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 containment in cellular networks. Unfortunately, understanding this network 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 the network structure, through a series of changes, by only using the structure identified from previous network snapshots, but also the ability of tracing the evolving of its communities over time. To illustrate the effectiveness of our algorithm, we extensively test QCA on real-world dynamic social networks including ENRON email, arXiv e-print citation and Facebook networks. Finally, we demonstrate the bright applicability of our algorithm via two practical applications in MANETs routing strategies and worm containment on social networks. The comparative results reveal that social-aware routing strategies in MANETs and worm containment methods in social networks, which employ QCA as a community detection core, outperform the current available methods.

I. INTRODUCTION

Many social networks exhibit the property of containing community structure [14], [27], 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 input in graph clustering, whereas the number of communities is typically unknown in community detection. Detecting communities in a network provides us a meaningful insight to the network structure as well as its organization principles. Furthermore, knowing the community structures could provide more helpful points of view to some uncovered parts of the network, thus helps in preventing potential diseases such as virus or worm propagation. Studies on community detection on static networks can be found in an excellent survey [22] as well as in the work of [25], [1], [26], [4] 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 network 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 friend and friend 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 re-identification. However, the rapidly and unpredictably changing topology of dynamic social networks makes it an extremely complicated and challenging problem.

Although one can possibly run any of the static community detection methods, which are widely available [25], [8], [1], [26], 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. Figure 1 briefly generalizes the idea of dynamic network structure adaptation.

Detecting community structure in 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. 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 exists groups of nodes which are densely connected inside than outside in the underlying MANET as a reflection, and therefore, forms community structure in that underlying 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 closely related) community as the destination. By doing this way, we can avoid unnecessary messages forwarding through nodes in different communities, thus lowering down the number of duplicate messages as well as reducing overhead information, which are essential in MANETs.

Another practical application of community detection includes worm containment in dynamic social networks. With the introduction of the World Wide Web and online social networks, people now have sought ways to socialize and make new friends online over a greater distance. Popular social network sites such as Facebook, Twitter and Bebo have witnessed rapid increases in space and the number of online users over a short period of time. However, alongside with these fast expands comes the threat of malicious softwares such as viruses, worms or false information propagation (Facebook, one of the most famous online social networks, experienced a wide propagation of a trojan worm in late 2008 and “Koobface”-popularly known as “Facebook virus” - was the name of the worm that spread out widely through not only Facebook but also Bebo, MySpace and Friendster social networks [12], [16]). A natural approach to this problem is to send some sort of patches to all users in the network to stop the worm or virus propagation. However, doing so is too costly and almost impossible on a giant social network such as Facebook. Another possible solution is to prevent their propagation to a larger population by sending patches to only most influential users and let them redistribute these patches to the others. Naturally, the smaller number of patched users the better. But how to choose that set of influential users of a small size in such a large social network? This is where community structure coming into the picture. As we will see in Section VII, a good and up-to-date community structure will help in providing us a tighter set of important users, thus lowers down the number of sending patches and reduces overhead information sent over the social networks.

The contributions of this paper are:

- We propose QCA, a fast and adaptive algorithm for efficiently identify 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 structure in a timely manner.
- We employ QCA as a community identification core in routing strategies in MANETs and in worm containment methods on social networks. 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 data sets. In sections V and VI, we present two practical applications of our approach in routing in MANETs and worm containment on social networks, respectively. In Section VII, we discuss the related work and finally conclude our work in Section VIII.

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 \( C = \{C_1, C_2, ..., C_k\} \) denote a collection of disjoint communities of \( G \) where \( C_i \subseteq C \) is a community. 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 \( e_S \) 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 \( \Delta V^s \) and \( \Delta E^s \) the set 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.,
A dynamic network $\mathcal{G}$ is a sequence of network snapshots evolving over time: $\mathcal{G} = (G^0, G^1, \ldots, G^s)$. To quantify the goodness of a network community structure, we take into account the network topology affect the structure of its communities. Efficiently detect and identify the network community structure snapshots as well as tracing the evolution of the network. Therefore, our objective is to find a community assignment for each vertex in the network such that $Q$ is maximized. Modularity, just like other quality measurements for community identifications, has some certain disadvantages such as its non-locality and scaling behavior [4] or resolution limit [13]. 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 network $\mathcal{G} = (G^0, G^1, \ldots, G^s)$ where $G^0$ is the original network and $G^1, G^2, \ldots, G^s$ are the network snapshots obtained through $\Delta G^1, \Delta G^2, \ldots, \Delta G^s$, 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 $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ 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 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 details the possible behaviors of dynamic network community structure in Section III-A.

In order to reflect changes introduced to a social network, its underlying graph representation 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, introducing or removing 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 thus 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 is 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 $\mathcal{C}$, which we refer to as a 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 one of the available static community detection methods [25], [8], [1]. To obtain a good basic structure, we choose the method proposed by Blondel et al in [1] which produces a network community structure of high modularity in a reasonable running time [22].

