Fall 1-31-2010

Social group discovery using using co-location traces

Steve Mardenfeld
New Jersey Institute of Technology

ABSTRACT

SOCIAL GROUP DISCOVERY USING USER CO-LOCATION TRACES

by
Steve Mardenfeld

Social information can be used to enhance existing applications and services or can be utilized to devise entirely new applications. Examples of such applications include recommendation systems, peer-to-peer networks, opportunistic data dissemination in ad hoc networks, or mobile friend finder. Social information can be collected from either online or mobile sources.

This thesis focuses on identifying social groups based on data collected from mobile phones. These data can be either location or co-location traces. Unfortunately, location traces require a localization system for every mobile device, and users are reluctant to share absolute location due to privacy concerns. On the other hand, co-location can be collected using the embedded Bluetooth interface, present on almost all phones, and alleviates the privacy concerns as it does not collect user location.

Existing graph algorithms, such as K-Clique and WNA, applied on co-location traces achieve low group detection accuracy because they focus on pair-wise ties, which cannot tell if multiple users spent time together simultaneously or how often they met.

This thesis proposes the Group Discovery using Co-location (GDC) algorithm, which leverages the meeting frequency and meeting duration to accurately detect social groups. These parameters allow us to compare, categorize, and rank the groups discovered by GDC. This algorithm is tested and validated on data collected from 141 active users who carried mobile phones on our campus over the duration of one month. GDC received ratings that were 30% better than the K-Clique algorithm.

SOCIAL GROUP DISCOVERY USING USER CO-LOCATION TRACES

by
Steve Mardenfeld

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer Science

January 2010

APPROVAL PAGE

Social Group Discovery Using User Co-Location Traces

Steve Mardenfeld

12/08/2009

Dr. Cristian M. Borcea, Thesis Advisor — Date
Associate Professor of Computer Science, NJIT

12/08/2009

Dr. Vincent Oria, Committee Member — Date
Associate Professor of Computer Science, NJIT

12/08/09

Dr. Reza Curtmola, Committee Member — Date
Assistant Professor of Computer Science, NJIT

12/08/2009

Dr. Adriana Iamnitchi, Committee Member — Date
Assistant Professor of Computer Science and Engineering, University of South Florida

BIOGRAPHICAL SKETCH

Author: Steve Mardenfeld

Degree: Master of Science

Date: January 2010

Undergraduate and Graduate Education:

- Master of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2010

- Bachelor of Arts in Sociology,
  Ithaca College, Ithaca, New York. US, 2005

Major: Computer Science

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and appreciate to everyone who has supported me.

First, I would like to thank my advisor, Dr. Cristian Borcea. This thesis would not have been possible without his guidance and support. I am indebted to his patience, knowledge, and encouragement.

Next, I would like to thank Dr. Reza Curtmola, Dr. Adriana Iamnitchi, and Dr. Vincent Oria for accepting to serve as members on my thesis committee. I am grateful for their time, questions and suggestions.

I would also like to thank Professor Quentin Jones, the director of the Smart Campus initiative, for providing support for running the data collection user study.

This work would not have been possible without the help of Daniel Boston, Susan Juan Pan, Gezhi Zhong, and Sara Gatmir Motahari.

Of course, I have to thank my supportive parents who have always been encouraging, no matter what context.

This thesis is based upon work supported by the National Science Foundation under Grants No. CNS-0831753, CNS-0454081, and IIS-0534520. Any opinions, findings, and conclusions or recommendations expressed in this thesis are those of the author and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

The incorporation of social networking information into online and mobile applications has started to become mainstream in the past several years. Social information can be expressed as the relationship between two individuals (i.e., social ties) or the relationship between multiple individuals (i.e., social groups). This information can be collected from either on-line sources (e.g., web sites, email) or mobile devices.

Not only has social networking become a genre of standalone websites, where the main objective is to allow users to interact with peers and friends (Facebook [48], MySpace [47], LinkedIn [49]) but social networking has been seamlessly integrated into most sites, regardless of their original intent. For example, both youTube [51] and Flickr [52] originally centered on user-driven content, but now incorporate social networking into their user's experience.

Yet, social networking has been used in more than just websites in a large panoply of applications and services. For example, Tribler [30] is a peer-to-peer collaborative download scheme that extracts online social data from users and adapts it to improve download speeds for its users. Community based forwarding schemes, such as LocalCom [17], are employed in ad-hoc networking to facilitate efficient packet dissemination. Mobius [34], a project developed in our lab, intends to utilize social information to improve network protocols and services for mobile social computing.

1.1 Problem Statement

This thesis focuses on identifying social groups based on information collected automatically from smart phones carried by mobile users. For the purposes of this thesis, a group is defined as a collection of users who spent a significant amount of time together during one or multiple meetings. Informally, the idea is to discover groups starting from group meetings identified from user location or co-location traces. The reliability of group detection using this type of information is higher than the self-reporting user data as user-reported social network data is historically troublesome [37]. This social information can be used separately, or in conjunction with other social information collected on-line, to improve applications, services, or protocols that involve groups of people.

Group discovery using data collected from mobile users is a difficult problem due to several factors:

i. Group members do not necessarily attend all group meetings.

ii. Guests or people that 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

v. Users may collect different data for the same group meeting.

This problem was initially solved by a different project at our lab using location traces with an algorithm known as Group-Place Identification (GPI) [35]. However, there were three issues which prompted the need for an additional algorithm. First, GPI requires a localization system on every mobile device, which, unfortunately is not always

available. Secondly, many users are often reluctant to share location traces for long periods of time due to privacy concerns. Even anonymous location traces are vulnerable to user identification through the application of data mining techniques, which can lead to target tracking and home identification [50]. Finally, Bluetooth-based data collection is expected to be more efficient from the point of view of energy consumption than GPS-based or WiFi-based solutions.

Another solution to the problem of social group detection is to utilize co-location information. Co-location information is a record of other users that are within certain proximity, at the same time, but does not contain data regarding exact location. This information is typically captured using the embedded Bluetooth interface, which is common on most mobile devices. Previous studies have employed this method of collecting co-location data. Reality Mining, is the largest of these studies and contains data regarding nine months worth of data for 94 different subjects [9] [10]. Using this data, P. Hui et al. created pair-wise links, thus effectively turning the co-location data into user graphs [16]. Then, using well-known graph algorithms, such as the K-Clique [28] and the weighted network analysis (WNA) method [23], was able to detect communities.

Unfortunately, there are a host of problems associated with using these methods for social group detection. The most important issue is their creation of graphs using a threshold for total time spent together by pairs of users to decide when to add an edge and the subsequent discarding of the rest of the data. The result is that there is no guarantee that their detected social groups spent any time together (i.e., the pairs of users may meet at different times). Furthermore, important parameters such as group meeting

frequency and total group meeting time are lost. Finally, these algorithms do not allow for weighted group analysis (K-Clique) or overlapping groups (WNA). Due to all these problems, these algorithms achieve low group detection accuracy.

1.2 Contributions

This thesis proposes a novel algorithm that leverages the meeting frequency and meeting duration to accurately detect social groups known as Group Discovery through Co-location (GDC). These parameters allow us to compare, categorize, and rank the groups discovered by GDC. GDC transforms the individual Bluetooth traces into meeting records between all pairs of users. Using these pair-wise meeting records, GDC discovers all permutations of users who appeared together at the same meeting. If these permutations meet the minimum criteria specified by the parameters and if they are not subgroups of other, more frequently meeting groups, then these permutations are considered final groups.

