Hindawi Publishing Corporation
Mobile Information Systems
Volume 2015, Article ID 402705, 18 pages
http://dx.doi.org/10.1155/2015/402705

Research Article

Context-Aware Community Construction in Proximity-Based Mobile Networks

Na Yu and Qi Han

Department of Electrical Engineering and Computer Science, Colorado School of Mines, Golden, CO 80401, USA

Correspondence should be addressed to Qi Han; qhan@mines.edu

Received 28 August 2014; Accepted 1 September 2014

Academic Editor: David Taniar

Sensor-equipped mobile devices have allowed users to participate in various social networking services. We focus on proximity-based mobile social networking environments where users can 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 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 contexts. To model the potential information influences, we construct an influence graph by integrating the space and time contexts into the proximity-based 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.

1. 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, the studies of community structures become even more important. For instance, people may collect information from the places they visit and share that information with other people they encounter in their *proximity* as they move. Proximity here implies their mobile devices are within each other's WiFi or Bluetooth transmission ranges. Identifying communities in such proximity-based mobile networks could often lead to more cost-effectiveness 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 how to provide more relevant information for community construction to more efficiently support applications in proximity-based mobile networks.

Existing work, as detailed in Section 2, 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 there is information to be shared via the contacts or not, hence losing critical information in proximity-based mobile networks: the contexts associated with these contacts, that is, the space 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 their space and time contexts.

In this paper, we address the drawbacks of existing work by building connections based on potential information

influence when people are in proximity to construct context-aware community structure. This information influence is tied to the 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 he/she gets the information, 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.

Similar to contact-only communities, context-aware communities are also constructed using long-term mobility patterns of mobile users where contact duration and contact frequency are considered along with places a user has visited. Therefore, context-aware communities are more informed and meaningful than the pure contact-based communities and can be utilized to better support data services in proximity-based mobile networks while consuming less energy. Consider the previous campus scenario, when Emma wants to know 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.

In this work, we have integrated space and time contexts with contacts to construct a novel graph,"influence graph," and developed a two-phase strategy to detect basic context-aware communities and track the evolving context-aware communities over time. Our evaluation results show that the context-aware community structure has good community properties using some standard community metrics. During evolution, it is consistent in these community properties, effective in reflecting information influence, and efficient in updating the communities. Further, it is effective and efficient in supporting two types of mobile social applications—event sharing and event query.

The rest of the paper is organized as follows: in Section 2, we present the related work. In Section 3, we present the assumptions and problem statement. In Section 4, our approach of constructing influence graphs is given. In Section 5, our approach of detecting and tracking context-aware communities is given. In Section 6, we present the results of context-aware community structure with empirical datasets. In Section 7, we present the results of applying context-aware communities to mobile social applications.

2. Related Work

Detection and evolution of community structures are a well-studied topic in traditional social networks [3]. One method for tracking community evolution is based on historical characteristics. 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 historical community structures; Evo-NetClus [5] studies the problem of evolutionary multityped communities in heterogeneous networks and proposes a model to generate net-clusters at each time window, which is achieved by deciding the natural cluster number and considering historical impacts from net-clusters of previous time windows simultaneously. Another method for tracking community evolution is based on significant discrete events. Evolutionary behavior of interaction graphs [6] captures interesting events from nonoverlapping snapshots of interaction graphs and uses these events to characterize complex behavioral patterns of individuals and communities over time; evolutionary events-based community tracking [7] describes a model for evolutionary communities with life-cycles characterized by a series of significant events and proposes a simple method involving matching communities found at consecutive time steps in the individual snapshot graphs to identify and track communities; similarly, GED [8] proposes community evolution discovery algorithms based on seven types of events, including continuing, shrinking, growing, splitting, merging, dissolving, and forming, and it takes into account both the quantity and quality of community members by providing a balance between the communities which contain many but less important members and communities which contain only a few but key members.

Although community structures play an important role in social network analysis [9], research on communities in mobile social networks is still largely incomplete. One of the most recent works [10] 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 [11] 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 [12, 13], distributed local encounters [14, 15], or simply user proximity [16, 17]. None of the existing work includes any space-time context in their community construction.

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

3. Problem Description

In order to represent the space and time contexts, we adopt the concept point of interests (PoIs) [18]. 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, in the campus scenario, 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 Figure 1, there are four PoIs (represented using small rectangles) and 20 mobile users (represented using small circles), where some users are moving outside PoIs (moving directions are represented using arrows). 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 other mobile users with the PoI events via their mobile devices if they encounter other 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 (i.e., influence other users). Therefore, the information influence is continued in a ripple carry fashion.

The problem we are studying in this work is how to take into account space and time contexts when forming communities so that the constructed communities can more efficiently support information influence. In other words, our focus is not on a new algorithm for community construction, rather it is about how to integrate the space and time contexts into community construction. This problem is broken down into three issues: (i) building the social graph based on user mobility profiles; (ii) constructing basic context-aware communities and dynamically updating the context-aware communities to reflect the evolving network; and (iii) using the dynamic communities to support mobile social applications.

4. Influence Graph

In this section, we introduce influence graphs, directed weighted graphs that embed all the contacts as well as the associated space and time contexts. The influence graph will be used for community construction in the next section.

Influence graphs are based on the *potential information influences* of PoI events between mobile users. The potential information influence when a contact occurs is determined as follows: the potential information 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 PoI 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 information influences regarding PoI p (i.e., user i can be influenced by other users), within user i's influence lifetime before this contact occurs. In other words, the influence is determined by both direct and indirect exposure to PoI events.

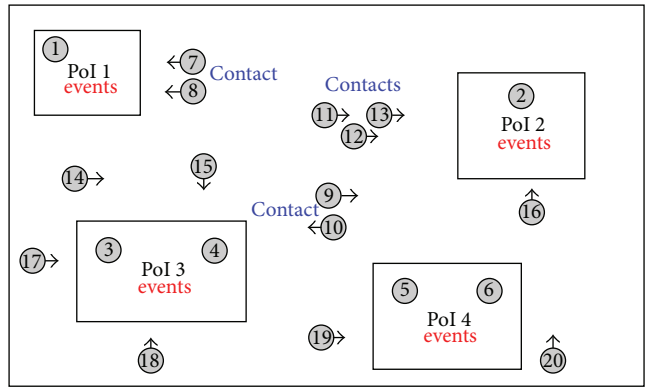


Figure 1: PoIs and mobile users.

Formally, 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. In other words, 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 user i's influence lifetime. More formally, when user i and user j encounter at time t, the potential information influences from user i to user j is denoted as a tuple $I_{i,j,t} = (I_{P_1}, I_{P_2}, \ldots, I_{P_p}, \ldots, I_{P_N})$, 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 information influences regarding PoI p when the contact occurs. Then, the weight of $e_{i,j}$ is a tuple accumulated for all contacts between user i and user j: $w_{i,j} = \sum_t I_{i,j,t} = (\sum I_{P_1}, \sum I_{P_2}, \ldots, \sum I_{P_p}, \ldots, \sum I_{P_N})$. 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.

4.1. Generation of 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 the time when the user enters or exits the PoI and the time when the user is in contact with other users. More formally, a user i's mobility profile can be represented using a sequence of timestamped events. There are three types of events: (i) a user enters a PoI $(t, \text{Start}, \text{User}_i, \text{PoI}_p)$, (ii) a user exits a PoI $(t, \text{End}, \text{User}_i, \text{PoI}_p)$, and (iii) two users are in contact $(t, \text{Contact}, \text{User}_i, \text{User}_j)$.

Figure 2 is a direct translation of 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.

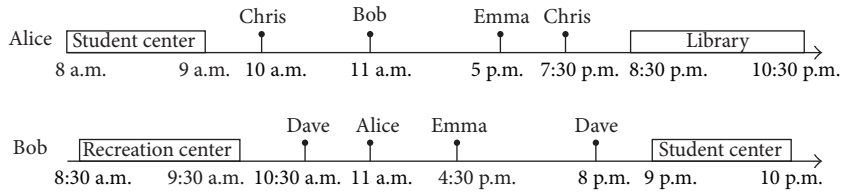


Figure 2: Sample individual user mobility profiles for Alice and Bob.

