Overlapping communities in dynamic networks: Their detection and Mobile applications

Nam P. Nguyen, Thang N. Dinh, Sindhura Tokala, My T. Thai
Department of Computer and Information Science and Engineering, University of Florida, USA
{nanguyen, tdinh, sindhura, mythai}@cise.ufl.edu

ABSTRACT

Many practical problems on Mobile networking, such as routing strategies in MANETs, sensor reprogramming in WSNs and worm containment in online social networks (OSNs) share an ubiquitous, yet interesting feature in their organizations: community structure. Knowledge of this structure provides us not only crucial information about the network principles, but also key insights into designing more effective algorithms for practical problems enabled by Mobile networking. However, understanding this interesting feature is extremely challenging on dynamic networks where changes to their topologies are frequently introduced, and especially when network communities in reality usually overlap with each other.

We focus on the following questions (1) Can we effectively detect the overlapping community structure in a *dynamic network*? (2) Can we quickly and adaptively update the network structure only based on its history without recomputing from scratch? (3) How does the detection of network communities help mobile applications? We propose $AFOCS$, a two-phase framework for not only detecting quickly but also tracing effectively the evolution of overlapped network communities in dynamic mobile networks. With the great advantages of the overlapping community structure, $AFOCS$ significantly helps in reducing up to 7 times the infection rates in worm containment on OSNs, and up to 11 times overhead while maintaining good delivery time and ratio in forwarding strategies in MANETs.

1. INTRODUCTION

The rapid and exceptional growing of mobile network has called for a deeper understanding of its organization principles, in order to develop better techniques for a wide range of problems enabled by mobile networking. Many practical problems, such as forwarding and routing strategies in MANETs [1], sensor reprogramming in WSNs [2] and worm containment in cellular networks [3, 4] appear to share an ubiquitous and interesting property: the property of containing community structure, i.e. there are groups of devices or people that frequently communicate more with each other than with the others in the underlying organizations.

In a general concept, a community is a group of tight-knit nodes having more internal than external connections [5, 6]. For instance, a community in MANETs often comprises of sensors or mobile devices that are frequently transmitting data to each other than to other devices. Similarly, since people have a natural tendency to form groups of communication, a community in a cellular network usually consists of mobile devices that often call or text each other. The detection of network community structure, as a result, provides us a better knowledge about its characteristics as well as its organization principles, thereby providing more efficient solutions for mobile networking problems such as forwarding in MANETs or worm containment in OSNs.

Particularly, how does community structure help in Mobile applications? Indeed, detecting network communities is of considerable advantage in mobile networks. Let us consider the worm containment in cellular networks [3], or in OSNs [4, 7]. Nowadays, many social applications such as Facebook, Twitter and FourSquare, are able to run on open-API enabled mobile devices like PDAs and Iphones. However, if such an application is infected with malicious software, such as worms or viruses, this openness will also make it easier for their propagation. A possible solution to prevent worms from spreading out wider is to send patches to critical users and let them redistribute to the others. Intuitively, the smaller the set of important users for sending patches, the better. But how can we effectively choose that set of minimal size? This is where community structure comes into the picture and helps. In particular, we show that selecting users in the boundaries of the overlapped nodes gives a tighter and more efficient set of influential users, thus significantly lowers the number of sent patches as well as overhead information, which are essential in cellular networks and OSNs.

Another great advantage of community structure can be found in the forwarding problem in communication networks. Due to the high mobility of devices, an apparent challenge for a forwarding method is to quickly forward the message from the source to the destination, without introducing too many duplicate messages or overhead information. Since people tend to form groups of communication, there are also communities of tightly connected devices in the underlying network as a refection. A good forwarding strategy, as soon as it discovers the network structure, can actively forward messages to devices sharing more common community labels with the destination, rather than simply sending messages to those in the same community as the destination. With the helpful knowledge of network communities, this strategy will considerably reduce the number of duplicate messages while maintaining good delivery ratios, as we shall see in Section 8. This example, again, amplifies the importance of an efficient community detection method in mobile networks.

Mobile networks in reality are highly dynamic and thus, their communities are not always disjoint from each other. Indeed, their communities often overlap with each other since some active devices can participate in multiple groups at the same time, thereby reassemble the concept of overlapping community structure. Furthermore, most practical models for mobile network problems evolve frequently over time due to the high mobility of participating devices. Although any slight change does not seem to have a significant effect on the network structure, the evolution of the mobile network over a long duration might lead to an unpredictable transformation of its communities, particularly when they can overlap. This drives a crucial need of reidentification. However, the rapid changing network topology makes this an extremely challenging problem, especially on dynamic mobile networks.

A naive solution to the above problem would try to repeatedly execute one of the available static methods [8, 9, 10] to find new communities whenever the network changes; doing so, nonetheless, suffers from some major disadvantages (1) the huge consumption of time and computing resources on large networks and (2) the almost same reactions to some local parts of the network. Intuitively, a much better approach should adaptively update the current community structure based on its history and the network changes only, thus can eventually avoid the hassle of redetection.

Motivated by this intuition and the applicability of overlapping community structure, we propose $AFOCS$ (Adaptive $FOCS$), an adaptive framework for detecting, updating and tracing the evolution of overlapping communities in dynamic mobile networks. Our two-phase framework first identifies basic network communities with $FOCS$ (Finding Overlapping Community Structure), and then employs $AFOCS$ to adaptively update these structures as the network evolves. Since only $AFOCS$ will stay up and handle all changes introduced to the network, this adaptive phase is the main focus of the paper, and hence composes the name of our framework.

In order to effectively handle the network changes, $AFOCS$ decomposes them into simpler events in such a way that each event can be quickly handled. Thanks to this feature, $AFOCS$ can eventually obviate the need of reidentifying the network community structure every time. Both $FOCS$ and $AFOCS$ require β, the *overlapping threshold*, as the only input for their entire operations. This requirement is essential since network communities can overlap at different scales and hence, we do need a control parameter in order to certify how much the overlap means to them.

The contributions of this paper are:

- We propose $AFOCS$, a two-phase adaptive framework for not only detecting and updating the overlapping network communities but also tracing their evolution over time. Theoretical analyses show $AFOCS$ partially achieves more than 0.83% internal density of the optimal solution.
- We evaluate $AFOCS$ on both synthesized and real-world networks in comparison to both the state-of-the art and the most popular static community detection methods $COPRA$ [10] and $CFinder$ [8], as well as to recent adaptive methods $FacetNet$ [11] and $iLCD$ [12]. Experimental results show that $AFOCS$ achieves both competitively results and high quality community structures in a timely manner.
- With $AFOCS$, we propose a new community based forwarding strategy for communication networks that reduces up to 11x overhead information while maintaining competitively delivery time and ratio. We also propose a new social-aware patching scheme for containing worms in OSNs, which helps reducing up to 7x the infection rates on Facebook network dataset.

Organization: In section 2, we discuss the related work and then state the basic notations and problem formulation in section 3. Sections 4, 5 give a complete description of our algorithms and their analyses. Section 6 shows experimental results of $AFOCS$ on synthesis and real-world traces. Sections 7, 8 present two practical applications of our methods in MANETs and OSNs. Finally, we conclude our work in section 9.

