Context-Aware Community: Integrating Contexts with Contacts for Proximity-Based Mobile Social Networking

Na Yu and Qi Han
Department of Electrical Engineering and Computer Science, Colorado School of Mines, Golden, CO 80401
{nyu, qhan}@mines.edu

***Abstract*—Sensor-equipped mobile devices have allowed users to participate in various social networking services while they are on the go. We focus on proximity-based mobile social networking environments where users share information obtained from different places via their mobile devices when they are in proximity. Since people are more likely to share information if they can benefit from the sharing or if they think the information is of interest to others, there might exist community structures where users who share information more often are grouped together. Communities in proximity-based mobile social networks represent social groups where connections are built when people are in proximity. We consider information influence (i.e., specify who shares information with whom) as the connection, and the space and time related to the shared information as the social contexts. To model the potential information influences, we construct an influence graph by integrating the social contexts into the contacts of mobile users. Further, we propose a two-phase strategy to detect and track context-aware communities based on the influence graph and show how the context-aware community structure improves the performance of two types of mobile social applications.**

I. Introduction

Identifying communities is an important research topic in traditional as well as online social networks [1], [2]. A community is a densely connected group of nodes/users such that connections between communities are sparse. With the increasing popularity of mobile devices such as smartphones, community structures are now affected by people's mobility. For instance, people may collect information from the places they visit and share that information with other people they encounter in their proximity (i.e., within their mobile devices' WiFi or Bluetooth transmission ranges) as they move around. Identifying communities in such proximity-based mobile social networks could often lead to more cost-effective information dissemination by sharing the information only with those who might be interested, and more targeted queries by only asking those people who may have the information. Reducing the communication overhead using the communities will ultimately reduce energy consumption of mobile devices. In this paper, we study the community structure as well as its applications in proximity-based mobile social networks.

Existing work, as detailed in section II, clusters mobile users into communities merely based on contacts that occur when users are in proximity. These pure contact-based communities only show that people inside the same communities are in contact more often or have longer contact durations. They are incognizant of where and when the contacts occur, or whether or not there is information to be shared via the contacts, hence losing critical information in proximity-based mobile social networks: the "social contexts" associated with these contacts, i.e., the space (e.g., library) and time related to information sharing. In other words, two users in contact more often or longer only imply they are within each other's communication range more, but this does not necessarily imply they can have more information to share. Therefore, to take one step further towards better support of information sharing in proximity-based social applications, a better community structure shall integrate contacts with social contexts of the contacts.

In this paper, we address the drawbacks of existing work by building connections based on potential information influence (i.e., the two end users may have information to share with each other) when people are in proximity to construct context-aware community structure. This information influence is tied to space and time contexts in terms of information sources. Let us look at a campus scenario. Alice visited the student center where she learned free movie tickets were being given away, while Bob visited the recreation center where he learned a game was going on. A student may share the information within a certain time duration after obtaining it, say 5 hours. Assume that within 5 hours of the visit to the student center, Alice encounters Chris and then Bob. While within 5 hours of the visit to the recreation center, Bob encounters Dave and then Alice. So, Alice could tell Chris and Bob about the student center event, while Bob could tell Dave and Alice about the recreation center event. A context-aware community for the student center consists of Alice, Bob, and Chris; a context-aware community for the recreation center consists of Alice, Bob, and Dave. However, assume that outside of the 5 hours window, Emma frequently encounters Alice, Bob, Chris, and Dave. If only considering contacts, then a pure contact-based community consists of all these five students.

Context-aware communities are more informed and can be utilized to reduce the cost of information sharing. For instance, when Emma wants information about the student center, with context-aware communities, her queries can be sent only to the student center related community consisting of three members; but with pure contact-based communities, her queries have to be sent to the community consisting of five members. The communication cost as well as the energy cost for getting the information is reduced by using context-aware communities.

Our preliminary work [3] has started exploring the integration of contexts with contacts, but it did not keep track of the

community structure or study its applications. In this paper, beyond introducing the novel influence graph which integrates contexts with contacts, we propose a two-phase strategy to detect basic context-aware communities and track the evolving context-aware communities over time, and we further demonstrate its effectiveness and efficiency in supporting two types of mobile social applications—event sharing and event query.

II. Related Work

Community structures are well studied in traditional social networks. For instance, Facetnet [4] studies the detection and evolution of communities in a uniform process, where the community structure at a given time step is determined by both the observed network data and the prior distribution given by historic community structures. [5] describes a model for evolutionary communities with life-cycles characterized by a series of significant events and proposes a community tracking method involving matching communities found at consecutive time steps in the individual snapshot graphs. [6] studies the problem of evolutionary multi-typed communities in heterogeneous networks, where the Evo-NetClus model is proposed to generate net-clusters at each time window.

Although community structures play an important role in social network analysis [7], research on communities in mobile social networks is still largely incomplete. One of the most recent work [8] proposes a two-phase framework for detection and evolution of overlapping communities in dynamic mobile networks, but it assumes an undirected unweighted graph and generates networks using LFR overlapping benchmark [9] for evaluation. It does not address how to build the graph for constructing communities in a mobile social network. Other recent work shows that communities in mobile social networks have been formed based on global contact graphs [10] [11], or distributed local encounters [12] [13], or simply user proximity [14] [15]. None of the existing work includes any social context in their community construction.