To test GDC, data was collected from 161 students of a medium-sized, urban university, 141 of which were considered active users. Subjects were instructed to carry a mobile device on campus, for a month, logging all co-presence interactions with other subjects of the study. Overall, a total of 384,512 discovery scans were completed, generating 404,105 co-presence records. Eight thousand one hundred and ninety-four unique Bluetooth devices were discovered overall, which includes 181 devices registered with the study (some of the subjects needed to swap phones while the study was in progress.)

To validate the results of GDC, a user survey was designed, which was deployed using Facebook. This survey prompted the users to blindly rate, using a five point Likert scale, the results of GDC and the K-Clique algorithm. Eighty-eight users responded to the Facebook questionnaire, 56 of which belonged to groups, composing 482 different ratings for 265 different groups. Of these 482 ratings, approximately 60 percent were for groups only from the K-Clique algorithm and 32 percent were for groups from GDC.

Examining these results, it was determined that GDC performed well, with approximately 62 percent of the ratings falling within the top two tiers of the rating scale, which corresponded to good or very good. Only 19 percent of the ratings were conclusively bad. The algorithm fares well when compared to the results generated by K-Clique, as K-Clique only generated 28 percent favorable ratings. The mean of all ratings was 3.62 contrasted against 2.73 for K-Clique. This difference between the algorithms is statistically significant at the .000 level.

Finally, this thesis takes a first step toward a distributed version of GDC by analyzing its feasibility. In this version, the phones store the co-location traces locally and potentially communicate with each other over the Internet. The main benefit of a distributed GDC is better privacy for the user, who does not need to share co-location data with a centralized "big-brother" entity. Additionally, such a version can improve GDC's resiliency and flexibility as there will be no central point of failure and users can run the algorithm each time they want. Our analysis shows, however, that a completely localized version does not achieve good accuracy due to the differences in the co-location traces among users (e.g., instead of detecting one larger group, the localized version could detect several smaller subgroups). Therefore, the users have to share data with

other users in the distributed GDC, but only with the users that have been recorded in her proximity; privacy is not a big issues in this case because users already saw each other at the meeting. Future work will determine the frequency and amount of data exchanged in the distributed GDC.

1.3 Contributors

Several students from the lab helped with the work. Gezhi Zhong and Daniel Boston implemented the Bluetooth scanning and data collection modules. Daniel also analyzed and provided statistics for the collected co-location data. Susan Juan Pan implemented the survey module. All of them also helped run the user study alongside Sara Gatmir Motahari.

1.4 Thesis Structure

The rest of this thesis is organized as follows. Chapter two presents an overview of the current methodologies used to capture social information and the various protocols and applications that utilize it. Chapter three presents a detailed description of the algorithm. Chapter four presents a comprehensive examination of the results of the experiment. Chapter five presents a discussion on the feasibility of a distributed version of the algorithm. Chapter six presents the final conclusion.

CHAPTER 2

RELATED WORK

This section provides an overview of the wide range of uses of social information in current services and protocols, covering popular Internet applications and budding mobile protocols alike. We then focus on the different technologies and methodologies used to catalogue the mobility of the social human and the various techniques to discover social groups.

2.1 Applications of Social Properties in Network Protocols and Services

The utilization of social information in Internet services and protocols has recently become mainstream. Not only are many of today's most popular and common websites based solely on their social networking capabilities, but other sites have begun to incorporate social information into their products. According to the market research firm, hitwise.com, for the week of 9/5/09 the top five most popular American websites were Google, Facebook, Yahoo Mail, Myspace, and Yahoo [1]. Two of these five sites, Facebook and Myspace, are exclusively online social networking sites (OSNs), while the other three sites incorporate the use of social information to better their services. Although Google does not currently incorporate social information as a method to personalize search results, other Google utilities, such as Google Reader, seamlessly integrate social communities with an RSS reader, allowing users to share and comment on articles that others have posted. Yahoo provides similar capabilities, allowing users to

establish connections with other users, automatically updating address books as contact information changes.

Social networks are used online to complement services, helping users organize and locate content. Sites such as Youtube [51] and Flickr [52] provide opportunity to peruse content through users channels and uploads, providing a simple way to search for similar material in addition to a tag-based system. Mislove et al. performed a simple experiment on a sample of photos found on Flickr and discovered that 80% of the views were a direct result of following links in the Flickr user graph or were from within a users set of photos [21]. Other sites such as Amazon and Netflix also provide the capabilities to friend and make recommendations to other users. Both Netflix and Amazon allow users to either friend other users, like most other online social networks, or become fans of other users, which does not require explicit permission of the other user. Within these communities, the user can view all activity made by their friends, including seeing all movies rented and rated in Netflix and wish lists in Amazon. This social information is distinct from the recommendation system in both services.

Tribbler, a peer-to-peer system, utilizes social information in order to establish groups of trusted connections. Users in these groups can then collaboratively download files as a group, thus speeding up the download process [30].

One of the first commercial attempts at exploiting mobile co-presence social connections was Lovegety, which was released in Japan in 1998. Lovegety was a device, which allowed users to select an activity of their choice, and would subsequently inform users of a possible nearby love match by beeping [42]. This idea was refined by Social

Net, a interest-matching application that detects patterns of co-presence with unknown users. If these two users share a mutual friend, then this friend is suggested to introduce them together [43]. Online social networks have been supplemented by physical mobility through the Serendipity application. This utility alerts users when they are within the presence of another whose profile and collected behavioral data is similar. Thus, strangers in the same location, who have no social ties in common, can become acquainted with each other based on mutual interests [44].

Applications such as Loopt [2], Dodgeball [45], Google Latitude [56], and Brightkite [3] are examples of location based mobile social networks, which are similar to online social networks, but are supplemented by the ability to share current location in addition to pictures and text. Researchers at the University of Massachusetts developed a multi-layered model that incorporates location, tags, and social information in order to accurately predict users who are likely to become friends [46].

MobiSoc, middleware developed by our lab, provides a common platform for sharing and managing social state information for mobile devices. MobiSoc allows developers to create applications that utilize social state information while maintaining privacy for users [36] [55].

Within the field of ad-hoc networking, Delay Tolerant Networks (DTN) has been a major focus, as new technology has greatly increased the volume of wireless computing devices [18]. Pocket Switched Networks (PSN) is a subclass of DTNs designed for mobile humans to communicate without network infrastructure [16]. Finding an optimal method for DTNs and PSNs to relay messaging between mobile humans is considered to

be a fairly difficult task, as there is no real way of knowing when a user will encounter another user with real accuracy. Nevertheless, there have been several different approaches to this problem that have successfully managed to forward packets throughout the network, with the most successful taking advantage of social information present in the networks [18].