2. RELATED WORK

Community detection in complex networks has attracted huge attention since its introduction. In general, one can classify detection methods in two main categories including non-overlapping versus overlapping communities, and on static networks versus on dynamic networks. Many efficient methods have been proposed for detecting both non-overlapping and overlapping communities on *static networks*, among which $CFinder$ [8] and $COPRA$ [10] have remarked themselves as the most popular and most effective methods once fed with correct parameters [13]. A recent work [14] detailed a survey and benchmark on those algorithms.

Detecting communities on *dynamic networks*, both on overlapping and disjoint structures, has so far been an untrodden area. [4] proposed QCA, an adaptive method that can update and trace the network structure through a series of changes. This method is quick and effective, however, is not able to detect overlapped communities. [11] proposed $FacetNet$, a framework for analyzing communities in dynamic networks based on the optimization of snapshot costs. $FaceNet$ is guaranteed to converge to a local optimal solution; however, its convergence speed is slow and its input asks for the number of network communities which are usually unknown in practice. [15] proposed $Stream-Group$, an incremental method to solve the community mining and detect the change points in weighted dynamic graphs. This method is modularity-based thus may inherit the resolution limit while discovering network communities. In another attempt, [16] suggested a particle-and-density based clustering method for dynamic networks, based on the extended modularity and the concepts of nano-community and l-quasi-clique-by-clique. Apart from that, [12] proposed $iLCD$ to find the overlapping network communities by adding edges and then merging similar ones. However, this model might not be sufficient in consideration with the dynamic behaviors of the network when new nodes are introduced or removed, or when existing edges are removed from the network.

3. PROBLEM FORMULATION

3.1 Basic notations

Let $G = (V, E)$ be an undirected unweighted graph representing a network where V is the set of N nodes and E is the set of M connections. Denote by $\mathcal{C} = \{C_1, C_2, ..., C_k\}$ the network community structure, i.e. a collection of subsets of V where each $C_i \in \mathcal{C}$ and its induced subgraph form a community of G. In contrast with the disjoint community structure, we do not restrict the intersection of any two communities to be empty, i.e. we allow $C_i \cap C_j \neq \emptyset$, so that network communities can overlap with each other.

For a node $u \in V$, let d_u, $N(u)$ and $Com(u)$ denote its degree, its neighbors and its set of community labels, respectively. For any $C \subseteq V$, let C^{in} and C^{out} denote the set of links having both endpoints in C and the set of links having exactly one endpoint in C, respectively. Finally, the terms *node-vertex* as well as *edge-link-connection* are used interchangeably.

3.2 Dynamic network model

Let $G_0 = (V_0, E_0)$ be the original input network and $G_t = (V_t, E_t)$ be a time dependent network snapshot recorded at time t. Denote by ΔV_t and ΔE_t the sets

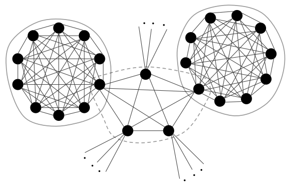


Figure 1: Overlapped v.s. non-overlapped community structure.

of nodes and edges to be added to or removed from the network at time t. Furthermore, let $\Delta G_t = (\Delta V_t, \Delta E_t)$ describe this change in terms of the whole network. The network snapshot at next time step $t + 1$ is expressed as a combination of the previous one together with the change, i.e. $G_{t+1} = G_t \cup \Delta G_t$. Finally, a *dynamic network* $\mathcal{G}$ is defined as a sequence of network snapshots changing over time: $\mathcal{G} = (G_0, G_1, G_2, ...)$.

3.3 Density function

In order to quantify the goodness of an identified community, we use the popular density function Ψ [17] defined as: $\Psi(C) = \frac{|C^{in}|}{\binom{|C|}{2}}$ where $C \subseteq V$. The more C approaches a clique of its size, the higher its density value $\Psi(C)$. In order to set up a threshold on the internal density that suffices for C to be a local community, we propose a function $\tau(C)$ defined as follows:

$$\tau(C) = \frac{\sigma(C)}{\binom{|C|}{2}} \text{ where } \sigma(C) = \binom{|C|}{2}^{1-\frac{1}{\binom{|C|}{2}}}$$

Here $\sigma(C)$ is the threshold on the number of inner connections that suffices for C to be a local community. Particularly, a subgraph induced by C is a local community iff $\Psi(C) \geq \tau(C)$ or equivalently $|C^{in}| \geq \sigma(C)$.

Several functions with the same purpose have been introduced in the literature, for instance, in the work of [9, 18] and it is worth noting down the main differences between them and ours. First and foremost, our functions $\tau(C)$ and $\sigma(C)$ locally process on the candidate community C only and neither require any predefined thresholds or user-input parameters. Secondly, by Proposition 1, $\sigma(C)$ and $\tau(C)$ are increasing functions and closely approach C's full connections as well as its maximal density. That makes $\sigma(C)$ and $\tau(C)$ relaxation versions of the traditional density function, yet useful ones as we shall see in the experiments.

Proposition 1. *The function* $f(n) = n^{1-\frac{1}{n}}$ *is strictly increasing for* $n \geq 4$ *and* $\lim_{n\to\infty} f(n) = n$.

3.4 Objective function

Our objective is to find a community assignment for the set of nodes V which maximizes the overall internal density function $\Psi(\mathcal{C}) = \sum_{C\in\mathcal{C}} \Psi(C)$ since the higher the internal density of a community is, the clearer its structure would be. Unlike the case of disjoint community structure, in which the number of connections crossing communities should be less than those inside them, our objective does not take into account the number of out-going links from each community.

To understand the reason, let's consider a simple example pictured in Figure 1. In the overlapping community structure point of view, it is clear that every clique should form a community on its own and each community shares with the central clique exactly one node. However, in the disjoint community structure point of view, any vertex at the central clique has n internal and $2n$ external connections, which violates the concept of a community in the strong sense. Furthermore, the internal connectivity of the central clique is also dominated by its external density, which implies the concept of a community in weak sense is also violated. (A community C is in *weak sense* if $|C^{in}| > |C^{out}|$, and in *strong sense* if any node in C has more links inward than outward C [19]).

3.5 Problem Definition

Given a dynamic network $\mathcal{G} = (G_0, G_1, G_2, ...)$ where G_0 is the input network and $G_1, G_2, ...$ are network snapshots obtained through a collection of network topology changes $\Delta G_1, \Delta G_2, ...$ over time. The problem asks for an adaptive framework to efficiently detect and update the network overlapping community structure $\mathcal{C}_t$ at any time point t by only utilizing the information from the previous snapshot $\mathcal{C}_{t-1}$, as well as tracing the evolution of the network communities.

In the next section, we present our main contribution: an adaptive framework for (1) identifying basic overlapped community structure in a network snapshot and (2) updating as well as tracing the evolution of the network communities in a dynamic network model. First, we describe $FOCS$, a procedure to identify the basic communities in a static network, and then discuss in great detail how $AFOCS$ adaptively updates these basic communities to cater with the evolution of the dynamic network.