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 $Q$ intact. The interesting case happens, as it always does, when $u$ comes with edges connecting one or more existing communities. In this latter situation, we need to determine which community $u$ should join in order to maximize the gained modularity. There are several local methods introduced for this task, for instance the algorithms of [25], [8]. Our method is inspired by a physical approach proposed in [32], in which each node is influenced by two forces $F^u_{in}(C)$ (to keep $u$ stays inside community $C$) and $F^u_{out}(C)$ (the force a community $C$ makes in order to bring $u$ to $C$) defined as follow: $F^u_{in}(C) = e_{in}u - \frac{d_{out}(d_{in} - d_{out})}{2M}$ and $F^u_{out}(u) = \max_{S \in NC(u)} \left( e^u_S - \frac{d^u_{in} - d^u_{out}}{2M} \right)$ where $d_{out}$ is of opposite meaning of $d_{in}$.

Taking into account the above two forces, a node $u$ can actively determine its best community membership by computing those forces and either let itself join in the community having the highest $F^u_{out}(u)$ (if $F^u_{out}(u) > F^C_{in}(u)$) or stays put in the current community otherwise. By Theorem 1, we show the connection between these forces and the objective function modularity, i.e., letting a new node joining in the community with the 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 Algorithm 1.

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

**Proof**: Let $D$ be a community of $G$ and $D \neq C$, we show that joining $u$ in $D$ contributes less modularity than joining $u$ in $C$. The overall modularity $Q$ when $u$ joins in $C$ is

$$Q = \frac{m_C + e^{u}_C}{M + p} \frac{(d_C + e^{u}_C + p)^2}{4(M + p)^2} + \frac{m_D}{M + p} \frac{(d_D + e^{u}_D)^2}{4(M + p)^2} + A$$

where $A$ is the summation of other modularity contributions. Similarly, joining $u$ to $D$ gives

$$Q' = \frac{m_C}{M + p} \frac{(d_C + e^{u}_C)^2}{4(M + p)^2} + \frac{m_D + e^{u}_D}{M + p} \frac{(d_D + e^{u}_D + p)^2}{4(M + p)^2} + A$$

and

$$Q - Q' = \frac{1}{M + p} \left( e^{u}_C - e^{u}_D + \frac{p(d_D - d_C + e^{u}_D - e^{u}_C)}{2(M + p)} \right)$$

Since $C$ is the community that gives the maximum $F^C_{out}(u)$, we obtain

$$e^{u}_C - e^{u}_D + \frac{p(d_D - d_C + e^{u}_D - e^{u}_C)}{2(M + p)} > 0$$

which implies

$$e^{u}_C - e^{u}_D + \frac{p(d_D - d_C + e^{u}_D - e^{u}_C)}{2(M + p)} > 0$$

Hence, $Q - Q' > 0$ and thus the conclusion follows.

**Algorithm 1 New_Node**

**Input**: New node $u$ with associated links; Current structure $C_i$.

**Output**: An updated structure $C_{i+1}$.

1: Create a new community of only $u$;
2: for $v \in N(u)$ do
3: let $v$ determine its best community;
4: end for;
5: for $C \in NC(u)$ do
6: find $F^{C}_{out}(u)$;
7: end for;
8: let $C_u = \arg \max_{C \in C} \{ F^{C}_{out}(u) \}$;
9: $C_{i+1} \leftarrow (C_i \backslash C_u) \cup (C_u \cup u)$;

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 $C$, its present 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 could 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 Lemma 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 clusters 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 for handling a new social link is described in Algorithm 2.

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

**Proof**: The portion $Q_C$ that community $C$ contributes to the overall modularity $Q$ is determined by:

$$Q_C = \frac{m_C}{M} - \frac{d^2_C}{4M^2}$$

When a new edge coming in, the new modularity $Q'_C$ is

$$Q'_C = \frac{m_C + 1}{M + 1} - \frac{(d_C + 2)^2}{4(M + 1)^2}$$
Now, taking the difference between the two expressions gives
\[
\Delta Q_C = Q_C - Q_C
\]
\[
= \frac{4M^3 - 4m_CM^2 - 4d_CM^2 - 4m_CM + 2d^2_CM + d_C^2}{4(M + 1)^2M^2}
\geq \frac{4M^3 - 6d_CM^2 - 2d_CM + 2d^2_CM + d_C^2}{4(M + 1)^2M^2}
\geq \frac{(2M^2 - 2d_CM - d_C)(2M - d_C)}{4(M + 1)^2M^2} \geq 0
\]
(since \(m_C \leq \frac{d_C}{2}\))

The last inequality holds since \(d_C \leq M - 1\) implies \(2M^2 - 2d_CM - d_C \geq 0\).

