Leveraging Bluetooth Co-location Traces in Group Discovery Algorithms

Daniel Boston[a], Steve Mardenfeld[a], Juan (Susan) Pan[a], Quentin Jones[b], Adriana Iamnitchi[c], Cristian Borcea[a]

[a] *Computer Science Department, New Jersey Institute of Technology, Newark, NJ, USA*
[b] *Information Systems Department, New Jersey Institute of Technology, Newark, NJ, USA*
[c] *Department of Computer Science and Engineering, University of South Florida, Tampa, FL, USA*

Abstract

Smart phones can collect and share Bluetooth co-location traces to identify ad hoc or semi-permanent social groups, which can enhance recommender systems or allow detection of epidemic events. Group discovery using Bluetooth co-location is practical due to low power consumption, short range, and applicability to decentralization. This paper presents the Group Discovery using Co-location traces (GDC) and Decentralized GDC (DGDC) algorithms, which leverage user meeting frequency and duration to accurately detect groups. GDC and DGDC are validated on one month of data collected from 141 smart phones carried by students on our campus, and by comparison against ground-truth groups.

Keywords: Mobile social computing, group discovery, co-location traces, smart phones, distributed computing

1. Introduction

The incorporation of social information into online and mobile applications or services has become mainstream in the past several years. However, collecting social networking information is generally application-dependent and limited to on-line activities. As such, a large amount of social information that results from face-to-face interactions is not captured. This is especially true for informal groups, which do not advertise their meetings on-line. Examples of such groups include a student study group, faculty who routinely have lunch together, or coworkers who sometimes play golf.

Capturing information about social groups formed through face-to-face interactions could be used in several types of services. Recommender services can benefit from a significant amount of additional social information to provide geo-social group recommendations such as meeting places or events of interest based on common user preferences [1]. Systems such as Mobius [2] can enforce socially-aware access control: multimedia content generated at a group meeting is only shared with group members, including those who were not present. The information about groups could also be leveraged in systems such as RoamWare [3] for mobile computer supported cooperative work. Forwarding in delay-tolerant ad hoc networks can be made socially aware by selecting a next hop device that belongs to a person who shares a group with the destination; in this way, the next hop has a higher probability to meet the destination [4]. Finally, disease epidemic events can be monitored [5] and potentially thwarted using social information, by alerting at-risk individuals before they come into contact with a source of infection. As well, the same information can be used to accelerate the delivery of treatment to those already suffering. Beyond biologic epidemics, the spread of mobile computer viruses could be monitored, contained, and reversed with the effective use of social group data.

Email addresses: djb38@njit.edu (Daniel Boston), sm424@njit.edu (Steve Mardenfeld), jp238@njit.edu (Juan (Susan) Pan), qjones@njit.edu (Quentin Jones), anda@cse.usf.edu (Adriana Iamnitchi), borcea@cs.njit.edu (Cristian Borcea)

One way to identify this type of social group is to leverage information collected automatically from smart phones carried by mobile users, such as location or co-location. Using this information, a group can be defined as a relatively small set of users who spend a significant amount of time together and meet for a significant number of times [6]. [1] Discovering such groups, however, is difficult: (i) group members do not necessarily attend all group meetings, (ii) guests or people who pass by the meeting location can appear to be part of groups, (iii) group members spend different amounts of time at meetings, (iv) the collected data is incomplete due to sampling frequency and mobility, and (v) users may collect different data for the same group meeting.

Our GPI algorithm [7] used location to identify, with high accuracy and low false positives, groups and their associated places. However, there are several issues which prompted the need for a different algorithm. First, GPI requires a localization system on every mobile device, which is not always available (especially indoors). Second, many users are often reluctant to share location traces for any length of time due to privacy concerns. Even anonymous location traces are vulnerable to user identification through data mining techniques, which can lead to user tracking and home identification [8]. Third, localization systems running on phones (e.g., GPS, Place Lab [9]) consume a significant amount of battery power. Finally, the accuracy of localization systems can vary significantly across different places.

This paper presents the Group Discovery using Co-location traces (GDC) and Decentralized GDC (DGDC) algorithms, which use Bluetooth co-location traces and policy based data sharing to identify groups. A co-location trace for a user is a set of records of other users who are within a certain proximity at the same time. GDC leverages the Bluetooth discovery protocol to collect these traces. GDC is practical and achieves good efficiency for several reasons. Unlike localization systems, Bluetooth is available on nearly all mobile phones and can work indoors. Although some mobile development platforms restrict use of Bluetooth either by limiting discovery mode or by requiring regular user feedback, solutions do exist such as in the Android 4.0 API [10], where developers can request unlimited Bluetooth discovery and educated users could consent. Sharing co-location information, instead of absolute location, may be a participation incentive for those users with concerns about their location being tracked. DGDC builds on this incentive by further reducing sharing to those other users with whom you already interact, while still providing reasonable group detection. Bluetooth consumes much less power than GPS and WiFi, while helping with the accuracy of the algorithm due to its short transmission range (i.e., 10m). Finally, while not guaranteeing face-to-face interactions, proximity-based interactions captured by Bluetooth combined with GDC's rejection of social tie transitivity leads to social interaction data similar to face-to-face encounters.

DGDC has some important benefits over GDC. As it is entirely decentralized, privacy concerns are specifically addressed. Users can control the extent of their sharing through simple (or complex) policies reflecting their willingness to share information. DGDC data collection could also be selectively disabled to prevent undesirable sharing, such as on a daily commute or while riding on public transportation. There are some potential drawbacks, as communication costs can be a greater than GDC, and some discovered groups may be incomplete due to perspective isolation. As our evaluation will show, DGDC achieves reasonable results over a variety of policy choices.

Well-known graph algorithms, such as K-Clique [11] and WNA [12], could be employed to detect groups based on co-location. They work by inserting an edge in the graph between any two users who have been around each other for a certain amount of time. However, since these algorithms use only pair-wise information, there is no guarantee that their detected groups spent any time together. Furthermore, important parameters that can be used to classify groups, such as group meeting frequency and total group meeting time, are lost. Finally, K-Clique does not allow for weighted edges, which leads to lower group detection accuracy, and WNA does not work for overlapping groups, thus missing many groups.

This paper makes the following contributions:

- it presents GDC, a practical algorithm that leverages Bluetooth co-location traces to accurately identify

[1] These solutions discover co-located people. Additional sensing (e.g. voice recognition) or user confirmation is required to determine if the co-located users did indeed interact. An alternative solution for detecting face-to-face interaction is the use of RFIDs [5], but this solution requires people to carry an artifact (e.g., badge) with an embedded RFID; furthermore, it does not scale well as it requires RFID readers everywhere.

social groups. GDC preserves the group meeting times and aggregate meeting frequency with each group, which can be used to compare, categorize, and rank groups.

- it describes DGDC, a distributed version of GDC involving data sharing only among other people who have actually been met, while still achieving reasonably similar group detection characteristics.
- it validates GDC with one-month of data collected from 141 smart phones carried by students on our campus. Additionally, GDC was tested on the Reality Mining dataset [13].
- it compares GDC against K-Clique on our dataset by asking the users to participate in a survey deployed on Facebook. Using a Likert scale, we observed that GDC performed well: GDC-discovered groups received 30% higher ratings than K-Clique's groups.
- it evaluates GDC and DGDC against a careful reconstruction of "true groups" created using a novel tool (TIE [14]) and deep investigation of the source data, a reconstruction named the ground truth data set (GTS). We show that GDC and DGDC outperform K-Clique in both accuracy and terseness.

The rest of this paper is organized as follows. Section 2 describes the GDC algorithm. Section 3 presents in detail DGDC, its policies and benefits. Section 4 gives a three part analysis of GDC, DGDC, and similar algorithms through a user study, a relative comparison, and comparison against ground truth. Section 5 discusses the related work, and Section 6 concludes the paper.

2. GDC Algorithm

2.1. Input Data

The GDC algorithm discovers social groups from a set of co-location traces. These traces are collected by a program running on every smart phone, which invokes the Bluetooth discovery protocol to periodically (1-3 minutes) query all nearby Bluetooth devices (i.e., in a 10m transmission range) for their MAC addresses. These data are typically uploaded to a server at certain time intervals. Every instance of a detected Bluetooth device is stored as a Bluetooth record, which includes the MAC address of the initiator, the MAC address of the discovered device, and a timestamp. Records including unknown devices (i.e., not registered with the data collection module) are discarded.

2.2. Definitions

A *Bluetooth record* consists of a pair of co-located users and a timestamp (i.e., the time when they were co-located). If a smart phone detects N users in one discovery query, then N Bluetooth records will be created, one for each device.

A *Bluetooth trace* is the collection of all Bluetooth records for a specific user resulting from queries performed by the smart phone carried by that user.

A *pair-wise meeting record* (or simply pair-wise meeting) is a collection of contiguous Bluetooth records for the same pair of users, where contiguous means each pair of records has at most a limited time gap controlled by the meeting granularity (MG) parameter. This parameter is related to the Bluetooth discovery frequency, and is explored further in Section 2.3's discussion of GDC's first phase.

A *group meeting* involves a set of at least three users, and it requires that all possible user pairs in the set have pair-wise meetings at the same time. Implicitly, this means that all the participants at a meeting must be in the Bluetooth transmission range of each other.

A *cluster* is a set of users, who have one or multiple meetings together.