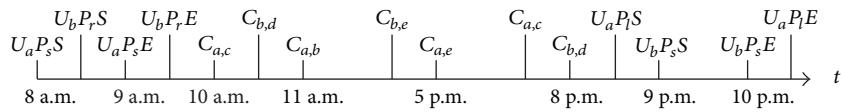


Figure 3: Sample integrated mobility profile of Alice and Bob.

4.2. Construction of Influence Graphs. Using the mobility profiles generated, we can construct influence graphs using the following steps.

(1) Do a merge sort on the mobility profiles based on time.

(2) Scan the integrated user mobility profile to find all the bidirectional potential information influences between any pair of users in contact for each PoI. Note that the number of potential information 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 information 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 events $(t, \text{End}, \text{User}_i, \text{PoI}_p)$ within the influence lifetime) 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 information influences for each PoI from one user to another as the weight of the directed edge on the influence graph.

Consider Figure 2 as an example. The integrated mobility profile can be shown as in Figure 3, where "a," "b," "c," "d," and "e" represent Alice, Bob, Chris, Dave, and Emma, respectively, and "l," "r," and "s" represent the library, the recreation center, and the student center, respectively. At each time instant, the event 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.

Figure 4 is the influence graph based on Figure 3, assuming both Alice and Bob have the influence lifetime $T_l = 5$ hours. The values in the weight tuples represent the number of potential information influences for the library, the recreation center, and the student center, respectively. For instance, Alice and Bob are in contact at 11 AM. Within 5 hours before 11 AM, Alice has visited the student center once while Bob has visited the recreation center once. As a result, Alice has a potential information influence to Bob regrading the student center while Bob has a potential information influence to Alice regarding the recreation center. Therefore, the weight tuple from Alice to Bob is (0, 0, 1), and the weight tuple from Bob to Alice is (0, 1, 0).

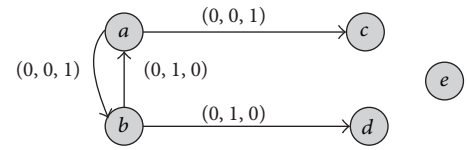


Figure 4: Sample influence graph.

5. Context-Aware Community Structure

5.1. Detecting Basic Context-Aware Communities. In the first phase of our approach, we form the basic communities using the influence graph. There are various ways to discover communities, most of which are based on either the centralized modularity optimization approach [19] or the local clique percolation approach [20]. We choose to apply the CPMd algorithm [21], which is the directed graph version of the well-known clique percolation method (CPM) [20]. The advantages of using a clique percolation based method are as follows: (i) it is local so it does not have resolution limit (i.e., it can detect small communities) as centralized modularity optimization approaches have, (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^* and *weak links* to represent other links.

The link fraction f^* is the ratio of the number of strong links over the total number of links in the influence graph. For each selected value of k, CPMd adjusts the link weight threshold w^* to the point where the largest community becomes twice as big as the second largest one. In addition, at least half of the links should be strong links for the optimal threshold of w^* (i.e., the link fraction 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 $C_c = (\text{PoI}_1 : (C_1, C_2, \ldots), \text{PoI}_2 : (C_1, C_2, \ldots), \ldots)$. The significance of the context-aware community structure is that users who have more potential information influences with each other 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 Figure 4, and by setting $k = 2$ and $w^* = 1$, we can construct the context-aware community structure as $C_c = (P_r : (\{a, b, d\}), P_s : (\{a, b, c\}))$. There are two communities, one for the recreation center with members Alice, Bob, and Dave and another for the student center with members Alice, Bob, and Chris. 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).

5.2. Tracking Evolving Context-Aware Communities. A good evolutionary model of communities should have flexible and automatically learned communities in each time step, while the communities in adjacent timestamps should be consistent [5]. In the second phase of our approach, we present a simple but effective method for efficiently maintaining our community structure, which adaptively updates the communities as user interactions change over time. The most commonly used notations in this paper are listed in Notations section.

According to the discussion in Section 2, our method falls into the category of community evolution methods based on important discrete events, but we also consider the historical inherited characteristics of communities which are reflected by keeping the same criteria of optimal k and w^* values in constructing communities. As discussed in Section 5.1, we construct the basic communities using CPMd. 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. However, 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 nontrivial while following the historical characteristics.

Since CPMd constructs communities based on strong links, 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. In the following subsections, we will address three cases involved in updating the communities.

5.2.1. 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) (Figure 5(a)); (ii) e is inside overlapping communities (i.e., both n_1 and n_2 are in the overlapping part of several existing communities) (Figure 5(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)) (Figures 5(c)-5(d)); (iv) e connects some noncommunity structures (i.e., at least one of n_1 and n_2 has links with other nodes that are not inside any community) (Figures 5(e)–5(g)).

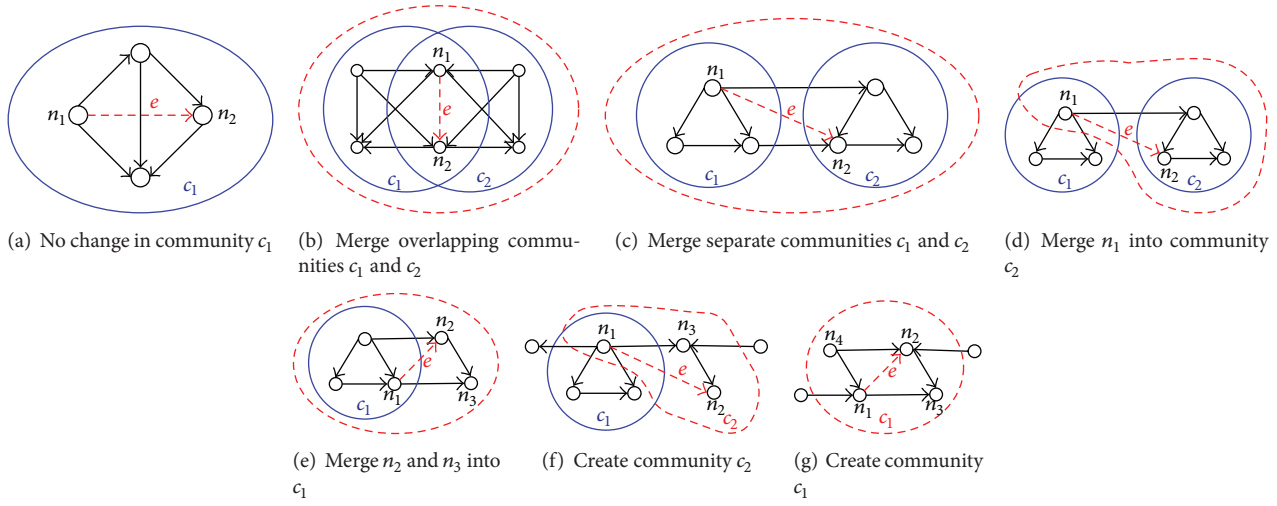
Theorem 1. *For communities constructed based on link density, a new strong link inside a community or connecting a community with other communities or noncommunity structures should not split or destroy the community.*

Proof. This is because community split or destroy only happens when the strong links inside the community are getting sparse for link density-based community construction. Since addition of a new strong link does not degrade the link density, it will not cause community split or destroy. □

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

(i) keep the current community structure intact (Figure 5(a)).

(ii) Community merge might happen for any subset of the overlapping communities (Figure 5(b)). For each pair of communities containing e, find local k-cliques containing e until a k-clique contains another



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

Figure 5: Addition of a strong link ($k = 3$).

Input: community structure, x, w^*, f^0, E_f
Output: an updated community structure and w^*
(1) **for** each of the x link updates **do**
(2) Update link weight;
(3) **if** an addition of strong link **then**
(4) Check for communities merge or creation;
(5) **end if**
(6) **end for**
(7) Calculate the new link fraction f^*;
(8) **if** $f^* - f^0 > E_f$ **then**
(9) Increase w^* so $f^* - f^0 \leq E_f$ and $f^* \geq 0.5$;
(10) **for** each strong link removal **do**
(11) Check for communities split or destroy;
(12) **end for**
(13) **else if** $f^* < 0.5$ **then**
(14) Decrease w^* so that $f^* - f^0 \leq E_f$ and $f^* \geq 0.5$;
(15) **for** each addition of strong link **do**
(16) Check for communities merge or creation;
(17) **end for**
(18) **end if**

Algorithm 1: Overall process of community updates.

