollary** 1: If the condition in Theorem 3 is not satisfied, then neither u (or v) nor its neighbors should be moved to D.*

3) Node removal: When an existing node u of a community C is removed at time t, all of its adjacent edges are removed as a result. This case is challenging in the sense that the resulting community is very complicated: it can be either unchanged or broken into smaller pieces and could probably be merged

Algorithm 2 New_Edge

Input: Edge $\{u, v\}$ to be added; Current structure $\mathcal{C}_t$.
Output: An updated structure $\mathcal{C}_{t+1}$.

if (u and v are new vertices) **then**
  $\mathcal{C}_{t+1} \leftarrow \mathcal{C}_t \cup \{u, v\}$;
else if $C(u) \neq C(v)$ **then**
  if $\Delta q_{u,C(u),C(v)} < 0$ and $\Delta q_{v,C(u),C(v)} < 0$ **then**
    return $\mathcal{C}_{t+1} \equiv \mathcal{C}_t$;
  else
    $w = \arg\max\{\Delta q_{u,C(u),C(v)}, \Delta q_{v,C(u),C(v)}\}$;
    Move w to the new community;
    for $t \in N(w)$ **do**
      Let t determine its best community;
    end for
    Update C_{t+1};
  end if
end if

with other communities. To give a sense of it, let's consider two extreme cases when a single degree node and a node with highest degree in a community is removed. If a single degree node is removed, it leaves the resulted community unchanged (Lemma 4). However, when a highest degree vertex is removed, the current community might be disconnected and broken in to smaller pieces which then are merged to other communities as depicted in Figure $2(c)$. Therefore, identifying the leftover structure of C is a crucial part once a vertex in C is removed.

To quickly and efficiently handle this task, we utilize the clique percolation method presented in [2]. In particular, when a vertex u is removed from C, we place a 3-clique to one of its neighbor and let the clique percolate until no vertices in C are discovered (Figure $2(d)$). We then let the remaining communities of C choose their best communities to merge in. The detailed algorithm is presented in Alg. 3.

Algorithm 3 Node_Removal

Input: Node $u \in C$ to be removed; Current structure $\mathcal{C}_t$.
Output: An updated structure $\mathcal{C}_{t+1}$.

$i \leftarrow 1$;
while $N(u) \neq \emptyset$ **do**
  $S_i = \{$Nodes found by a 3-clique percolation on $v \in N(u)\}$;
  if $S_i == \emptyset$ **then**
    $S_i \leftarrow \{v\}$;
  end if
  $N(u) \leftarrow N(u) \backslash S_i$;
  $i \leftarrow i + 1$;
end while
Let each S_i and singleton in $N(u)$ consider its best communities;
Update $\mathcal{C}_t$;

4) Edge removal: In the last case when an edge $e = (u, v)$ is about to be removed, we divide further into four subcases (1) e is a single edge connecting only u and v where u and v are of single degree (2) either u or v has degree one (3) e is an inter-community link connecting $C(u)$ and $C(v)$ and (4) e is an inter-community link. If e is an single edge, it is clear that removing e will result in the same community structure plus two singletons of u and v themselves. The same reaction applies to the second subcase when either u or v has single degree due to Lemma 4, thus results in the prior network structure plus u (or v). When e is an inter-community link, the removal of e will strengthen the current network communities (Lemma 3) and hence, we just make no

change to the community structure.

The last but most complicated case happens when an intra-community link is deleted. As depicted in Figure $2(b)$, removing this kind of edge often leaves the community unchanged if the community itself is densely connected; however, the target module will be divided if it contains substructures which are less attractive or loosely connected to each other. Therefore, the problem of identifying the structure of the remaining modules becomes very complicated. Theorem 4 provides us a convenient tool to test for community bi-division when an intra-community link is removed from the host community C. However, it requires an intensive look for all subsets of C, which may be time consuming. Note that prior to the removal of (u, v), the community C hosting this link should contain dense connections within itself and thus, the removal of (u, v) should leave some sort of 'quasi-clique' structure inside C. Therefore, we find all maximal 'quasi-cliques' within the current community and have them (as well as leftover singletons) determine their best communities to join in. The detailed procedure is described in Alg. 4.