Prophet is a probabilistic routing protocol that uses a delivery probability metric, which captures the likelihood that a certain node will be able to deliver its message to its final destination. As a node encounters other nodes, the delivery probability metric of both nodes is compared, and if the new node has a higher delivery probability metric, then the message is passed on. As nodes continuously detect each other, they share summary data on the eventual status of previously passed messages in order to continuously calculate accurate delivery probability metrics [18]

Based on the premise that pocket switched networks are governed by the mobility of humans, Bubble Rap expands upon the concept of delivery probability by integrating it with social community detection. A common metric in social networks is centrality, which measures how close to the center a certain node may be [33]. Bubble Rap is able to select nodes with a high centrality to be used as the major relays for dispersing packets by utilizing communities in a given network. Bubble Rap employs a tiered system and there are two different metrics for centrality: global and local. So for inter-community communication, the message will be pushed to the gateway with the higher global centrality value, whereas the message will be pushed to the nodes that have higher local

centrality for intra-community communication. In layman's terms, the popular people represent the most likely shortest path between two people [16].

As an improvement, Localcom [17] proposed a distributed community detection method and gateway pruning schema based on the real-time analysis of the encounter histories of each node. Their results clearly show that the delivery ratio can be improved by the utilization of social properties in DTN.

Social information has become so successful in ad-hoc networks that mobility models now generally include them. Within these studies that utilize simulation data, there are different methodologies for generating a dataset composed of co-location traces. Classic models include random walk, random waypoint, random direction, city section, column mobility, pursue mobility, and reference points [5]. Yet there are also models that are more specific to human social network theory. One such example first generates a weighted social relationship between all other users in the simulation and then generates a random waypoint model that is influenced by these social relationships [22].

2.2 Inferring Social Properties from Mobility Traces

Social data can currently be extracted from users through three main methodologies: having users declare it, either online in OSNs or in a survey; analyzing interaction patterns in online social networks; or analyzing user interactions in real life through mobility traces. Simply using social data declared by users in online social networks, such as Facebook or Myspace, can lead to errors due to hidden social pressures of friendship declarations. Users do not frequently delete relationships, allowing for increased stability

for graphs constructed from these types of declared user friendships [12]. This is clearly evident as Facebook users routinely have hundreds of publicly declared friends. Clearly all of these online friends display varying degrees of friendship, which can be ranked through their interactions. However, not all of these online social networking sites focus on socializing, instead they can revolve around specific topics, which can influence the extracted relationships.

Yet mobility traces, collected from mobile phones, are not without their problems. These include distinguishing between co-location and social interaction in crowded environments as well as the possibility of users not maintaining full contact with their own phones. Nevertheless, mobility traces can provide an accurate measure of social groups, as they can determine co-presence and the total amount of time spent together by members of a group.

The differences between social graphs extracted from user reported data and user interaction data have been studied within the context of delay-tolerant ad hoc networking. These comparisons focus on the gaps that would affect routing, rather than the inclusion and absence of different groups. There have been at least two separate studies that have investigated this issue. The first, conducted by Mtibaa et al., compared social graphs, compiled by surveying the friendship status of each subject with every other subject in the study, to contact graphs, which represented inter-contact time between pairs of subjects. This showed that subjects generally spend more time with those that are declared friends, than they do with other subjects, therefore providing similar results between the different social graphs [39]. However, this experiment was performed at a

conference environment, over the course of a single day, therefore these results cannot truly be broadened to anything other than a contained event. Of course, it is expected that subjects will spend more time with their friends when they are given an option, but unfortunately, in real life, free time is diminished by obligations, such as work. That being the case, one would imagine spending more time with fellow employees than with one's friends. Expanding upon this idea, researchers from the University of St. Andrews compared social graphs gleaned from Facebook to a contact graph collected over the course of 79 days. Conversely, their research showed that these two graphs are noticeably different in regards to both role and structural equivalence, which are two popular techniques from traditional social network analysis [40]. Therefore it is expected that social graphs will be different depending on which method is used to create them, despite the fact that subjects can be expected to choose to spend more time with their self-declared friends when that is an option, such as when they have a free moment during a conference.

Using existing mobile technology in an effort to capture social data is not a new idea. Historically, there have been two different methods of capturing this information through mobile devices, through the collection of location or co-location traces. Location traces retain the location of each mobile device, either through GPS, Bluetooth localization systems, or triangulation based on access points/base stations. These location traces can be used to place users within the same area at the same time, however the accuracy and viability of these techniques vary. Unfortunately, GPS does not work indoors and Bluetooth localization systems require the installation of static Bluetooth stations in

order to work, requiring significant investment [26] [29] [27] [14] [4] [19]. Nevertheless, there have been many projects that have successfully employed location traces using a variety of methods. Office buildings have been equipped with static Bluetooth dongles, which work well but require an intensive training phase in addition to the costly setup [14]. Other projects have utilized mixed methods by incorporating location traces alongside co-location traces [26] [29].

On the other hand, a set of co-location traces is a collection of every visible device that is discovered by the mobile device using a Bluetooth discovery program. This method has several advantages including: increased co-presence accuracy due to the smaller range of Bluetooth (10m), better scalability as there are no infrastructure requirements, less reliance on persistent internet availability, and better privacy protection as the location of individual users is not stored. The results of co-location traces are simply other Bluetooth devices in range at a certain time, whereas location traces result in the absolute location of each user at a certain time. With location traces we can quantify the distances between users in a meaningful way, however the absence of such data means that users privacy is better protected, as something that is not collected cannot be compromised.

From a privacy point of view, one could imagine that the centralized entity can try to detect the absolute location of the users from the co-location traces. For example, the centralized entity can find out the location of a few fixed devices that appear in the co-location traces (i.e., "landmarks" in the language of localization researchers). Localization methods based on landmarks [54] have been proposed in the literature for

determining the position of nodes in an ad hoc network. However, these methods do not work in this type of Bluetooth environment. First, only a few smart phones would be in multi-hop connectivity range from the landmarks (i.e., a potential Bluetooth-based smart phone ad hoc network is sparse and disconnected across large areas). This is because it is assumed that the central entity does not deploy a very dense network of Bluetooth devices for tracking purposes. Therefore, only the position of very few smart phones can be determined with any accuracy. Over time, their old location can be used only as a very rough estimate of their current location (i.e, the user might have walked, biked, or drove away in any possible direction). Second, the landmark-based methods require orientation or angle-of-arrival information from the smart phones in order to achieve good location accuracy, but the smart phones do not provide this type of information. Therefore, absolute user location is relatively well protected in the general case.

Unlike location-based solutions, where the mobile device collects data by itself, co-location based solutions involve one-hop ad hoc communication between devices. Thus, one potential problem could be that a co-location trace for a mobile user can be empty or have little data because the Bluetooth discovery process takes about 20 seconds. However, the user is not in a group meeting in such a situation, and therefore, the lack of data is irrelevant for group discovery algorithms.

Other means have been used to detect co-location traces besides Bluetooth. For example, Hummingbird employs short-wave radio frequency in order to detect the presence of other members of a social group. Once detected, the detection device notifies the user by emitting a sound, allowing the user to notice other nearby group members in otherwise impossible situations [14].

Other studies have taken a completely different approach, such as focusing exclusively on the mobility models over users in a specific network by examining a network's traces. The Wireless Topology Discovery project at UCSD collected wireless trace data for PDAs assigned to 275 students modeling their movements and wireless activity throughout campus for 11 weeks [20]. Similarly, on the Dartmouth campus, researchers have measured the use and traffic on their wireless local area network (WLAN) and have determined that most users generally do not stray from their home location [13].

There have been several projects, all of which have utilized co-presence information in order learn social properties, that have been conducted by researchers at the Intel Research Laboratory, the University of Cambridge, and MIT.