A *group* is a cluster of users who spend a significant amount of time together over a series of meetings. Not all members of a group have to be present at every meeting, but they must be present at a significant number of meetings. A set of groups can be overlapping and thus share members, but each group must be notably different from the others in terms of total time spent together (and meeting times). A cluster becomes a group if it meets the minimum group time (MGT) and minimum group meeting frequency (MGMF) parameters and does not significantly overlap other groups, as controlled by the group time parameter (GTP). GDC's phase three and four explain in more detail these parameters, found in the next section.

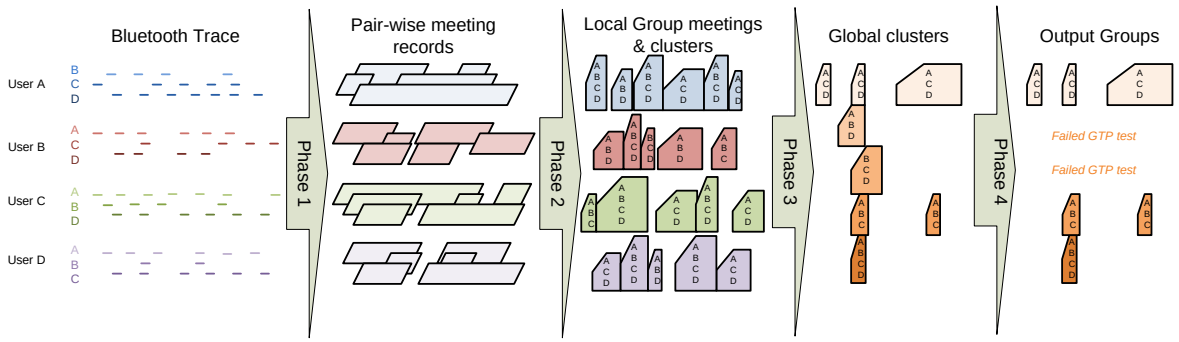


Figure 1: A visual overview input and output diagram of the four GDC phases.

2.3. GDC Description

Given a set of pair-wise co-location records, GDC identifies groups of users. Naturally, groups may share users, and consequently, groups can be overlapping. However, the groups outputted by GDC are notably different from each other, whether in terms of members or total time spent together. Not only does GDC find all groups that spend at least a specified minimum amount of time together for at least a specified number of meetings, but these parameters are maintained in the final output to allow groups to be compared, categorized, and ranked.

GDC discovers these groups in four different phases, each dependent on the results from the previous phase as illustrated in Figure 1. The first phase transforms individual Bluetooth trace records into meeting records, which detail the start and end times of individual meetings between two users. Reference Figure 1 to see how a set of Bluetooth records becomes a set of pair-wise meetings. The second phase takes these meeting records and calculates the time spent by all permutations of users who appeared together over different meetings. This phase results in a raw list of potential user clusters; note in the figure how the set of meetings records represented as horizontal boxes become vertical boxes with the user names embedded to represent a user's local set of clusters. The third phase compares the views of different users and removes some of the clusters that do not appear in all the cluster members' views. In Figure 1, only those clusters shared and overlapping across all user perspectives appear in the phase three output. The fourth phase takes the global list of clusters and identifies the final groups by eliminating subgroups of little significance. In the figure, a few clusters are discarded by phase four as they do not contribute significant additional information.

In the first phase, GDC goes linearly through all the Bluetooth records to identify pair-wise meetings between users and the durations of these meetings. Each meeting record represents a collapsed time-frame for several "chained" Bluetooth records. This chaining is reflected in pseudo-code lines 4 through 18, where if the pair-wise users are the same, the end-time of the meeting is updated until a new user pair is encountered. In this way, many one-time encounters (e.g., people passing each other) are discarded, as no chaining occurs. To make sure new chained records represent a single meeting, the meeting granularity (MG) parameter is used.It represents the maximum amount of time that two Bluetooth records can be apart and still be considered part of the same meeting. Ideally, this parameter should be large enough such that missing a single Bluetooth record (e.g., the user stepped out of the meeting to take a phone call) would not result in two different meetings, but small enough to capture real changes in group structure. In our initial evaluation, we set MG to 15 minutes based on empirical observations of the clusters created by GDC's second phase. At this value, the number of additional clusters started to level off. One drawback of this technique is it tunes the algorithm parameter using an intermediate and hidden output of the algorithm, instead of tuning the algorithm based on characteristics of the input data. Additionally, tuning in this way requires repeatedly running the first two phases of GDC, which is a significant time cost to tune a single parameter. An alternate approach that can be applied dynamically to auto-tune GDC or DGDC is based

Phase 1 Creating Pair-wise Meeting Records

```
Sort union of Bluetooth records r_i by user, userwith, time
endtime = 0, starttime = r_0.time
for all r_i in records do
    if (r_i.user == r_{i-1}.user) AND (r_i.userwith == r_{i-1}.userwith) then
        if (r_i.time − r_{i-1}.time > MG) then
            if (endtime ≠ 0) then
                addmeeting(r_{i-1}.user.meetings, r_{i-1}.userwith, starttime, endtime)
            end if
            starttime = r_i.time, endtime = 0
        else
            endtime = r_i.time
        end if
    else
        if endtime ≠ 0 then
            addmeeting(r_{i-1}.user.meetings, r_{i-1}.userwith, starttime, endtime)
        end if
        starttime = r_i.time, endtime = 0
    end if
end for
if endtime ≠ 0 then
    addmeeting(r_i.user.meetings, r_i.userwith, starttime, endtime)
end if
```

on the distribution of next-sightings of the same user. Upon review we discovered that 95% of same user next-sightings occur within 29 minutes of first sighting. Using a value of 29 minutes for MG results in a more complete discovery of meetings. Many groups which were fragmented into short meetings by an MG of 15 minutes became noticeably more continuous with an MG of 29 minutes. For the purposes of this paper, figures using the newer 29 minute meeting granularity parameter will be marked as such.

Two users can have different views of the same meeting due to the randomness of the periodic Bluetooth discovery, potential wireless collisions, and the varying distance between users. GDC merges the user views of the same meeting by taking the union of their individual views.

Phase 2 Creating User Clusters

```
for all x in users do
    lasttime = 0, currentwith = new minheap([user, endtime] keyed on endtime)
    for all m in x.meetings do
        while (currentwith[0].endtime ≤ m.starttime) do
            for all cluster c in allCombinations(x, allUsers(currentwith)) do
                c.totaltime += currentwith[0].endtime - lasttime
                c.frequency += 1
                updateClusters(x.clusters, c)
            end for
            lasttime = currentwith[0].endtime
            remove(currentwith[0])
        end while
        for all cluster c in allCombinations(x, allUsers(currentwith)) do
            c.totaltime += m.starttime - lasttime
            c.frequency += 1
            updateClusters(x.clusters, c)
        end for
        currentwith.add([m.userwith, m.endtime])
        lasttime = m.starttime
    end for
end for
```

The second phase analyzes the correlation between pair-wise meeting records to identify all user clusters formed during the entire period analyzed. It is necessary to consider all possible combinations of users at this stage because there is not enough information to tell if overlapping clusters should be put together in

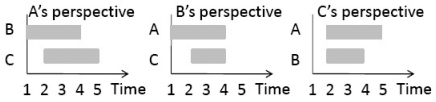


Figure 2: Co-location for three users according to each user's perspective

one group or kept separate. This information will be derived later based on the total meeting time and the number of meetings for each cluster. For example, in Figure 2, according to A's perspective, GDC selects the following clusters: (AB:3, AC:3, ABC:2). The same figure illustrates that users may have different perspectives of their clusters. According to B, the results are (AB:3, BC:2, ABC:2). This can be due to the fact that users separated by a distance close to the transmission range may have fewer records of each other than a users located between them.

For each user, GDC goes linearly through its meeting recordsand constructs its clusters, calculating how much time she spent and how many times she met with every permutation of users clustered together for a meeting. The min-heap in the pseudo-code is used to organize all the pair-wise meetings that coincide in time (see lines 18-19). Once found, all permutations of coincident users ordered by when they left the meeting are used to update the permuted cluster's time spent together and frequency of meeting (see the loops starting at lines 5 and 13). Note that at the end of this phase, each user has a localized version of each cluster as users can have different perspectives on group meetings.

Phase 3 Creating Global Clusters

```
 1: globalclusters = []
 2: for all x in users do
 3:    for all c in x.clusters do
 4:       if (minTime(allUsers(c)) > MGT) AND (minFreq(allUsers(c)) > MGMF) then
 5:          c.totaltime = minTime(allUsers(c).totaltime)
 6:          c.frequency = minFreq(allUsers(c))
 7:          globalclusters.add(c)
 8:       else
 9:          for all user y in c do
10:             delete(c, y.clusters)
11:          end for
12:       end if
13:    end for
14: end for
```

The third phase consolidates each user's perspective of clusters into a global view of all clusters.Additionally, it eliminates all clusters that met for a total time less than a threshold called minimum group time (MGT) and have a meeting frequency less than the minimum group meeting frequency (MGMF). This is shown in lines 8-11. By including MGMF in the algorithm, we can relax MGT to provide better identification accuracy for groups with frequent but brief encounters. We take a conservative approach and set the meeting time and meeting frequency for a cluster to the minimum values across all members.