***Lemma** 3: If C_1 and C_2 are two communities of G, then the removal of an inter-community link connecting them will strengthen modularity contributions of both C_1 and C_2.*

***Lemma** 4: The removal of (u, v) inside a community C where only u or v is of degree one will not separate C.*

***Lemma** 5: (Separation of a community) Let $C_1 \subseteq C$ and $C_2 = C \backslash C_1$ be two disjoint subsets of C. $(\mathcal{C} \backslash C) \cup \{C_1, C_2\}$ is a community structure with higher modularity when an edge crossing C_1 and C_2 is removed, i.e, C should be separated into C_1 and C_2, if and only if $e_{12} < \frac{d_1 d_2 - d_C + 1}{2(M-1)} + 1$.*

***Theorem** 4: (Testing of module separation) For any community C, let α and β be the lowest and the second highest degree of vertices in C, respectively. Assume that an edge e is removed from C. If there do not exist subsets $C_1 \subseteq C$ and $C_2 \equiv C \backslash C_1$ such that*

1) *e is crossing C_1 and C_2*
2) $\frac{\min\{\alpha(d_C - \alpha), \beta(d_C - \beta)\}}{2M} < e_{12} < \frac{(d_C - 2)^2}{8(M-1)} + 1$

then any bi-division of C will not benefit the overall $\mathcal{Q}$.

Algorithm 4 Edge_Removal

Input: Edge (u, v) to be removed; Current structure $\mathcal{C}_t$.
Output: An updated clustering $\mathcal{C}_{t+1}$.

if (u, v) is a single edge **then**
  $\mathcal{C}_{t+1} = (\mathcal{C}_t \backslash \{u, v\}) \cup \{u\} \cup \{v\}$;
else if Either u (or v) is of degree one **then**
  $\mathcal{C}_{t+1} = (\mathcal{C}_t \backslash C(u)) \cup \{u\} \cup \{C(u) \backslash u\}$;
else if $C(u) \neq C(v)$ **then**
  $\mathcal{C}_{t+1} = \mathcal{C}_t$;
else
  % Now (u, v) is inside a community C %
  L = {Maximal "quasi-cliques" in C};
  Let the singletons in $C \backslash L$ consider their best communities;
end if
Update $\mathcal{C}_{t+1}$;

Finally, our main algorithm QCA for quickly updating community structures of dynamic social networks is presented in Alg. 5.

IV. Experimental results

In this section, we present the experimental results of our QCA algorithm on identifying and updating the network community structures of dynamic social networks. To illustrate the strength and effectiveness of our approach, we choose three popular real-world social networks including *ENRON* email network [11], *arXiv* e-print citation network (provided by the KDD cup 2003 [12]) and Facebook online social network [13]. The static method we are comparing to is the one proposed by Blondel *et al* [5], which we refer to as Blondel method or static method. In addition to the static method, we further compare QCA to a recent dynamic adaptive method called MIEN proposed in [14]. Basically, MIEN tries to compress and decompress network modules into nodes in order to adapt with the changes and uses fast modularity method [4] to keep the network structure updated. In particular, we will show in the experiments the following quantities (1) modularity values (2) the quality of the identified network communities through NMI scores and (3) the processing time of our QCA in comparison with other methods. The above networks expose to contain clear community structures due to their high modularities, which is the main reason for them to be chosen.

Algorithm 5 Quick **C**ommunity **A**daptation (QCA)

Input: $G \equiv G_0 = (V_0, E_0)$, $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, .., \mathcal{E}_s\}$ a collection of simple events
Output: Community structure $\mathcal{C}_t$ of G^t at time t.

