GDC: Group Discovery using Co-location Traces

Steve Mardenfeld, Daniel Boston, Susan Juan Pan, Quentin Jones*, Adriana Iamnitchi+, and Cristian Borcea
*Computer Science Department, *Information Systems Department, New Jersey Institute of Technology, Newark, USA*
+Department of Computer Science and Engineering, University of South Florida, Tampa, USA
Email: sm424@njit.edu, dbj38@njit.edu, jp238@njit.edu, qjones@njit.edu, anda@cse.usf.edu, borcea@cs.njit.edu

***Abstract*—Smart phones can collect and share Bluetooth co-location traces to identify ad hoc or semi-permanent social groups. This information, known to group members but otherwise unavailable, can be leveraged in applications and protocols, such as recommender systems or delay-tolerant forwarding in ad hoc networks, to enhance the user experience. Group discovery using Bluetooth co-location is practical because: (i) Bluetooth is embedded in nearly every phone and has low battery consumption, (ii) the short wireless transmission range can lead to good group identification accuracy, and (iii) privacy-conscious users are more likely to share co-location data than absolute location data.**

This paper proposes the Group Discovery using Co-location traces (GDC) algorithm, which leverages user meeting frequency and duration to accurately detect groups. GDC is validated on one month of data collected from 141 smart phones carried by students on our campus. Users rated GDC's groups 30% better than groups discovered using the K-Clique algorithm. Additionally, GDC lends itself more easily to a distributed implementation, which achieves similar results with the centralized version while improving user's privacy.

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

I. 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. Finally, 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].

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 [5]. 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 [6] 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 long 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 [7]. Third, localization systems running on phones (e.g., GPS, Place Lab [8]) 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) algorithm, which uses Bluetooth co-location traces 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. Sharing co-location information, instead of absolute location, may be a participation incentive for those users with concerns about their location being tracked. Finally, 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).

Well-known graph algorithms, such as K-Clique [9] and WNA [10], 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 introduces GDC, a novel and practical algorithm that leverages Bluetooth co-location traces to accurately identify social groups. GDC associates a group meeting time and meeting frequency with each group, which can be used to compare, categorize, and rank groups.
- it presents the validation and evaluation of 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 [11].
- 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 presents a distributed version of GDC, in which users do not share data with a centralized server, but only with people they have met. This implementation achieves similar results with the centralized version and significantly contributes to user privacy protection.

The rest of this paper is organized as follows. Section II describes the GDC algorithm. Section III presents the experimental results and analysis. Section IV describes the distributed GDC version and presents comparison results between the two GDC versions. Section V discusses the related work, and Section VI concludes the paper.

II. GDC Algorithm

A. 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.

B. 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 (i.e., without time gaps between them, where time gap is a function of the Bluetooth discovery frequency).

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 fulfills all of the necessary requirements, defined by GDC parameters, and if it does not significantly overlap other groups.

C. 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. The first phase transforms individual Bluetooth trace records into meeting records, which detail the start and end times of individual meetings between two users. 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. 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. The fourth phase takes the global list of clusters and identifies the final groups by eliminating subgroups of little significance.

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. In this way, many one-time encounters (e.g., people passing each other) are discarded. The meeting granularity (MG) parameter is used to indicate the maximum amount of time that two Bluetooth records can be apart and still be considered part of the same meeting (pseudo-code line 5). 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 evaluation, we set MG to 15 minutes based on empirical observations of the clusters

Phase 1 Creating Pair-wise Meeting Records

```
1: Sort union of Bluetooth records r_i by user, userwith, time
2: endtime = 0, starttime = r_0.time
3: for all r_i in records do
4:     if (r_i.user == r_{i-1}.user) AND (r_i.userwith == r_{i-1}.userwith) then
5:         if (r_i.time − r_{i-1}.time > MG) then
6:             if (endtime ≠ 0) then
7:                 addmeeting(r_{i-1}.user.meetings, r_{i-1}.userwith, starttime, endtime)
8:             end if
9:             starttime = r_i.time, endtime = 0
10:        else
11:            endtime = r_i.time
12:        end if
13:    else
14:        if endtime ≠ 0 then
15:            addmeeting(r_{i-1}.user.meetings, r_{i-1}.userwith, starttime, endtime)
16:        end if
17:        starttime = r_i.time, endtime = 0
18:    end if
19: end for
20: if endtime ≠ 0 then
21:    addmeeting(r_i.user.meetings, r_i.userwith, starttime, endtime)
22: end if
```