In short, integrating contexts with contacts to build communities for proximity-based mobile social networking is what distinguishes our work from existing work.

III. Problem Formulation

In order to represent the space and time contexts, we adopt the concept Point of Interests (PoIs). PoIs are popular places with frequent user stays, and they can be identified using various localization technologies. We assume that users who are inside a PoI can obtain events of the PoI and store the events on their mobile devices. Each user has a unique user id, a mobility profile with a series of contacts and PoI visits, a set of PoI events with the time when the user obtained them, and a user influence lifetime T_l to specify the time duration the user is willing to share the event. For instance, Bob in the recreation center hears about a game event, and he is willing to use his mobile device to store the event for 5 hours in order to share with future encountered students. In this case, the user influence lifetime for Bob is $T_l = 5$ hours. In general, mobile users visit the PoIs at different times and can be influenced by the PoI events during their stays in the PoIs. Later, they can further influence the PoI events to encounter users outside a corresponding PoI anytime before the expiration of their influence lifetimes. Once the event information reaches a user outside its PoI, it may be further forwarded to other users. Therefore, the information influence is continued in a ripple carry fashion.

The problem we are studying in this work is to form, maintain and utilize the social structure which represents the potential information influence among mobile users. This problem is broken down into three issues: (1) constructing basic context-aware communities based on user mobility profiles; (2) dynamically updating the context-aware communities to reflect the evolving network; and (3) using the dynamic communities to support mobile social applications.

The basic context-aware community structure is constructed and updated on a centralized server based on the user mobility profiles collected, and it is then distributed to all users for future usage in supporting applications.

IV. Basic Context-Aware Community Structure

The first phase is constructing basic context-aware communities based on influence graphs, directed weighted graphs that embed all the contacts as well as the associated contexts.

A. Influence Graph

An influence graph is represented by $G = (V, E, P)$, where V is a set of users, E is a set of directed weighted edges between two users, and P is a set of PoIs. There is a directed edge from vertex i to vertex j ($e_{i,j}$) in the influence graph if user i can potentially influence user j (i.e., if user i has visited some PoIs or has been influenced by other users with PoI events, and then encounters user j before the expiration of i's influence lifetime). This definition of an edge is different from purely using contacts, where an edge exists between two users if there is a contact between them. In the influence graph, contact occurrence between two users does not necessarily imply an edge between them if none of the users have potential of information influence. For instance, if user i did not visit any PoI or was not influenced by other users with PoI events, or user i's influence lifetime has expired before encountering user j, then there is no edge from user i to user j. The weight of $e_{i,j}$ is a tuple: $w_{i,j} = (I_{P_1}, I_{P_2}, \cdots, I_{P_p}, \cdots, I_{P_N})$, where N is the total number of PoIs in the network, P_p is PoI p, and I_{P_p} is the number of potential influences regarding PoI p within user i's influence lifetime cumulated for all contacts. The potential influences when a contact occurs is determined as follows: the potential influence for PoI p from user i to user j exists only when the contact location is outside PoI p, and it is calculated as the sum of the number of times user i has visited p (i.e., user i can directly obtain PoI events) and the number of times user i has been in contact with other users who have potential influences regarding p (i.e., user i can be influenced by other users), within user i's influence lifetime.

User mobility profiles: In order to construct the influence graph, we need to generate user mobility profiles first. A

typical mobility profile of a user includes when the user is inside or outside the PoI and when the user is in contact with other users. Fig. 1 shows sample mobility profiles for Alice and Bob, where solid boxes tagged with a PoI (Student Center, Recreation Center, or Library) show the duration that Bob or Alice is inside the PoI, and the connections with other users show the contacts with those users at that time.

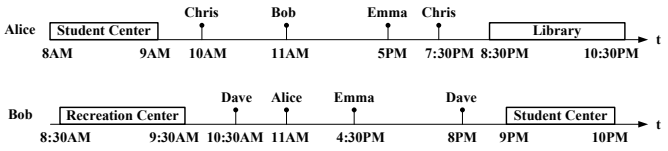


Fig. 1. Sample individual user mobility profiles for Alice and Bob

Constructing the influence graph: (1) Do a merge sort on the mobility profiles based on time. Fig. 2 shows the integrated mobility profile of Fig. 1, where 'a', 'b', 'c', 'd', 'e' represent Alice, Bob, Chris, Dave, and Emma, respectively, and 'l', 'r', 's' represent the library, the recreation center, and the student center, respectively. At each moment, the user action is represented by "U_iP_pS" meaning user i enters PoI p, "U_iP_pE" meaning user i exits PoI p, or "$C_{i,j}$" meaning user i and user j are in contact. (2) Scan the integrated user mobility profile to find all the bidirectional potential influences between any pair of users in contact for each PoI. Note that, the number of potential influences includes two parts: i) the number of PoI visits when the user can directly obtain events from the PoI, and ii) the number of contacts with other users who have potential influences regarding the PoI, so the user can also be influenced via contacts. Both i) and ii) should happen within the user's influence lifetime. For the number of PoI visits, we count the number of times user i exits PoI p (i.e., the number of moments with user action "U_iP_pE") in the integrated mobility profile. However, if the contact occurs inside PoI p (i.e., the users have not yet exited PoI p), then the users are considered self-influenced with events regarding PoI p because these users can directly get the information from PoI p, not from another user. In this case, the influence regarding PoI p is not counted towards the weight. (3) Put all the users as the vertices and put the count of potential influences for each PoI from one user to another as the weight of the directed edge on the influence graph. Fig. 3 shows the sample influence graph based on Fig. 2, assuming both Alice and Bob have the influence lifetime $T_l = 5$ hours. The values in the weight tuples represent the number of potential influences for the library, the recreation center, and the student center, respectively.