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 (Figure 5(c)), or one of n_1 and n_2 might be included in the other's communities (Figure 5(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.

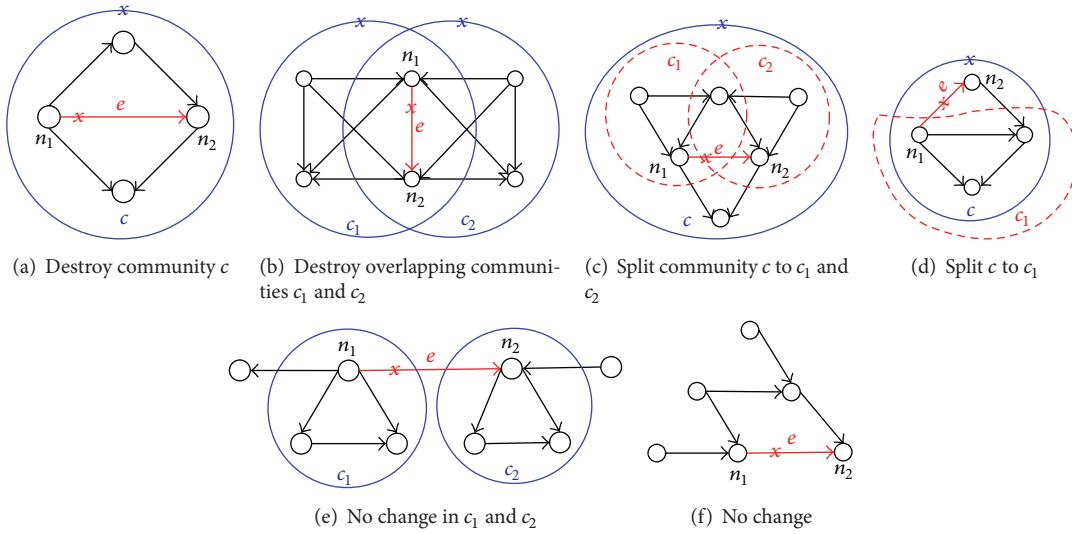
(iv) If e connects communities with a noncommunity structure, noncommunity structures of n_1 or n_2 might be included into the communities of n_1 or n_2 (Figure 5(e)) or might form new communities (Figure 5(f)). To do this, for each community including n_1 or n_2, "merge" the adjacent k-cliques into 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 noncommunity structures, the noncommunity structures of n_1 and n_2 might form new communities (Figure 5(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.

5.2.2. 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) (Figures 6(a)–6(d)); (ii) e is connecting different communities or noncommunity structures (i.e., n_1 and n_2 do not share any community) (Figures 6(e)-6(f)).

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

Proof. This is also because community split or destroy only happens when the strong links inside the community are



(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

FIGURE 6: Removal of a strong link ($k = 3$, except in (b) $k = 4$).

Input: community structure, strong link $e(n_1, n_2)$ to be removed
Ouput: communities that are split or destroyed
(1) **for** each community including e **do**
(2) Find all local k-cliques $\{K_1\}$ including n_1 after removing e;
(3) Find all local k-cliques $\{K_2\}$ including n_2 after removing e;
(4) **if** $\{K_1\} = \varnothing$ and $\{K_2\} = \varnothing$ (Figure 6(a)) **then**
(5) "Destroy" the community;
(6) **else if** $\{K_1\} \neq \varnothing$ and $\{K_2\} \neq \varnothing$ (Figure 6(c)) **then**
(7) **if** none of the k-cliques in $\{K_1\}$ are adjacent k-cliques with any k-clique in $\{K_2\}$ **then**
(8) "Split" the community into two communities including $\{K_1\}$ and $\{K_2\}$, respectively;
(9) **end if**
(10) **else if** $\{K_1\} \neq \varnothing$ (Figure 6(d)) **then**
(11) "Split" the community by excluding n_2;
(12) **else**
(13) "Split" the community by excluding n_1;
(14) **end if**
(15) **end for**

ALGORITHM 2: Identify community split/destroy.

getting sparse. Since removal of a strong link outside a community does not degrade the link density inside the community, it will not cause the community split or destroy. □

According to Theorem 2, we can handle each destroyed strong link as follows: (i) community destruction (Figures 6(a)-6(b)) or spliting (Figures 6(c)-6(d)) might happen for each community involved; (ii) keep current community structure (Figures 6(e)-6(f)). Algorithm 2 shows how to identify communities that should be split or destroyed due to the removal of a strong link.

5.2.3. 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.

6. Performance Evaluation

For performance studies, we need to use empirical datasets that have both location/PoI and contact information. For extracting location information especially indoor PoIs, using WiFi AP accesses is better than using Cell Tower accesses or GPS locations; for extracting contact information, using Bluetooth contacts is better than using other proximity estimation techniques since it has the most appropriate communication range. To the best of our knowledge, there

Input: community structure, w^*, f^0, E_f, m, s
Output: w^*, m, s
(1) **for** each link update considered **do**
(2) Get the weight increment Δw from the link update
(3) **if** the current weight of the link is $w = 0$ **then**
(4) $s = s + 1$;
(5) **end if**
(6) **if** $w < w^*$ and $w + \Delta w \geq w^*$ **then**
(7) $m = m + 1$;
(8) **end if**
(9) $w = w + \Delta w$;
(10) **end for**
(11) $f^* = m/s$;
(12) **if** $f^* - f^0 > E_f$ or $f < 0.5$ **then**
(13) Sort all the links by decreasing order of weight;
(14) Count the sorted links as strong links until $f^* - f^0 \leq E_f$ and $f^* \geq 0.5$, and then update w^* accordingly;
(15) **end if**

Algorithm 3: Update link weight threshold.

are two available datasets containing both WiFi AP accesses and Bluetooth contacts: UIM [22, 23] and Strath Nodobo [24]. UIM was collected via a dedicated joint WiFi/Bluetooth framework [25] in March 2010 with 27 users on UIUC campus, it has most of the WiFi and Bluetooth traces periodically recorded; Nodobo was collected via a general mobile phone sensing software for mobile phone usage traces from September 2010 to February 2011 with 27 high school students, and it has less records of WiFi accesses and Bluetooth contacts for each user than UIM due to less activity of high school students and less focus on collecting WiFi/Bluetooth traces. We use UIM for all evaluations since it contains enough records for this work.

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 [26] on the WiFi APs; we get about 200 clusters, and each cluster represents a PoI. Note that a PoI can be as small as an office suite inside a building.

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 other 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 March 1, 2010, to March 10, 2010) to construct the basic community structure and the test set (from March 11, 2010, to March 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 other users which can be obtained from the available Bluetooth traces. They still can be influenced and also further influence others via contacts. Therefore, all users are added to the influence graph.

In the literature, there is no standard criteria for evaluating community structures. The evaluation metrics most commonly adopted are modularity Q [19] and normalized mutual information [27]. However, the modularity Q has different variations in different community detection algorithms that support overlapping communities. The normalized mutual information comes from information theory and uses ground truth (the actual communities) as the base line; thus it is not applicable to our work where ground truth is unknown. 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 [28]. 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. According to our previous studies [29, 30], T_l heavily impacts the community structure, so we keep evaluating the community structure by varying T_l. Moreover, we vary w^* in constructing the basic community structure while keeping $k = 3$ since the constructed influence graph only has very few cliques with higher k. The chosen range of w^* along with the chosen k satisfies the criteria discussed