4. BASIC COMMUNITY STRUCTURE

We describe $FOCS$, the first phase of our framework that quickly discovers the basic overlapping network community structure. In general, $FOCS$ works toward the classification of network nodes into different groups by first locating all possible densely connected parts of the network, and then combining those who highly overlap with each other, i.e. those share a significant substructure. In $FOCS$, β (the input overlapping threshold) defines how much substructure two communities can share. Note that $FOCS$ fundamentally differs from [20] in the way it allows $|C_i \cap C_j| \geq 2$ for any subsets C_i, C_j of V, and consequently allows network communities to overlap not only at a single vertex but also at a part of the whole community

4.1 Locating local communities

Local communities are connected parts of the network whose internal densities are greater than a certain level. In $FOCS$, this level is automatically determined based on the function $\tau(\cdot)$ and the size of each corresponding part. Particularly, a local community is defined based on a connection (u, v) when the number of internal connections within the subgraph induced by $C \equiv \{u, v\} \cup (N(u) \cap N(v))$ exceeds $\sigma(C)$, or in other words, when $\Psi(C) \geq \tau(C)$ (Figure 2(a)).

However, there is a problem that might eventually arise during this procedure: the containment of sub communities in an actual bigger one. Intuitively, one would like to detect a bigger community unified by smaller ones if the bigger community is itself densely connected. In order to filter this undesired case, we impose $\Psi\big(\bigcup_{i=1}^{s} C_i\big) < \tau\big(\bigcup_{i=1}^{s} C_i\big) \quad \forall s = 1...|\mathcal{C}|$ (note that some of these unifications do not contain all the nodes). In addition, we allow this locating procedure to skip over tiny communities of size less than 4. This condition is carried out from Proposition 1. This makes sense in terms of mobile or social networks where a group of mobile devices or a social community usually has size larger than 3, and intuitively agrees with the finding of [21, 22]. Those tiny communities will then be identified later. Alg. 1 describes this procedure.

Algorithm 1 Locating local communities

Input: $G = (V, E)$
Output: A collection of raw communities $\mathcal{C}_r$.

for $(u, v) \in E$ **do**
  if $Com(u) \cap Com(v) = \emptyset$ **then**
    Let $C = \{u, v\} \cup N(u) \cap N(v)$;
    if $|C^{in}| \geq \sigma(C)$ and $|C| \geq 4$ **then**
      Define C a local community;
      /*Include C into the raw community structure*/
      $\mathcal{C}_r = \mathcal{C}_r \cup \{C\}$;
    end if
  end if
end for

LEMMA 1. *The time complexity of Alg. 1 is $O(dM)$ where $d = \max_{v \in V} d_v$ (Note: All proofs are excluded due to space constraint).*

LEMMA 2. *All local communities C's detected by Alg. 1 satisfy $|C| \geq 4$ and $\Psi(C) \geq \tau(4) \approx 0.83$. Furthermore, other communities satisfying these conditions will also be detected by Alg. 1.*

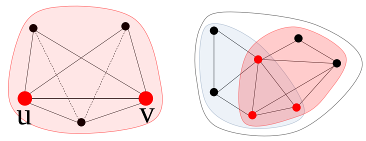
THEOREM 1. *The local community structure $\mathcal{C}_r$ detected by Alg. 1 satisfies $\Psi(\mathcal{C}_r) \geq 0.83 \times \Psi(OPT)$ where OPT is the optimal community assignment that maximizes the overall internal density function.*

4.2 Combining overlapping communities

After Alg. 1 finishes, the raw network community structure is pictured as a collection of (possibly overlapped) dense parts of the network together with outliers. As some of those dense parts can possibly share significant substructures, we need to merge them if they are highly overlapped. To this end, we introduce the overlapping score of two communities defined as follow

$$OS(C_i, C_j) = \frac{|I_{ij}|}{\min\{|C_i|, |C_j|\}} + \frac{|I_{ij}^{in}|}{\min\{|C_i^{in}|, |C_j^{in}|\}}$$

where $I_{ij} = C_i \cap C_j$. Basically, $OS(C_i, C_j)$ values how important the common nodes and links shared between C_i and C_j mean to the smaller community. In comparison with the distance metric suggested in [23], our overlapping score not only takes into account the fraction of common nodes but also values the fraction of common connections, which is crucial in order to combine network communities. Furthermore, $OS(\cdot, \cdot)$ is symmetric and scales well with the size of any community, and the higher the overlapping score, the more those communities in consideration should be merged. In this merging process, we combine communities C_i and C_j if $OS(C_i, C_j) \geq \beta$ (Figure 2(b)).



(a) (b)

Figure 2: (a) A local community C defined by a link (u, v). Here $\Psi(C) = 0.9 > \tau(C) = 0.725$ (b) Merging two local communities sharing a significant substructure (OS score $= 1.027 > \beta = 0.8$)

Algorithm 2 Combining local communities

Input: Raw community structure $\mathcal{C}_r$
Output: A refined community structure $\mathcal{C}_f$.

```
C_f ← C_r;
for C_i, C_j ∈ C_r and !Done do
    if OS(C_i, C_j) > β then
        C ← Combine C_i and C_j;
        /*Update the current structure*/
        C_f = (C_f \ {C_i ∪ C_j}) ∪ C;
        Done ← False;
    end if
end for
```

The time complexity of Alg. 2 is $O(N_0^2)$ where N_0 is the number of local communities. Clearly, $N_0 \leq M$ and thus, it can be $O(M^2)$. However, when the intersection of two communities is upper bounded, by Lemma 3 we know that the number of local communities is also upper bounded by $O(N)$, and thus, the time complexity of Alg. 2 is $O(N^2)$. In our experiments, we observe that the running time of this procedure is, indeed, much less than $O(N^2)$.

LEMMA 3. *The number of raw communities detected in Alg. 1 is $O(N)$ when the number of nodes in the intersection of any two communities is upper bounded by a constant α.*

5. DETECTING EVOLVING NETWORK COMMUNITIES

We describe $AFOCS$, the second phase and also the main focus of our detection framework. In particular, we use $AFOCS$ to adaptively update and trace the net- work communities, which were previously initialized by $FOCS$, as the dynamic network evolves over time. Note that $FOCS$ is executed only once on G_0, after that $AFOCS$ will take over and handle all changes introduced to the network.

Let us first discuss the various behaviors of the community structure when the network topology evolves over time. Suppose $G = (V, E)$ and $\mathcal{C} = \{C_1, C_2, .., C_n\}$ is the current network and its corresponding overlapping community structure, respectively. We use the term *intra links* to denote edges whose two endpoints belong to the same community, *inter links* to denote those with endpoints connecting different disjoint communities and the term *hybrid links* to stand for the others. For each community C of G, the number of connections joining C with the others are lesser than the number of connections within C itself by definition

Intuitively, the addition of intra links or removal of inter links between communities of G will strengthen them and consequently, will make the structure of G more clear. Similarly, removing intra links from or introducing inter links to a community of G will decrease its internal density and as a result, loosen its internal structure. However, when two communities have less distraction to each other, adding or removing links makes them more attractive to each other and therefore, leaves a possibility that they can overlap with each other or can be combined to form a new community. The updating process, as a result, is very complicated and challenging since any insignificant change in the network topology could possibly lead to an unpredictable transformation of the network community structure.