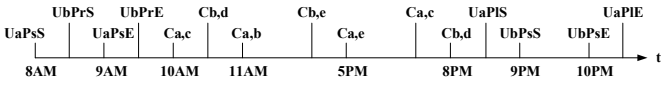


Fig. 2. Sample integrated mobility profile of Alice and Bob

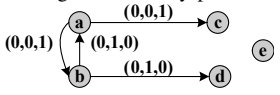


Fig. 3. Sample influence graph

B. Context-Aware Communities

Once we have constructed the influence graph, we can form communities. There are various ways to discover communities, most of which are based on either the centralized modularity optimization approach [16] or the local clique percolation approach [17]. We choose to apply the CPMd algorithm [18], which is the directed graph version of the well known Clique Percolation Method (CPM) [17]. The advantages of using a clique percolation based method are i) it is local so it does not have resolution limit (i.e., it can detect small communities) as centralized modularity optimization approaches have, and ii) the definition of modules based on k-cliques is actually based on link density, so connections are more concentrated inside communities, and iii) it allows overlaps between communities.

CPMd uses directed k-cliques (complete subgraphs of size k) to discover modules with directed links. It also uses a link weight threshold w^* to indicate the strength of directed links. We use *strong links/edges* to denote those links whose weight is no smaller than w^*. The link fraction f^* is the ratio of the number of strong links over the total number of links. For each selected value of k, CPMd adjusts w^* to the point where the largest community becomes twice as big as the second largest one. In addition, f^* should be no less than 0.5.

To make the communities context-aware, we find communities regarding each PoI by running the CPMd algorithm on the PoI-specific subgraph of the influence graph. The subgraph only contains strong links related to the PoI. In this way, the context-aware community structure is constructed with PoI related communities, represented as $C_c = (PoI_1 : (C_1, C_2, \cdots), PoI_2 : (C_1, C_2, \cdots), \cdots)$. Its significance is that users who have more potential influences with PoI-specific information are grouped together, so users can determine their communities based on the contexts of information they have. In addition, communities can overlap, so a user may belong to multiple communities for different PoIs or the same PoI.

Since the link weights depend on influence lifetimes and mobility profiles, the influence lifetime is the most important factor in forming community structures. For instance, using the influence graph in Fig. 3, and by setting $k = 2$ and $w^* = 1$, we can construct the context-aware community structure as $C_c = (P_r : (\{a, b, d\}), P_s : (\{a, b, c\}))$. However, if Bob changes his influence lifetime to 8 hours, then the community regarding the student center will be constructed with four members (i.e., Emma is added in addition to Alice, Bob, and Chris).

V. Evolving Context-Aware Community Structure

A good community evolutionary model should have automatically learned communities in each time step, while the communities in adjacent timestamps should be consistent [6]. In the second phase, we present a simple but effective method for efficiently updating the community structure.

As discussed about CPMd in the previous section, we choose the optimal k and w^* values in a way that the link fraction f^* of the influence graph is no less than 0.5, and the largest community is no larger than twice as big as the second largest one among all communities regarding all PoIs. Then, the evolving influence graph is updated with actual information influences among users. To keep track of the evolving communities based on the evolving influence graph,

we use the same k value as used in the initial formation of communities, but we update w^* so that f^* only deviates from the initial value within a certain error range E_f. All these are to make sure there are enough strong links in the entire influence graph and the constructed communities are non-trivial.

Since only strong links are considered for constructing communities, addition or removal of a node itself will not cause instability of existing communities, unless that leads to significant changes in link weights (i.e, causing a new strong link due to link additions, or a strong link to become weak due to increasing of w^*). More specifically, handling a new strong link might involve handling one or two new nodes, and the new nodes might become part of some communities. Similarly, removal of a strong link might lead to removal of the relevant nodes from their communities. Therefore, we do not need to explicitly deal with any node addition or removal, but only need to cover them when dealing with addition or removal of strong links. We also note that addition and removal of strong links are caused by not only changes in link weights, but also change of w^*. The overall process of community updates (regarding the PoI involved in each link update) with x link updates is shown in Algorithm 1.

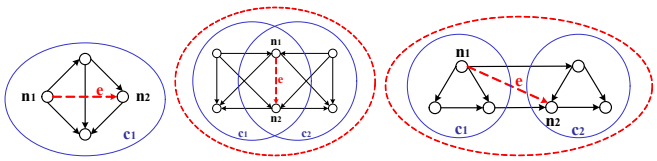
Algorithm 1 : Overall Process of Community Updates

Input: community structure, x, w^*, f^0, E_f
Output: an updated community structure and w^*