Figure 1. Co-location for three users according to each user's perspective

created by GDC's second phase. At this value, the number of additional clusters started to level off. The same value can be applied to other datasets as long as the Bluetooth discovery frequency remains similar (once every 1-3 minutes).

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

```
1: for all x in users do
2:     lasttime = 0, currentwith = new minheap([user, endtime] keyed on endtime)
3:     for all m in x.meetings do
4:         while (currentwith[0].endtime ≤ m.starttime) do
5:             for all cluster c in allCombinations(x, allUsers(currentwith)) do
6:                 c.totaltime += currentwith[0].endtime - lasttime
7:                 c.frequency += 1
8:                 updateClusters(x.clusters, c)
9:             end for
10:            lasttime = currentwith[0].endtime
11:            remove(currentwith[0])
12:        end while
13:        for all cluster c in allCombinations(x, allUsers(currentwith)) do
14:            c.totaltime += m.starttime - lasttime
15:            c.frequency += 1
16:            updateClusters(x.clusters, c)
17:        end for
18:        currentwith.add([m.userwith, m.endtime])
19:        lasttime = m.starttime
20:    end for
21: 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 (lines 5 and 13) because there is not enough information to tell if overlapping clusters should be put together in 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 1, 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). For instance, 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 records and 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. These values are updated for each new meeting record processed. The min-heap in the pseudo-code (line 2) is used to process all these permutations for each meeting. 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 (updated in line 7). 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 (lines 5-6).

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. 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. The second phase of the algorithm, in which GDC generates all the possible cluster permutations, guarantees that two clusters are either disjoint or one includes the other. 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. The output of this phase is the final output of GDC.

D. 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. For our dataset, the maximum number of users found together was 15 and the average was 6.8.

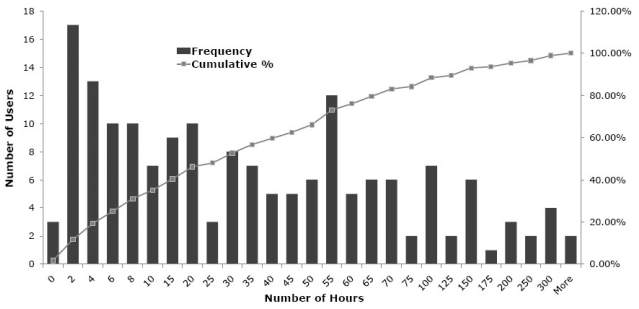

Figure 2. Number of hours of collected data per user

III. 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 [9].

A. 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 [12]. The local records were uploaded to a server periodically.

Figure 2 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 [11], which is denser than our dataset, as illustrated in Table I.

Table I
REALITY MINING DATASET VS. NJIT DATASET (SCALED UP TO 9 MONTHS FOR THIS COMPARISON)

	NJIT Dataset	Reality Mining
Number of users	141	96
umber 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

For a fair comparison, we scaled the NJIT dataset up to nine months (the same as the Reality 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.

B. Validation Methodology

The output of GDC was verified by surveying the participants from the data collection phase. 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 [11]. 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 [13].

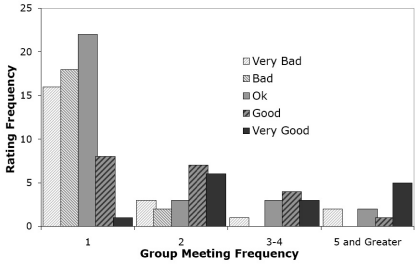

Figure 3. Distribution of user ratings for GDC groups function of group meeting frequency. MGT=2000s.

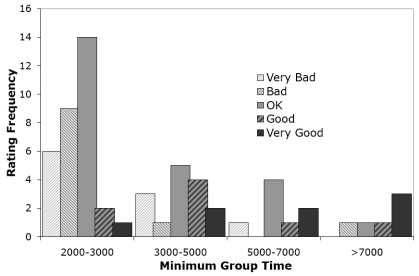

Figure 4. Distribution of user ratings for GDC groups function of group meeting time. MGMF=1.

For the validation, GDC was run using the following parameters: MG = 15min (threshold for considering a pairwise 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.

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 $K \geq 3$.

C. Results

Effect of parameters on group detection accuracy. Figures 3 and 4 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 5 presents a comparison of the groups