Researchers at Intel Research Laboratory in Cambridge [15] and the University of Cambridge distributed iMotes, a pocket-sized Bluetooth device, which logged all visible Bluetooth devices every two minutes for eleven days. Participants initially included students and researchers at Cambridge [15] and then was subsequently expanded to cover a conference environment (InfoCom06) [6]. This follow-up study followed the same discovery procedure involving iMotes, but drastically increased the number of participants for a duration of four days. Contact times for users in these experiments followed a power-law distribution where there were more contacts for a shorter duration. More applicable to social networking was the finding that users were more likely to pair with specific users over time than with other random users, most likely because the users were adhering to their underlying social network.

Similarly, researchers at MIT deployed one hundred smart phones to users over the duration of an academic year (nine months), logging all visible Bluetooth devices every five minutes, as well as call logs, cell tower IDs, application status, and phone status in the Reality Mining project. By calculating the entropy of specific users, they were able to generate a probability, with accuracies of up to 90%, that a user will come into contact with another user within a certain timeframe. By converting the pair-wise meetings into a time series and then by converting this time series into a frequency domain, Reality Mining was able to show that the two most popular meeting frequency was one day, closely followed by seven days. Reality Mining also showed the differences between friends compared to frequently-seen work acquaintances can largely be attributed to specific timeframes. For example, most users see friends most often outside of the workplace, even though they may see them often at work [9]. By analyzing this proximity data in conjunction with location and phone logs, Reality Mining was able to develop the behavioral characteristics of friendship according to their users [10].

There have also been numerous studies that investigate social patterns of the mobile human without employing Bluetooth technology; rather they focus on modeling and simulating human mobility and social behavior. One such study uses student class schedules from the National University of Singapore to model the expected contact times between other students over the course of a single semester [31].

2.3 Inferring Social Groups

The process of finding and generating communities in a graph is a widely-studied problem and has applications in many disciplines, including computer science, physics, sociology, and anthropology. Traditionally, these general community-detection techniques have been applied in a similar fashion to the problem of group discovery in Bluetooth generated data. The typical method for deriving social information from a graph is done by transforming pair-wise co-presence data into a graph and then deciding whether users should be linked or not. It is from this graph that social properties can be extracted through analysis.

Three popular algorithms that are commonly used for inferring human communities in social graphs are the K-Clique method, Weighted Network Analysis (WNA), and Spectral Clustering [16] [32]. These methods are complementary in their discovery of human communities as they utilize different methods as well as focus on very different aspects of communities.

A clique is defined as a sub-graph in which every vertex is connected to every other vertex in the graph, and a K-Clique is a component made of complete sub-graphs of K vertices [7]. The K-clique algorithm is particularly useful for examining social networks because of its inherent ability to discover overlapping communities, which mimics the community structure of real human networks. Unfortunately, the K-clique algorithm was not designed for weighted networks; rather, it was designed with simple binary networks in mind. Unfortunately, social network data is not binary in nature, as groups time can greatly vary in both quantity and quality. In order to utilize this algorithm on social data

gleaned from Bluetooth devices, the data must first be examined and thresholded, thus ignoring all subtleties present in the data [28] [16]. This process removes the option to compare cliques to each other, as cliques can either be considered groups or not groups, but within these groups there is no defined ranking. This is clearly incorrect as co-workers and family are both social groups but they are certainly not on the same social ranking.

FAST [24] [41] and WNA [23] [11] are both hierarchical methods for detecting communities in graphs that can be easily modified to deal with weighted networks. They both are driven by a property of networks known as modularity, which is the difference between the current fraction of the edges within the communities and, the fraction of the edges that would be expected to fall within the communities if the edges were assigned randomly and the degrees of the vertices unchanged. The guiding principle for these algorithms is the maximization of this modularity. Yet, these algorithms are also inappropriate for social group detection as they do not allow the discovery of overlapping communities. Unfortunately, overlapping communities are a major facet of social groups and by only allowing a user to belong to a single group, we exclude many features of a true social network.

Spectral clustering [32] is another well studied and widely used clustering mechanism that can be used to discover communities in social networks. By arranging all of the members of a graph into a matrix, based on a weight, typically the distance between vertices, one can apply the k-means clustering algorithm to the top eigenvectors of this matrix in order to segment the graph into communities [32] [8] [24]. Unfortunately, this algorithm is also unable to cope with overlapping communities.

There is, however, another more specific method for deriving social groups from location traces. Group-Place Identification (GPI), an algorithm developed in our lab, solved this problem by using location and time to discover group meetings and then calculating whether a user attends at least the minimum amount of required meeting time to be considered in that group. This is a necessary step, as not all members of a group will attend every meeting [35]. The end result of this algorithm is the set of all groups discovered by the location traces. By substituting co-location traces for location traces, the algorithm developed in this thesis can be used in place of GPI.

The author was compelled to develop GDC to provide a single solution to the problem of community discovery that is specific to human social networks. This is necessary because, unfortunately, social networks are dissimilar from other types of networks by their increased network transitivity and their natural proclivity for multiple communities [25]. Furthermore, the process of transforming these Bluetooth traces into a social graph results in a loss of valuable information which is extremely pertinent to human communities, specifically co-presence between many users. The notion of real-life social communities inherently implies that everyone in the community be present at the same time, however this information is lost in all current forms of community detection algorithms, leaving communities that contain people who were never co-present together. Perhaps this transitivity is appropriate when dealing with communities of very strong ties, such as family or friends, however it is inappropriate when considering all people that a person spends time with, because we are capturing all social relationships: strong and weak; familiar, stranger, and somewhere in between.

Theoretically, it could be possible to analyze all the groups generated by a method such as K-Clique that allows for overlapping groups and check if their members spend enough time together to be considered groups according to the definition presented in this thesis. However, the verification becomes relatively complex for larger groups that could end up with many subgroups that spent time together. Since this could be as complex as GDC, it is preferable to start from a clean slate.

CHAPTER 3

ALGORITHM DESCRIPTION

This chapter describes the GDC algorithm. First, the input data and the methods for its collection are presented. Then we define the basic concepts used in GDC and briefly overview the algorithm. Finally, each of the four GDC phases is described in further detail. The first phase creates pair-wise meeting records, the second phase creates the user clusters, the third phases deals with clusters that are viewed differently by group members, and the fourth and final phase selects the final user groups.

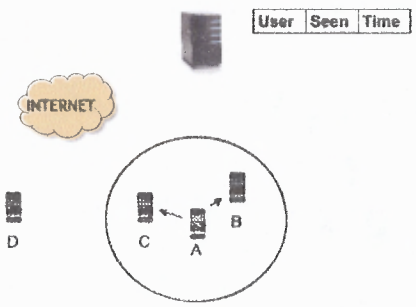
3.1 Input Data

The GDC algorithm discovers social groups from a set of co-location traces. These co-location traces are collected through a Bluetooth discovery program, which runs on the mobile devices of all users.