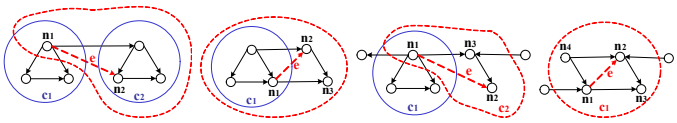
```
for each of the x link updates do
    Update link weight;
    if an addition of strong link then
        Check for communities merge or creation;
    end if
end for
Calculate the new link fraction f*;
if f* − f^0 > E_f then
    Increase w* so f* − f^0 ≤ E_f and f* ≥ 0.5;
    for each strong link removal do
        Check for communities split or destroy;
    end for
else if f* < 0.5 then
    Decrease w* so that f* − f^0 ≤ E_f and f* ≥ 0.5;
    for each addition of strong link do
        Check for communities merge or creation;
    end for
end if
```

A. Adding a Strong Link

Assume there is no directed link from node n_1 to n_2, or e from n_1 to n_2 is a weak link. When the link weight grows to w^* or above, e becomes a new strong link. There are four possibilities of a new strong link: i) e is inside exactly one community (i.e., both n_1 and n_2 are within the same existing community) (Fig. 4 (a)); ii) e is inside overlapping communities (i.e., both n_1 and n_2 are in the overlapping part of several existing communities) (Fig. 4 (b)); iii) e connects different communities (i.e., n_1 and n_2 are inside different communities (both n_1 and n_2 can be in overlapping communities)) (Fig. 4 (c)-(d)); iv) e connects some non-community structures (i.e., at least one of n_1 and n_2 has links with other nodes that are not inside any community) (Fig. 4 (e)-(g)).



(a) No change in community c_1 (b) Merge overlapping communities c_1 and c_2 (c) Merge seperate communities c_1 and c_2



(d) Merge n_1 into community c_2 (e) Merge n_2 and n_3 into c_1 (f) Create community c_2 (g) Create community c_1

Fig. 4. Addition of a strong link ($k = 3$)

Theorem 1. *For communities constructed based on link density, a new strong link inside a community or connecting a community with other communities or non-community structures should not split or destroy the community.*

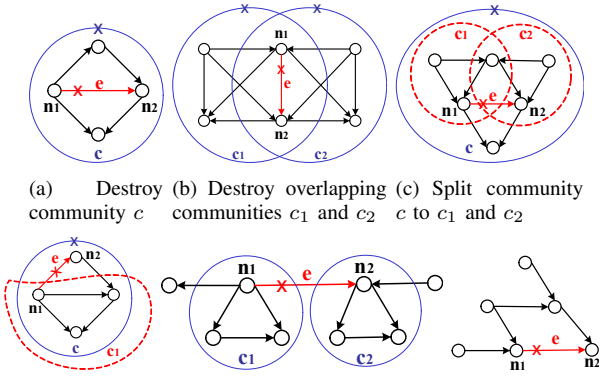
According to Theorem 1, we can handle each possibility of a new strong link e as follows.

- i) keep the current community structure intact (Fig. 4 (a)).
- ii) community merge might happen for any subset of the overlapping communities (Fig. 4 (b)). For each pair of communities containing e, find local k-cliques containing e until a k-clique contains another overlapping node of the two communities (excluding n_1 and n_2), and mark the two communities as "to merge" if found.
- iii) community merge might happen for any subset of communities containing n_1 and n_2 (Fig. 4 (c)), or one of n_1 and n_2 might be included in the other's communities (Fig. 4 (d)). For each pair of communities including n_1 and n_2, respectively, find all local k-cliques K_e including e, and mark the two communities as "to merge" if there exists a k-clique chain constructed from K_e that includes any other nodes in both the two communities; otherwise, "merge" the other node of e into the community if there exists a k-clique in K_e that includes any other nodes in only one of the two communities.
- iv) if e connects communities with a non-community structure, non-community structures of n_1 or n_2 might be included into the communities of n_1 or n_2 (Fig. 4 (e)), or might form new communities (Fig. 4 (f)). To do this, for each community including n_1 or n_2, "merge" the adjacent k-cliques into the community if e is in adjacent k-cliques of the community, otherwise, find all local k-cliques K_e including e and "create" a new community for each k-clique chain constructed from K_e. If e connects two non-community structures, the non-community structures of n_1 and n_2 might form new communities (Fig. 4 (g)). To do this, find all local k-cliques K_e including e, and "create" a new community for each k-clique chain constructed from K_e.

For the communities marked as "to merge", "Merge" them into larger groups in which every community is identified "to merge" with at least one of the other communities.

B. Removing a Strong Link

For a directed link e from node n_1 to n_2, when its link weight becomes lower than the increased w^*, it becomes weak. There are four possibilities of a removed strong link: i) e is inside a community or overlapping communities (i.e., both n_1 and n_2 are within the same community or overlapping communities) (Fig. 5 (a)-(d)); ii) e is connecting different communities or non-community structures (i.e., n_1 and n_2 do not share any community) (Fig. 5 (e)-(f)).



(a) Destroy community c (b) Destroy overlapping communities c_1 and c_2 (c) Split community c to c_1 and c_2

(d) Split c to c_1 (e) No change in c_1 and c_2 (f) No change

Fig. 5. Removal of a strong link ($k = 3$, except in (b) $k = 4$)

Theorem 2. *For communities constructed based on link density, removal of a strong link connecting a community with other communities or non-community structures should not split or destroy the community.*

According to Theorem 2, we can handle each destroyed strong link as follows: i) community destruction (Fig. 5 (a)-(b)) or spliting (Fig. 5 (c)-(d)) might happen for each community involved; ii) keep current community structure (Fig. 5 (e)-(f)). Algorithm 2 shows how to identify communities that should be split or destroyed due to the removal of a strong link.