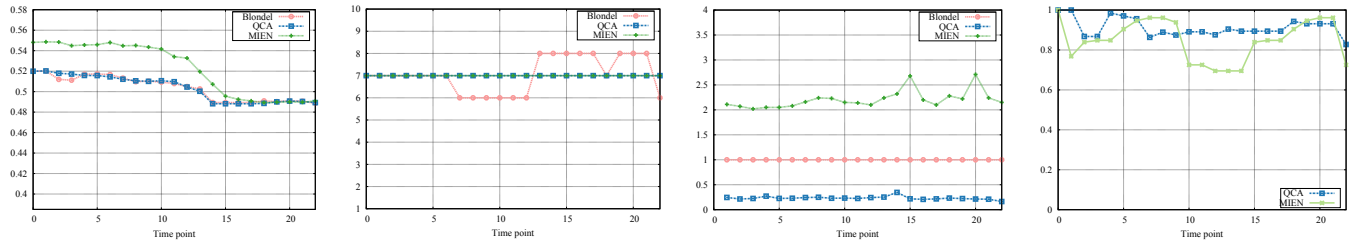
Use [5] to find an initial community clustering $\mathcal{C}_0$ of G_0;
for ($t \leftarrow 1$ to s) **do**
  $\mathcal{C}_t \leftarrow \mathcal{C}_{t-1}$;
  if $\mathcal{E}_t$ = *newNode(u)* **then**
    New_Node($\mathcal{C}_t, u$);
  else if $\mathcal{E}_t$ = *newEdge((u,v))* **then**
    New_Edge($\mathcal{C}_t, (u, v)$);
  else if $\mathcal{E}_t$ = *removeNode*(u) **then**
    Remove_Node($\mathcal{C}_t, u$);
  else
    Remove_Edge($\mathcal{C}_t, (u, v)$);
  end if
end for

In order to quantify the quality of the identified community structure, i.e., the similarity between the identified communities and the ground truth, we adopt a well known measure in Information Theory called *Normalized Mutual Information (NMI)*. NMI has been proven to be reliable and is currently used in testing community detection algorithms [3]. Basically, $NMI(U, V)$ equals 1 if structures U and V are identical and equals 0 if they are totally separated. Due to space limit, the readers are encouraged to read [3] for NMI formulas.

For each network, time information is extracted in different ways and a portion of the network data (usually the first network snapshot) is collected to form the basic network community structure. Our QCA algorithm takes into account that basic community structure and runs only on the network changes whereas the static method has to be performed on the whole network snapshot up to each time point.

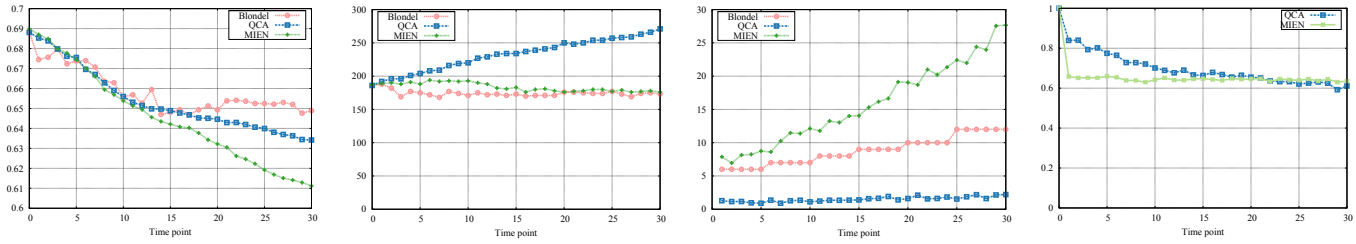
A. ENRON email network

The *Enron* email network contains email messages data from about 150 users, mostly senior management of *Enron*



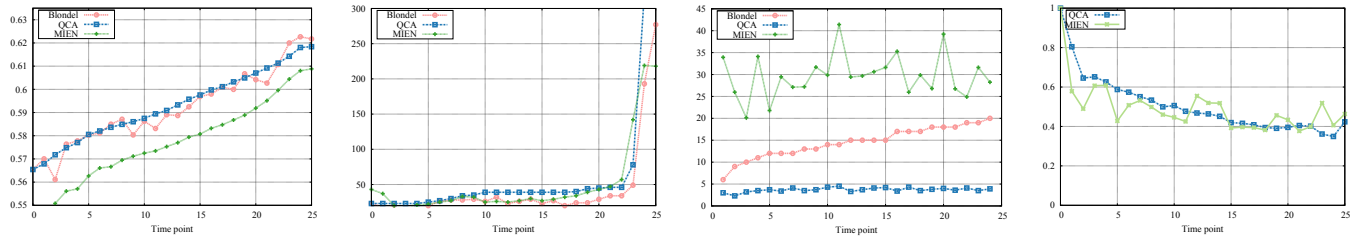
(a) Modularity (b) Number of Communities (c) Running Time(s) (d) NMI

Fig. 3. Simulation results on Enron Email Network dataset



(a) Modularity (b) Number of Communities (c) Running Time(s) (d) NMI

Fig. 4. Simulation results on ArXiv e-Print Citation Network



(a) Modularity (b) Number of Communities (c) Running Time(s) (d) NMI

Fig. 5. Simulation results on Facebook Social Network

Inc., from Jan 1999 to July 2002 [11]. Each email address is represented by an unique identification number in the dataset and each link corresponds to a message sent between the sender and the receiver. After a data refinement process, we choose 50% of total links to form a basic community structure of the network with 7 major communities, and simulate the network evolution via a series of 21 growing snapshots in which roughly 10^3 links are added at a time.