One significant optimization of GDC can be achieved here, at the boundary between phase two and three. Phase two already touches every user's perspective on clusters; after finding the clusters of a particular user, we can immediately compare them against a "global" min-heap that keeps track of the current minimum meeting time perspectives on each unique cluster. In this way Phase three can traverse a consolidated list of clusters contributed by all users which already represents the minimum perspective. This removes the need to always re-traverse each user's distinct perspective on clusters, representing a significant running time improvement. Ultimately, Phase three can simply verify that the minimum perspective on each cluster in the global heap meets the thresholds, and passing that test, check that each member of the cluster has a record of the cluster.

Although each cluster should appear in the cluster list of each of its members, this is not always the case because cluster members might not have attended all meetings for that cluster. GDC removes a cluster

that does not appear in the lists of all its members (see lines 9 - 11). Unfortunately, this means that groups whose members are located within a range larger than the Bluetooth transmission range, such as a large auditorium, are not detected. However, many times, by including groups "linked" through one or a few users, GDC would erroneously merge different groups that meet at the same location. Our current version may miss a few large groups, outputted as overlapping subgroups, but it guarantees that the identified groups exist in reality (in the sense that their members have spent time together). One possible optimization is to check the group meeting times at the end of GDC's execution, and merge groups that meet with a certain frequency at the same time and share a significant percentage of their users.

Phase 4 Selecting the User Groups

```
 1: Sort globalclusters by descending cluster length
 2: finalclusters = [], addcluster = true
 3: for all i in globalcluster do
 4:     addcluster = true
 5:     for all j in finalcluster do
 6:         if (allUsers(i) is subset of allUsers(j)) AND (i.totaltime*GTP < j.totaltime) then
 7:             addcluster = false: break
 8:         end if
 9:     end for
10:     if addcluster then
11:         finalclusters.add(i)
12:     end if
13: end for
```

The final phase determines whether each of the global clusters should be considered a significant group or a subgroup of another significant group. Such a check is necessary due to the second phase of the algorithm, in which GDC generates all the possible cluster permutations, guaranteeing that two clusters are either disjoint or one includes the other. Additionally, this last stage is necessary for two reasons. First, most people may not arrive at or leave from a meeting at the exact same time. Second, many users may miss a few group meetings. GDC must tolerate both these issues and remove insignificant subgroups as they are included in other groups.

However, GDC must also allow for subgroups to be considered as standalone groups if their members meet much more frequently than any of the other members of a larger group. For example, a group of students who hangout together frequently could play weekly basketball games with other students. In this case, both groups should be considered. GDC uses the total time a group spent together as a weight to determine whether a group should be kept or removed. Specifically, a group A is removed (i.e., it is a subgroup) if all its members belong to another group B and B's total time is more than half A's total time (the GTP parameter in line 6 is set to 0.5). This parameter practically denotes how much more time the subgroup members must spend together compared to all the group members in order to be considered a standalone group. By stepping through each candidate group in descending size order, GDC tests smaller groups against all accepted larger groups (lines 5 through 9), ensuring that all relevant subgroups and disjoint groups are preserved and all trivial subgroups are discarded. The output of this phase, a set called "finalclusters" in the pseudo-code, is the final output of GDC.

One particularly challenging edge-case involving larger groups is where two or three members of a group see a majority of the members, but not all members. By example, consider two groups, both meeting from 1pm to 5pm. Group one has members A, B, C, and D. Group two has members B, C, D, and E. Because A and E do not see each other between 1pm and 5pm in the pair-wise data, GDC produces multiple groups. This situation arises most commonly in larger groups, and is a more complicated case than simple subset or superset detection. While techniques can be used to address simple instances of this complication, a general solution is more difficult to derive. Consequently, GDC does not address this situation directly, instead focusing on correctness and accuracy of detecting smaller groups.

2.4. Computational Complexity

The complexity of GDC is $O(R \times 2^L)$, where R is the total number of Bluetooth records and L is the maximum number of users in a group. The assumption is that $R >> N$ (the number of users) and $MR = R/const$ (the number of meeting records). This complexity is due to the second phase of the algorithm, which has a complexity of $O(MR \times 2^L)$. Although this complexity is exponential in L, *the value of L is relatively low* because of the limited number of users who can be found together at any one time in a transmission range of 10m. Additionally, even with longer record times, GDC's rejection of transitivity (all group members must be mutually co-present) constrains L to the number of people within a single 10 meter radius. Some more recently explored algorithm optimizations could also restrict this exponential complexity by replacing the enumeration of group member combinations with a carefully constructed search against known groups. Initial tests of this algorithm construction resulted in running times nearly a tenth that of combinatorial GDC, but will require more testing to ensure equivalence. For our dataset, the largest output group GDC creates has 13 members, and the average is 4 members. During phase two, the largest cluster considered has 19 members, but those larger clusters are not preserved as output groups by phase three and four, due to a failure to meet MGMT and MGF parameter constraints.

Although our dataset is limited, it provides an excellent baseline for real GDC deployments. The design of GDC and the use of Bluetooth act to limit group sizes even for longer recording times. As well, applied models of GDC and DGDC will likely consider only limited periods of time, not the full record history. Groups would be built against a month or two of time, and the results compared and normalized against prior months. As well, infrequently observed strangers would be discarded due to group time and frequency requirements. These considerations would help in restricting overall complexity.

3. Distributed GDC

So far, GDC was assumed to run on data collected at a central location. By modifying GDC's design, the algorithm can run in a distributed manner such that each mobile phone stores co-location traces and calculates groups for its owner without relying on a centralized server. Phones can exchange data directly with other phones of interest. This section presents results that quantify the privacy benefits and the cost in terms of overhead of different sharing policies in DGDC.

3.1. Benefits

The main benefit of DGDC is better privacy for users, who are protected from sharing data with a centralized server, which can be viewed as a "big brother". Each user must share data with other users, but these are the same users who have been recorded in her proximity. In general, this is not private information as the users physically saw each other. Two users who do not have any co-location records will never directly exchange information. Depending on the application, final social groups might be pooled together; however, co-location traces still remain local, thus specific details, such as time of encounter, can remain private. Interactions with users who are not in any final social groups will remain confidential as well.

With GDC, all users must place complete trust in the central server, and any single compromise will result in total compromise of all users' privacy. In DGDC's distributed model, users place trust in those they meet that satisfy their privacy policy. While the overall number of entities who, if compromised, would result in a privacy loss is greatly increased, the scope of each loss is limited. Only if all users across the entire network of DGDC participants were compromised would a loss occur equivalent to a privacy failure at the single central server. Arguably, the potential of catastrophic privacy loss is reduced in the distributed model.

Other benefits of a distributed version are resilience and flexibility. DGDC is more resilient as there will be no central point of failure: both data and processing are distributed. It is also more flexible as it allows users to run the algorithm at different time intervals according to their needs. This is important because groups change over time, but at different rates for individual users.

3.2. Changes

In the centralized version of GDC, the first phase simply takes a union of all Bluetooth records, creating shared perspectives among users with mutual co-location data. Having shared perspectives is vital because of errors associated with recording co-location via Bluetooth. Thus, the first phase of DGDC must involve Bluetooth record swapping between co-located users such that each user can resolve most of the errors their local data would otherwise contain. This information exchange is achieved via the Internet or opportunistically in an ad hoc manner. In a different paper [15], we proved that data exchanges in ad hoc manner over Bluetooth work for walking-speed mobility. The same paper demonstrated that thousands of exchanges can be done within the battery power limits assuming the phones are charged at night.

In the centralized version, clock synchronization between mobile phones was not necessary as the server associated server-side timestamps with each Bluetooth record. In the distributed version, mobile phones have to timestamp their own records. However, we can rely on the cellular network, which synchronizes the clocks on all phones connected to the network.

Using intelligent policies, it is possible to exchange some limited sets of Bluetooth records between frequently co-located users. To enable this exchange, users should record mutual co-location through a "seen with" list for each co-located user; this is possible since more than one user is often seen in a single scan. Ultimately, this list can be used to control which records are shared. The advantage of this method is that the privacy goals of the system can be reflected through policies.

These policies can help reduce undesired sharing of co-location traces with infrequently observed strangers. In cases where automatic policies similar to the ones proposed here would still result in undesired sharing, more configurable policies could be established that disable sharing for predefined time periods (such as during bus rides). Additionally, implementations could present users with early estimates of groups and allow the removal of groups without clear social meaning. Such controls might limit the effectiveness of certain applications of DGDC, such as group recommendation engines, but could increase user adoption.

Each user develops her own perspective on groups (the quality of these perspectives is presented in Section 3.3), with information exchange being controlled early in the process. As a simple example, when a user, Alice, initiates a request for Bluetooth records with another user, Bob, she can request data on all the users in the "seen with" list she has on Bob. Bob, in turn, could either simply fulfill her request, or only send her data on users in his "seen with" list on Alice. Through this simple message exchange step, users will aggregate a representative subset of Bluetooth records. The remainder of GDC runs as previously described.

3.3. Results

To evaluate DGDC, we developed a system to replay the collected Bluetooth data while emulating message exchange between users. Each user is represented as a thread. The system allows a number of policies controlling when message exchanges occur and what data to send. While the initial set of policies we developed involved a number of decisions about when and how much data to share, after testing we determined that the following represents a minimally viable set of policies for use with DGDC in everyday use. They represent a reasonable trade off between privacy concerns and accuracy of DGDC results, discussed further in the evaluation section.