Algorithm 2 : Identify Community Split/Destroy

Input: community structure, strong link $e(n_1, n_2)$ to be removed
Output: communities that are split or destroyed

for each community including e **do**
  Find all local k-cliques $\{K_1\}$ including n_1 after removing e;
  Find all local k-cliques $\{K_2\}$ including n_2 after removing e;
  if $\{K_1\} = \emptyset$ and $\{K_2\} = \emptyset$ (Fig. 5 (a)) **then**
    "Destroy" the community;
  else if $\{K_1\} \neq \emptyset$ and $\{K_2\} \neq \emptyset$ (Fig. 5 (c)) **then**
    if none of the k-cliques in $\{K_1\}$ are adjacent k-cliques with any k-clique in $\{K_2\}$ **then**
      "Split" the community into two communities including $\{K_1\}$ and $\{K_2\}$, respectively;
    end if
  else if $\{K_1\} \neq \emptyset$ (Fig. 5 (d)) **then**
    "Split" the community by excluding n_2;
  else
    "Split" the community by excluding n_1;
  end if
end for

C. Updating Link Weight Threshold

We set an error threshold E_f for the link fraction f^*. When there are enough link updates leading to the situation when link fraction difference is greater than the error threshold E_f or f^* falls below 0.5, we update the link weight threshold w^* (as shown in Algorithm 3) so there are enough strong links to form communities. Recall in Algorithm 1, the increase of w^* (only happens when $f^* - f^0 > E_f$) may result in the removal of strong links, and the decrease of w^* (only happens when $f < 0.5$) may lead to addition of strong links.

Algorithm 3 : Update Link Weight Threshold

Input: community structure, w^*, f^0, E_f, m, s
Output: w^*, m, s

for each link update considered **do**
  Get the weight increment Δw from the link update
  if the current weight of the link is $w = 0$ **then**
    $s = s + 1$;
  end if
  if $w < w^*$ and $w + \Delta w \geq w^*$ **then**
    $m = m + 1$;
  end if
  $w = w + \Delta w$;
end for
$f^* = m/s$;
if $f^* - f^0 > E_f$ or $f < 0.5$ **then**
  Sort all the links by decreasing order of weight;
  Count the sorted links as strong links until $f^* - f^0 \leq E_f$ and $f^* \geq 0.5$, and then update w^* accordingly;
end if

VI. Performance Evaluation

For performance studies, we use the UIM dataset [19] that has both location/PoI and contact information. Interested readers may refer to our technical report [20] for the justifications of using this dataset.

UIM WiFi Traces: The WiFi traces are collected every 30 minutes. There are 5614 WiFi AP MACs in the WiFi traces of all the 27 users we considered, and the number of valid WiFi AP MACs which do not occasionally occur is 985 (i.e., occur at least 27 times in the entire data set in our evaluation). Then, we apply the star clustering algorithm [21] on the WiFi APs; we get about 200 clusters, each cluster represents a PoI.

UIM Bluetooth Traces: The Bluetooth traces are collected every minute. In addition to the Bluetooth MACs of the 27 users we have previously considered, there are another 7671 Bluetooth MACs in the Bluetooth traces of these 27 users, and the number of Bluetooth MACs that have frequent contacts is 123 (i.e., at least 100 contacts in the whole dataset in our evaluation). Therefore, we also take these 123 bluetooth MACs into consideration, and then we have 150 users in total.

By combining the WiFi and Bluetooth traces, we can get the user mobilities with PoI visits (updated every 30 minutes for each user) and user contacts (updated every minute for each user). We construct an integrated user mobility profile and divide it into two parts—the training set (from 03/01/2010 to 03/10/2010) to construct the basic community structure and the test set (from 03/11/2010 to 03/19/2010) to study the evolving community structure and its impact on the applications as well. Note that, those users without mobility profiles only have contacts with users who have mobility profiles, they still can be influenced and also further influence others via contacts.

In the literature, there is no standard criteria for evaluating community structures. In order to exploit the local feature

of the community structure based on CPMd, we adopt the Internal Pairwise Similarity (IPS) metric used in a greedy local optimization based community detection approach [22]. IPS measures the average similarity between pairs of nodes within community. It can be used to evaluate community detection algorithms which support overlapping communities and does not need the ground truth as the baseline. In addition to average IPS, we use average community size and average number of communities per PoI to show the community structure.

To study the evolution with actual information influences, we let each user be self-influenced with a PoI event when visiting the PoI. We set $w^* = 2$ in constructing the basic community structure and $k = 3$ since the constructed influence graph does not have many cliques with higher k. In the following evaluation, E_f is set to be 0.006, the community structure is updated daily. Different update frequency may be used and interested readers may refer to our technical report for more results and discussions [20].

Fig. 6 shows the impact of T_l on the final community structure. When T_l increases, the average IPS increases because there are more PoI influences among users within higher T_l; the average community size increases because more users are involved within higher T_l, and the average communities per PoI increases because there are more PoI influences and also more users within higher T_l. Since overlaps are allowed, the product of the average community size and the average communities per PoI also grows when T_l increases.