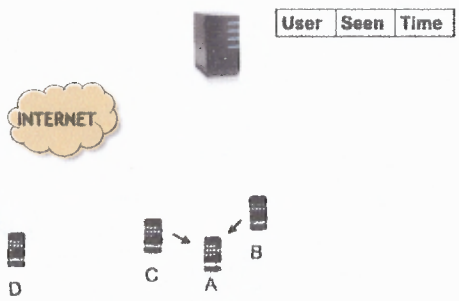
Bluetooth devices are proficient at detecting other nearby Bluetooth devices through a device discovery mechanism. This mechanism allows a Bluetooth device to collect specific information, including MAC address, device name, and device type for all other Bluetooth devices that have not disabled this feature, within a range as large as 10 meters. By exploiting this knowledge, one can systemically query all nearby Bluetooth devices to determine what other devices are within range at a certain time [9].

This data is typically retrieved from the user at certain intervals and sent to a centralized server. Every instance of a detected Bluetooth device is recorded as a

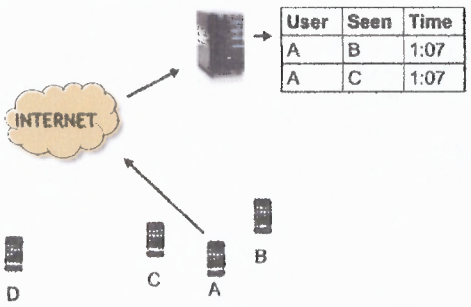
Bluetooth record, which includes the MAC address of the initiated device, the MAC address of the seen device, and a timestamp. Because GDC does not consider devices that do not belong to unknown devices, these records can be discarded (i.e., the assumption is that only known Bluetooth addresses/users are considered by GDC). Thus the input data for the GDC algorithm is the set of each user's Bluetooth traces, where a trace is defined as the set of all records for an individual user.



User A initiates discovery process with all other devices within range



User B and user C respond to user A's discovery request



User A sends discovered user information through the Internet to a central server. Each discovered user is stored as a Bluetooth record.

Figure 3.1 Using Bluetooth device detection to discover nearby users.

Figure 3.1 illustrates the process of utilizing Bluetooth to create a set of Bluetooth traces at a centralized server. In this example, user A initiated Bluetooth discovery for all devices within its Bluetooth range. Both user B and user C receive the request and respond. User D is not within transmission range; hence, it does not receive nor respond to the request. Once user A receives the results, it transmits them through the Internet (e.g., using WiFi connectivity to an access point in our experiments) to a centralized server. Each detected user becomes a separate record.

3.2 Definitions

For a clear understanding of GDC, it is necessary to define several concepts used throughout this chapter.

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 for the same Bluetooth discovery scan, then N Bluetooth records will be created.

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

A *pair-wise meeting record* (or simply pair-wise meeting) is a collection Bluetooth records without time gaps between them for the same pair of users. The time gap is function of the Bluetooth discovery frequency. This definition extends to *cluster/group meetings* (or simply meetings) when all user pairs in the group 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 collection of users, who have one or multiple meetings together.

A *group* is a cluster of users who spend a significant amount of time together throughout a dataset over a series of meetings. Not all members of a group have to be present for every meeting, but they must be present for a specified percentage of them. A set of groups can be overlapping and thus share members, but each group must be notably different from each other in terms of total time spent together. A cluster becomes a group if it fulfills all of the necessary requirements, defined by GDC parameters, and if it does not significantly overlaps other groups.

3.3 GDC Overview

Given a set of pair-wise co-location records, GDC identifies groups of users. For this purpose, a group is defined as a collection of people who are all together for a significant amount of time during one or multiple meetings. Naturally, groups may share users, and consequently, groups can be overlapping. Group members must be present at one or multiple group meetings together. Specifically, this means that group members must be in the Bluetooth transmission range (ten meters) of each other for each group meeting. However, the groups outputted by GDC are notably different from each other, whether in terms of members or total time spent together throughout the period in question. Not only will GDC find all groups that spend at least a specified minimum number of time together overall for at least a specified amount of meetings, but these cumulative meeting times and meeting frequencies will be maintained in the final output. Therefore, groups can be compared, categorized, and ranked.

Phase 1: Creating pair-wise meeting records

```
1. Sort union(bluetooth records) by user, userwith, time
2. Set endtime = time[0], starttime = time[0]
3. For each record n
4.      If user[n]==user[n-1] and user_with[n]==user_with[n-1]
5.              If time[n]-time[n-1] > MG
6.                      Add user[n-1], user_with[n-1], starttime, endtime as meeting record
7.              Else
8.                      Starttime = time[n]
9.      Else
10.             Starttime = time[n]
11.     Endtime = time[n]
12. Add meeting record
```

Phase 2: Creating User Clusters

```
13. For each user x
14.     Set lasttime = 0
15.     Set currentwith = heap( [user,endtime] keyed on endtime)
16.     For each record r(x)
17.             While heap[0] <= record(starttime)
18.                     Create/modify clusters for All_Combinations(users in heap, user x)
19.                             time=heap[0][endtime]-lasttime
20.                             Remove heap[0]
21.             Create/modify clusters for All_Combinations(users in heap, user x)
22.                     time=record(starttime) – lasttime
23.             currentwith.add( [record(user),record(endtime)])
24.             Lasttime = record(starttime)
25.     For each cluster
26.             Keep cluster if total_time(cluster) >= MG

All_Combinations(userlist,mainuser)
27.     Set r = 2
28.     While r < length(userlist):
29.             output each combination of size r + mainuser
30.             r+=1
```

Phase 3: Dealing with Clusters Viewed Differently by Their Members

```
31. for each user x
32.     for each cluster(x) c
33.             if min(groupmeetingtime for each user in c) > MGT AND Group Meeting Frequency(least
                frequent user in c) > MGMF
34.                     Add cluster c to globalclusters
35.                     c_total_time = min(groupmeetingtime for each user in c)
36.                     c_min_user = user with min(groupmeetingtime)
37.             else
38.                     Delete c
39. Delete user clusters
```

Phase 4: Selecting the User Groups

```
40. Sort globalclusters by size of group (max to min)
41. Set finalclusters = []
42. For each cluster(i) in globalcluster:
43.     For each cluster(j) in finalcluster:
44.             If all members of cluster(i) in cluster(j) and time(i)*GTP< time(i): Break
45.             If j==len(finalcluster): Add cluster(j) to finalcluster
```

Figure 3.2 Pseudo code for the GDC algorithm.

Group Discovery through Co-location discovers these groups through 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 between all permutations of all users who appeared together over each different meeting. The third phase creates a list of all clusters where each member of each cluster detects that group and eliminates all groups that do not meet the minimum group frequency threshold. The fourth and final phase takes this list and identifies the groups differentiating them from any subgroups.

The complexity of GDC is $O(R * 2^L)$, 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 $R = Const * MR$ (the number of meeting records). The complexity is not as high as it seems because L is a relatively small number (the number of users in a 10m range with each other). More details and the complexity analysis for each phase of the algorithm will be presented in each of the following sections.

3.4 Creating Pair-Wise Meeting Records

The GDC pseudo-code is presented in Figure 3.2. The goal of the first phase is to identify pair-wise meetings between users and the durations of these meetings. A pair-wise meeting record represents one meeting between two users and contains the identities of these users and the total time they spent together during that meeting. The algorithm goes linearly through all the Bluetooth co-presence records (periodic user-centered reports of other users "seen" in proximity) to create pair-wise meeting records.