There are two aspects of every privacy policy; what to do when a user – known or unknown – is encountered, and how to respond when queried by a user. While a great number of policies can be contrasted, it is clearer to distill them down to a simple set. Consider Table 1, where the fundamental policy construction is divided into four distinct actions or responses.

We can define "Well Known" as a user that passes the "seen with" test described previously. Anyone who does not yet pass the test is "Not Well Known"; this includes people who were not previously observed and those who have been observed infrequently (below the test threshold). Given that definition and the decision matrix, a great variety of policies can be developed to specify what related data updates as mentioned in the table are, or how to determine what data matches a request.

The phrase "related data updates" refers to the query made to a user who was seen. Instead of listing all the kinds of queries possible under this model, we can distill related data updates to two general types of queries:

Table 1: DGDC Basic Policy Decision Matrix

	Well Known	**Not Well Known**
Encounter	Ask for related data updates	Record encounter; ask for nothing
Queried By	Respond with data that matches request	Ignore

Ask for a little involves querying user A only for information on interactions involving user A and other users seen with user A.

Ask for a lot involves querying user A for information on interactions involving all users encountered, regardless of user A's presence in those encounters.

Naturally, the second query potentially reveals more about the behavior of the querying user, but as the nature of the policy in use is not transmitted in the request, this can be determined only by inference.

Likewise, data that matches requests can be distilled to two responses:

Give a little involves sending to user A a subset of requested information. User A may request data on a large number of users, but the responder only returns information on interactions that were observed as mutual between user A, each requested user, and the responder.

Give a lot just attempts to fulfill user A's request, sending whatever information is available about interactions involving the users queried.

By skipping the gradient of policy decisions in between little and lot, we can gain an overall picture of the impact policy has on privacy, transmission cost, and group detection accuracy (described in section 4) without getting bogged down with fine-grained policy distinctions.

Ultimately, we can derive four fundamental policies involving the interaction of the above definitions. They are as follows:

Policy Lele Users following this policy ask for a little and give only a little.

Policy Lelo Users on this policy ask for little but give a lot.

Policy Lole These users ask for a lot but give only a little.

Policy Lolo Adherents to this policy ask for a lot and give a lot.

To evaluate DGDC when run with these policies, it is helpful to define a few simple metrics.

Self Exposure Risk Index (SERI) is a normalized measure between 0 and 100 that helps quantify the amount of information about herself revealed to other people during exchanges against our dataset using a particular policy. 100 represents worst case, where an individual reveals, as a policy consequence, all records about themselves to all participating users. By definition, GDC has a SERI of 100, as all members reveal all information about themselves to some central authority. This is a measure computed per user, and for a set of users the average of their SERI scores. In computing SERI, we define COR as the sum of the number of records about themselves sent by users to other users under a particular policy. NREC is the sum of social interactions between all users, and NOP is the number of people in the system. The denominator represents the worst-case, where all users reveals all records about themselves to all other users.

$$SERI = 100 * \frac{COR}{(NOP - 1) * NREC}$$

Table 2: DGDC Policy Comparison based on SERI, XERI, and TCI)

	SERI (Max)	**XERI (Max)**	**TCI (Max)**
Local-Only	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
DGDC Lele	2.46 (8.31)	2.45 (9.20)	2.46 (6.35)
DGDC Lelo	3.03 (8.37)	3.01 (9.52)	2.82 (6.39)
DGDC Lole	2.63 (8.31)	2.61 (9.32)	4.57 (8.35)
DGDC Lolo	4.42 (9.46)	4.39 (11.71)	5.66 (9.97)
GDC	100.0 (100.0)	100.0 (100.0)	0.31 (0.38)

External Exposure Risk Index (XERI) is a normalized measure between 0 and 100 that quantifies the amount of information revealed about a particular person by other users. This is a consequence of pair-wise information exchange; every time a user reveals something about themselves (a self exposure event) they also reveal something about another user (an external exposure event). Again, GDC is defined as having a XERI of 100, as all members reveal everything they know about every other member. XERI is something that cannot be controlled by a user's policy, but is moderated as a consequence of the overall policy use by every participating user. In computing XERI, we define CRM as the sum of the number of records sent by users to other users that reveal something about another user. SREC is the sum of people's social interactions over all users. NOP is as defined previously. The denominator represents the worst-case, where every user reveals all the social interactions they observed of others to all users.

$$XERI = 100 * \frac{CRM}{NOP * SREC}$$

Transmission Cost Index (TCI) is a normalized measure between 0 and 100 that quantifies the cost of transmitting records between users. 0 indicates no transmission activity (no communication at all), while 100 indicates worst case transmission activity (all people ask each person they encounter about everyone every time, and all people transmit all their records to everyone). We count transmissions on a field basis; a field is a discrete piece of data such as a timestamp or an identifier. This construction weighs record transfer more heavily (and more appropriately) than information query (asking for records). Merely counting transmission events does not truly capture transmission cost, which led to the construction of this metric. To compute TCI, we define the cost of sending a record as RC, in our calculations set to 4, as 4 fields are sent in each record (three IDs and one timestamp). The cost to query someone about all people, named QC, is $1 + NOP$ where NOP is as previously defined. NXM is defined as the count of fields sent. NQM is defined as the count of fields involved in querying about others. Worst case transmission is defined in the denominator as sending all records to all people, and always querying about everyone whenever someone is seen.

$$TCI = 100 * \frac{NXM + NQM}{NREC * RC * (NOP - 1) + NREC * QC}$$

Each of these metrics is computed per-user, and averages are used to represent the SERI, XERI, and TCI of a particular algorithm or policy.

With these definitions in mind, we can directly compare policies based on their scores in each Index. Table 2 shows the comparative SERI, XERI, and TCI values for each of the policies described above, as well as Local-Only and GDC, for baseline. Note that Local-Only describes running the GDC algorithm without data sharing, instead exclusively using data collected locally by a single device. Also for convenience, the maximum values of each index experienced by any single person is shown in parenthesis next to the population average.

This table shows the contrast between GDC and DGDC in terms of privacy and communication costs. Even the most permissive and greedy policy, Policy Lolo, still shows significant privacy improvement in both average and max cases. As expected, this does come with significantly increased communication costs, although that can be moderated by choosing a less permissive policy, such as the Lele policy.

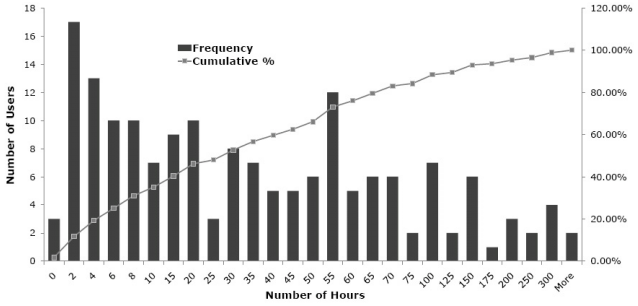


Figure 3: Number of hours of collected data per user

We will further discuss DGDC policy choices and their impact not only on privacy and communication, but also on accuracy in the ground truth evaluation of section 4.4.

4. Experimental Evaluation

The goal of this evaluation was to understand if GDC performs well in real-world settings and to compare GDC's performance against a well-known graph algorithm for community detection, K-Clique [11]. We also wanted to investigate GDC, D-GDC and K-Clique's absolute performance against the ground truth. We constructed the ground truth of our data set through a careful timeline survey using a tool of our own construction called TIE, the Temporal Interaction Explorer [14].

4.1. Data Collection

We collected one month of Bluetooth co-location data for a set of 141 students at NJIT. The study took place on our medium size urban campus, and the subjects were representative of the various majors offered on campus; 75% were undergraduates and 25% were graduates. Also, 28% were women and 72% were men.

We equipped the participants with HTC Windows Mobile 8595 and 8525 phones, which come preloaded with Windows Mobile 5. Under supervision, each participant installed a custom application that quietly recorded the Bluetooth addresses of nearby devices using the Bluetooth discovery protocol. Discovery queries occurred on each device at a random interval between 1 and 3 minutes. The randomness was introduced to minimize the potential delays due to wireless collisions (which could lead to losing records). Each discovery query took approximately 20-30 seconds to finish, which is in line with previous studies [16]. The local records were uploaded to a server periodically.

In newer phones, Bluetooth discovery is better managed, and in some cases it may be difficult to enable the appropriate Bluetooth mode (e.g., discovery). However, in Android 4.0, applications can request indefinite discovery by setting a value of 0 to the maximum Bluetooth discovery duration, to which informed users could consent [10]. While other mobile operating systems could present more fundamental challenges, this is an evolving technology. Bluetooth is indeed pervasive, and as use cases emerge, appropriate permission classes will become available to support legitimate use cases.

Figure 3 depicts the total number of hours recorded for each user as well as the cumulative distribution function. This figure shows that less than 24 hours worth of data was collected for 78 users. There were, however, some users for whom large amounts of data were collected (over 300 hours). The typical user provided a few hours of data per day, especially during the week days. Essentially, this figure shows that our dataset is sparse. There are several causes: Our sample size (141 users) is small compared to the university student population of 9000. Furthermore, our users were volunteers; we did not select them based on friendships. Therefore, many of them did not meet each other. Finally, many students are commuters and come on campus just for courses.