Fig. 7 shows how the community properties evolve every day. Day 0 represents the basic community structure constructed from the training set of the UIM dataset. The properties of evolving community structure fluctuate every day and do not vary drastically between two adjacent days relative to the overall fluctuation. It means that *the evolving community structure is consistent over time*, which meets the objectives of community evolution as discussed in [6]. Further, we compare the properties of the context-aware community structure ("Context-Aware") with those of the pure contact-based community structure ("Contact-Only") using the same dataset and the same algorithms. The results show that the pure contact-based community structure has much lower average IPS. Since the pure contact-based community structure is the same for all PoIs, it has a larger average community size as well as more communities, and it changes more frequently during evolution due to more updates of w^*, leading to more community splits and merges (i.e., more and larger fluctuations). In summary, *the context-aware community structure has higher similarity within communities and is more consistent in evolution than the pure contact-based community structure.*

Table I shows the number of links added every day and the time of updating the community structure on a server with Intel(R) Core(TM) 3.40GHz CPU. The results show that the update time of the evolving community structure is much less than the construction time of the basic community structure. Further, we compare the number of links added in each day to the influence graph with the number of links added to the contact graph. The results show that the number of links in the influence graph is almost always below 20% of the number of links in the contact graph. This validates our intuition that the number of links in the influence graph is much smaller than that in the contact graph. This is because no information influence exists for many contacts. This also implies that communities built using pure contact graphs as in most existing work are not meaningful since links in contact graphs do not always imply potential information influence. In summary, *our evolving community structure is effective in reflecting information influence and efficient in updating the communities. Note that fewer relevant links in the influence graph leads to smaller and less communities than using contact graphs, thereby reducing communication overhead for information sharing in proximity-based mobile social applications.* We will validate this intuition using two mobile social applications in the next section.

TABLE I
LINK UPDATES AND COMMUNITY UPDATE TIME ($T_l = 5, E_f = 0.006$)

	Context-Aware			Contact-Only
Day	Links Added	Strong Links Added	Time (s)	Links Added
0	28537	23812	3.5402	242352
1	3876	3325	0.5977	30741
2	4105	3557	0.6125	34456
3	3143	2388	0.5694	20123
4	2273	1854	0.5116	14051
5	3730	3098	0.5856	24657
6	3306	2699	0.5751	21510
7	2837	2213	0.5442	18068
8	3052	2681	0.5608	19233
9	1777	1430	0.4233	8167

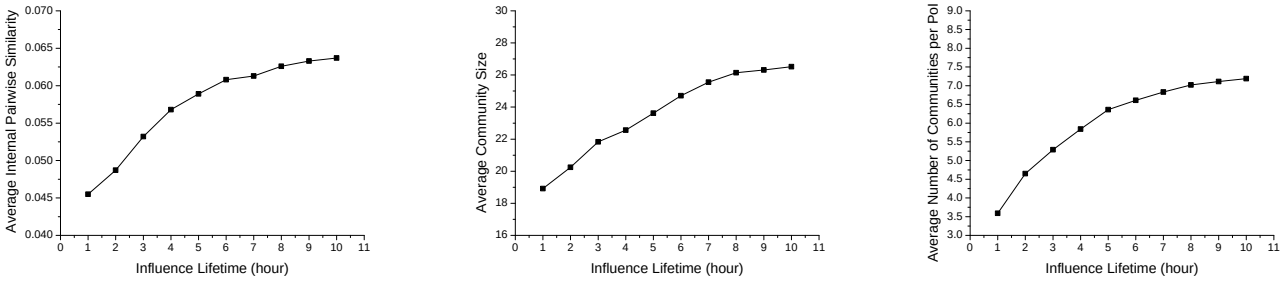
VII. COMMUNITY-BASED MOBILE SOCIAL APPLICATIONS

The applications can determine the frequency of community updates. For instance, it can use an online server to collect user profile updates in real time and then trigger updates of the community structure immediately, or it can use an offline server to periodically collect user profile updates and then update the community structure periodically. In the following applications, we assume an offline server is being used and the community structure is updated daily. For each application, we evaluate its performance for each day with updated communities using the test set of UIM. For the events happening in PoIs, we do not specify any event lifetime or update period. We assume that when a user visits a PoI each time, only one new event is generated during the user's stay.

A. Don't Miss: Community-based event sharing

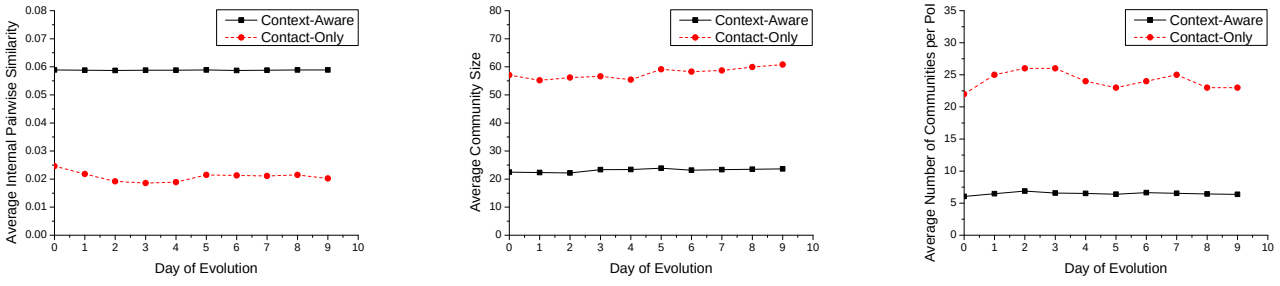
Each user who holds an active event (i.e., the event is still within the user's influence lifetime) forwards it to those encountered users who are within the same community regrading the PoI of the event. If the receiver has visited the PoI of the event before, he will keep the event; otherwise, the event is discarded. In addition to implementing this application using "Context-Aware" communities and "Contact-Only" communities, we also implement a baseline approach—"Epidemic" where a user forwards the events to all encountered users.