**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.

**Proof:** Assume the contradiction, i.e, \(C\) should be divided into smaller modules when an edge is added into it. Let \(X_1, X_2, \ldots, X_k\) be disjoint subsets of \(C\) representing these modules. Denote \(d_i\) and \(e_{ij}\) the degree of vertices inside \(X_i\) and the links going from \(X_i\) to \(X_j\), respectively. Assume that, W.L.O.G, when an edge is added inside \(C\), it is added to \(X_1\).

We will show that
\[
\sum_{i \neq j} \frac{d_id_j}{2M} < \sum_{i \neq j} e_{ij} < \sum_{i \neq j} \frac{d_id_j}{2M} + 1
\]
which cannot happen since \(\sum_{i \neq j} e_{ij}\) is an natural number.

Recall that
\[
Q_C = \frac{m_C}{M} - \frac{\sum_{i \neq j} d_id_j}{4M^2}
\]
\[
q(X_i) = \frac{m_i}{M} - \frac{\sum_{i \neq j} e_{ij}}{4M^2}
\]
Since \(C\) is indivisible before adding an edge, we have \(Q_C > Q_{X_1} + Q_{X_2} + \ldots + Q_{X_k}\), or equivalently,
\[
\sum_{i \neq j} e_{ij} < \frac{\sum_{i \neq j} d_id_j}{2M}
\]
\[
\sum_{i \neq j} d_id_j < 2M\sum_{i \neq j} \frac{d_i d_j}{2M} + \sum_{i \neq j} d_i d_j
\]
Now, assume that the new edge is added to \(X_1\) and \(C\) is split into \(X_1, X_2, \ldots, X_k\) which implies that dividing \(C\) into \(k\) smaller communities will increase the overall modularity, i.e,
\[
Q_C' < Q_{X_1} + Q_{X_2} + \ldots + Q_{X_k}. \quad \text{Now, } Q_C' < \sum_{i = 1}^k Q_{X_i}
\]
\[
\sum_{i = 1}^k m_i + \sum_{i \neq j} e_{ij} + 1 = \left( \sum_{i = 1}^k d_i + 2 \right) - \frac{4(M + 1)^2}{4(M + 1)^2} + \sum_{i = 2}^k \frac{m_i}{M + 1} - \frac{d_i^2}{4(M + 1)^2}
\]
\[
\leq \frac{m_1 + 1}{M + 1} - \frac{(d_1 + 2)^2}{4(M + 1)^2} + \sum_{i = 2}^k \frac{m_i}{M + 1} - \frac{d_i^2}{4(M + 1)^2}
\]

Moreover, since \(\sum_{i = 1}^k d_i - 2d_1 < 2M\), we have
\[
\frac{\sum_{i = 1}^k d_i - 2d_1 + \sum_{i \neq j} d_id_j}{2M} < \frac{\sum_{i = 1}^k d_i - 2d_1 + \sum_{i \neq j} d_id_j}{2M} + 1
\]
and thus the conclusion follows.

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

**Proof:** Let \(C \equiv C(u)\) and \(D \equiv C(v)\). Recall the outer force that a community \(S\) applies the vertex \(u\) is
\[
F_{out}^S(u) = e_u^S - \frac{d_u d_{outS}}{2M}
\]
We will show that the present of edge \((u, v)\) will strengthen \(F_{out}^D(u)\) while weaken the other outer forces \(F_{out}^S(u)\), i.e, we will show that \(F_{out}^D(u)\) increases while \(F_{out}^S(u)\) decreases.

\[
F_{out}^D(u)_{new} - F_{out}^D(u)_{old} = (e_u^D + 1 - (d_u + 1)(d_{outD} + 1)) - (e_u^D - d_u d_{outD})
\]
\[
= \frac{2M + d_u d_{outD}}{2M} - d_u d_{outD} + d_{outD} + d_u + 1
\]
\[
\geq \frac{2M + d_u d_{outD}}{2M} - d_u d_{outD} + d_{outD} + d_u + 1 > 0
\]
and thus \(F_{out}^D(u)\) is strengthened when \((u, v)\) is introduced.

Furthermore, for any community \(S \not\in \{C, D\},
\[
F_{out}^S(u)_{new} - F_{out}^S(u)_{old} = (e_u^S - \frac{(d_u + 1)(d_{outS})}{2(M + 1)}) - (e_u^S - \frac{d_u d_{outS}}{2M})
\]
\[
= d_u \left( \frac{d_{outS} + d_{outS} + 1}{2M} - \frac{d_{outS} + 1}{2(M + 1)} \right) < 0
\]
which implies \(F_{out}^S(u)\) is weaken when \((u, v)\) is connected. Hence, the conclusion follows.