In order to reflect these changes to a complex network, its underlying graph model is frequently updated by either inserting or removing a node or a set of nodes, or an edge or a set of edges. Scrutiny into these events reveals that the introduction or removal of a set of nodes (or edges) can furthermore be decomposed as a collection of node (or edge) insertions (or removals), in which only a node (or only an edge) is inserted (or removed) at a time. Therefore, changes to the network at each time step can be viewed as a collection of simpler events whose details are as follow:

- *newNode* $(V + u)$: A new node u and its adjacent edge(s) are introduced
- *removeNode* $(V - u)$: A node u and its adjacent edge(s) are removed from the network.
- *newEdge* $(E + e)$: A new edge e connecting two existing nodes is introduced.

- *removeEdge* $(E - e)$: An edge e in the network is removed.

As we mentioned earlier, our adaptive framework initially requires a basic community structure $\mathcal{C}_0$. To obtain this basic structure, we apply *FOCS* algorithm at the first network snapshot and then let *AFOCS* adaptively handle this structure as the network evolves.

5.1 Handling a new node

Let us discuss the first case when a new node u and its associated links are introduced to the network. Possibilities are (1) u may come with no adjacent edge or (2) with many of them connecting one or more possibly overlapped communities. If u has no adjacent edge, we simply join u in the set of outliers and preserve the current community structure.

The interesting case happens, and it usually does, when u comes with multiple links connecting one ore more existing communities. Since network communities can overlap each other, we need to determine which ones u should join in in order to maximize the gained internal density. But how can we quickly and effectively do so? By Lemma 4, we give a necessary condition for a new node in order to join in an existing community, i.e. our algorithm will join node u in C if the number of connections u has to C suffices: $d_{ui} > \frac{2|C^{in}|}{|C|-1}$. However, failing to satisfy this condition does not necessarily imply that u should not belong to C, since it can potentially gather some substructure of C to form a new community (Figure 3). Thus, we also need to handle this possibility. Alg. 3 presents the algorithm.

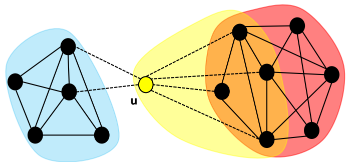


Figure 3: When a new node u is introduced, u could gather some nodes from an existing community (red) to form a new community (yellow)

LEMMA 4. *Suppose u is a newly introduced node with d_{ui} connections to each adjacent community C_i. u will join in C_i if $d_{ui} > \frac{2|C_i^{in}|}{|C_i|-1}$.*

The analysis of Alg. 3 is shown by Lemma 5. In particular, we show that this procedure achieves at least 0.83% internal density of the optimal assignment for u, given the prior community structure.

LEMMA 5. *Alg. 3 produces an assignment that, prior to the community combination, achieves $\Psi(\mathcal{C}_t) \geq \tau(4) \times \Psi(OPT(u)_t)$ where $OPT(u)_t$ is the optimal community assignment for u at time t, given the prior community structure $\mathcal{C}_{t-1}$.*

Algorithm 3 Handling a new node u

Input: The current community structure $\mathcal{C}_{t-1}$
Output: An updated structure $\mathcal{C}_t$.

$C_1, C_2, ..., C_k \leftarrow$ Adjacent communities of u;
for i = 1 **do** to k
    if $d_{ui} > \frac{2|C_i^{in}|}{|C_i|-1}$ **then**
        $C_i \leftarrow C_i \cup \{u\}$;
    else
        $C \leftarrow N(u) \cap C_i$;
        if $\Psi(C) \geq \tau(C)$ and $|C| \geq 4$ **then**
            $C_i \leftarrow C_i \cup \{u\}$;
        end if
    end if
end for
/*Checking new community from outliers*/
for $v \in C_0$ and $Com(v) \cap Com(u) = \emptyset$ **do**
    $C \equiv N(u) \cap N(v)$;
    if $\Psi(C) \geq \tau(C)$ and $|C| > 4$ **then**
        Define C a new community;
    end if
end for
Merging overlapping communities on $C_1, C_2, ..., C_k$ and C_0;
Update $\mathcal{C}_t$;

5.2 Handling a new edge

In case where a new edge $e = (u, v)$ connecting two existing vertices u and v is introduced, we divide it further into two four smaller cases: (1) e is solely inside a single community C (2) e is within the intersection of two (or more) communities (3) e is joining two separated communities and (4) e is crossing overlapped communities. If e is totally inside a community C, its presence will strengthen C's internal density and by Lemma 6, we know that adding e should not split the current community C into smaller substructures. The same reaction applies in the second subcase when e is within the intersection of two communities since their inner densities are both increased. Thus, in these first two cases, we leave the current network structure intact.

Handling the last two subcases is complicated since any of them can either have no effect on the current network structure or unpredictably form a new network community, and furthermore can overlap or merge with the others (Figure 4). By Lemma 7, we know that when a new edge (u, v) crossing between two disjoint communities is introduced, its two endpoints should not be moved. However, there is still a possibility that the introduction of this new link, together with some substructure of C_u or C_v, suffices to form a new community that can overlap with not only C_u and C_v but also with some of the others. The other subcases can be handled similarly. Alg. 4 describe this procedure.

LEMMA 6. *If an new edge (u, v) is introduced solely inside a community C, it should not split C into smaller substructures.*

LEMMA 7. *When a new edge (u, v) is introduced between two disjoint communities C_u and C_v, neither u nor v should be moved to C_v or C_u.*

5.3 Removing an existing node

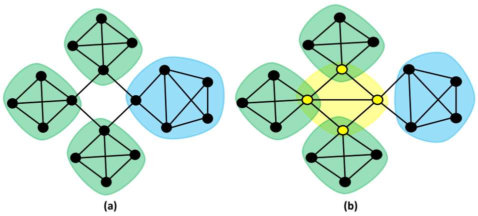


Figure 4: (a) The network with 4 disjoint communities (b) When the central edge is added, the central nodes form a new community (yellow)

Algorithm 4 Handling a new edge (u, v)

Input: The current community structure $\mathcal{C}_{t-1}$.
Output: An updated community structure $\mathcal{C}_t$.

```
if ((u, v) ∈ a single community OR (u, v) ∈ C_u ∩ C_v) then
    C_t ← C_{t-1};
else if Com(u) ∩ Com(v) = ∅ then
    C ← N(u) ∩ N(v);
    if Ψ(C) ≥ τ(C) then
        Define C a new community;
        Check for combining on Com(u), Com(v) and C;
    else
        for D ∈ Com(u) (or D' ∈ Com(v)) do
            if Ψ(D ∪ {v}) ≥ τ(D) (or Ψ(D' ∪ {u}) ≥ τ(D')) then
                D ← D ∪ {v} (or D' ← D' ∪ {u})
            end if
        end for
        Merging overlapping communities on D's (or D');
    end if
    Update C_t;
end if
```

When an existing node u is about to be removed from the network, all of its adjacent edges will also be removed as a consequence. If u is an outlier, we can simply exclude u and its corresponding links from the current structure and safely keep the network communities unchanged.

In unfortunate situations where u is not an outlier, the problem becomes very challenging in the sense that the resulting community is complicated: it can either be unchanged, or broken into smaller communities, or could probably be merged with the other communities. To give a sense of this effect, let's consider two examples illustrated in Figure 5. In the first example, when C is almost a full clique, the removal of any node will not break it apart. However, if we a remove node that tends to connect the others within a community, the leftover module is broken into a smaller one together with a node that will later be merged to one of its nearby communities. Therefore, identifying the leftover structure of C is a crucial task once a vertex u in C is removed.