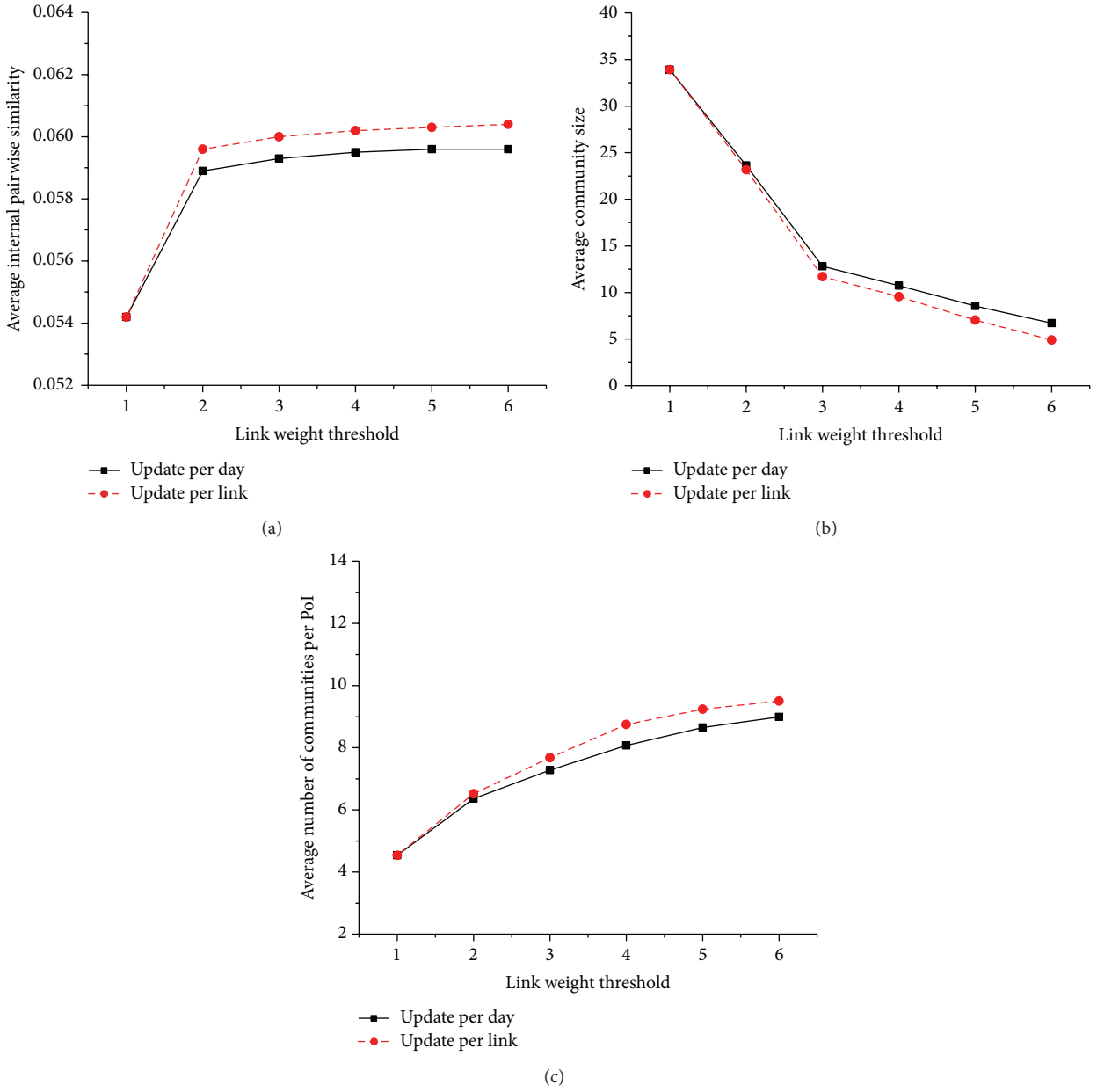


FIGURE 7: Impact of link weight threshold w^* (influence lifetime $T_l = 5$, error threshold $E_f = 0.006$).

in Section 5. Further, we also vary E_f to study its impact on the community structure. In addition, we consider two community updating frequencies—"update per day" (i.e., the community structure is updated daily after processing all link updates occurring on that day) and "update per link" (i.e., the community structure is updated for each link update).

We first evaluate the final community structure that has evolved during the 9 days' test of the UIM dataset with various values of w^*. Since bigger w^* results in less new strong links which further leads to less merging and creation of communities, it finally leads to more and smaller communities with higher IPS inside communities.

Figure 7 shows the impact of w^* on the final community structure with moderate values of T_l and E_f (as suggested in the discussion of Figure 8). We observe (a) as shown in Figure 7(a), when w^* increases, the average IPS increases; "update per link" leads to higher average IPS than "update per day." (b) As shown in Figure 7(b), the average community size decreases when w^* increases, and "update per day" leads to larger average community size than "update per link." (c) As shown in Figure 7(c), the average communities per PoI increase when w^* increases and "update per link" leads to more average communities per PoI than "update per day." Note that when $w^* = 1$, the original link fraction f^0 after constructing the basic community structure is 1, which leads to no update of f^* or w^* in community evolution and also leads to the same community structure for "update per link" and "update per day." Therefore, the community structure

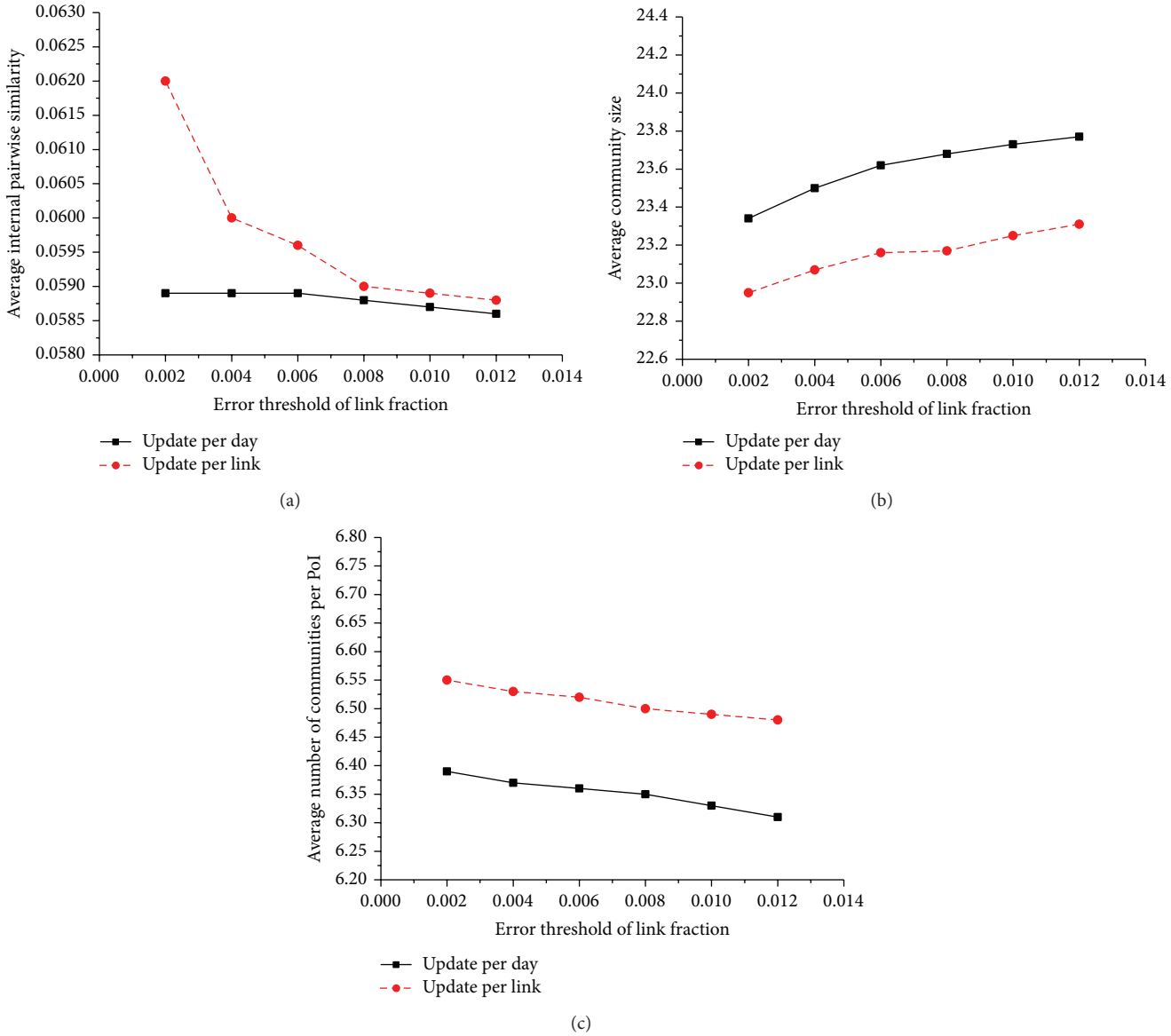


FIGURE 8: Impact of error threshold E_f (influence lifetime $T_l = 5$, link weight threshold $w^* = 2$).

with $w^* = 1$ has much lower average IPS and much larger average community size than higher w^*. When $w^* \geq 2$, w^* will be updated since f^* keeps updating. We observe that when $w^* > 2$, the average IPS does not increase much while the average community size decreases significantly which may lead to insufficient user participations. Therefore, we choose the optimal value $w^* = 2$ for the following evaluations.