Each meeting record represents a collapsed timeframe for several Bluetooth records. Each Bluetooth record can be thought of as a probe, observing all surrounding users at a specific time instance. However each individual probe is useless by itself; rather, it is the collection of all the probes that informs us whether two users spent any significant time together at all. For example, two users could be passing each other in the hallway and a Bluetooth record could be generated. Yet, these two users spent only a few seconds within range of each other. Therefore, it is the chaining of these traces together that can tell us the real story. Since users can easily leave each other's Bluetooth range for a minute and not appear in the next record, it is important to have a time buffer such that a probe is considered part of the same meeting as long as a user re-appears within a certain timeframe.

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 so that missing a single Bluetooth scan would not result in two different meetings, but small enough to capture real changes in group structure.

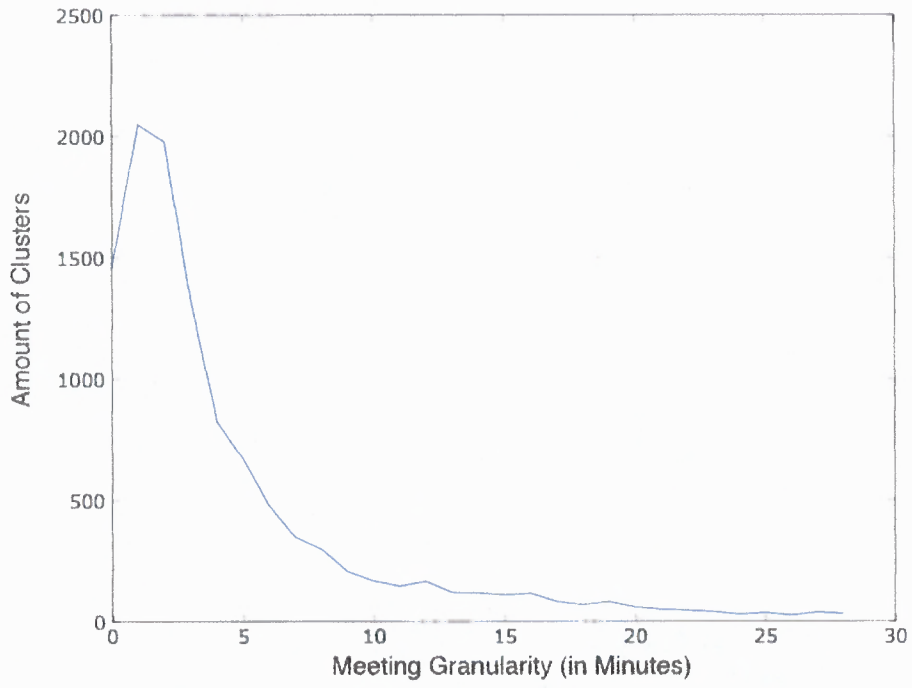


Figure 3.3 Number of clusters function of meeting granularity. After an initial peak, the number of clusters quickly drops and then levels off. The initial peak is due to the time involved in querying other Bluetooth devices.

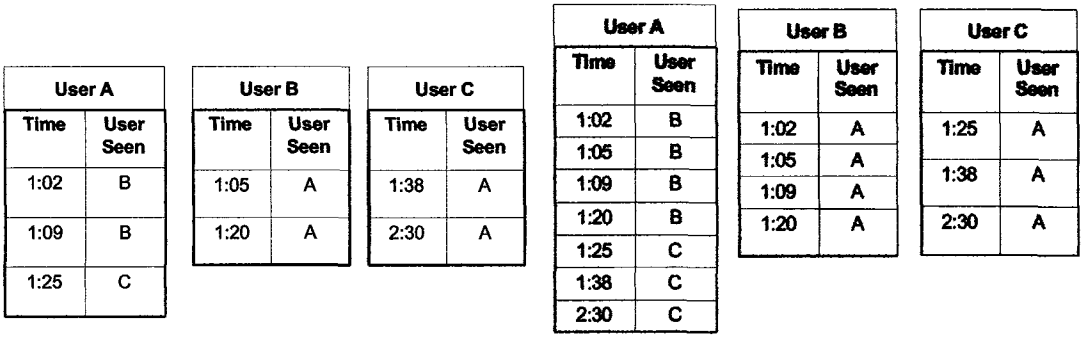
We determined the value of MG empirically, based on the quantity of clusters that would incur with different granularities. Clusters (created in GDC's second phase) are collection of users that were co-present for a certain duration during the study. However, these clusters are not the final output groups of the algorithm; rather they represent the potential for a collection of users to become a group. The fourth phase of the algorithm determines what clusters are groups, until then each potential group is referred to as a cluster. For this study, MG was set to 15 minutes. At 15 minutes, the number of additional clusters started to level off. By choosing a higher granularity, such as 25 or 30, more clusters would be included; however those granularities seemed too high as they

have the potential to link together two separate meetings. Figure 3.3 shows the additional number of meetings discovered compared to the previous granularity. This graph also shows that the vast majority of Bluetooth traces for a meeting occur within the six-minute range, however a granularity higher than six minutes was chosen as some user's Bluetooth scans occur once every four minutes. Therefore missing a single scan would erroneously result in two different meetings.

The same MG value can be applied to other datasets as long as the Bluetooth discovery frequency remains similar. Since MG is inherently linked to this frequency, however, lower frequencies will lead to higher MG values (and vice-versa).

Two users can have different views of the same meeting. This is due to the randomness of the periodic Bluetooth discovery and the varying distance between users (in the ideal case, they should report the same data). Therefore, GDC has to unify the views of two users about the same meeting by taking the union of the individual views. At the end of this phase, the output is all meetings between every pair of users (ordered by time).

Figure 3.4 illustrates the transformation of individual Bluetooth traces into unified meeting records. The first set of records shows that each user has a slightly different perspective of each encounter. The second set of records shows a unified view of these encounters, so both users have a record of either time one of them detected each other. The third set of records shows the meeting records for user A. These records are greatly condensed compared to the initial co-location records.



User A	
Time	User Seen
1:02	B
1:09	B
1:25	C

User B	
Time	User Seen
1:05	A
1:20	A

User C	
Time	User Seen
1:38	A
2:30	A

Co-Location Records for Each User

User A	
Time	User Seen
1:02	B
1:05	B
1:09	B
1:20	B
1:25	C
1:38	C
2:30	C

User B	
Time	User Seen
1:02	A
1:05	A
1:09	A
1:20	A

User C	
Time	User Seen
1:25	A
1:38	A
2:30	A

Unified Co-Location Records for Each User

User Seen	Start Time	End Time
B	1:02	1:20
C	1:25	1:38

Meeting Records for User A

Figure 3.4 The transformation of Bluetooth traces into meeting records with a 15 MG.

The final number of meeting records depends on the density of the data. The worst case would be many meetings of groups that only met for the duration of the MG. For the experiment's dataset, there were 71,677 initial Bluetooth records resulting in 8,522 meeting records.

The computational complexity of this phase is O(R) where R is the number of Bluetooth records.

3.5 Creating User Clusters

The goal of the second phase is to analyze 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 or kept separate. Figure 3.3 illustrates this point. This information will be derived later based on the total meeting time of a user cluster and its number of meetings. In the end, a user cluster could form a group (if the users met for a certain amount of time), could be part of a group (i.e., it is possible that some group members miss some meetings leading to multiple user clusters that belong to the same group), or could be discarded if its total meeting time is inadequate.