To quickly handle this task, we first examine the internal density of C excluding the removed node u. If the number of internal connections still suffices, we can safely keep the current network communities intact. Otherwise, we apply Alg. 1 on the subgraph induced by $C\backslash\{u\}$ to quickly identify the leftover modules in C, and then let these modules hire a set of unassigned nodes $\Psi(C)$ that help them increasing their inner densities. Finally, we locally check for community combination, if any, by using an algorithm similar to Alg. 2.

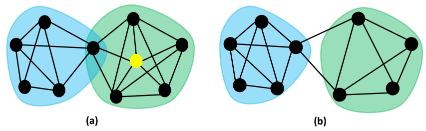


Figure 5: (a) Two overlapped communities (b) When the central node is removed, the new structure consists of two disjoint communities

Algorithm 5 Removing a node u

Input: The current community structure $\mathcal{C}_{t-1}$.
Output: An updated structure $\mathcal{C}_t$.

```
for C ∈ Com(u) and Ψ(C\{u}) < τ(C\{u}) do
    LC ← Local communities by Alg 1 on C\{u};
    for C_i ∈ LC and |C_i| ≥ 4 do
        S_i ← Nodes such that Ψ(C_i ∪ S_i) ≥ τ(C_i ∪ S_i);
        C_i ← C_i ∪ S_i;
    end for
    Merging overlapping communities on LC;
end for
Update C_t;
```

5.4 Removing an edge

In the last situation when an edge $e = (u, v)$ is about to be removed, we divide it further into four subcases similar to those of a new edge (1) e is between two disjoint communities (2) e is inside a sole community (3) e is within the intersection of two (or more) communities and finally (4) e is crossing overlapping communities.

In the first subcase, when e is crossing two disjoint communities, its removal will make the network structure more clear (since we now have less connections between groups), and thus, the current communities should be keep unchanged. When e is totally within a sole community C, handling its removal is complicated since this can lead to an unpredictable transformation of the host module: C could either be unchanged or broken into smaller modules if it contains substructures which are less attractive to each other, as depicted in Figure 6. Therefore, the problem of identify the structure of the remaining module becomes the central part for not only this case but also for the others.

To quickly handle these tasks, we first verify the inner density of the remaining module and, again utilize the local community location method (Alg. 1) to locally identify the leftover substructures. Next, we check for community combination since these structures can pos-

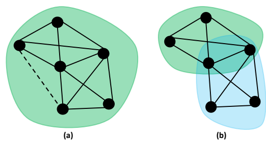


Figure 6: (a) The original community (b) When the dotted edge is removed, the community is broken into two overlapped communities

sibly overlap with existing network communities. The detailed procedure is described in Alg. 6.

Algorithm 6 Removing an edge (u, v)

Input: The current structure $\mathcal{C}_{t-1}$.
Output: An updated community structure $\mathcal{C}_t$.

if (u, v) is an isolated edge **then**
  $\mathcal{C}_t = (\mathcal{C}_{t-1} \backslash \{u, v\}) \cup \{u\} \cup \{v\}$;
else if $d_u = 1$ (or $d_v = 1$) **then**
  $\mathcal{C}_t = (\mathcal{C}_{t-1} \backslash C(u)) \cup \{u\} \cup C(v)$;
else if $C \equiv C(u) \cap C(v) = \emptyset$ **then**
  $\mathcal{C}_t = \mathcal{C}_{t-1}$;
else if $\Psi(C \backslash (u, v)) < \tau(C \backslash (u, v))$ **then** /*Here $C \neq \emptyset$*/
  $LC \leftarrow$ Local communities by Alg 1 on $C \backslash (u, v)$;
  Define each $L \in LC$ a local community of $\mathcal{C}_{t-1}$;
  Merging overlapping community on L's;
end if
Update $\mathcal{C}_t$;

5.5 Remarks

Note that the ultimate goal of our framework is to adaptively detect and update the community structure as the network evolves, i.e. to mainly deal with the dynamics of a mobile network. As a result, we mainly put our focus on $AFOCS$. Although $FOCS$, the first detection phase, appears to be a centralized algorithm, it is executed only once at the very first network snapshot whereas $AFOCS$ stays up and locally handles all changes as the network evolves over time. That said, we do not execute $FOCS$ again. Furthermore, $AFOCS$ can be run independently with $FOCS$, i.e. one can use any localized detection algorithm to identify a basic community structure at the first phase. Thus, $AFOCS$ can be easily apply to mobile network problems, as presented in sections 7 and 8.

6. EXPERIMENTAL RESULTS

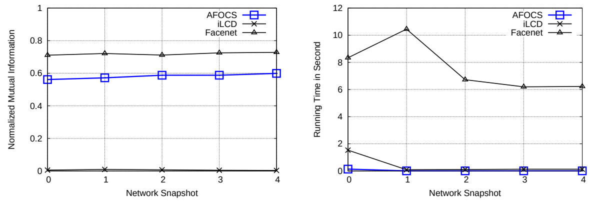
We first compare the performance of $AFOCS$ to two *static* detection methods: $CFinder$ - the most popular method [8] and $COPRA$ - the most effective method [10]. Both of them can detect overlapping communities in *static* networks and hence, are excellent resources for comparison purposes; however, they are not designed to work on dynamic mobile networks.

Data Sets: We use networks generated by the well-known LFR overlapping benchmark [14], the 'de facto' standard for evaluating overlapping community detection algorithms. Generated networks follow power-law degree distributions and contain embedded overlapping communities of varying sizes that capture the internal characteristics of real-world networks.

Metrics: We evaluate following metrics.

(1) The generalized *Normalized Mutual Information (NMI)* [9] specially built for overlapping communities. NMI is an standardized measure since $NMI(U, V) = 1$ if structures U and V are identical and 0 if they are totally separated.

(2) The *number of communities*, ignoring singleton communities and unassigned nodes. A good community detection method should produce roughly the same number of communities with the known ground truth.



(a) (b)

Figure 7: Comparison among $AFOCS$, $FacetNet$ and $iLCD$ (a) NMI scores (b) Running time

Set up: To fairly compare with $COPRA$ and to avoid being biased, we keep the parameters close to [10]: the minimum - maximum community size is $c_{min} = 10$ and $c_{max} = 50$, each vertex belongs to at most two communities, $o_m = 2$. $N = 1000$ or $N = 5000$ and the mixing rate is $\mu = 0.1$ and $\mu = 0.3$. The *overlapping fraction*, which determines the fraction of overlapped nodes of the generated network, is from 0 to 0.5. Since the output of $COPRA$ is nondeterministic, we run it 10 times on each instance and select the best result. We put no time constraint on $CFinder$.