**Theorem 3:** Assume that a new edge \((u, v)\) is added to the network. Let \(C \equiv C(u)\) and \(D \equiv C(v)\). If \(\Delta q_{u,C,D} \equiv \Delta q_{u,C,D} = \frac{4(M + 1)(e_u^D + 1 - e_u^C) + e_u^C(d_{outD} - 2d_u - d_C) - 2(d_u + 1)(d_u + 1 + d_D - d_C)}{2} > 0\) then joining \(u\) to \(D\) will increase the overall modularity.
Proof: Node $u$ should leave its current community $C$ and join in $D$ if $Q_{D+u} - Q_{C-u} > Q_{C} + Q_{D}$ or

$$m_D + e_D + 1 - \frac{(d_D + d_u + 2)^2}{4(M+1)^2} + \frac{m_C - e_C}{M+1} - \frac{(d_C - d_u - e_C)}{4(M+1)^2} > 0.$$ 

Corollary 1: If condition in Theorem 3 is not satisfied, then neither $u$ nor its neighbors should be moved to community $D$.

Proof: The proof follows immediately from Theorem 3.

Algorithm 2 New Edge

Input: Edge $(u, v)$ to be added; Current structure $C_t$.  
Output: An updated structure $C_{t+1}$.

1: if $(u$ and $v$ are new vertices) then
2:     $C_{t+1} \leftarrow C_t \cup \{u, v\}$;
3: else if $C(u) \neq C(v)$ then
4:     if $\Delta q_{C(u), C(v)}(u, v) < 0$ and $\Delta q_{C(u), C(v)}(v, u) < 0$ then
5:         return $C_{t+1} \equiv C_t$;
6:     else
7:         $w = \arg \max \{\Delta q_{C(u), C(v)}(w, u), \Delta q_{C(u), C(v)}(w, v)\}$;
8:         Move $w$ to the new community;
9:         for $t \in N(u)$ do
10:             Let $t$ determine its best community;
11:         end for
12:     Update $C_{t+1}$;
13: end if
14: end if

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 with other communities. To give a sense of it, let’s consider two extreme cases when a single degree node and a node with a single degree in a community is removed. If a single degree node is removed, it leaves the resulted community unchanged. If a single degree edge is removed, the current community might be disconnected and broken into smaller pieces which then are merged to other communities as depicted in Figure 2(c). Therefore, identifying the leftover structure of $C$ is an crucial part once a vertex in $C$ is removed. To quickly and efficiently handle this task, we utilize the clique percolation method presented in [27]. 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 these left over communities of $C$ choose their best communities to merge in. The detailed algorithm is presented in Algorithm 3.

4) Edge removal: In the last case when an edge $e = (u, v)$ is about to be removed, we divide further into four subcases

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 $v$ itself. When $e$ is an inter-community link, the removal of $e$ will strengthen the current community structure as in Lemma 3 and hence, we just remove $e$ and keep the previous network community structures intact.

The last but most complicated case happen 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 identify structure of the remaining module becomes 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 result in 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 $(u, v)$ leaves some kind of “quasi-clique” structures inside $C$. Therefore, we find all maximal “quasi-cliques” within the current community and have them as well as leftover singletons determine their best community to join in. The detailed procedure is described in Algorithm 4.

Lemma 3: If $C_1$ and $C_2$ are two communities in the current snapshot of $G$, then the removal of an inter-community link connecting $C_1$ and $C_2$ will strengthen the modularity contribution of $C_1$ and $C_2$.

Proof: Let $Q_1$ and $Q_1'$ be the modularities of $C_1$ before and after the removal of that link. We show that $Q_1' > Q_1$ (and similarly, $Q_2' > Q_2$) and thus, $C_1$ and $C_2$ contribute higher modularity values to the network. Now,

$$Q_1 - Q_1' = \left(\frac{m_1}{M-1} - \frac{(d_1 - 1)^2}{4(M-1)^2}\right) - \left(\frac{m_1}{M} - \frac{d_1^2}{4M^2}\right)$$

$$= m_1 \left(\frac{1}{M-1} - \frac{1}{M}\right) + \frac{1}{4} \left(\frac{d_1}{M} - \frac{d_1 - 1}{M-1}\right) \left(\frac{d_1}{M} + \frac{d_1 - 1}{M-1}\right)$$

$$> 0$$

since all terms are all positive. The same technique applies to
show that $Q'_2 > Q_2$. Hence, the condition follows.

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

**Proof:** Assume the contradiction, i.e., after the removal of $(u, v)$, $C$ is broken into two modules $C_1$ and $C_2$ which yields higher modularity: $Q_1 + Q_2 > Q_C$. In terms of formulation, we have

$$Q_1 + Q_2 > Q_C$$

$$\iff \frac{m_1 - d_1^2}{M-1} + \frac{m_2 - 2d_2}{4(M-1)^2} + \frac{m_1 - (d_2 - 2)^2}{4(M-1)^2} > \frac{m_1 + m_2 + e_{12} - 1}{M-1} - \frac{m_1 - m_2 - 1}{M-1}$$

$$\iff e_{12} > \frac{d_1 d_2 - d_1^2}{2(M-1)}$$

Thus, the conclusion follows.