Fig. 8(a) shows the sharing success ratio, which is the number of successful event sharings (i.e., those events which have been delivered to at least one user who has visited the



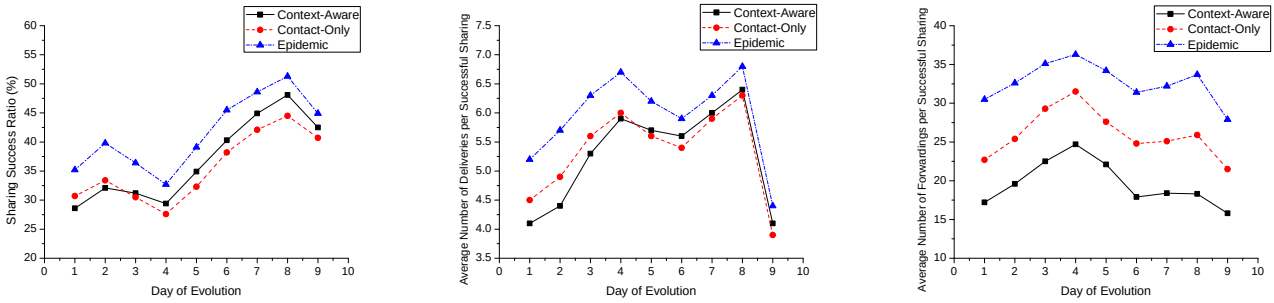
(a) (b) (c)

Fig. 6. Impact of T_l ($E_f = 0.006$)



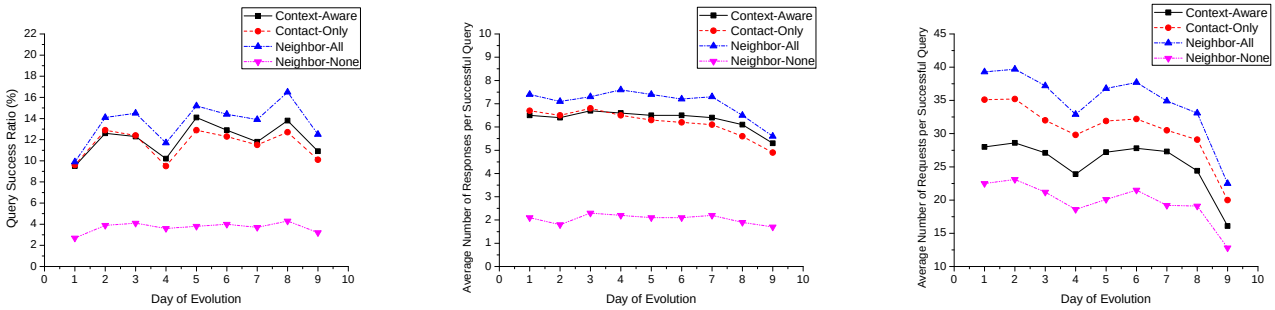
(a) (b) (c)

Fig. 7. Community Evolution ($T_l = 5, E_f = 0.006$)



(a) (b) (c)

Fig. 8. Performance of Event Sharing with Daily Update ($T_l = 5, E_f = 0.006$)



(a) (b) (c)

Fig. 9. Performance of Event Query with Daily Update ($T_l = 5, E_f = 0.006$)

PoI of the event before) over the number of total events being shared (i.e., the number of PoI visits of all users since one event per PoI visit is assumed). "Contact-Only" has higher sharing success ratios in the first two days, but then it falls below "Context-Aware". This is because the update of context-aware community structure is PoI related; it gathers more relevant users for each PoI than "Contact-Only". Fig. 8(b) shows that "Context-Aware" shares events with more interested users than "Contact-Only" during evolution. Fig. 8(c) shows that "Context-Aware" introduces much lower cost than "Contact-Only". The average reduction of the event forwarding cost is about 24.7%. This is because "Contact-Only" gathers users regardless of PoI relevance, leading to more irrelevant users participating in event sharing. In summary, *the context-aware community-based event sharing is effective in event delivery and efficient in terms of event forwarding cost.*

B. What's Up: Community-based event query

Each user who is going to a PoI initiates a query of events regarding the PoI and requests other users who are in contact

to respond. An encountered user who receives a request from the querist further requests his neighbors who are in the same community regarding the PoI to respond. In the meantime, each user receiving a request responds with events regarding the PoI if he has such events within the influence lifetime. In addition to implementing the application using "Context-Aware" communities and "Contact-Only" communities, we implement two baseline approaches —"Neighbor-All" where a user who receives a request from the original querist further requests all his neighbors to respond and "Neighbor-None" where a user does not further request his neighbors to respond.