Figure 8 shows the impact of E_f on the final community structure with $T_l = 5$ and $w^* = 2$. We observe (a) as shown in Figure 8(a), when E_f increases, the average IPS decreases; "update per link" leads to higher average IPS than "update per day," especially for smaller value E_f. (b) As shown in Figure 8(b), the average community size increases when E_f increases, and "update per day" leads to larger average community size than "update per link." (c) As shown in Figure 8(c), the average communities per PoI decrease when E_f increases and "update per link" leads to more average communities per PoI than "update per day." All these are because bigger E_f results in slower and less spliting and destruction of communities even though these communities are already getting loose. In general, the impact of E_f on the above metrics is not significant. Therefore, we choose the moderate value $E_f = 0.006$ for the following evaluations.

Figure 9 shows the impact of T_l on the final community structure with $w^* = 2$ and $E_f = 0.006$. We observe that "update per link" leads to slightly higher average IPS, smaller average community size, and more average communities per PoI than "update per day." These observations are consistent with the conclusions about the impact of w^* and E_f. In general, the differences between the two community updating frequencies are small. Therefore, we

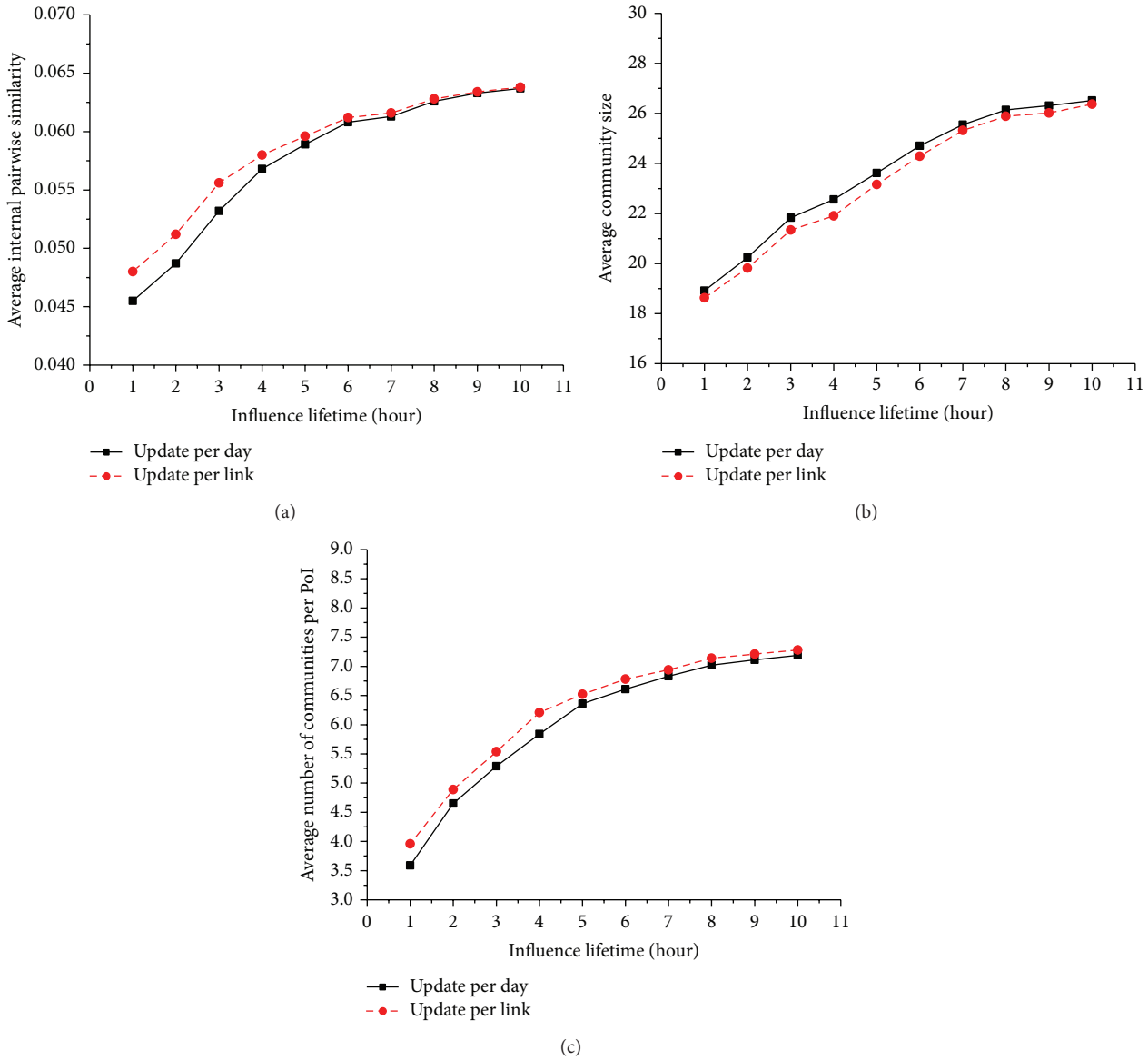


FIGURE 9: Impact of influence lifetime T_l (link weight threshold $w^* = 2$, error threshold $E_f = 0.006$).

can just use "update per day" for the following evaluation of community evolution. Moreover, 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 increase because there are more PoI influences and also more users within higher T_l. Since overlaps are allowed in the context-aware community structure, the product of the average community size and the average communities per PoI also grows when T_l increases.

Figure 10 shows how the community properties evolve every day using "update per 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 too much 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 [5]. In Figure 10, we also compare the community properties of the context-aware community structure ("context-aware") with those of the pure contact-based community structure ("contact-only") using the same dataset. The pure contact-based community structure is based on an undirected contact graph in which each edge along with its weight represents the number of contacts between two users. Its first phase uses CPM to construct the basic communities with the same k and w^* as CPMd in constructing the basic context-aware community structure, and its second phase uses the algorithms presented in Section 5 but with undirected links and k-cliques. The results of Figure 10 show that the pure contact-based community structure has much lower average IPS. Note that the pure contact-based community structure is