GDC was also tested on the Reality Mining dataset [13], which is denser than our dataset, as illustrated in Table 3. For a fair comparison, we scaled the NJIT dataset up to nine months (the same as the Reality

Table 3: Reality Mining dataset vs. NJIT Dataset (scaled up to 9 months for this comparison)

	NJIT Dataset	**Reality Mining**
Number of users	141	96
Number of different co-located user pairs	3510	6219
Average number of meetings per pair	19.51	30.44
Std dev of number of meetings per pair	29.57	66.58
Average total meeting minutes per pair	683.83	917.55
Std dev of total meeting minutes per pair	4090.23	4116.55

Mining data collection period). Despite their differences, both datasets exhibit large and similar standard deviations for the pairwise number of meetings and total time spent together.

4.2. User Study Validation

In order to verify the results of the data collection phase, we designed an interactive survey to measure users' reactions to the groups we discovered using both GDC and K-clique.

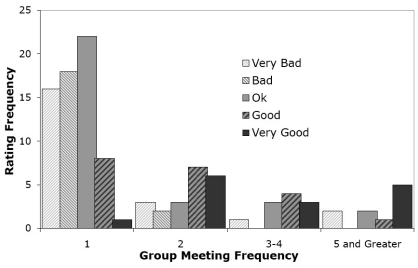
4.2.1. Methodology

A Facebook application was created to present each user with a selection of groups generated by GDC and K-Clique (the users were not aware of what algorithm produced which group). Users ranked the groups on a Likert scale, from 1 to 5, where 1 is very bad and 5 is very good. Users were also encouraged to mark groups as "don't know" if they were not sure. This was included to address the well established fact that people are not always able to recall interactions or accurately report data on all their relationships [13]. Each user was presented with a variety of groups that pertained to him, including at least two groups found only by GDC and two groups found only by the K-Clique algorithm, and when applicable, a group that was found by both algorithms. Users had the option to continue to rank groups until they exhausted all groups in which they were considered a member.

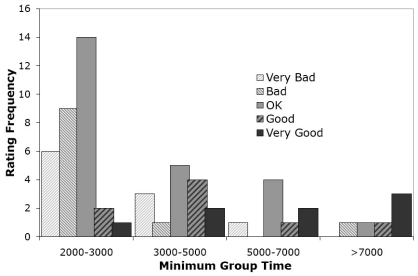
Every study participant who indicated in a pre-survey that they used Facebook were invited to use our Facebook application. Eighty-eight users responded to the Facebook questionnaire, with 56 belonging to at least one group from either algorithm, thus composing 482 different ratings for 265 different groups. Of these ratings, approximately 60% were for groups only from K-Clique and 32% were for groups only from GDC (as we will see later, K-Clique discovers more groups). Two hundred twenty-one groups received ratings within the Likert scale, with the other 44 groups receiving only "don't know" ratings. 93 groups received two or more ratings, while 128 groups received only one rating. Each participant rated 8.6 groups and each participant was a member of 12.9 groups on average.

It is important to note that these metrics are self-reported user ratings and do not represent whether these groups actually met or not, but rather represent the user's perception of the group. It is known with certainty that all the subjects in a GDC-discovered group were co-located for at least the minimum required parameters, but the subjects may not be aware of this co-location. For example, they may be familiar strangers, who meet with certain frequency and recognize each other's faces, but do not know each other's name. Thus, it is difficult for users to accurately self-report on these groups [17]. In some cases, groups with a low rating may be informal groups with no strong bonds like commuters on a train. While these groups will receive low user ratings, they may be useful for some applications such as new group recommendations and ad hoc networking.

For the validation, GDC was run using the following parameters: MG = 15min (threshold for considering a pair-wise meeting), MGT = 2,000s (threshold to consider a group), and MGMF = 1 meeting (threshold for meeting frequency). These parameters were chosen to be as inclusive as possible, but with the intention of increasing their values post-survey to better understand their effect on groups.

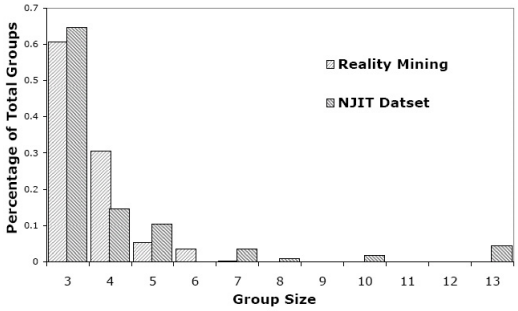


(a) Distribution of user ratings for GDC groups function of group meeting frequency. MGT=2000s.

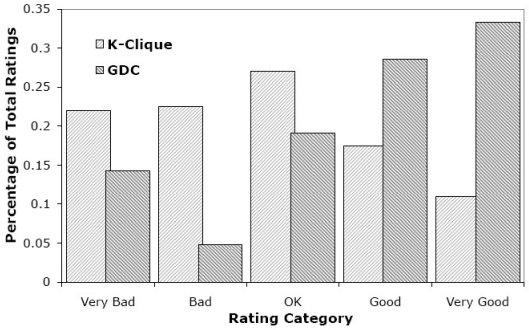


(b) Distribution of user ratings for GDC groups function of group meeting time. MGMF=1.

Figure 4: User rating distributions for different parameter values.



(a) Groups discovered by GDC for our dataset (MGT=2000s, MGMF=1) and the Reality Mining dataset (MGT=18000s, MGMF=9).



(b) User rating comparison between GDC and K-Clique.

Figure 5: Comparison of GDC against Reality Mining (Figure a) and rating comparisons of GDC against K-Clique (Figure b).

K-Clique was chosen because of its shared ability with GDC to detect overlapping communities. Two users who spent at least 2,000s together have an edge in the graph analyzed by K-Clique. The groups used in our survey represent the union of K-Clique results for values of $K \geq 3$.

4.2.2. Results

Effect of parameters on group detection accuracy. Figures 4a and 4b show how the user ratings vary with the group meeting frequency (MGMF) and group meeting time (MGT). As expected, the ratings are much better for higher frequencies and larger amounts of time. What is interesting, however, is that even groups that meet only 2 times receive significantly higher ratings than groups meeting only once. The results also show that meeting frequency is a better predictor of group quality than meeting time. As MGMF is increased, the poorly rated groups virtually disappear. These results also emphasize the importance of GDC's ability to associate MGMF and MGT with the groups it detects. These parameters can be used as tuning knobs for the algorithm, allowing users to trade off the number of detected groups for detection accuracy.

Comparison of results for NJIT and Reality Mining datasets. Figure 5a presents a comparison of the groups detected by GDC when applied on the two datasets. We increased linearly the values of the two GDC parameters for the Reality Mining dataset to account for the longer data collection period. Although many more groups are detected for the Reality Mining dataset, the results in terms of group distribution

as a function of their size are relatively similar. For example, approximately 60% of the groups in both categories have just 3 members, and most of the other groups have 4-5 members. Unlike Reality Mining, our dataset shows a few larger groups (e.g., 13 members).

GDC vs. K-Clique comparison. For this comparison, the values of the parameters for GDC were: MGT=2000 and MGMF=2. The threshold for a K-Clique edge was 2000s. Figure 5b presents the user ratings for the groups detected by GDC.

GDC discovered 65 groups, and K-Clique discovered 292 groups. Twenty one groups were discovered by both algorithms, but only 13 of them received ratings within the Likert scale, with a 2.62 average score (indicating results between poor and okay). Since these groups are common and our goal is to understand the difference between GDC and K-Clique, we exclude them from further comparisons. If these groups would be included in the comparisons, the K-Clique accuracy would remain almost the same, while GDC's would decrease slightly.

Within the 271 remaining K-Clique groups, only 51 are real groups according to our definition, which is that all members of a group spend at least some time together throughout the study. This result is expected because K-Clique's group detection is based on social tie transitivity and does not account for time spent together, which is a major characteristic of co-location. Consequently, users rate K-Clique discovered groups low.

GDC, on the other hand, detects fewer groups, but it guarantees that all these groups exist (i.e., their members spent time together). The results show that it outperforms K-Clique. A χ^2 *test* proves that the difference is statistically significant: χ^2 *(4,N=260) = 21.807,* $p < .001$. Additionally, by treating the Likert data as continuous, the means of the ratings can be compared: GDC has a mean of 3.62, compared to K-Clique's 2.73. A *t-test* proves that this difference is statistically significant: *t(258) = -4.057,* $p < .001$. Finally, we believe that the low meeting frequency considered in this experiment prevented GDC from achieving even higher ratings; our conjecture is that low meeting frequency groups included familiar strangers and users rated these groups low because they did not know the names of the group members.

4.3. Similarity Validation

One issue encountered with DGDC is how to properly evaluate the results it generates. Specifically, we would like to evaluate it based on its comparative similarity to GDC, the impact more sharing has on the quality of results, the risk to privacy, and the cost in communication overhead. We would also like to evaluate GDC, DGDC, and other methods such as K-Clique against a mutual baseline, or ground truth, which we will discuss in the subsequent subsection 4.4.

4.3.1. Methodology

To adequately evaluate the results of different group detection algorithms, we need to define metrics of comparison. When surveying existing comparison metrics, the majority were inadequate at taking into account all the output characteristics of GDC, namely groups with meetings. Specifically, most comparison metrics involve reducing the algorithm output to a graph and comparing the graphs, or generating some values based on the graphs and comparing those values.

While these metrics are useful in many cases, for our purposes they provide an inadequate means of comparison. To that end, we propose the following constructs to enable either in-depth or summary comparison of the more complex outputs that modern group detection algorithms produce, such as GDC.