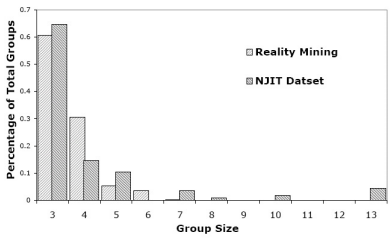

Figure 5. Groups discovered by GDC for our dataset (MGT=2000s, MGMF=1) and the Reality Mining dataset (MGT=18000s, MGMF=9).

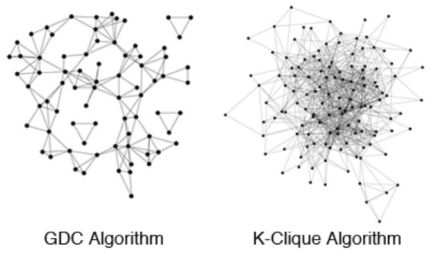

Figure 6. Visualization of the groups detected by GDC and K-Clique

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 6 shows the social network graphs created from the groups detected by the two algorithms, and Figure 7 presents the user ratings for these groups.

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

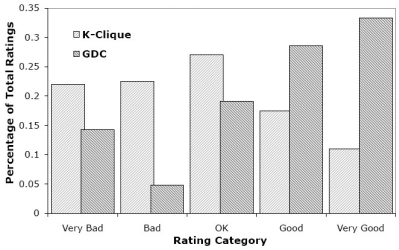

Figure 7. User rating comparison between GDC and K-Clique

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.

IV. 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 the benefits of a distributed version of GDC (D-GDC), describes the changes required by this version, and highlights a few initial results.

A. Benefits

The main benefit of D-GDC is better privacy for users, who are protected from sharing data with 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 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.

Other benefits of a distributed version are resilience and flexibility. D-GDC 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.

B. 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

Table II
SIMILARITY OF GROUPS DETECTED BY D-GDC AND GDC

	Request using "seen with"	Request everyone ever seen	Local only
Average similarity	74.38%	77.33%	58.24%
Groups with similarity > 90%	56.53%	59.77%	19.14%

shared perspectives is vital because of errors associated with recording co-location via Bluetooth. Thus, the first phase of D-GDC 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 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.

Each user develops her own perspective on groups (the quality of these perspectives is presented in Section IV-C), 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.

C. Results

To evaluate D-GDC, 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. For example, a user attempts communication with another if they have seen each other for half of the meeting granularity (MG) threshold. Then, the requester asks for records on all mutually co-located users based on her "seen with" list. Alternatively, a user could simply ask for records on every user she has ever seen; while this represents a lax personal privacy policy for the requesting user, it may ensure a more complete response.

Table II shows how similar the results of GDC and D-GDC are. Since many groups share a significant number of members, but they are not exactly the same, we define the similarity between two groups (A and B) as follows:

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

For every user *U*, we compute the similarity between her GDC groups (i.e., groups in which U is a member) and the groups discovered on U's local phone by D-GDC. For each GDC group, we considered the D-GDC group with the highest similarity. Then, we compute the average similarity for all groups of each user, and finally, the average similarity across all users.

The results show that D-GDC groups have a high similarity with GDC groups. We also observe that the "request using seen with" policy of D-GDC performs similarly with the "request everyone ever seen"; thus, the first policy should be used for better privacy protection. Additionally, the results demonstrate that D-GDC outperforms a localized version of the algorithm (i.e., "local only" policy), in which GDC runs only on the locally collected data. This difference is even clearer when we look at the second row of the table, where we see how many GDC groups have a similarity of 90% or greater with D-GDC groups.

D-GDC can achieve better performance for higher values of the MGT and MGMF parameters as well as for denser data. For example, due to our sparse data, several users ended up with a few groups in the centralized version, but no groups in the decentralized version. If these users are discarded because their group membership is borderline, the performance of D-GDC increases by over 10%.

V. RELATED WORK