Fig. 9(a) shows the query success ratio, which is the number of successful queries (i.e., those queries which have at least one response) over the number of total queries (i.e., the number of PoI visits of all users), and Fig. 9(b) shows the average number of users who have responded for each successful query. The higher average number of users who have the right answers to the queries, the more effective the community is. "Context-Aware" reachers more users who have the right answers to the queries than "Contact-Only" since it groups users based on potential information influences. Fig. 9(c) shows the average number of users who have been requested for each successful query. "Context-Aware" incurs much lower cost than "Contact-Only". The average reduction of the cost is about 16.5%. This is because the contact-only community structure involves lots of users who are not relevant to the PoIs of the queries, leading to many more unnecessary requests to users who cannot respond to the queries. In summary, *the context-aware community-based event query is effective in query response and efficient in terms of query request cost*.

More Discussions: In both applications, the performance of using the context-aware community structure is better than using the contact-only community structure. The reduction of event forwarding cost or query request cost validates our intuition in section VI that context-aware communities will lead to lower communication overhead due to smaller and less communities constructed using influence graphs. On the other hand, since over 80% of the contacts in the UIM dataset cannot contribute to information influence as discussed in section VI, the advantage of context-aware communities has not been fully revealed. Actually, the performance could be even better with more regular mobility patterns such as repeated daily mobilities which can contribute to significant information influence and lead to higher relevance within communities.

VIII. Conclusions

In this paper, we have first integrated social contexts into user contacts to construct an influence graph, then constructed the basic context-aware community structure using the influence graph. We have further developed algorithms to update the evolving community structure. Our performance studies indicate that the context-aware community structure has good community properties (i.e., high average IPS, reasonable average community size and average communities per PoI). It is consistent in evolution, and it is also effective in reflecting information influence while efficient in updating the communities. We have further applied the evolving community structure to two types of mobile social applications—event sharing and event query. Our evaluation results show that the context-aware community structure is effective in both event delivery and query response, while also efficient in terms of event forwarding cost and query request cost.

Acknowledgement

This project is supported in part by NSF grant CNS-0915574.

References

[1] W. Chen, Z. Liu, X. Sun, and Y. Wang, "A game-theoretic framework to identify overlapping communities in social networks," *Data Mining and Knowledge Discovery*, 2010.

[2] Y. Wang, G. Cong, G. Song, and K. Xie, "Community-based greedy algorithm for mining top-k influential nodes in mobile social networks," in *SIGKDD*, 2010.

[3] N. Yu and Q. Han, "Context-aware communities and their impact on information influence in mobile social networks," in *PerCol*, 2012.

[4] Y. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, "Facetnet: A framework for analyzing communities and their evolutions in dynamic networks," in *WWW*, 2008.

[5] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *ASONAM*, 2010.

[6] Y. Sun, J. Tang, J. Han, M. Gupta, and B. Zhao, "Community evolution detection in dynamic heterogeneous information networks," in *MLG*, 2010.

[7] B. Furht, *Handbook of Social Network Technologies and Applications*. Springer, 2010.

[8] N. P. Nguyen, T. N. Dinh, S. Tokala, and M. T. Thai, "Overlapping communities in dynamic networks: Their detection and how they can help mobile applications," in *MobiCom*, 2011.

[9] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis," *Phys. Rev.*, 2009.

[10] A. Pietilainen and C. Diot, "Dissemination in opportunistic social networks: the role of temporal communities," in *MobiHoc*, 2012.

[11] T. Hossmann, T. Spyropoulos, and F. Legendre, "Putting contacts into context: Mobility modeling beyond inter-contact times," in *MobiHoc*, 2011.

[12] D. Zhang, Z. Wang, B. Guo, X. Zhou, and V. Raychoudhury, "A dynamic community creation mechanism in opportunistic mobile social networks," in *SocialCom*, 2011.

[13] P. Hui, E. Yoneki, S. Chan, and J. Crowcroft, "Distributed community detection in delay tolerant networks," in *MobiArch*, 2007.

[14] E. G. Boix, A. L. Carreton, C. Scholliers, T. V. Cutsem, and W. D. Meuter, "Flocks: Enabling dynamic group interactions in mobile social networking applications," in *SAC*, 2011.

[15] R. Lubke, D. Schuster, and A. Schill, "Mobilisgroups: Location-based group formation in mobile social networks," in *PerCol*, 2011.

[16] J. Reichardt and S. Bornholdt, "Statistical mechanics of community hierarchies in large networks," *Phys. Rev.*, 2006.

[17] G. Palla, I. Dernyi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, 2005.

[18] G. Palla, I. Farkas, P. Pollner, I. Dernyi, and T. Vicsek, "Directed network modules," *New Journal of Physics*, 2007.

[19] UIUC, "Uim," http://crawdad.cs.dartmouth.edu/meta.php?name=uiuc/uim.

[20] N. Yu and Q. Han, "Context-aware community," Colorado School of Mines, Tech. Rep., 2012, http://pecs.mines.edu/trac/attachment/wiki/Students/NaYu/file/CONTEXT.pdf.

[21] J. Aslam, K. Pelekhov, and D. Rus, "Static and dynamic information organization with star clusters," in *the Conference on Information Knowledge Management*, 1998.

[22] M. Goldberg, S. Kelley, M. Magdon-Ismail, K. Mertsalov, and A. Wallace, "Finding overlapping communities in social networks," in *SocialCom*, 2010.