Total meeting time is determined to be inadequate or not according to a threshold known as the minimum group time (MGT). This parameter determines the minimum time that a cluster must spend together in order to qualify as a significant cluster and should be changed according to the length of time that the data is collected from. For this experiment, we set MGT to 2000 seconds (approximately 30 minutes). This was determined by analyzing the total number of groups the algorithm would result in with different MGT values.

For each user, the algorithm goes linearly through its meeting records and constructs the clusters in which this user is a member, calculating how much time this particular user spends with every permutation of every cluster (pseudo-code lines 16 to 23). However, only other users who spend at least MGT with the user in question need to be considered; all clusters that include pairs that meet for time values lower than MGT are removed. Because each user's device can discover other Bluetooth devices only within a certain radius, each user will have a different perspective of their surroundings, unless all members of a cluster are standing within a very close proximity. Therefore it is necessary to calculate cluster meetings times for each user.

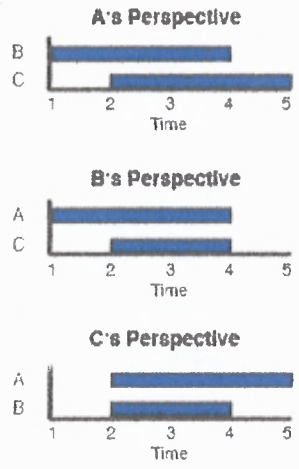


Figure 3.5 Co-presence for three users according to each user's perspective.

For example, Figure 3.5 shows that according to user A's perspective, A was with user B from one to four and with user C from two to five. For each of the clusters, we calculate the total time spent together, so for our example we have the following results (AB:3, AC:3, ABC:2). But because the amount of all possible clusters can be quite large, we only calculate the total time that the clusters spent together for clusters that we actually encounter in the data. Therefore, for this example, BC is excluded. According to user B, we find the following results (AB:3, BC:2, ABC:2). AC is excluded because B is not a part of that clusters.

These values are updated for each new meeting record processed. If the total time for a cluster at the end of this phase is less than MGT, then we discard this cluster (pseudo-code line 26).

At the end of this phase, the algorithm outputs for each user every cluster in which this user is a member. Note that at this phase, each user has a localized version of each cluster, as each user can have a different perspective on group meetings.

The computation complexity of this phase is $O(MR * 2^L)$, where MR is the total number of meeting records and L is the number of users together at any one time. MR is upper-bounded by R, the total number of Bluetooth records. Therefore, the complexity becomes $O(R * 2^L)$. Although this complexity is exponential in L, the value of L is relatively low because there are a limited number of users that 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.

3.6 Dealing with Clusters Viewed Differently by Their Members

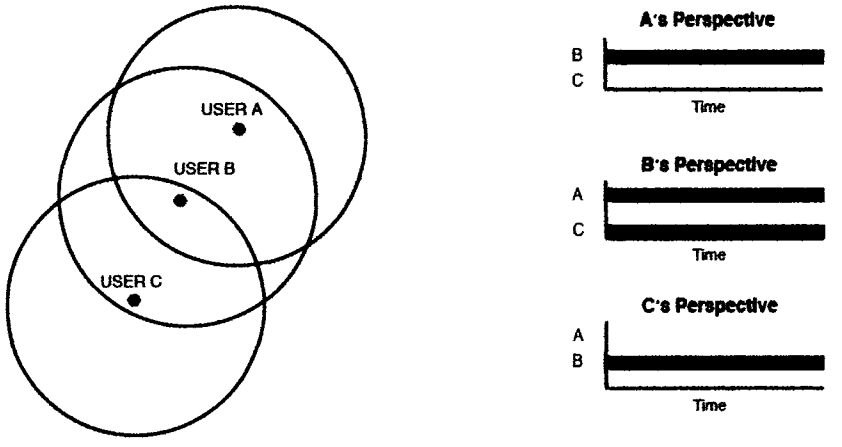
The goals of the third phase are to consolidate each user's perspective of clusters into a global view of all clusters and to eliminate all clusters that meet with a frequency less than a specified parameter known as the minimum group meeting frequency (MGMF) parameter.

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 attented all meetings for that cluster. Therefore, only clusters that are consistent for every member are kept.

By consolidating each Bluetooth record into meetings, the algorithm removes much of the ambiguity involved with fleeting encounters, but it is still left with the problem of continually linking multiple users together whose Bluetooth devices cannot all see each other. This can happen when a centrally located user can see other users, who because of their location cannot see each other. Therefore this user has information regarding a cluster that not all users have. Note, however, that if a cluster meets with a

certain frequency, GDC will be able to detect it as the users have different relative positions at the meetings.

In order to ensure that each cluster is representative of a set of users who were all together for at least MGT, we create a global view of all groups. In this global view, the total time spent together by a group is set to the minimum value recorded by the individual group members (pseudo-code lines 33 to 36). Since meetings between two users are transitory, this will only affect users that are linked through a third user. Clusters that contain a user who does not see that cluster at all are removed at this point.



Three Users Linked Together Resulting Perspectives For Each User

Figure 3.6 The resulting perspectives of three users who are chained together by a central user.

Unfortunately, this means that groups whose members are located within a range larger than our Bluetooth range, such as a large auditorium or a restaurant, are not detected (the positive aspect associated with this issue is that that value of L cannot be large). This is a tradeoff as the goal of this algorithm is to find social groups based on co-

location; by including groups that span multiple users, GDC would erroneously merge two different groups that meet at the same location into one large group. This method guarantees that all groups discovered are real groups and not just two separate groups linked together by one central user. However, sprawling groups that met infrequently and share common users during those meetings (such as research talks in CS departments) can result in several smaller groups rather than one large group. 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 users.

Figure 3.6 demonstrates a typical scenario and the resulting views of each user. According to user B, the cluster ABC exists. However, neither user A nor user C has any recorded meetings of cluster ABC; hence, the cluster is not considered a valid cluster and is not added to global view.

Before adding a cluster to the global view, it must be ensured that a cluster meets more often than MGMF. This is based on the view of the least frequent member of that cluster. If a cluster meets more than MGMF, added to the global view; otherwise, it is removed.

MGMF is a necessary attribute for determining the importance of social groups. This is true for several reasons, including the fact that some groups may meet for a long period of time once and then never spend time together (e.g., weekend retreat, wedding, or conference). Other groups may meet very frequently for minimum amounts of time, such at the local coffee shop in the mornings. By including group meeting frequency in GDC, the MGT threshold can be relaxed and a better sense of frequent but brief encounters is retained, thus providing a more accurate view of social groups.

It is also necessary to capture group meeting frequency as the total group meeting time is not related to the group meeting frequency. One would imagine that groups that meet often would tend to have fairly stable meeting times. This is not the case. A simple analysis from the experiment's dataset shows that pairs that meet more than once maintain a fairly high level of volatility in the length of their meetings. To calculate this, the mean and standard deviation of the total length of meeting (in minutes) were calculated, excluding meetings that were shorter than five minutes, for all pairs. Figure 3.7 depicts this standard deviation of each pair as function of the total number of meetings for that pair. As expected, fewer meetings maintain a high standard deviation, but so do the meetings that occur more often. In fact, one of the highest pairs that had the biggest difference in meeting times was a pair that met 18 times. In order to put these statistics into perspective, the average standard deviation is 23.5, whereas the average mean turned out to be 31.8. What this means is that pairs that frequently spent long periods of time together, also had brief encounters. Therefore, the total group time cannot exclusively be used as an indicator of the frequency of group meetings.