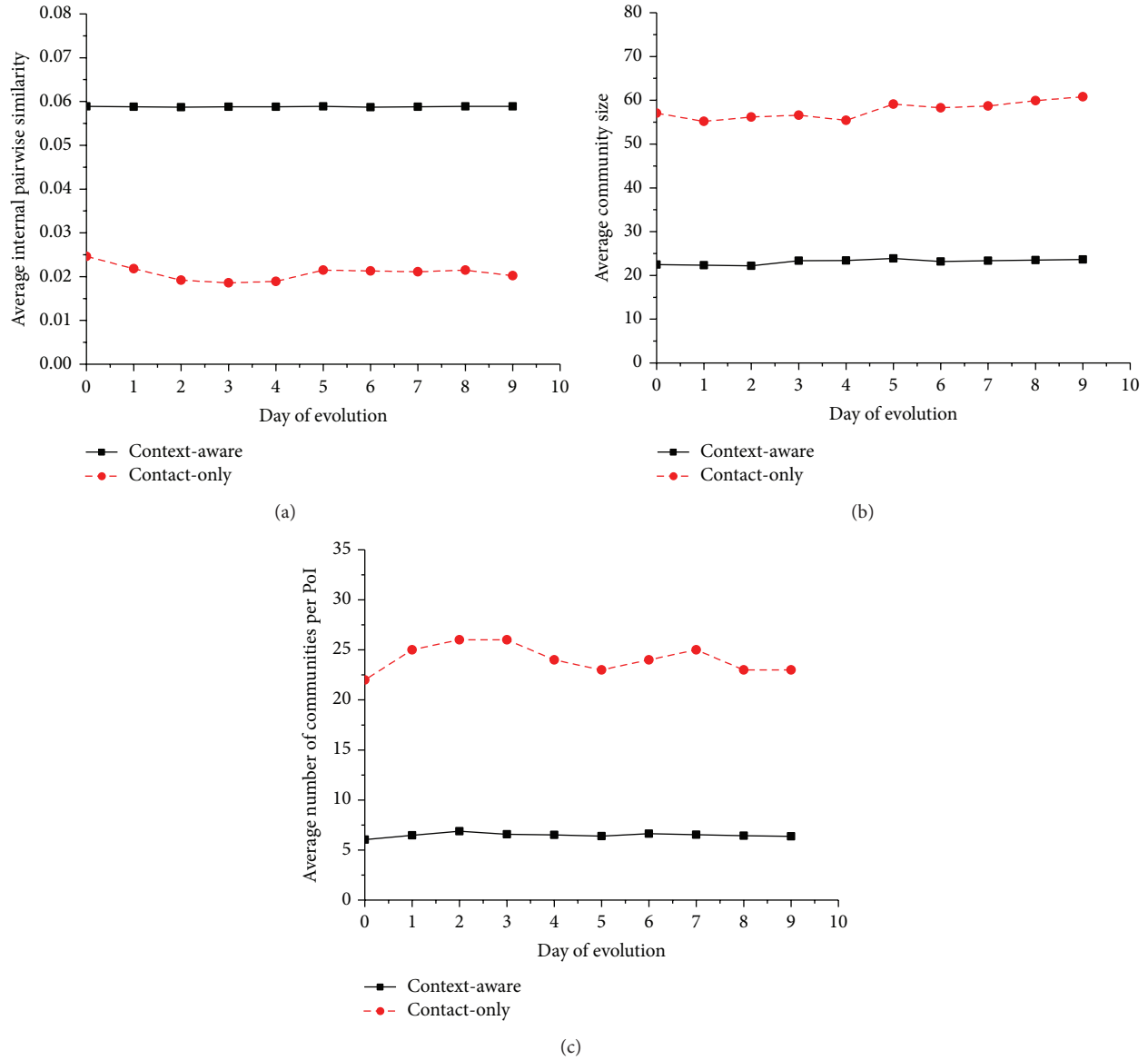
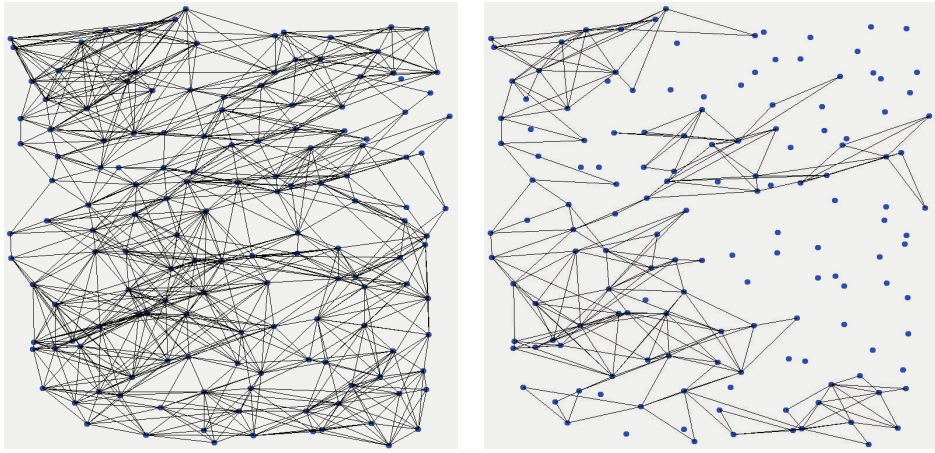


FIGURE 10: Community evolution ($T_l = 5$, $w^* = 2$, $E_f = 0.006$, update per day).

the same for all PoIs; thus, 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*. Moreover, we have also evaluated the community properties when using "update per link" (not shown in Figure 10 since the results are close to those of using "update per day"). It is noteworthy that the average IPS when using "update per link" keeps increasing during evolution. This implies that *the evolving community structure gets more and more cohesive by frequently updating the communities*.

Finally, Table 1 shows the number of links added every day and the time of updating the community structure every day on a server with Intel(R) Core(TM) 3.40 GHz CPU using both "update per day" (UpD) and "update per link" (UpL). The results show that (i) the update time of the evolving community structure is much less than the construction time of the basic community structure and (ii) "update per link" consumes slightly more time than "update per day" due to a few more updates of w^* and more frequent handling of destroyed strong links. These results indicate that our community update techniques are indeed helpful in reducing the community tracking time. 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. The links in the contact graph and the influence graph of the most popular PoI



(a) Contact graph (b) Influence graph of one PoI

Figure 11: Graphs of the entire dataset.

Table 1: Link updates and community update time ($T_l = 5$, $w^* = 2$, $E_f = 0.006$).

Day	Context-aware				Contact-only
	Links added	Strong links added	Time (UpD)	Time (UpL)	Links added
0	28537	23812	3.5402 (s)	3.5402 (s)	242352
1	3876	3325	0.5977 (s)	0.6115 (s)	30741
2	4105	3557	0.6125 (s)	0.6272 (s)	34456
3	3143	2388	0.5694 (s)	0.5825 (s)	20123
4	2273	1854	0.5116 (s)	0.5221 (s)	14051
5	3730	3098	0.5856 (s)	0.5982 (s)	24657
6	3306	2699	0.5751 (s)	0.5883 (s)	21510
7	2837	2213	0.5442 (s)	0.5569 (s)	18068
8	3052	2681	0.5608 (s)	0.5737 (s)	19233
9	1777	1430	0.4233 (s)	0.4348 (s)	8167

(i.e., the most visited PoI by the mobile users) using the entire dataset are shown in Figures 11(a) and 11(b), respectively. 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 accurate representation of the ground truth since links in contact graphs do not always imply potential information influences. 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 lead 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.

7. Community-Based Mobile Social Applications

The 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. We present two applications in proximity-based mobile networks. The advantages of using communities in these applications are twofold: (i) users are divided into various subsets for more localized information sharing and (ii) information sharing can be processed in different ways according to user community memberships.

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 "update per day" is adopted. For each application, we evaluate its performance for each day with updated communities using the test set of the UIM dataset. 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. Then, we compare the application performance when context-aware community structure is used versus when

contact-only community structure (i.e., pure contact-based) is used.

7.1. Do Not Miss: Community-Based Event Sharing. In this application, a user who visits a place shares events happening there with encountered users who have visited the place before but are currently outside that place. 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 view the event; otherwise, the event is discarded. In addition to implementing this application using "context-aware" communities and "contact-only" communities, for comparison, we also implement a baseline approach—"epidemic" where a user forwards the events to all encountered users.

Figure 12(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 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 it falls below "context-aware" in the following days. This is because the update of context-aware community structure is PoI related; it gathers more relevant users for each PoI than contact-only community structure. Figure 12(b) shows the average number of interested users reached per successful sharing. "Context-aware" shares events with more interested users than "contact-only" during evolution. Figure 12(c) shows the cost of the application through the average number of event forwardings per successful sharing. "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.

The performance for each day of evolution varies based on the PoI visits and contacts happening on each day, so the performance cannot be compared from day to day. Moreover, the focus of this work is not on designing an efficient forwarding protocol for mobile social applications, so we simply adopt a basic dissemination protocol in this work. If better forwarding strategies are used, the performance of the application may be improved. In summary, *the context-aware community-based event sharing is effective in event delivery and efficient in terms of event forwarding cost*.

7.2. What Is Up: Community-Based Event Query. In this application, users who are heading specific places query events regarding those places. More specifically, a query of a user is active before the user arrives at a PoI, and a new query is created after the user leaves the PoI.

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" (upper bound) where a user who receives a request from the original querist further requests all his neighbors to respond and "neighbor-none" (lower bound) where a user does not further request his neighbors to respond.