Traditional social network studies involve self-reported relational data, which may have accuracy problems [13]. 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 [4], [11], [14], [15].

The Reality Mining project at MIT [11] 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 [16], [17] 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.

Community detection in complex networks plays an important role in social computing [18], [19]. Several community detection methods have been proposed and examined

in the literature. FAST and WNA [10] 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 [9], 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 III explained this issue in detail).

In [17], 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. 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 [17] extend graph algorithms such as K-Clique, which are not appropriate for detecting such groups.

VI. Summary

This paper presented GDC, an algorithm for group discovery based on co-location traces collected from mobile phones. 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?

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, and the most relevant groups can be selected by increasing the values of the meeting frequency and meeting time parameters. Finally, this paper presented a distributed version of GDC, which runs on mobile phones and achieves good group detection accuracy while protecting the privacy of the users.

Acknowledgment

This material is based upon work supported in part by the National Science Foundation under Grant Numbers CNS-0831753 and CNS-0834585. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] A. Gupta, A. Kalra, D. Boston, and C. Borcea, "Mobisoc: A middleware for mobile social computing applications," *ACM/Springer MONET*, vol. 14, no. 1, pp. 35–52, Jan 2009.

[2] C. Borcea and 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)*, Oct 2008, 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)*, Sep 2001, pp. 288–297.

[4] P. Hui, J. Crowcroft, and 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)*, May 2008, pp. 241–250.

[5] G. C. Homans, *The Human Group*. Harcourt, Brace and Company, 1950.

[6] A. Gupta, S. Paul, Q. Jones, and C. Borcea, "Automatic identification of informal social groups and places for geo-social recommendations," *International Journal of Mobile Network Design and Innovation*, vol. 2, no. 3/4, pp. 159–171, Dec 2007.

[7] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady, "Enhancing security and privacy in traffic-monitoring systems," *IEEE Pervasive Computing*, vol. 5, no. 4, pp. 38–46, Oct 2006.

[8] (2010) Place lab website. [Online]. Available: http://www.placelab.org/

[9] 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, Jun 2005.

[10] M. Newman, "Detecting community structure in networks," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 38, no. 2, pp. 321–330, Mar 2004.

[11] N. Eagle, A. S. Pentland, and D. Lazer, "Inferring friendship network structure by using mobile phone data," *Proceedings of the National Academy of Sciences*, vol. 106, no. 36, pp. 15 274–15 278, Sep 2009.

[12] N. Ravi, P. Stern, N. Desai, and L. Iftode, "Accessing ubiquitous services using smart phones," in *Proc. of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Mar 2005, pp. 383–393.

[13] E. Paulos and 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)*, Apr 2004, pp. 223–230.

[14] A. Madhavapeddy and A. Tse, "A study of bluetooth propagation using accurate indoor location mapping," in *Proc. of the 7th International Conference on Ubiquitous Computing (UbiComp '05)*, Sep 2005, pp. 105–122.

[15] M. McNett and G. M. Voelker, "Access and mobility of wireless pda users," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 2, pp. 40–55, Apr 2005.

[16] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and 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)*, Aug 2005, pp. 244–251.

[17] P. Hui, E. Yoneki, S. Chan, and 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)*, Aug 2007, pp. 1–8.

[18] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, no. 09, p. P09008, Sep 2005.

[19] M. Newman, "Analysis of weighted networks," *Physical Review E*, vol. 70, no. 5, pp. 56 131–56 140, Nov 2004.