**Theorem 4:** (Testing of module separation) For any community $C$, let $\alpha$ and $\beta$ denote the lowest and the second highest degree of vertices in $C$, respectively. Assume that an edge $e$ is removed from $C$. If there does not exist subsets $C_1$ and $C_2 \subseteq C \setminus \{u\}$ such that

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

then dividing $C$ into two subsets will not benefit the overall modularity.

**Proof:** From Lemma 5, it follows that in order to really benefit the overall modularity we must have

$$\frac{d_1 d_2}{2M} < e_{12} < \frac{d_1 d_2 + 1}{2(M-1)} + 1$$

Now we find an upper bound for the RHS inequality. Since $d_1 + d_2 = d$, we have

$$e_{12} < \frac{d_1 d_2 - d + 1}{2(M-1)} + 1 \leq \frac{(d_1 + d_2)^2 - d + 1}{2(M-1)} + 1$$

$$\leq \frac{d_1^2 - d + 1}{2(M-1)} + 1 = \frac{(d_C - 2)^2}{8(M-1) + 1}$$

For a lower bound of the LHS inequality, we rewrite $d_1 d_2$ as

$$d_1 d_2 = d_1(d - d_1) = d_1 d - d_1^2$$

and find the non-zero minimum value on the range $d_1 \in [\alpha, \beta]$. By taking the derivative and solving for critical point, $d_1 d - d_1^2$ is minimized either at $d_1 = \alpha$ or $d_1 = \beta$. Thus,

$$\min\{\alpha(d - \alpha), \beta(d - \beta)\} \leq \frac{d_1 d_2}{2M} < e_{12} \leq \frac{d_1 d_2 + 1}{2(M-1)} + 1$$

$$\leq \frac{(d_C - 2)^2}{8(M-1) + 1}$$

**Algorithm 4 Edge Removal**

**Input:** Edge $(u, v)$ to be removed; Current structure $C_t$.

**Output:** An updated clustering $C_{t+1}$.

1. if $(u, v)$ is a single edge then
2. $C_{t+1} = (C_t \setminus \{u, v\}) \cup \{u\} \cup \{v\}$;
3. else if Either $u$ or $v$ is of degree one then
4. $C_{t+1} = (C_t \setminus \{u\}) \cup \{u\} \cup (C_t \setminus \{u\})$;
5. else if $C(u) \neq C(v)$ then
6. $C_{t+1} = C_t$;
7. else
8. % Now $(u, v)$ is inside a community $C$
9. $L = \text{(Maximal "quasi-cliques" in $C$)}$;
10. Let the singletons in $C \setminus L$ consider their best communities;
11. end if
12. Update $C_t$.

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

---

**Algorithm 5 QCA**

**Input:** Current structure $C_t$.

**Output:** Updated structure $C_{t+1}$.

1. if $Q_t^2 > Q_{t+1}$ then
2. $C_{t+1} = (C_t \setminus \{u, v\}) \cup \{u\} \cup \{v\}$;
3. else if Either $u$ or $v$ is of degree one then
4. $C_{t+1} = (C_t \setminus \{u\}) \cup \{u\} \cup (C_t \setminus \{u\})$;
5. else if $C(u) \neq C(v)$ then
6. $C_{t+1} = C_t$;
7. else
8. % Now $(u, v)$ is inside a community $C$
9. $L = \text{(Maximal "quasi-cliques" in $C$)}$;
10. Let the singletons in $C \setminus L$ consider their best communities;
11. end if
12. Update $C_t$.

---

**Note:** The above algorithms are designed for dynamic social networks where communities evolve over time due to the addition or removal of edges. The goal is to efficiently update the community structure in a way that maximizes modularity, a common measure of community structure. These algorithms use a combination of greedy and heuristic approaches to ensure that the community structure remains high-quality even as the network evolves.
IV. EXPERIMENTAL RESULTS

In this section, we present the experimental results of our QCA algorithm on identifying and updating the network community structure of dynamic social networks. To illustrate the strength and effectiveness of our approach, we choose three popular real-world networks including ENRON email network [20], arXiv e-print citation network (provided by the KDD cup 2003 [7]) and Facebook online social network [29]. The static method we are comparing to is the one proposed by Blondel et al [1], which we refer to as Blondel method or static method. In addition to the static method, we further compare ours to an recent dynamic adaptive method called MIEN proposed in [11]. Basically, MIEN tries to compress and decompress network modules into nodes in order to adapt with the changes and uses fast modularity method [25] to keep the network structure updated. In particular, we will show in the experiments the following quantities (1) the modularity values of our adaptive method versus those of the static method (2) the quality of the identified network community structures via NMI scores and (3) the processing time of our approach in comparison with the static method. The above networks expose to contain community structure due to their high modularity values, which is the main reason for them to be chosen.