We first evaluate the modularity values computed by QCA, MIEN and Blondel method. As shown in Figure $3(a)$, our QCA algorithm archives competitively higher modularities than the static method but a little bit less than MIEN method while maintaining the nearly same number of communities of the other two (Figure $3(b)$). In particular, the modularity values produced by QCA very well approximate those found by static method with lesser variation. There are reasons for that. Recall that our QCA algorithm takes into account the basic community structures detected by the static method (at the first snapshot) and processes on network changes only. Knowing the basic network community structure is a great advantage of our QCA algorithm: it can avoid the hassle of searching and computing from scratch to update the network with changes. In fact, QCA uses the basic structure for finding and quickly updating the local optimal communities to adapt with changes introduced during the network evolution.

The running time of QCA and the static method in this small network are relatively close: the static method requires one second to complete each of its tasks while our QCA does not even ask for one (Figure $3(c)$). In this dataset, MIEN requires a little more time (1.5 second in average) to complete the task. Time and computational cost are significantly reduced in QCA since our algorithms only take into account the network changes while the static method has to work on the whole network every time.

Due to the lack of the proper information about real communities in *Enron* Inc. (and in other datasets), we use community structure identified by the static method as a reference to the ground truth. The quantity $NMI(QCA, Blondel)$ (or NMI(QCA) in short) indicates how community labels assigned by QCA similar to those of the ground truth computed at every timepoint. A NMI value of 1 means two assignments are identical and 0 implies the opposite. As one can see in

Figure $3(d)$, both the NMI scores of MIEN method and ours are very high and relatively close to 1, indicating that in this Enron email network, both our QCA and MIEN algorithm are able to identify high quality community structure with high modularity; however, only our method significantly reduces the processing time and computational requirement.

B. arXiv e-print citation network

The arXiv e-print citation network [12], which was initialized in 1991, has become an essential mean of assessing research results in various areas including physics and computer sciences. At the time the dataset was collected, the network already contained more than 225K fulltext articles and was growing of over 40K submissions each year, ranging from Jan 1996 to May 2003.

In our experiments, citation links of the first two years 1996 and 1997 were taken into account to form the basic community structure of our QCA method. In order to simulate the network evolution, a total of 30 time dependent snapshots of the arXiv citation network are created on a two-month regular basis in the duration between Jan 1998 and Jan 2003.

We compare modularity results obtained by QCA algorithm at each network snapshot to Blondel method as well as to MIEN method. It reveals from Figure 4(a) that the modularities returned by QCA are very close to those obtained by the static method with much more stabler and are far higher than those of MIEN. In particular, the modularity values produced by QCA algorithm cover from 94% up to 100% that of Blondel method and from 6% to 10% higher than MIEN. Moreover, our method reveals a much better network community structure since it discovers many more communities than both the static method and MIEN as the network evolves (Figure 4(b)). This can be explained based on the resolution limit of modularity [9]: the static method might disregard some small communities and tend to combines them in order to maximize the overall network modularity.

Second observation on running time shows that QCA outperforms the static method as well as its competitor MIEN: QCA takes at most 2 seconds to complete updating the network structure while Blondel method requires more than triple that amount of time (Figure $4(c)$) and MIEN asks for more than 5x time. In addition, higher NMI scores of QCA than MIEN methods (Figure $4(d)$) implies network communities identified by our approach are not only of high similarity to the ground truth but also more precise than that detected by MIEN, meanwhile the computational cost and the running time are significantly reduced.

C. Facebook social network

This data set contains friendship information (i.e., who is friend with whom and wall posts) among New Orleans regional network on Facebook, spanning from Sep 2006 to Jan 2009 [13]. To collect the information, the authors created several Facebook accounts, joined each to the regional network, started crawling from a single user and visited all friends in a breath-first-search fashion. The data set contains more than 60K nodes (users) connected by more than 1.5 million friendship links with an average node degree of 23.5. In our experiments, the nodes and links from Sep 2006 to Dec 2006 are used to form the basic community structure of the network and each network snapshot is recored after every month during Jan 2007 to Jan 2009 for a total of 25 network snapshots.