Group Similarity is a measure of how similar two groups are to each other. For the purposes of GDC generated groups, we have two principal dimensions to compare: meeting time and membership. In this way, given two groups A and B, we can compute their similarity with the following equation:

$$s\left(A,B\right) = 0.5 \times \left(\frac{A.members \cap B.members}{A.members \cup B.members} + \frac{A.meetings \cap B.meetings}{A.meetings \cup B.meetings} \right)$$

Members and meetings are equally weighted in the similarity computation. When comparing algorithms such as GDC and K-Clique, it is possible to consider only the members component, as K-Clique does not preserve the kind of meeting data necessary to compute this complete metric. A simple "time together" is

insufficient; the start and end times of each meeting is necessary to perform the intersections and unions described above. This definition of similarity produces a value between 0 and 1, with 1 being perfectly similar, 0 being totally dissimilar.

Group Set Similarity is a measure of how similar two sets of groups are to each other. This metric makes use of a maximum matching of the similarity scores between group set U and group set X. As a consequence, $s(U.u, X.x) \forall u \in U \wedge x \in X$ should be precomputed. From these values, finding the maximum matching is fairly straightforward. With high likelihood there will be some unmatched groups from U or X; we ignore zero value matches in this metric. They are considered in the representativeness metric described subsequently. Ultimately, Group Set Similarity is computed against the similarity scores of the maximum matching set $s(u, x) = m \in M, u \in U, x \in X$ as follows:

$$sSet\left(U, X\right) = \frac{\sum\limits_{m \in M(U,X)} m}{\left\| M\left(U, X\right) \right\|}$$

Group Set Representativeness measures how complete the matching is. A value of 0 means that there was no match at all between the two group sets under consideration. A value of 1 means that every group has some kind of match, regardless of quality. Note that since a maximum matching involves, strictly, a 1 to 1 mapping between the two sets of groups, set X and set Y must be the same size for representativeness to be its highest. A consequence of this definition is that representativeness can be a good measure, in the context of evaluation against ground truth, of either the wastefulness or incompleteness of a matching set. Algorithms that find adequate matches in terms of similarity may do so at great cost in terms of representativeness. For clarity, we define representativeness as follows:

$$rSet\left(U, X\right) = \frac{\left\| M\left(U, X\right) \right\|}{max\left(\left\| U \right\|, \left\| X \right\|\right)}$$

Algorithm Similarity is a similarity measure between two algorithms G and H which return sets of groups where each set is keyed on a particular user. This is exactly the format of DGDC's output, which returns for each participating user a set of groups. Note also that Algorithm Similarity is built using the results of computing Group Set Similarity over each users' groups across both algorithms. In the case of GDC, we can trivially phrase its output as a set of groups sets by assigning to each user the set of groups for which GDC considers that user a member. Although one could argue this may cause some inflation of comparison as each group, good or bad, will appear many times in the result sets, we believe this expansion is valid as many use cases for GDC's output involve distributing the results over the membership, and as such an evaluation metric that factors in this use case is justified. The construction of this similarity metric assumes that every $g \in G$ and $h \in H$ represents a user, and the function $groups(x)$ represents the set of groups associated by the algorithm with that user. Consequently, algorithm similarity is defined as follows:

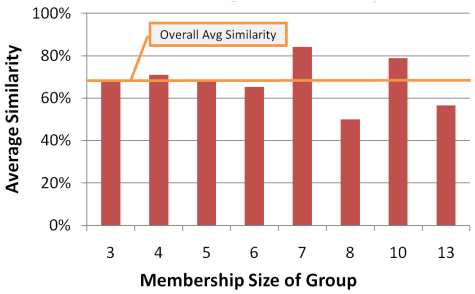
$$sAlg\left(G, H\right) = \frac{\sum\limits_{g,h \forall g=h}^{G,H} sSet\left(groups\left(g\right), groups\left(h\right)\right)}{\left\| \left\{ \left(g, h\right) \forall g = h \right\} \right\|}$$

It can also be useful, and in some cases far more informative, to construct a histogram categorized on the membership size of all the individual group matches drawn from the union of all the matchings backing each sSet result. This allows comparison of group detection similarity based on the size of the group, and can reveal otherwise hidden performance characteristics.

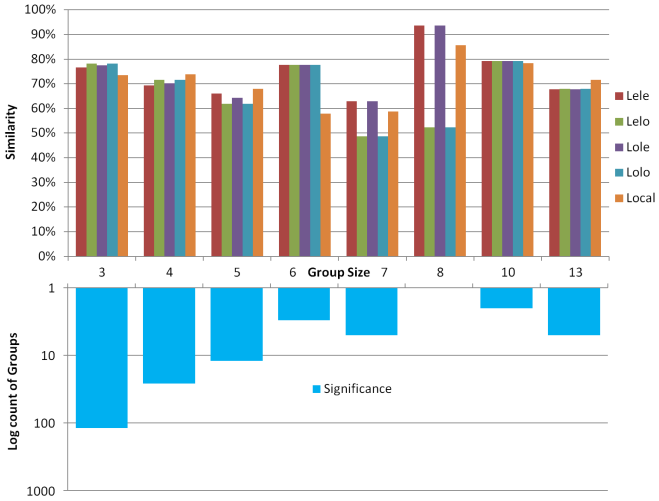
Algorithm Representativeness has a similar construction, as follows:

$$rAlg\left(G, H\right) = \frac{\sum\limits_{g,h \forall g=h}^{G,H} rSet\left(groups\left(g\right), groups\left(h\right)\right)}{\left\| \left\{ \left(g, h\right) \forall g = h \right\} \right\|}$$

This gives an idea of the average representativeness between the two algorithms, as it is the average over the Group Set Representativeness scores for each users' groups as generated by both algorithms. A low



(a) A relative comparison of GDC against K-clique on the basis of average similarity by group membership size. The orange line indicates the overall average similarity across all groups. Note that similarity in this case only considers membership, not meeting time.



(b) A relative comparison of similarity of DGDC policies to GDC. The bars descending are a logarithmic representation of the number of groups discovered by GDC with the category's membership size. This gives some idea of the importance of the similarity results in each category, which is shown by the upward facing bars.

Figure 6: Relative comparisons of GDC against K-clique, and DGDC against GDC

score indicates that algorithm results H discovers either far too few or far too many groups when compared against algorithm G.

With these metrics and constructions in hand, a deeper exploration of the similarity and representativeness between algorithms becomes possible and straightforward. Note that all the following results were generated using GDC and DGDC with a meeting granularity of 29 minutes (see Section 2.3 for an explanation of this granularity choice). GDC discovers 172 groups at that granularity, with a minimum size of 3 members, a maximum of 13, and an average group size of 3.86. The minimum number of meetings per group is 1 with a maximum of 6 and an average of 1.5.

4.3.2. Results

In Figure 6a we see that in relative terms, GDC and K-Clique produce results that are marginally similar (overall average of 68.4%). Additionally, K-Clique against GDC has an algorithm representativeness of just 58.9%. This is as expected, since K-Clique discovers some 292 groups compared to GDC's 172. Independent of the survey results, it would be difficult to make strong conclusions from this comparison alone, beyond the observations that GDC and K-Clique produce marginally similar results, yet GDC's output is considerably more concise. From our user study, we can conclude more strongly that not only are GDC's results more concise, they are also qualitatively better, and that the average algorithm similarity of 68.4% between GDC and K-Clique indicates in a computational sense how divergent their results are. If the membership of the groups detected by each algorithm were closely related (convergent), we would expect a higher average similarity. Note that we will explore in more detail the differences suggested by these results in our ground truth discussion.

In Figure 6b, we see the relative similarity of DGDC against GDC. The most notable feature of this graph is that, regardless of policy, the results for the most important classes of groups are approximately similar. Specifically for groups of size 3, DGDC with some manner of sharing is about 77% similar to GDC, and without sharing (labeled Local on the graph) is about 73% similar. On the average across all group matches, DGDC is about 3% more similar to GDC than using only local data. Concerning representativeness, DGDC

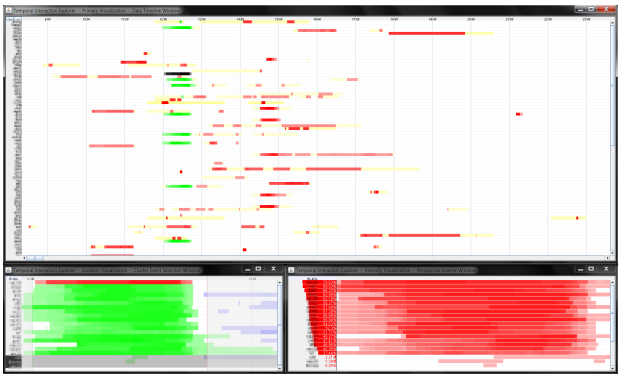


Figure 7: An overview of TIE's various windows showing perspectives on source data and events in temporal "heat-maps". TIE allows deeper investigation of interaction-based temporal data, and enables some measure of ground truth construction in the absence of user confirmation, provided the collected data is reliable.

outperforms Local with Policy Lele scoring 66.5% representativeness on the low end, Policy Lolo at the high end with 69.2%, and Local only at 62.6%. Overall, these results begin to suggest that while sharing data can produce results closer to GDC's, more permissive sharing mechanisms are not overly beneficial. The question remains if these relative results are confirmed in an absolute comparison of GDC, DGDC, and Local only against Ground truth, as discussed in the following section.