Algorithm 5 Quick Community Adaptation (QCA)

Input: $G \equiv G_0 = (V_0, E_0)$, $\xi \equiv \{\xi_1, \xi_2, \ldots, \xi_n\}$ a collection of simple events
Output: Community structure $C_t$ of $G_t$ at time $t$
1: Use [1] to find an initial community clustering $C_0$ of $G_0$;
2: for ($t \leftarrow 1$ to $s$) do
3: $C_t \leftarrow C_{t-1}$
4: if $\xi_t = \text{newNode}(u)$ then
5: $\text{New}_\text{Node}(C_t, u)$;
6: else if $\xi_t = \text{newEdge}(u,v)$ then
7: $\text{New}_\text{Edge}(C_t, (u,v))$;
8: else if $\xi_t = \text{removeNode}(u)$ then
9: $\text{Remove}_\text{Node}(C_t, u)$;
10: else
11: $\text{Remove}_\text{Edge}(C_t, (u,v))$;
12: end if
13: end for

In order to quantify the similarity of the identified community structure with the ground truth, i.e., the quality of the identified community structure, we adopt a well known measure in Information Theory called Normalized Mutual Information (NMI). Normalized mutual information has been proven to be reliable and is currently very often used in testing community detection algorithms [22]. $NMI(U,V)$ is 1 if structures $U$ and $V$ are identical and is 0 if they are totally separated. Due to space limit, the readers are encouraged to read [22] 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 structures. Our QCA algorithm takes into account that basic community structures 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 Inc., from Jan 1999 to July 2002 [20]. 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 simulative 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 outputted 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 network evolution.

The running time of QCA algorithm and the static method in this small network are relatively close: the static method requires 1 second to complete each of its tasks while our QCA does not even ask for a second (Figure 3(c)). In this data set, MIEN requires a little more time (1.5 second in average) to complete the task. Time and computational cost are significantly reduced in our QCA algorithm due to the fact that our approach only processes on network changes while static method has work on the whole network every time.

Due to the lack of the proper information about real communities in Enron Inc., we use community structure identified by the static method as a reference to the ground truth. The quantity $NMI(QCA, Blondel)$ indicates how community labels assigned by QCA method 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 meaning. 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 cost.

B. arXiv e-print citation network

The arXiv e-print citation network [7], 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 MIEN method. It reveals from Figure 4(a) that the modularity values returned by QCA are very close to those obtained by the static method with much more stable and are weight 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 detects and discovers many more communities than both the static method and MIEN as the network evolves 4(b). This can be explained based on the resolution limit of modularity [13]: 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 algorithm outperforms the static method as well as its adaptive competitor: 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 5 times. In addition, higher NMI scores of QCA than MIEN methods (Figure 4(d)) implies community structure identified by our approach are not only of high similarity to those detected by applying the static method directly 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 [29]. 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 recorded after every month during Jan 2007 to Jan 2009 for a total of 25 network snapshots.

Evaluation depicted in Figure 5(a) reveals that our QCA algorithm achieves competitive modularity values 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 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 of both MIEN method and ours. 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 Facebook social network, tend to come and go rapidly and local adaptation procedures are not enough to capture or 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 [17] [9] [5] 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 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 [9]-[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 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 they reach 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 [17] (4) QCA: A Label version utilizing QCA as the dynamic community detection method and lastly, (5) MIEN: A recently proposed social-aware routing method on MANETs [11].

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 [24] 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 (40 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
Online social networks have become more and more popular nowadays. Since their introduction, popular social network sites such as Facebook, Twitter, Bebo and MySpace have attracted millions of users worldwide, many of whom have integrated those sites into their everyday lives. On the bright side, online social networks are ideal places for people to keep in touch with friends and colleagues, to share their common interests, to hold discussions in forums or just simply to socialize online. However, on the other side, social networks are also fertile grounds for the rapid propagation of malicious softwares (such as viruses or worms) and false information.

Facebook, one of the most famous online social networks, experienced a wide propagation of a trojan worm named “Koobface” in late 2008. Koobface made its way not only through Facebook but also Bebo, MySpace and Friendster social networks [12], [16]. Once an user is infected with Kooface, this worm scans through the current user’s profile and sends out fake messages or wall posts to everyone in the user’s friend list with titles or comments appeal to people’s curiosity. If one of the user’s friends, attracted by the comments without a shadow of doubt, clicks on the link and installs the fake “flash player”, he will be infected. Koobface’s life will then cycle on this newly infected user’s machine. Since people are able to access social network sites via cell phones nowadays, worm’s targets are now not only computers but also mobile devices.

The problem of worm containment becomes more and more complicated on a dynamic social network as this kind of network evolves and changes rapidly over time. The dynamics of social networks thus gives worms more chances to spread out faster and wider as they could flexibly switch between new and existing users in order to propagate. Therefore, the problem of containing worm propagation on social networks is extremely challenging in the sense that a good solution at the previous time step might not be sufficient or effective at the next time step. Although one can recompute a new solution at each time the network changes, doing so would result in heavy computational costs and time consuming as well as worms spreading out wider during the recomputing process. A better solution should quickly and adaptively update the current worm containing strategy based on changes in network.
topology, thus could avoid the hassle of recomputing from scratch whenever the network evolves.