Evaluation depicted in Figure $5(a)$ reveals that QCA algorithm achieves competitive modularities in comparison with the static method, and again far better than those obtained by MIEN. In the general trend, the line representing QCA results closely approximates that of the static method with much more stable. Moreover, the two final modularity values at the end of the experiment are relatively the same, which means that our adaptive method performs competitively with the static method running on the whole network.

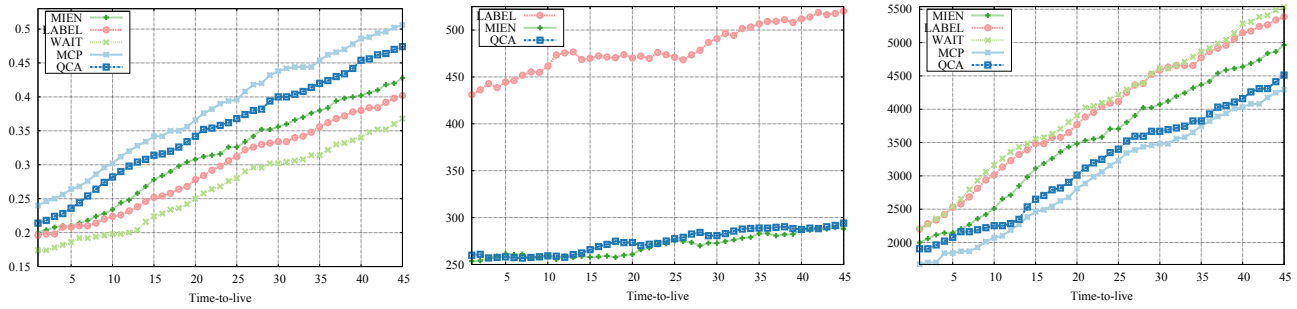
Figure $5(c)$ describes the running time of the three methods on the Facebook data set. As one can see from this figure, QCA takes at least 3 seconds and at most 4.5 seconds to successfully compute and update every network snapshot whereas the static method, again, requires more than triple processing time. MIEN method really suffers on this large scale network when requiring more than 10x amount of QCA running time. This result confirms the effectiveness of our adaptive method when applied to real-world social networks where a centralized algorithm may not be able to detect a good network community structure in a timely manner.

However, there is a limitation of QCA algorithm we observe on this large network and want to point out here: As the the duration of network evolution lasts longer over time (i.e., the number of network snapshots increases), our method tends to divide the network into smaller communities to maximize the local modularity, thus results in an increasing number of communities and a decreasing of NMI scores. Figure $5(b)$ and $5(d)$ describes this observation. For instance, at snapshot #12 (a year after Dec 2006), the NMI score is approximately $1/2$ and gets decaying after this timepoint. It implies a refreshment of network community structure is required at this time, after a long enough duration. This is reasonable since activities on an online social network, especially on Facebook social network, tend to come and go rapidly and local adaptive procedures are not enough to reflect the whole network topology over a long period of time.

V. Application: Social-aware routing in MANETs

In this section, we present an application where the detection of network community structures plays an important role in routing strategies in Mobile Ad Hoc Networks. A MANET is a dynamic wireless network with or without the underlying infrastructure, in which each node can move freely in any direction and organize itself in an arbitrary manner. Due to nodes mobility and unstable links nature of a MANET, designing an efficient routing scheme has become one of the most important and challenging problems on MANETs.

Recent researches have shown that MANETs exhibit the properties of social networks [15][16][17] and social-aware algorithms for network routing are of great potential. This is due to the fact that people have a natural tendency to form



(a) Delivery Ratio (b) Average Duplicate Message (c) Average Delivery Time

Fig. 6. Experimental results on the Reality Mining data set

groups or communities in communication networks, where individuals inside each community communicate with each other more frequent than with people outside. This social property is nicely reflected to the underlying MANETs by the existence of groups of nodes where each group is densely connected inside than outside. This resembles the idea of *community structure* in Mobile Ad hoc Networks.