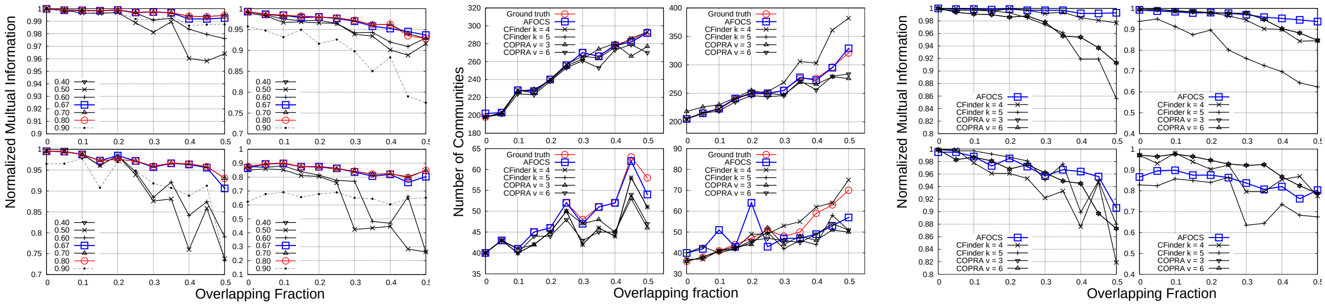
6.1 Choosing the Overlapping Threshold

The overlapping threshold β is the unique input parameter required by our framework and thus, determining its appropriate value plays an important role in assessing $AFOCS$'s performance on both synthesized and real-world datasets. To best determine this threshold, we run $AFOCS$ on all generated networks with different values of β and record the similarities between embedded and detected communities via NMI scores in Figure 8(a). Of course, the higher NMI scores imply the better β values. As depicted in this figure, the best values for β are ranging from 0.67 to 0.80, among which $\beta = 0.70$ yields the best community similarity in all of the generated networks. Therefore, we fix the overlapping threshold in $AFOCS$ to be 0.70 hereafter.

6.2 Overlapping Communities Quality

We show our results in groups of four. For each case we vary the overlapping fraction γ from 0 to 0.5 and analyze the results found by $AFOCS$, $CFinder$, and $COPRA$. We only present results when corresponding parameters give top performance for $CFinder$ (clique size $k = 4, 5$) and $COPRA$ (max. communities per vertex $v = 3, 6$).

Number of communities: Figure 8(b) shows the number of communities found by $AFOCS$, $COPRA$ and $CFinder$ and the ground truth. It reveals from this figure that the numbers of communities found by $AFOCS$, marked with squares, are the closest and almost identical to the ground truth as the overlapping fraction gets higher. There is an exception when $N = 1000$ and $\mu = 0.3$ which we will discuss later.



(a) (b) (c)

Figure 8: Top: $N = 5000$, Bottom: $N = 1000$, Left: $\mu = 0.1$, Right: $\mu = 0.3$. (a) NMI scores for different values of β (b) Number of communities found by $AFOCS$, $COPRA$ and $CFinder$. The closer to the ground truth, the better. (c) NMI of $AFOCS$, $COPRA$, $CFinder$. The higher NMI, the better

Normalized Mutual Information: As one can infer from Figure 8(c), $AFOCS$ achieves the highest performance among all methods with much more stable. A common trend in this test is the performances of all methods degrade (1) when the mixing rate μ increases, i.e. when the community structure becomes more ambiguous or (2) when the size of network decreases while the mixing rate μ stays the same. Even though $AFOCS$ is not very competitive only when both negative factors happen in the bottom-right char as $N = 1000$ and $\mu = 0.3$, it is in general the best performer.

The significant gap is observed when the mixing rate gets higher ($\mu = 0.3$) and the network size gets smaller ($N = 1000$). $AFOCS$ provides less numbers of communities than those of the ground truth but with much higher overlapping rates. The reason is with a larger mixing rate μ, a node will have more edges connecting vertices in other communities, thus increases the chance that $AFOCS$ will merge highly overlapped communities. Hence, $AFOCS$ creates less but with larger size communities. We note that this 'weakness' of AFOCS is controversial as when the mixing rate increases, the ground truth does not necessarily coincide with the structure implied by the network's topology.

Extensive experiments show the ability of $AFOCS$ in identifing high quality overlapping communities. In addition, we found $AFOCS$ runs substantially faster than the other competitors: on the network containing 63K nodes, $AFOCS$ is 150x faster than $COPRA$ while $CFinder$ is unable to finish.

6.3 Reference to other dynamic methods

We next observe the performance of $AFOCS$ in reference to two dynamic methods $FacetNet$ [11] and $iLCD$ [12]. Since the ground-truth communities are known on synthesized datasets, fair comparisons among three methods can be obtained via their NMI scores and running times. Of course, the higher its NMI scores with less time consuming, the better the method seems to be. Each synthesized dynamic network is simulated via 5 snapshots, in which the basic communities are formed by using 90% of the network data with approximately 200 edges added to each growing snapshot at a time.

The NMI and time results are presented in Figure 7(a) and 7(b). It reveals from these figures that the NMI scores of $AFOCS$ are very competitive to those of $FacetNet$ and are far better than those of $iLCD$. In particular, the NMI scores of $AFOCS$ are about just 5-7% lag behind that of $FacetNet$ while the running times are significantly lower: $AFOCS$ requires barely a second to finish updating each network snapshot whereas $FacetNet$ asks for more than 5 seconds (5x more time consuming). $iLCD$ performs fast in these generated datasets; however, the similarity of the detected communities and the ground-truth is surprisingly poor, as revealed from the results. Thus, we strongly believe that $AFOCS$ achieves competitive community detection results in a timely manner.

7. COMMUNITY-BASED FORWARDING IN COMMUNICATION NETWORKS

We present a practical application where the detection of overlapping network communities plays a vital role in forwarding strategies in communication networks. With the helpful knowledge of the network community structure, we propose a new community-based forwarding algorithm that significantly reduces the number of duplicate messages while maintaining competitive delivery times and ratios, which are essential factors of a forwarding strategy.

Many routing methods based on the discovery of network community structure have been proposed in the literature [24, 25, 26]. However, the community detection cores in those strategies encounter (1) the lack of knowledge about overlapping communities and (2) the repeated identification of communities as the network evolves. The second issue is computationally costly and time consuming, thus may dramatically reduce the performance of those forwarding strategies.

Let us first discuss how our new forwarding algorithm works in practice and then how $AFOCS$ helps it to overcome the above limitations. We use $AFOCS$ to detect overlapping communities and keep it up-to-date as the network changes. Each node in a community is assigned the same label and each overlapped node u has a set of corresponding labels $Com(u)$. During the network operation, if a devices u carrying the message meets another device v who indeed shares more common community labels with the destination than u, i.e. $|Com(v) \cap Com(dest)| > |Com(u) \cap Com(dest)|$, then u will forward the message to v. The same actions then apply to v as well as to devices that v meets.

The intuition behinds this strategy is that if v shares more communities with the destination nodes, it is likely that v will have more chances to deliver the message to the destination. By doing this, we not only have higher chances to correctly forward the messages but also generate much less duplicate messages. Due to its adaptive nature and the ability of identifying overlapping communities, $AFOCS$ helps our algorithm to overcome the above shortcomings naturally. This explains why our forwarding algorithm can significantly reduce the number of duplicate messages while maintaining very competitive delivery times and ratios.

We compare six forwarding strategies (1) $MIEN$: A recently proposed social-aware routing strategy on MANETs [1] (2) $LABEL$: A node will forward the messages to another node if it is in the same community as the destination [27] (3) $WAIT$: The source node waits and keeps forwarding the message until it meets the destination (4) MCP: A node keeps forwarding the messages until they reach the maximum number of hops (5) QCA: A $LABEL$ version utilizing QCA [4] as the adaptive disjoint community detection method and lastly (6) $AFOCS$: Our newly proposed forwarding algorithm equipped with $AFOCS$ as an community detection and update core.