There are many proposed methods dealing with worm containment on computer networks by either using a multi-resolution approach to enhance the power of threshold-based detection [28], or using a simplification of the Threshold Random Walk scan detector [31], or by measuring the velocity of the number of new connections and infected hosts [10], or using fast and efficient worm signature generation [21], [23]. There are also several method proposed for cellular and mobile networks [30], [3], [2]. However, all of these above approaches fail to take into account the community structure as well as the dynamics of social networks, thus might not be appropriate for our problem. A recent work [34] proposed a social-based patching scheme for worm containment on cellular networks. However, this method encounters the following limitations on a real social network (1) its clustered partitioning does not necessarily reflect the natural network community structure (2) it requires the number of clusters k (which is generally unknown for social networks) must be specified beforehand and (3) it exposes weaknesses when dealing with dynamics of the network.

To overcome these limitations, our approach first utilizes QCA to identify the network community structure and then adaptively keeps this structure fresh and updated as the network evolves. Once the network communities are detected, our patch distribution procedure will select the most influential users from different communities in order to sending patches. These users, as soon as they receive patches, will apply them to first disinfect the worm and then redistribute them to all friends in their communities. These actions will contain worm propagation to only some communities of the social network and prevent it from spreading out to a larger population. To this end, a quick and precise community detection method will definitely help the network administrator to select a more sufficient set of critical users to send patches, thus lowers down the number of sent patches as well as overhead information over the social network.

We next describe our patch distribution procedure. This procedure takes into account the community structure identified from the previous step and selects a set of influential users from each community of the network in order to distribute patches. Influential users of a community are ones having the most relationships or connections to other communities. In the point of an attacker view, these influential users are potentially vulnerable since they not only interact actively within their communities but also with people outside, thus, they can easily fool (or be fooled by) people both inside and outside of their communities. On the other point of view, these users are the best candidates for the network defender to distribute patches since they can easily announce and forward patches to other members and non-members.

We present here a quick greedy algorithm for selecting the set of most influential users in each community. In particular, for each community of G, the algorithm starts with picking the user whose number of social connections to outside communities is the highest and temporarily remove this user from the considering community. This process repeats until there is no connection crossing among communities of G. This set of influential users is the candidate for the network defender for distributing patches. The distribution procedure is presented in Algorithm 6. Note that we can directly apply the algorithm for Vertex Cover problem to the patch distribution to get a 2-approximation algorithm; however, doing so would result in heavy computation and much time consuming, especially on very large social networks.

Algorithm 6 Patch Distribution Procedure

| Input: A community structure S = {S1, S2, ..., Sp} |
| Output: Set of influential users. |
| 1: IS = ∅; |
| 2: for Si ∈ S do |
| 3: while maxv∈Si {dο(v)} > 0 do |
| 4: Let α ← arg maxv∈Si {dο(v)}; |
| 5: IS = IS ∪ α; |
| 6: Temporary remove α from Si; |
| 7: end while |
| 8: end for |
| 9: Send patches to users in IS; |

Experimental Results

We present the experimental results of our method on the Facebook network dataset [29] and compare the results with the social based method (Zhu’s method [34]) via a weighted version of our community update algorithms. This Facebook dataset of 63K users contains data of over 1.5M wall posts and friendship information spanning from Sep 2006 to Jan 2009. One notable feature of this dataset is time information (stamped at every moment the information was recorded) represents the dynamics of the network, which nicely suits to our adaptive method.

The worm propagation model in our experiments mimics the behavior of the famous “Koobface” worm which once spread out widely on Facebook. In our model, worms are able to explore their victim’s friend list and then send out fake messages containing malicious links for propagating. The probability of a victim’s friend activating the worm is proportional to communication frequency between the victim and his friends. The time taken for worms to spread out from one user to another is inversely proportional to the communication frequency between this user and his particular friend. Finally, when a worm has successfully infected a user’s computer, it will start propagating as soon as this computer connects to a specific social network (Facebook in this case).

When the fraction of infected users (the number of infected users over the number of all users) reaches a threshold α, the detection system raises an alarm and patches will automatically be sent to most influential users selected by Algorithm 6. Once an influential user receives a patch, he will first apply the patch to disinfect the worm and then have an option to forward this patch to all friends in his community. Each experiment on this dataset is seeded with 0.02% of users to be infected by worms and worm propagation is simulated through the duration of 2 days. In each experiment,
we compare infection rates of the social-based method of Zhu’s and ours. The infection rate is computed as a fraction of the remaining infected users over the overall infected ones. The number of clusters $k$ in Zhu’s method is set to be 150 (in a static network), 200 and 250 (in dynamic networks), respectively. For each value of $k$, the alarming threshold $\alpha$ is set to be 2\%, 10\% and 20\%, respectively. Each experiment is repeated 1000 times for consistency.