4.4. Ground Truth Validation

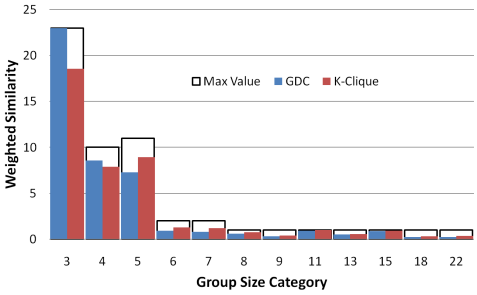
As our research into distributed and more computationally efficient forms of GDC continued, the best approach to evaluate new forms became complicated. Additional user surveys would not be an acceptable evaluation approach as too much time had passed since the original study, and the user feedback collected from the initial evaluation cannot be reliably applied to the new results.

Our first methods of comparison evaluated DGDC against GDC, which gave us a measure of the relative similarity between the two algorithms. However, this technique does not provide any way to say if the differences are "better" or "worse", only an intuition as to the extent of the differences. In an effort to better evaluate the results of DGDC against GDC, some form of ground truth would be necessary.

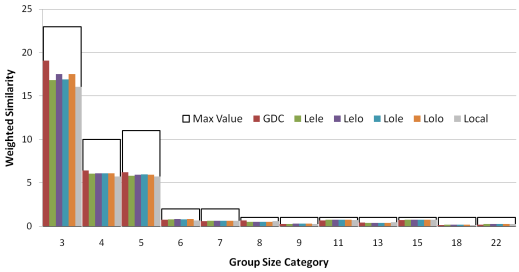
4.4.1. Methodology

Generating ground truth from our sparse but multi-feature dataset is a challenging endeavor. To enable ground truth discovery, we created a tool named the Temporal Interaction Explorer (TIE). TIE visualizes the source data to show the recorded interactions among users. It also enables close review of particular co-presence events so that an investigator can form a clear picture of who saw whom, for how long, and with what degree of overlap. Without this tool, it would be nearly impossible to generate ground truth data reliably.

TIE works by displaying a temporal "heat-map" of interactions between users, where specific interactions can be selected for deeper investigation. When an interaction event is selected, two important views of the event are generated. For reference, see Figure 7. First, the perspectives of all users who might have been present (were sighted by at least one other user at the event) are shown in a timeline. Each perspective is represented as a "smart" heat-map, where hovering the cursor over a perspective shows the identifiers of the other users seen by that perspective at that time. This allows an investigator to carefully analyze who was observed at an event, as represented by the source data. The second view shows an inverse of this perspective information. Namely, it shows a "smart" heat-map timeline revealing how many and which users saw each person who contributed a perspective in the first view. With these two views, it is possible to discover not only a user's contributed perspective on an event, but also the cumulative event-level perspective on that user's presence at an event.



(a) Absolute comparison of member-only GDC and K-Clique against GTS. This graph shows the average similarity per size of group, scaled by importance. Importance is the number of groups in ground truth of that size.



(b) Absolute comparison of GDC, DGDC Policies, and Local against GTS. This graph shows the average similarity per size of group, scaled by importance. Importance is the number of groups in ground truth of that size.

Figure 8: Comparisons of GDC, K-Clique, and DGDC Policies against GTS.

Even with this helpful tool, performing a complete survey of nearly a month of data contributed by more than a hundred users is tedious. As accurate social group detection will become more important in the coming years, continuing to develop effective visualizations and tools to validate the algorithmic results is a necessity.

Using TIE and the views mentioned above, we generated a set of groups and meetings that represents as near as we can determine the ground truth for out data set. We will refer to the ground truth data set as GTS throughout the paper for convenience. GTS has 55 groups, with the largest group having 22 members and an average group size of 5.2. The largest meeting frequency was 6, with an average of 2.

4.4.2. Results

The results in the case of K-Clique and GDC against GTS is dramatic. Figure 8a shows the weighted average similarity of each algorithm's results, when membership is the only consideration. The average in each category is weighted by the number of groups in GTS with membership of that size. GDC strongly outperforms K-Clique for groups of size 3 and 4, which are two of the most important kinds of groups (based on their prevalence in GTS). In these categories, GDC has an average similarity of 100% and 85.7% respectively, against K-Clique's 80.6% and 78.9%. While K-Clique edges ahead of GDC for groups of size 5 or more, this is due to K-Clique's larger result set (292 groups to GDC's 172), an observation supported strongly by the algorithm representativeness of just 36.8% for K-Clique against GDC's 55.0%. These numbers, based on our definition of algorithm representativeness, reveal that both GDC and K-Clique discover "too many" groups, but GDC finds significantly fewer excess groups. Our initial user study results are further validated by these data, as GDC outperforms K-Clique while also achieving greater conciseness, observations now empirically supported.

An analysis of DGDC against GDC is less compelling, but no less important, than that of GDC against K-Clique. Figure 8b illustrates these results. Note that a significant decrease in the similarity score of GDC between this figure and Figure 8a is due to the inclusion of meeting time data. Ultimately, large variations in perspectives on shared meetings degrades the results considerably. This is due to the GDC and DGDC algorithm design, which uses the group's minimum agreed upon perspective. In many cases during the ground truth construction the minimum perspective was observed to be entirely too conservative; this does impact the apparent performance of GDC and DGDC significantly. Assuming denser source data, this issue would be lessened.

Similar to earlier observations, figure 8b supports the perspective that while some sharing can on average improve the results of DGDC over only local data, the gains of greater sharing are not clear. Therefore it is our recommendation that a more conservative policy, such as Policy Lele, gives acceptable results while maximizing effective privacy and minimizing the distributed communication cost overhead.

Another interesting observation is that while GDC outperforms DGDC in nearly every group size category, the gains are not outstanding. For groups of size 3, GDC achieves 83% similarity to GTS, while the DGDC policies average to 74.8%. Local Only reaches 69.9% similarity for that category. Groups of size 4 are detected with 64.3%, 60.8%, and 57.3% between GDC, DGDC on average, and Local Only, respectively. The differences in result between centralized and distributed methods are largely from more isolated users in DGDC discovering fewer groups than the collective record indicates. Altogether, these results illustrate that distributing GDC produces comparatively similar results without releasing all data to a central collector.

5. Related work

Traditional social network studies involve self-reported relational data, which may have accuracy problems [17]. Observing co-location data directly can provide more accurate insights into human mobile social patterns. Over the past few years, there have been several projects that have utilized co-location information to learn social properties [18, 19, 13, 4].

The Reality Mining project at MIT[13] deployed one hundred mobile phones to users over the duration of an academic year (nine months). By calculating the entropy of specific users, they were able to compute the probability, with accuracies of up to 90%, that a user will come into contact with another user within a certain time frame. Similarly, researchers at Intel Research Lab and University of Cambridge [20, 21] used iMotes to collect co-location data. Participants initially included students and researchers at Cambridge; a subsequent study covered a conference environment, $InfoCom'06$. Contact times for users in these experiments followed a power-law distribution. More applicable to social networking was the finding that contact times adhered to the users' underlying social network. All these studies focused on incorporating social ties between individuals into applications. Instead of this microscopic view, our work concentrates on the macro perspective, identifying communities in the social network.

Beyond physical groups, online groups have received significant attention. [22] studied the relationships that exist between online and offline groups over the same set of people. Their results showed that online and physical groups represent distinct classes of social engagement, suggesting that the fusion of both types of groups could be a source of strong social connection, if the nature of that connection is unimportant. Social groups within online social networks are investigated in [23], measuring the proximity of groups from different networks and using that information to recommend new groups to join. In all likelihood, this method could be expanded to include physical groups detected using multiple methods. Networks formed by online interactions and their association to physical groups is also explored in [24], with a focus on potential applications, largely in the area of publication and research communities.

Community detection in complex networks plays an important role in social computing [25, 26]. Several community detection methods have been proposed and examined in the literature. FAST and WNA [12] are both hierarchical methods for detecting communities in graphs that can easily be modified to deal with weighted graphs. However, they cannot detect overlapping communities, which are natural in human networks. This problem is overcome by the K-clique algorithm [11], which on the other hand does not work for weighted graphs. Unlike these algorithms, GDC is able to solve both problems: it detects overlapping communities, and it uses the amount of time spent together and meeting frequencies as weights. Even more importantly, GDC guarantees that the detected communities are groups according to our definition (i.e., their members spend time together). All the other algorithms mentioned above are based on social tie transitivity and end up with many non-existent groups, whose members never spend time together (Section 4 explained this issue in detail).

In [21], the authors propose three distributed community detection algorithms, which share certain goals with D-GDC. The phones exchange the complete local sets of familiar users (i.e., their contact time is above a threshold). The algorithms differ in the way they merge this information into the local data structures. Similarly, [27] introduces a distributed community detection algorithm. Their approach involves local community detection and opportunistic encounters to exchange all local-knowledge community information. After these exchanges, each local node integrates the shared knowledge and discovers its global community membership. Unlike these algorithms, D-GDC allows users to specify sharing policies that can restrict the information being exchanged, thus improving their privacy. The most important difference, however, comes

from the very nature of these algorithms: D-GDC detects groups that actually spend time together, while those in [21] extend graph algorithms such as K-Clique and [27] is a transitivity-preserving graph algorithm, which are not appropriate for detecting such groups.