Results of $WAIT$ and MCP algorithms provide us the lower and upper bounds of important factors: message delivery ratio, time redundancy and message redundancy. Our experiments are performed on the Reality Mining dataset provided by the MIT Media Lab [28]. This dataset contains communication, proximity, location, and activity information from 100 students at MIT over the course of the 2004-2005 academic year. In particular, we take into account the Bluetooth information to construct the underlying communication network and evaluate the performance of the above six routing strategies.

In each experiment, 500 message sending requests are randomly generated and distributed in different time points. To control the forwarding process, we use *hop-limit*, *time-to-live*, and *max-copies* parameters. A message cannot be forwarded more than *hop-limit* hops in the network or exist in the process longer than *time-to-live*, otherwise it will be automatically discarded. Moreover, the maximum number of same messages a device can forward to the others is restricted by *max-copies*. Experiments results are repeated and results are averaged for consistency.

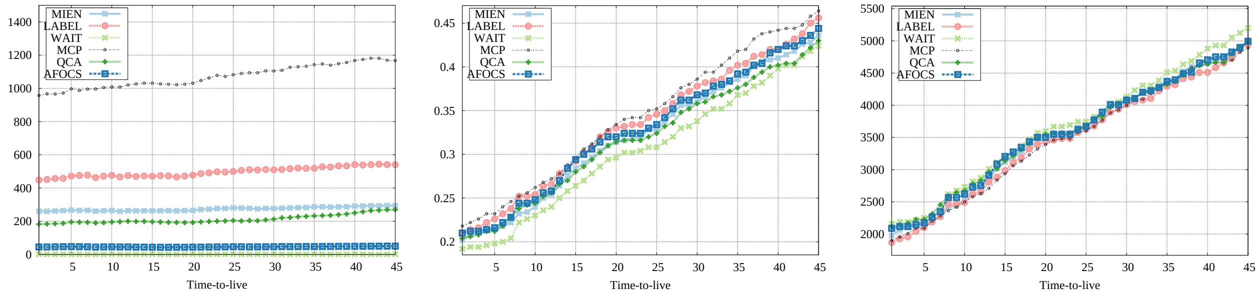
Our results are presented in Figures 9(a), 9(b), 9(c). The first observation reveals that our proposed forwarding algorithm achieves the lowest number of duplicate messages as depicted in Figure 9(a), and even far better than the second best method QCA. On average, only 46.5 duplicate messages are generated by $AFOCS$ during evaluation process in contrast with 212.2 of QCA, 274.2 of $MIEN$, 496.4 of $LABEL$ and the huge 1071.0 overhead messages of MCP. Thus, on the number of duplicate messages, $AFOCS$ strikingly achieves improvement factors of 4.5x, 5x, 11x and 23x over these mentioned strategies, respectively. These extremely low overhead strongly imply the efficiency of $AFOCS$ in communication networks.

Figures 9(b) and 9(c) present our results on the other two important factors, the message delivery ratios and delivery times. These figures supportively indicate that $AFOCS$ achieves competitive results on both of these vital factors. In general, $AFOCS$ is the second best strategy with almost no noticeable different between itself and the leader method $LABEL$. On average, $AFOCS$ gets 33% of the total messages delivered in 3569.2s and only a little bit lags over MCP (34% in 3465.3s) and $LABEL$ (slightly over 33% in 3462.7s), and is far better than $MIEN$ (32% in 3537.6s) and QCA (32% in 3572.2s). This can be explained by the advantages of knowing the overlapping community structure: the disjoint network communities in QCA and $MIEN$ can possibly have messages forwarded to the wrong communities when the destination changes its membership. With the ability of quickly updating the network structure, $AFOCS$ can efficiently cope with this change and thus, can still provide the most updated forwarding information.

In summary, $AFOCS$ helps our forwarding strategy to reduce up to 11x the number of duplicate messages while keeping good average delivery ratio and time. These experimental results are highly competitive and supportively confirm the effectiveness of $AFOCS$ and our new routing algorithm on communication networks.

8. CONTAINING WORMS USING OVERLAPPING COMMUNITIES

We show another application of $AFOCS$ in worm containment problem on OSNs. OSNs are good places for people to socialize online or to stay in touch with friends and colleagues. However, when some of the users are infected with malicious softwares, such as viruses, worms or false information, OSNs are also fertile grounds for their rapid propagations. Since mobile devices are able to access online social applications nowadays, worms and viruses now can target computers [4] and probably



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

Figure 9: Experimental results on the Reality Mining data set

mobile devices [3].

Recently, community structure-based methods have been proven to be effective solutions to prevent worms from spreading out wider on not only social networks [4, 7] but also cellular networks [3]. Due to the high and low frequencies of interactions inside and between communities, worms spread out quicker within a community than between communities. Therefore, an appropriate reaction should first contain worms into only infected communities, and then prevent them from getting outside. This strategy can be accomplished by patching the most *influential members* who are well-connected not only to members of their community but also to people in other communities.

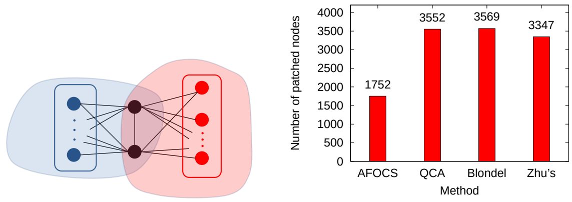
In our experiments, we use Facebook network dataset collected in [29]. This data set contains friendship information and wall posts among New Orleans regional network, spanning from Sep 2006 to Jan 2009. The data set contains more than 63.7K nodes (users) connected by more than 1.5 million friendship links with an average node degree of 23.5. We keep other parameters as well as the "Koobface" worm propagation model the same as [7] for comparison convenience. With the advantages of knowledge overlapping communities, we are able to develop a better and more efficient patching scheme. In particular, we enhance the patching scheme in [7] to take the advantage of the overlaps: nodes in the boundary of overlapped regions are selected for patching (Figure 11(a)). Alg 7 details the adjusted scheme.

Algorithm 7 *OverCom* Patching Scheme

Input: $G = (V, E)$ and $\mathcal{C} = \{C_1, C_2, ..., C_k\}$ detected by $AFOCS$
Output: A set of patched nodes IS.

```
IS = ∅;
for C_i, C_j ∈ C do
    if C_i ∩ C_j ≠ ∅ then
        IS = IS ∪ N(u)    ∀u ∈ C_i ∩ C_j;
    end if
end for
for u ∈ IS do
    Send patches to u;
    Let u redistribute patches to w ∈ IS\N(u);
end for
```

We compare the *OverCom* patching scheme and overlapping communities found by *AFOCS* to those using