Figure 7, 8 and 9 show the results of our experiments for three different values of $k$ and $\alpha$. We first observe that the longer we wait (the higher the alarm threshold is), the higher number of users we need to send patches to in order to achieve the expected infection rate. For example, with $k = 150$ clusters and an expected infection rate of 0.3, we need to send patches to less than 10\% number of users when $\alpha = 2\%$, to more than 15\% number of users when $\alpha = 10\%$ and to nearly 90\% of total influential users when $\alpha = 20\%$.

Second observation reveals that our approach achieves better infection rates than the social-based method of Zhu’s in a static version of the social network as depicted in Figure 7. In particular, the infection rates obtained in our method are from 5\% to 10\% better than those of Zhu’s. When the network evolves as new users join in and new social relationships are introduced, we resize the number of cluster $k$ and recompute the infection rates of the social based method with the number of cluster $k = 200$ and $k = 250$, and the alarm threshold $\alpha = 2\%$, 10\% and 20\%, respectively. As depicted in Figures 8 and 9, our method, with the power of quickly and adaptively update the network community structure, achieves better infection rates than Zhu’s method meanwhile the computational costs and running time is significantly reduced. As we discussed in Section I, detecting and updating the network community is the central part of a social based patching scheme: a good and up-to-date network community structure will provide the network defender a tighter set of vulnerable users who patches will be sent to and thus, will help to achieve a lower infection rate. Our adaptive algorithms, instead of recomputing the network structure from scratch once changes are introduced, quickly and adaptively update the community structure on-the-fly. Thanks to this frequently updated basic community structure, our patch distribution procedure is able to select a better set of influential users, thus helps in lowering down the number of infected users once patches are sent.

We further look more into the behavior of Zhu’s method when the number of clusters $k$ is running. We compute and compare the infection rates on Facebook dataset for various $k$ ranging from 1K to 2.5K with our approach. We first hope that the more predefined clusters, the better infection rates clustered partitioning method will achieve. However, the experimental results depicted in Figure 10 reveal the opposite. In particular, with a fixed alarming threshold $\alpha = 10\%$ and 60\% patched nodes, the infection rates achieved by Zhu’s method do not decrease but ranging near 28\% while ours are far better (20\%) with much less computational time.

Finally, a comparison on running time on the two approaches shows that time taken for Clustered Partitioning procedure is much more than our community updating procedure and thus, may prevents this method to complete in a timely manner. In particular, our approach takes only 3 seconds for obtaining the basic community structure and at most 30 seconds to complete all the tasks whereas [34] requires more than 5 minutes performing Clustered Partitioning to divide the communication network into modules and selecting the vertex separators. In that delay time, the worm propagation may spread out to a larger population and thus, the solution may not be effective. These above experimental results confirm the robustness and efficiency of our approach on social networks.

VII. 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 [27] proposed an innovative method for detecting communities on dynamic social networks based on a $k$-clique percolation methodology. This approach has the benefit of discovering overlapping communities, however, faces time consuming on large networks. Another recent work of [33] 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 [20] presented a parameter-free methodology for detecting clusters on time-evolving graphs based on mutual information and entropy functions of Information Theory. [19] proposed a method for community detection in a distributed manner, in which modularity was used as an measurement instead of objective function. A part from that, [15] attempts to track the evolving of communities over time, using a few static network snapshots.

In [6], the authors present a framework for identifying dynamic communities with a constant factor approximation. However, this method does not make much sense on real social networks since it requires some predefined penalty costs which are generally unknown on dynamic networks. Additionally, this method also asks for a certain amount of network snapshots in order to process further. A recent work [11] 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 [25] to update the network modules. However, this method performs slowly on large scale dynamic networks due to the high complexity of [25].

VIII. CONCLUSION AND FUTURE WORK

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. In addition, we prove some theoretical results, which are the basic observations of our approach. Finally, via two practical applications in MANETs routing strategies and worm containment on social networks, we show that our algorithm promises enormous realistic applications in mobile computing as it can be combined or integrated into many community detection modules.

Future research directions would include

- Extending our QCA method so that it will be able to identify and update overlapping network community structure, which would be more concrete for dynamic social networks.
- Given the Internet follows power law with exponent $\beta \approx 2.1 \rightarrow 2.3$, will social networks follow power law? If it is, then what would be the power law exponent?
- How would our proposed algorithm behave in the long term? Will it still be able to identify network communities with giant sizes? If not, what would be the upper bounded on the community size and the approximation ratio achieved by our algorithm?

REFERENCES

Infection rates at $\alpha = 10\%$ using 60\% of the patches nodes

**Fig. 10.** Infection rates at $\alpha = 10\%$ using 60\% of the patches nodes