Multiple routing strategies [16]-[18] based on the discovery of network community structures have provided significant enhancement over traditional methods. However, the community detection methods utilized in those strategies are not applicable for dynamic MANETs since they have to recompute the network structure whenever changes to the network topology are introduced, which results in significant computational costs and processing time. Therefore, employing an adaptive community structure detection algorithm as a core will provide speedup as well as robustness to routing strategies in MANETs.

We evaluate the following five routing strategies (1) *WAIT*: the source node waits and keeps sending or forwarding the messages until it meets the destination node (2) *MCP*: A node keeps forwarding the messages until they reach the maximum number of hops (3) *LABEL*: A node forwards or sends the messages to all members in the destination community [15] (4) *QCA*: A Label version utilizing QCA as the dynamic community detection method and lastly, (5) *MIEN*: A social-aware routing strategy on MANETs [14].

Even thought the *WAIT* and *MCP* algorithms are very simple and straightforward to understand, they provide us helpful information about the lower and upper bounds for message delivery ratio, time redundancy as well as message redundancy. *LABEL* forwarding strategy works as follow: it first finds the community structure of the underlying MANET, assigns each community with the same label and then exclusively forwards messages to destinations, or to next-hop nodes having the same labels as the destinations. *MIEN* forwarding method utilizes MIEN algorithm as a subroutine. *QCA* routing strategy, instead of using a static community detection method, utilizes QCA algorithm for adaptively updating the network community structure and then uses the newly updated structure to inform the routing strategy for forwarding messages.

We choose Reality Mining data set [19] provided by the MIT Media Lab to test our proposed algorithm. The Reality Mining data set contains communication, proximity, location, and activity information from 100 students at MIT over the course of the 2004-2005 academic year. In particular, the data set includes call logs, Bluetooth devices in proximity, cell tower IDs, application usage, and phone status (such as charging and idle) of the participated students of over 350,000 hours ($\tilde{4}0$ years). In this paper, we take into account the Bluetooth information to form the underlying MANET and evaluate the performance of the above five routing strategies.

For each routing method, we evaluate the followings (1) Delivery ratio: The portion of successfully delivered over the total number of messages (2) Average delivery time: Average time for a message to be delivered. (3) Average number of duplicated messages for each sent message. In particular, a total of 1000 messages are created and uniformly distributed during the experiment duration and each message can not exist longer than a threshold *time-to-live*. The experimental results are shown in Figure $6(a)$, $6(b)$ and $6(c)$.

Figure $6(a)$ describes the delivery ratio as a function of *time-to-live*. As revealed by this figure, *QCA* achieves much better delivery ratio than *MIEN* as well as *LABEL* and far better than *WAIT*. This means that *QCA* routing strategy successfully delivers many more messages from the source nodes to the destinations than the others. Moreover, as *time-to-live* increases, the delivery ratio of *QCA* tends to approximate the ratio of *MCP*, the strategy with highest delivery ratio.

Comparison on delivery time shows that *QCA* requires less time and gets messages delivered successfully faster than *LABEL*, as depicted in Figure $6(c)$. It even requires less delivery time in comparison with the social-aware method *MIEN*. This can be explained as the static community structures in *LABEL* can possibly get message forwarded to a wrong community when the destinations eventually change their communities during the experiment. Both *QCA* and *MIEN*, on the other hand, captures and updates the community structures on-the-fly as changes occur, thus achieves better results. Since *MIEN* needs to compress and decompress the network communities whenever network evolves, it may disregard the existence of newly formed communities and thus, may requires extra time to forward the messages.

The numbers of duplicate messages presented in Figure $6(b)$

indicate that both *QCA* and *MIEN* achieves the best results. The numbers of duplicated messages of *MCP* method are way higher than those of the others and are not plotted. In fact, the results of *QCA* and *MIEN* are relatively close and tend to approximate each other as *time-to-live* increases.

In conclusion, *QCA* is the best social-aware routing algorithm among five routing strategies since its delivery ratio, delivery time and redundancy outperform those of the other methods and are only below *MCP* while the number of duplicate messages are much lower. *QCA* also shows a significant improvement over the naive *LABEL* method which uses a static community detection method and thus, confirms the applicability of our adaptive algorithm to routing strategies in MANETs.