An important facet of GDC and DGDC is their preservation of group features over time. Frequency of meeting, length of meeting, and the changing nature of a group can be tracked over the lifetime of user participation. This record of group dynamics over time is similar to the research in [5], which looks at whole network dynamics. As demonstrated in their work, such analysis over time is good for epidemic tracking or prediction, delay tolerant networking, and network resilience testing.

Applications for group detection and more broadly, neighbor detection, are becoming increasingly available. Group membership detection in combination with mobility over time can help track group motion loyalty, as briefly described in [28]. Knowing something about an individual's existing groups or preferences allows recommending group membership, as described in [29], where known preferences are leveraged in constructing a target number of output groups. Once groups are known, it is possible to use multicast or anycast systems to communicate with the group, even securely (as demonstrated by [30, 31, 32]). As well, systems exist to interactively determine activity recommendations suitable for the entire group as in [33]. A survey of such systems for group recommendations was recently performed by [34] in the context of voting. Labeling groups can improve message forwarding decisions, as demonstrated by Hui, et. al [35]. Groups and group membership can enable effective socially-aware communication and storage services such as Prometheus [36], which also leverages labels to describe and organize types of interactions. Our work in identifying group membership in both centralized and decentralized ways is essential to enable these applications and systems as they rely on the availability of group knowledge.

6. Summary

The paper presented two algorithms, GDC and DGDC. The first is centralized and uses a global view of user co-location to discover groups. The second is distributed and may miss some groups, depending on the user's perspective. These groups can be used to provide new socially-aware features in applications, middleware, and ad hoc network protocols. Additionally, they can be used to address social science questions such as: do online social networks and co-location based social networks reinforce each other or do they capture different types of social ties, which we explored using a fused network [22].

GDC was validated on two different types of datasets, and, according to the user ratings, it outperforms the K-Clique algorithm by 30%. Its groups are guaranteed to exist because their members have spent time together. More complete group meetings can be discovered by tuning the meeting time parameter, and more relevant groups can be selected by increasing the value of the meeting frequency parameter.

DGDC was evaluated for privacy characteristics based on different policy choices and against GDC and DGDC without data exchange. This analysis helps form an understanding that minimal data exchange achieves desirable results while preserving privacy.

Finally, we evaluated GDC, DGDC and K-Clique against ground truth data to allow absolute comparison of group detection methods. From this we concluded that GDC outperforms K-Clique in both terms of average performance and in terms of output conciseness. Concerning GDC and DGDC, we conclude that ultimately a more conservative approach to exchange yields "good enough" results while maximizing privacy gains and minimizing communication costs.

References

[1] A. Gupta, A. Kalra, D. Boston, C. Borcea, Mobisoc: A middleware for mobile social computing applications, ACM/Springer MONET 14 (2009) 35–52.

[2] C. Borcea, A. Iamnitchi, P2p systems meet mobile computing: A community-oriented software infrastructure for mobile social applications, in: Proc. of the Workshop on Decentralized Self Management for Grids, P2P, and User Communities (SELFMAN '08), pp. 242–247.

[3] M. Wiberg, Roamware: An integrated architecture for seamless interaction in between mobile meetings, in: Proc. of the International ACM SIGGROUP Conference on Supporting Group Work (Group '01), pp. 288–297.

[4] P. Hui, J. Crowcroft, E. Yoneki, Bubble rap: social-based forwarding in delay tolerant networks, in: Proc. of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '08), pp. 241–250.
[5] L. Isella, J. Stehl, A. Barrat, C. Cattuto, J. Pinton, W. Van den Broeck, What's in a crowd? analysis of face-to-face behavioral networks, Journal of Theoretical Biology 271 (2011) 166–180.
[6] G. C. Homans, The Human Group, Harcourt, Brace and Company, 1950.
[7] A. Gupta, S. Paul, Q. Jones, C. Borcea, Automatic identification of informal social groups and places for geo-social recommendations, International Journal of Mobile Network Design and Innovation 2 (2007) 159–171.
[8] B. Hoh, M. Gruteser, H. Xiong, A. Alrabady, Enhancing security and privacy in traffic-monitoring systems, IEEE Pervasive Computing 5 (2006) 38–46.
[9] I. Smith, J. Tabert, T. Wild, A. Lamarca, A. Lamarca, Y. Chawathe, Y. Chawathe, S. Consolvo, S. Consolvo, J. Hightower, J. Hightower, J. Scott, J. Scott, T. Sohn, T. Sohn, J. Howard, J. Howard, J. Hughes, J. Hughes, F. Potter, F. Potter, P. Powledge, P. Powledge, G. Borriello, G. Borriello, B. Schilit, B. Schilit, Place lab: Device positioning using radio beacons in the wild, in: In Proceedings of the Third International Conference on Pervasive Computing, Springer, 2005, pp. 116–133.
[10] Google, Android developer api, WWW Page, http://developer.android.com/guide/topics/connectivity/bluetooth.html, 2012.
[11] G. Palla, I. Derényi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (2005) 814–818.
[12] M. Newman, Detecting community structure in networks, The European Physical Journal B-Condensed Matter and Complex Systems 38 (2004) 321–330.
[13] N. Eagle, A. S. Pentland, D. Lazer, Inferring friendship network structure by using mobile phone data, Proceedings of the National Academy of Sciences 106 (2009) 15274–15278.
[14] D. Boston, C. Borcea, Tie: Temporal interaction explorer for co-presence communities, in: Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on, pp. 854 –863.
[15] M. Talasila, R. Curtmola, C. Borcea, Collaborative bluetooth-based location authentication on smart phones, in: Under submission.
[16] N. Ravi, P. Stern, N. Desai, L. Iftode, Accessing ubiquitous services using smart phones, in: Proc. of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 383–393.
[17] E. Paulos, E. Goodman, The familiar stranger: anxiety, comfort, and play in public places, in: Proc. of the SIGCHI conference on human factors in computing systems (CHI '04), pp. 223–230.
[18] A. Madhavapeddy, A. Tse, A study of bluetooth propagation using accurate indoor location mapping, in: Proc. of the 7th International Conference on Ubiquitous Computing (UbiComp '05), pp. 105–122.
[19] M. McNett, G. M. Voelker, Access and mobility of wireless pda users, SIGMOBILE Mob. Comput. Commun. Rev. 9 (2005) 40–55.
[20] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, C. Diot, Pocket switched networks and human mobility in conference environments, in: Proc. of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking (WDTN '05), pp. 244–251.
[21] P. Hui, E. Yoneki, S. Chan, J. Crowcroft, Distributed community detection in delay tolerant networks, in: Proc. of 2nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture (MobArch '07), pp. 1–8.
[22] S. J. Pan, D. Boston, C. Borcea, Analysis of fusing online and co-presence social networks, in: PerCom Workshops'11, pp. 496–501.
[23] B. Saha, L. Getoor, Group proximity measure for recommending groups in online social networks, in: In 2nd ACM SIGKDD Workshop on Social Network Mining and Analysis.
[24] G. Groh, T. U. Muenchen, Groups and group-instantiations in mobile communities detection, modeling and applications, in: In Proceedings of the International Conference on Weblogs and Social Media 2007.
[25] L. Danon, A. Díaz-Guilera, J. Duch, A. Arenas, Comparing community structure identification, Journal of Statistical Mechanics: Theory and Experiment (2005) P09008.
[26] M. Newman, Analysis of weighted networks, Physical Review E 70 (2004) 56131–56140.
[27] M. Williams, R. Whitaker, S. Allen, Decentralised detection of periodic encounter communities in opportunistic networks, Ad Hoc Networks 10 (2012) 1544 – 1556.
[28] Y.-C. Chen, E. Rosensweig, J. Kurose, D. Towsley, Group detection in mobility traces, in: Proceedings of the 6th International Wireless Communications and Mobile Computing Conference, IWCMC '10, pp. 875–879.
[29] L. Boratto, S. Carta, M. Satta, Groups identification and individual recommendations in group recommendation algorithms, Computer (2010) 27.
[30] S. Banerjee, B. Bhattacharjee, Scalable secure group communication over ip multicast, in: In Proceedings of Internation Conference on Network Protocols, pp. 1511–1527.
[31] Y. Amir, Y. Kim, C. Nita-Rotaru, J. Schultz, J. Stanton, G. Tsudik, Secure group communication using robust contributory key agreement, Parallel and Distributed Systems, IEEE Transactions on 15 (2004) 468 – 480.
[32] Y. Amir, C. Nita-Rotaru, S. Stanton, G. Tsudik, Secure spread: an integrated architecture for secure group communication, Dependable and Secure Computing, IEEE Transactions on 2 (2005) 248 – 261.
[33] K. Mccarthy, M. Salamó, L. Coyle, L. Mcginty, B. Smyth, P. Nixon, Group recommender systems: a critiquing based approach, in: Proceedings of the 11th international conference on Intelligent user interfaces, ACM, 2006, pp. 267–269.
[34] G. Popescu, P. Pu, Group recommender systems as a voting problem, in: EPFL Technical report.
[35] P. Hui, J. Crowcroft, How small labels create big improvements, in: Proc. of the International Workshop on Intermittently Connected Mobile Ad hoc Networks, pp. 65–70.

[36] N. Kourtellis, J. Finnis, P. Anderson, J. Blackburn, C. Borcea, A. Iamnitchi, Prometheus: User-Controlled P2P Social Data Management for Socially-Aware Applications, in: Proceedings of the 11th ACM/IFIP/USENIX International Middleware Conference(Middleware 2010), pp. 212–231.