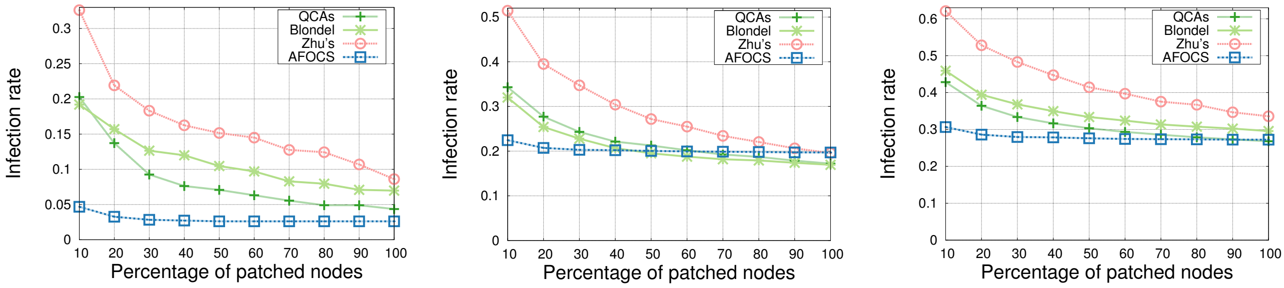


(a) (b)

Figure 11: ***OverCom*** **influential users selection and Number of patched nodes in four methods**

disjoint communities proposed by Blondel *et al* [30], QCA by Nguyen *et al* [4] and Clustering based method suggested by Zhu *et al* [3]. The number of patched nodes is shown in Figure 11(b). Both the number of patched nodes and the infection rates decline remarkably. In particular, the number of nodes to send patch in $AFOCS$ is substantially smaller by half of those required by *Blondel*, QCA as well as $Zhu's$ methods: only 1725 nodes over 63K nodes in the networks are needed to be patched by *OverCom* patching scheme, while the other schemes require nearly twice (≥3,300 nodes). The reason behind this improvement is due to the nature of our $AFOCS$ framework, the neighbors of the overlapped nodes should not be to far away from the center of each community, thus they can easily redistribute the patches once received.

We next present the achieved infection rates with alarming thresholds (the fraction of infected nodes over all nodes) $\alpha = 2\%$, 10% and 20%, respectively. This threshold alarms the distribution process as soon as the infected rate goes beyond α. The results are reported in Figures 10(a), 10(b), 10(c), respectively. In general, the higher α (i.e. the longer we wait), the more nodes we have to send patches and the higher infection rate. OverCom with AFOCS achieves the lowest infection rates in almost all the experiments and just a little bit lag behind when $\alpha = 10\%$. In particular, when $\alpha = 2\%$, $AFOCS$ helps *OverCom* to remarkably reduce from 1.6x up to 4.3x the infection rates of QCA, from 2.6x up to 4x the infection rates of *Blondel* and 3.2x to 7x those of $Zhu's$ method. When $\alpha = 10\%$, $AFOCS + OverCom$ achieves average improved rates of 9% over QCA, 5% over *Blondel* and 43% over $Zhu's$



(a) $\alpha = 2\%$ (b) $\alpha = 10\%$ (c) $\alpha = 20\%$

Figure 10: Infection rates between four methods

methods. As $\alpha = 20\%$, the average improvements are 12%, 23% and 53%, respectively. Due to the nature of the event handling processes, the neighbors of overlapped nodes are not located far away from the rest of their communities. As a result, they can help to distribute patches to more users in the communities, hence the lower infection rates of *AFOCS*. These improvement factors, again, confirm the strength and applicability of our adaptive community detection framework to mobile networks.

9. CONCLUSION

In this paper, we presented *AFOCS*, a two-phase framework for detecting network overlapping communities as well as tracing their evolution in dynamic mobile networks. Analyses show that *AFOCS* partially achieves no less than 83% internal density of the optimal community assignment. Experiments on synthesis and real-world data traces show good results. We show two mobile applications, namely forwarding and routing in MANETs and worm containment on OSNs, in which *AFOCS* significantly helps to increase the performances up to 11x and 7x, respectively. These results confirm the effectiveness of *AFOCS* as well as its applicability in mobile applications.

Acknowledgment

We thank Dr. Cecilia Mascolo for being our shepherd. We thank anonymous reviewers for valuable review. This work is partially supported by the NSF 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.

10. REFERENCES

[1] T. N. Dinh, Y. Xuan, and M. T. Thai. Towards social-aware routing in dynamic communication networks. *IPCCC*, '09.
[2] B. Pasztor, L. Mottola, C. Mascolo, G. Picco, S. Ellwood, and D. Macdonald. Selective reprogramming of mobile sensor networks through social community detection. *Wireless Sensor Networks*, '10.
[3] Z. Zou, G. Cao, S. Zhu, S. Ranjan, and A. Nucci. A social network based patching scheme for worm containment in cellular networks. In *INFOCOM*, '09.
[4] N. P. Nguyen, T. N. Dinh, Y. Xuan, and M. T. Thai. Adaptive algorithms for detecting community structure in dynamic social networks. *INFOCOM*, '11.
[5] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, '02.
[6] M. A. Porter, J-P. Onnela, and P. J. Mucha. Communities in networks. *Notices of the AMS*, '09.
[7] N. P. Nguyen, Y. Xuan, and M. T. Thai. A novel method for worm containment on dynamic social netowrks. *MILCOM*, '10.
[8] G. Palla, I. Derenyi, I. Farkas1, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, '05.
[9] A. Lancichinetti, S. Fortunato, and K. Jnos. Detecting the overlapping and hierarchical community structure in complex networks. *New J. of Phys.*, '09.
[10] Steve Gregory. Finding overlapping communities in networks by label propagation. *New J. of Physics*, '10.
[11] Y-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng. Anlyzing communities and their evolutions in dynamic social networks. *ACM TKDD*, '09.
[12] R. Cazabet, F. Amblard, and C. Hanachi. Detection of overlapping communities in dynamical social networks. In *SocialCom*, '10.
[13] L. Peel. Estimating network parameters for selecting community detection algorithms. *FUSION*, '10.
[14] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Phys. rev. E. 80*, '09.
[15] D. Duan, Y. Li, Y. Jin, and Z. Lu. Community mining on dynamic weighted directed graphs. In *CNIKM*, '09.
[16] M-S. Kim and J. Han. A particle-and-density based evolutionary clustering method for dynamic networks. *VLDB Endow.*, 2009.
[17] S. Fortunato and C. Castellano. Community structure in graphs. *arXiv*, '07.
[18] A. Lzr, D. bel, and T. Vicsek. Modularity measure of networks with overlapping communities. *Euro. Let.*, '10.
[19] F. Radicchi, C. Castellano, F .Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proc. Natl. Acad. Sci. USA*, '04.
[20] J. P. Bagrow Y-Y Ahn and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, '10.
[21] S. Fortunato. Community detection in graphs. *Phys. Reports*, '10.
[22] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW08*, '08.
[23] C. Lee, F. Reid, A. McDaid, and N. Hurley. Detecting highly overlapping community structure by greedy clique expansion. In *KDD*, '10.
[24] P. Hui, E. Yoneki, S. Y. Chan, and J. Crowcroft. Distributed community detection in delay tolerant networks. In *MobiArch*, '07.
[25] E. M. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *MobiHoc '07*, '07.
[26] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *MobiHoc '08*, '08.
[27] P. Hui and J. Crowcroft. How small labels create big improvements. *PERCOMW*, '07.
[28] E. Nathan and A. Pentland. Reality mining: sensing complex social systems. *Personal Ubiquitous Comput.*, '06.
[29] 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*, '09.
[30] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.: Theory and Experiment*, '08.