VI. Related work

Community detection on static networks has attracted a lot of attentions and many efficient methods have been proposed for this type of networks. Detecting community structure on dynamic networks, however, has so far been an untrodden area. Recent work [2] proposed an innovative method for detecting communities on dynamic networks based on the k-clique percolation technique. This approach can detect overlapping communities; however, it is time consuming, especially on large scale networks. Another recent work of [20] proposed a detection method based on contradicting the network topology and the topology-based *propinquity*, where *propinquity* is the probability of a pair of nodes involved in a community. A work in [11] presented a parameter-free methodology for detecting clusters on time-evolving graphs based on mutual information and entropy functions of Information Theory. [21] proposed a distributed method for community detection in which modularity was used as a measure instead of objective function. A part from that, [22] attempted to track the evolving of communities over time, using a few static network snapshots.

In [23], the authors present a framework for identifying dynamic communities with a constant factor approximation. However, this method does not seem to make sense on real-world social networks since it requires some predefined penalty costs which are generally unknown on dynamic networks. A recent work [14] proposes a social-aware routing strategy in MANETs which also makes uses of a modularity-based procedure name MIEN for quickly updating the network structure. In particular, MIEN tries to compose and decompose network modules in order to keep up with the changes and uses fast modularity algorithm [4] to update the network modules. However, this method performs slowly on large scale dynamic networks due to the high complexity of [4].

VII. Conclusion

In this paper, we presented QCA, an adaptive algorithm for detecting and tracing community structures in dynamic social networks where changes are introduced frequently. We show that our adaptive algorithms are not only effective in updating and identifying high quality network community structure but also has the great advantage of fast running time, which is suitable for large and rapidly changing online social networks. We prove some theoretical results which are the basic observations of our approach. Finally, via a practical social-aware routing strategy in MANETs, we show that our QCA algorithm promises enormous realistic applications in mobile computing as it can be combined or integrated as a community detection core.

Acknowledgment

This work is partially supported by the National Science Foundation under CAREER Award grant number 0953284, by the DTRA Young Investigator Program under grant number HDTRA1-09-1-0061 and by the DTRA under grant number HDTRA1-08-10.

References

[1] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99, 2002.
[2] G. Palla, P. Pollner, A. Barabasi, and T. Vicsek. Social group dynamics in networks. *Adaptive Networks*, 2009.
[3] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Physical review. E. 80*, 2009.
[4] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E 69*, 2003.
[5] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.: Theory and Experiment*, 2008.
[6] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phy. Rev. E 69*, 2004.
[7] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2), 2008.
[8] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E 70*, Aug 2004.
[9] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *PNAS*, 104, 2007.
[10] Z. Ye, S. Hu, and J. Yu. Adaptive clustering algorithm for community detection in complex networks. *Physical Review E*, 78, 2008.
[11] S. Jimeng, C. Faloutsos, S. Papadimitriou, and Philip S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *KDD*, 2007.
[12] ArXiv dataset. http://www.cs.cornell.edu/projects/kddcup/datasets.html. *KDD Cup 2003*, Feb 2003.
[13] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in facebook. In *2nd ACM SIGCOMM Workshop on Social Networks*, 2009.
[14] T. N. Dinh, Y. Xuan, and M. T. Thai. Towards social-aware routing in dynamic communication networks. *IPCCC*, 2009.
[15] P. Hui and J. Crowcroft. How small labels create big improvements. *PERCOMW*, 2007.
[16] E. M. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *MobiHoc '07*, 2007.
[17] A. Chaintreau, P.Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6, 2007.
[18] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *MobiHoc '08*, 2008.
[19] E. Nathan and A. Pentland. Reality mining: sensing complex social systems. *Personal Ubiquitous Comput.*, 10(4), 2006.
[20] Y. Zhang, J. Wang, Y. Wang, and L. Zhou. Parallel community detection on large networks with propinquity dynamics. In *KDD '09*. ACM, 2009.
[21] P. Hui, E. Yoneki, S. Chan, and J. Crowcroft. Distributed community detection in delay tolerant networks. In *Proc. MobiArch*, 2007.
[22] J. Hopcroft, O. Khan, B. Kulis, and B. Selman. Tracking evolving communities in large linked networks. *PNAS*, 101, 2004.
[23] T. Chayant and B. Tanya. Constant-factor approximation algorithms for identifying dynamic communities. In *KDD '09*, 2009.