Figure 13(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 Figure 13(b) shows the average number of users who have responded for each successful query. This is to show the average number of users reached by the queries who have the right answers to the queries. This metric demonstrates how effective the communities have been constructed, the higher average number of users, the more effective the community structure is. Using "context-aware" communities reaches more users who have the right answers to the queries than using "contact-only" queries. This is because context-aware communities group people based on potential information influences, eliminating those links that may not contribute to any information influence. Figure 13(c) shows the average number of users who have been requested for each successful query. This metric is the cost measurement. Using "context-aware" communities incurs much lower cost than using "contact-only" communities. 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 addition, the query request cost of "context-aware" is closer to the lower bound than the upper bound.

Moreover, we observe that the query success ratio is low. This is not an indication of the quality of community structure nor the impact of communities on performance. Instead, this is because users move to specific PoIs and may rarely encounter those who have information regarding the same PoIs. If more users can respond to the queries, the performance could be improved. Similar to the event sharing application, no day to day comparison of the performance should be made. In summary, *the context-aware community-based event query is effective in query response and efficient in terms of query request cost*.

7.3. More Discussion of the Results. 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 in these applications validates our intuition in Section 6 that applications using the context-aware community structure will have lower communication overhead than using pure contact-based community structure 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 6, the advantage of context-aware

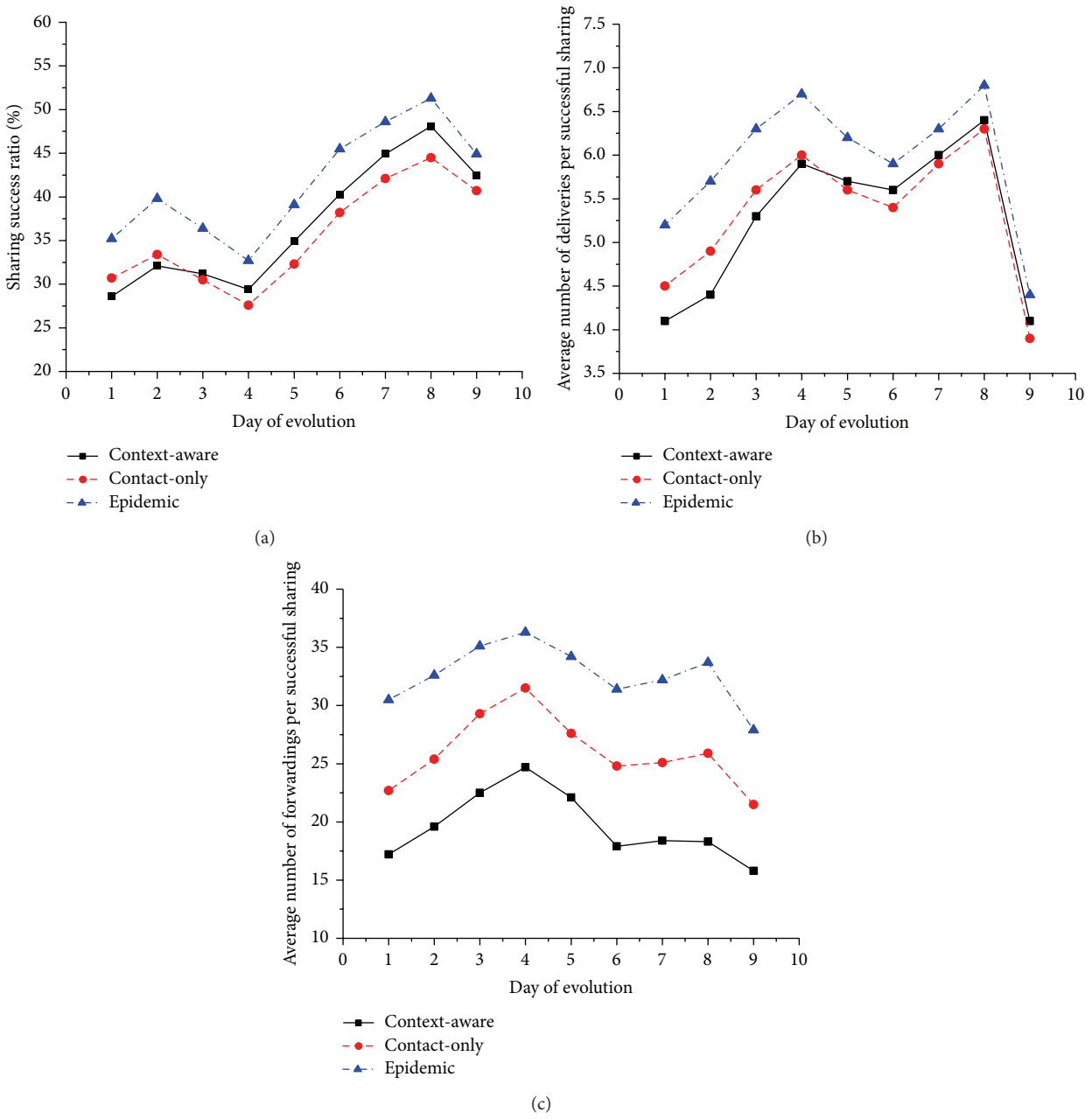


FIGURE 12: Performance of event sharing with daily update (influence lifetime $T_l = 5$, link weight threshold $w^* = 2$, and error threshold $E_f = 0.006$).

communities has not been fully revealed. Actually, the performance of using the context-aware community structure 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. For instance, based on the mobility profiles in Figure 2, let us add the mobility profile of Emma that she visits the library from 5:30 PM to 6:30 PM and encounters Chris and Dave at 7 PM. Assume Emma has the influence lifetime $T_l = 5$ hours, so she can influence library events to Chris and Dave at 7 PM. Then, when Chris encounters Alice at 7:30 PM, he can further influence the library events to Alice. Similarly, Dave can also influence the library events to Bob when they encounter at 8 PM. For the library, there exists two communities: one with members Emma, Chris, and Alice and another with members Emma, Dave, and Bob. If these mobility patterns repeat every day, then these two communities become more and more stable (i.e., links inside communities become stronger), resulting in Chris usually influences the library events to Alice, while Dave usually influences the library events to Bob. But assume, at certain days, Bob occasionally encounters Chris right after encountering Dave. In this case, Bob only needs to be influenced by Dave within their community regarding

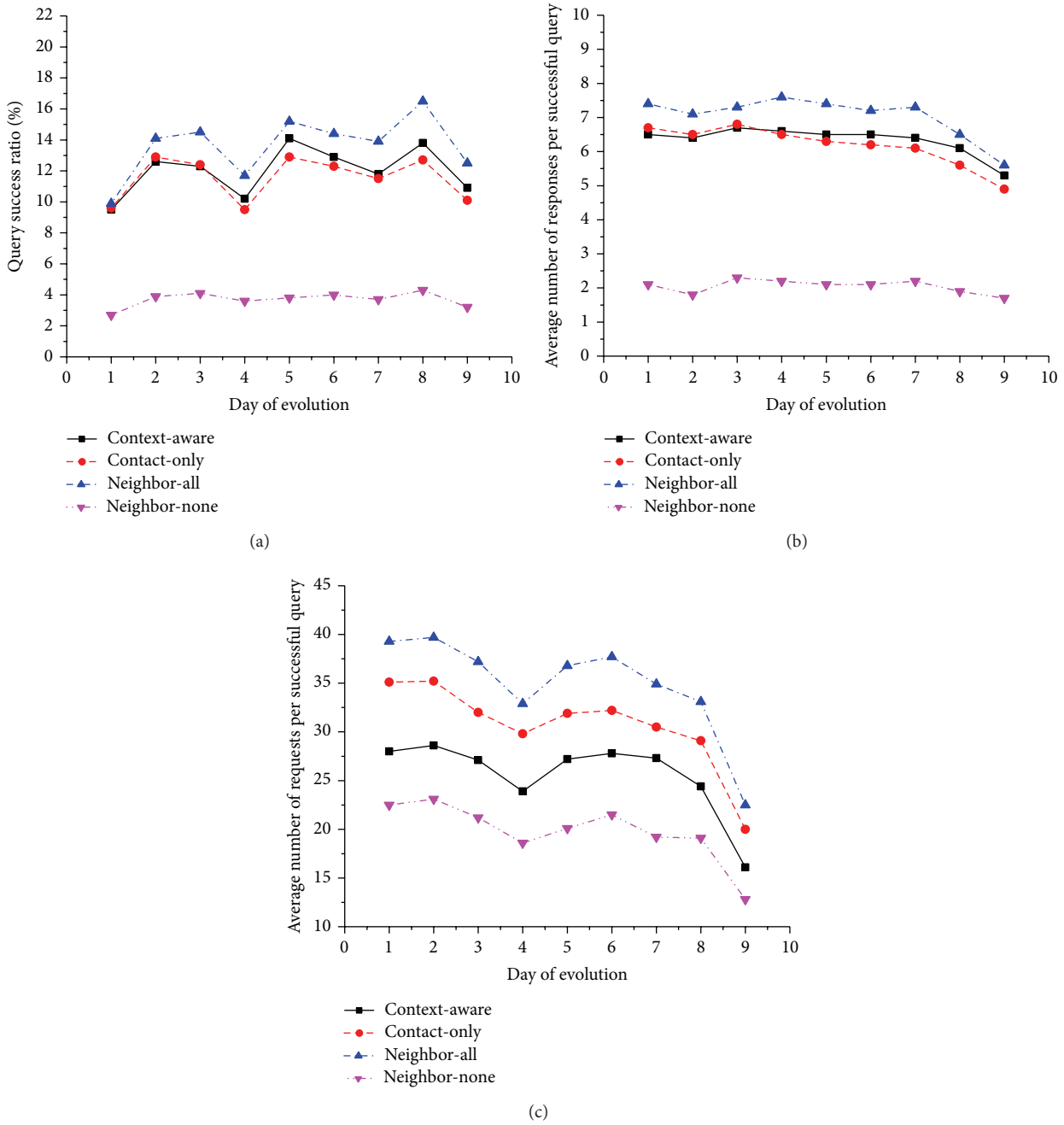


FIGURE 13: Performance of event query with daily update (influence lifetime $T_l = 5$, link weight threshold $w^* = 2$, and error threshold $E_f = 0.006$).

the library. There is no need for Bob to be influenced again with the same library events by Chris, so the unnecessary cost is avoided.

8. Conclusions

In this paper, we have first integrated space and time contexts into user contacts to construct an influence graph. We have then constructed the basic context-aware community structure using the influence graph and 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 being 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.

Notations

k: Size of a complete subgraph
w^*: Link weight threshold
f^*: Link fraction
f^0: Original link fraction
E_f: Error threshold of link fraction
x: Number of link updates
T_l: Influence lifetime.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work 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*, vol. 21, no. 2, pp. 224–240, 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 *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*, pp. 1039–1048, 2010.

[3] G. Palla, A.-L. Barabási, and T. Vicsek, "Quantifying social group evolution," *Nature*, vol. 446, no. 7136, pp. 664–667, 2007.

[4] Y.-R. Lin, H. Sundaram, Y. Chi, S. Zhu, and B. L. Tseng, "Facetnet: a framework for analyzing communities and their evolutions in dynamic networks," in *Proceedings of the 17th International Conference on World Wide Web (WWW '08)*, pp. 685–694, Beijing, China, April 2008.

[5] Y. Sun, J. Tang, J. Han, M. Gupta, and B. Zhao, "Community evolution detection in dynamic heterogeneous information networks," in *Proceedings of the 8th Workshop on Mining and Learning with Graphs (MLG '10)*, pp. 137–146, July 2010.

[6] S. Asur, S. Parthasarathy, and D. Ucar, "An event-based framework for characterizing the evolutionary behavior of interaction graphs," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)*, pp. 913–921, 2007.

[7] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *Proceedings of the International Conference on Advances in Social Network Analysis and Mining (ASONAM '10)*, pp. 176–183, August 2010.

[8] P. Brodka, S. Saganowski, and P. Kazienko, "GED: the method for group evolution discovery in social networks," *Social Network Analysis and Mining*, vol. 3, no. 1, pp. 1–14, 2012.

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

[10] N. P. Nguyen, T. N. Dinh, S. Tokala, and M. T. Thai, "Overlapping communities in dynamic networks: their detection and mobile applications," in *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking (MobiCom '11)*, pp. 85–95, September 2011.

[11] A. Lancichinetti and S. Fortunato, "Community detection algorithms: a comparative analysis," *Physical Review E*, vol. 80, no. 5, Article ID 056117, 2009.

[12] A.-K. Pietiläinen and C. Diot, "Dissemination in opportunistic social networks: the role of temporal communities," in *Proceedings of the 13th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '12)*, pp. 165–174, June 2012.

[13] T. Hossmann, T. Spyropoulos, and F. Legendre, "Putting contacts into context: mobility modeling beyond inter-contact times," in *Proceedings of the 12th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '11)*, 2011.

[14] D. Zhang, Z. Wang, B. Guo, X. Zhou, and V. Raychoudhury, "A dynamic community creation mechanism in opportunistic mobile social networks," in *Proceedings of the IEEE International Conference on Social Computing (SocialCom '11)*, pp. 509–514, October 2011.

[15] P. Hui, E. Yoneki, S. Chan, and J. Crowcroft, "Distributed community detection in delay tolerant networks," in *Proceedings of the 2nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture (MobiArch '07)*, 2007.

[16] E. G. Boix, A. L. Carreton, C. Scholliers, T. van Cutsem, W. de Meuter, and T. D'Hondt, "Flocks: Enabling dynamic group interactions in mobile social networking applications," in *Proceedings of the 26th Annual ACM Symposium on Applied Computing (SAC '11)*, pp. 425–432, March 2011.

[17] R. Lubke, D. Schuster, and A. Schill, "MobilisGroups: location-based group formation in Mobile Social Networks," in *Proceedings of the 9th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM '11)*, pp. 502–507, March 2011.

[18] L. Vu, K. Nahrstedt, and M. Hollick, "Exploiting schelling behavior for improving data accessibility in mobile peer-to-peer networks," in *Proceedings of the 5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services (Mobiquitous '08)*, July 2008.

[19] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Physical Review E*, vol. 74, no. 1, Article ID 016110, 2006.

[20] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[21] G. Palla, I. J. Farkas, P. Pollner, I. Derényi, and T. Vicsek, "Directed network modules," *New Journal of Physics*, vol. 9, article 186, 2007.

[22] UIUC, Uim, 2012, http://crawdad.cs.dartmouth.edu/uiuc/uim/.

[23] L. Vu, Q. Do, and K. Nahrstedt, "Jyotish: a novel framework for constructing predictive model of people movement from joint Wifi/Bluetooth trace," in *9th IEEE International Conference on Pervasive Computing and Communications (PerCom '11)*, pp. 54–62, March 2011.

[24] Strath Nodobo, 2011, http://crawdad.cs.dartmouth.edu/strath/nodobo/.

[25] L. Vu, K. Nahrstedt, S. Retika, and I. Gupta, "Joint bluetooth/wifi scanning framework for characterizing and leveraging people

movement in university campus," in *Proceedings of the13th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '10)*, pp. 257–265, October 2010.

[26] J. Aslam, K. Pelekhov, and D. Rus, "Static and dynamic information organization with star clusters," in *Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM '98)*, pp. 208–217, 1998.

[27] M. Meilă, "Comparing clusterings—an information based distance," *Journal of Multivariate Analysis*, vol. 98, no. 5, pp. 873–895, 2007.

[28] M. Goldberg, S. Kelley, M. Magdon-Ismail, K. Mertsalov, and A. Wallace, "Finding overlapping communities in social networks," in *Proceedings of the 2nd Conference on Social Computation (SocialCom '10)*, pp. 104–113, 2010.

[29] N. Yu and Q. Han, "Context-aware communities and their impact on information influence in mobile social networks," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM '12)*, 2012.

[30] N. Yu and Q. Han, "Context-aware community: integrating contexts with contacts for proximity-based mobile social networking," in *Proceedings of the IEEE 9th International Conference on Distributed Computing in Sensor Systems (DCoSS '13)*, pp. 141–148, Cambridge, Mass, USA, May 2013.