The result from this phase is the set of all groups, where the time each group spent together is dictated by its least frequent member.

The computational complexity of this phase is $O(N*2^L)$ where N is the total number of users in the dataset and 2^L is the maximum possible number of clusters for each user. However, in practice the amount of valid clusters attached to each user is quite small as many of the original clusters generated in the second phase meet for less than MGT, and therefore they have already been excluded. For example, in the experimental

dataset, there were 1,415,233 total clusters but only 22,510 clusters with a total time greater than MGT.

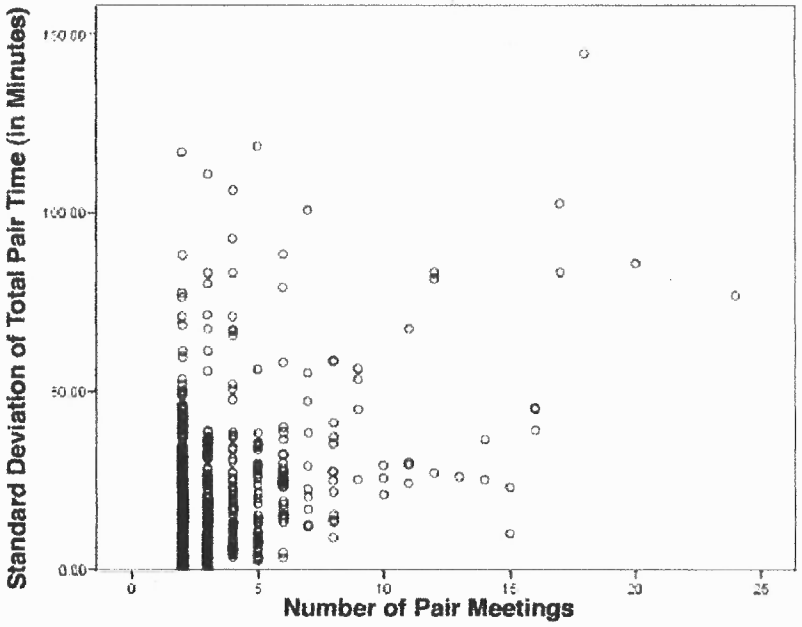


Figure 3.7 Standard deviation of total pair time in minutes by the total number of pair meetings. Regardless of whether or not pairs met many times, the average pair had a standard deviation of 23.5 minutes per meeting. This is extremely high as the average meeting time was 31.8 minutes long.

3.7 Selecting the User Groups

The fourth phase of the algorithm is to determine whether each of these remaining clusters should be considered a significant group or a subgroup of another significant group. This step is necessary for two reasons. First, the collected information is from all group permutations, such that if all users of a specific group only meet when all members are present, then each permutation of that group has the same total meeting time as the entire group. Therefore, each of these permutations is a subgroup of the entire, original group. However, this situation is unlikely as most people will not arrive at a location at

the exact same time, allowing early users to accrue more time together than the entire group.

Another problem is that groups that meet frequently, such as classes or sports team, will often not contain all participating members at individual meetings; rather, different users will be missing from different meetings throughout the course of the data. For that reason, it is necessary to build a tolerance of absences into our algorithm.

However, we must also allow for smaller subgroups of a group to be considered their own group if they meet much more frequently than any of the other members of the original group. This clearly happens in social settings with families, as a couple could share multiple groups together and could be considered a group in their own right. The total time a group spent together is used as a weight to determine whether a group should be considered a subgroup.

Therefore a group B is a subgroup of A, if (1) B contains less members than A but each member of B is also a member of A, and (2) B's total cumulative time multiplied by the group time percentage parameter (GTP) is less than the total cumulative time associated with A. 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 separate group (pseudo-code lines 44 to 47). This is necessary to determine whether a subgroup is meeting frequently without the main group or whether those users simply arrive earlier to regular group meetings. We set GTP to 0.5, such that if a smaller group, B, meets more than twice the time of a larger group, A, then we do not consider B to be a subgroup of A. The smaller group, B in this instance, always has a total time equal

or greater than the larger group, A in this instance, due to the fact that every time we record a meeting of A, we must also record a meeting of B.

Note that GDC avoids the potential issue of having to merge two overlapping clusters, which have both common and different users. Because all possible combinations of clusters are generated during the second phase, if two groups have some users in common, there will always be a group that is the combination of both of those two groups.

The output of this phase of the algorithm is the set of all unique groups for all users. The total time that each group met throughout the dataset is not lost through this process. Therefore, groups can be further refined without re-implementing the algorithm in its entirety. This is an important feature of our algorithm because these groups can then be categorized into different degrees of friendship.

The computational complexity of this last phase is $O(N*2^L)$ where N is the total number of users in the dataset and 2^L is the number of clusters for each user.

CHAPTER 4

ALGORITHM EVALUATION

In order to test the validity of our algorithm an experiment was performed, which consisted of two separate phases: a data collection phase and a data validation phase. The main goal of this experiment was to evaluate Group Discovery through Co-location according to user perception and to fine-tune the parameters to find the best results. Although GDC results in different types of groups than other social community detection algorithms, there was an interest in comparing these results to the results of another algorithm to quantify the differences. For this comparison, the K-Clique algorithm was chosen because of its ability to detect overlapping communities.

In addition to running GDC on a dataset of our own construction, known as the NJIT dataset, the algorithm was applied to another well-known dataset of co-presence data known as the Reality Mining dataset. Although both datasets capture the same information, the user population was chosen differently, thereby providing two distinct possibilities of social information data. The NJIT dataset includes users who are not as well connected as the Reality Mining dataset; therefore we can use these differences to examine our algorithm's results in different types of data.

4.1 Data Collection

In this data collection phase, mobile phones were distributed to students from the New Jersey Institute of Technology campus with the intention of collecting co-location traces through a Bluetooth discovery program. This program quietly recorded the MAC

addresses of nearby Bluetooth devices every few minutes, stored them locally, and periodically transmitted them as a package to a remote server. The program accomplished this by initiating contact with all nearby Bluetooth devices and then recording the responses received. By storing the co-location traces on the phones, co-presence was detected in the absence of a wireless connection. The frequency of Bluetooth scans was fixed, depending on the user's device, either scanning every one, two, or three minutes. These intervals, as well as the initial timeslots of the scans, were determined randomly in order to minimize the signal interference of the Bluetooth devices due to collisions. Each scan took approximately 20 to 30 seconds on average to complete from start to finish (these values are in line with previous studies of Bluetooth discovery times [53]). Because the MAC addresses of each mobile phone in the study were recorded, the devices could be linked to other users in the study.

Table 4.1 Bluetooth Discovery Scan Statistics

Total Scans	384,512
No Registered Devices Found	82.7%
One Other User Device	14.5%
More Than One Registered Device	2.8%
Total Pair-Wise Records	404,105
Unique Devices Discovered	8,194

All subjects of the study were students of the New Jersey Institute of Technology, a medium-sized urban university. These subjects carried our Windows-based mobile phones while on campus for four weeks. Although, 168 students registered for the study, only 141 were considered active participants throughout the entire four-week experiment. Subjects were exclusively university students representative of the various majors offered

on campus. Seventy-five percent were undergraduates with the remainder graduate students, and 28% percent of the subjects were female.

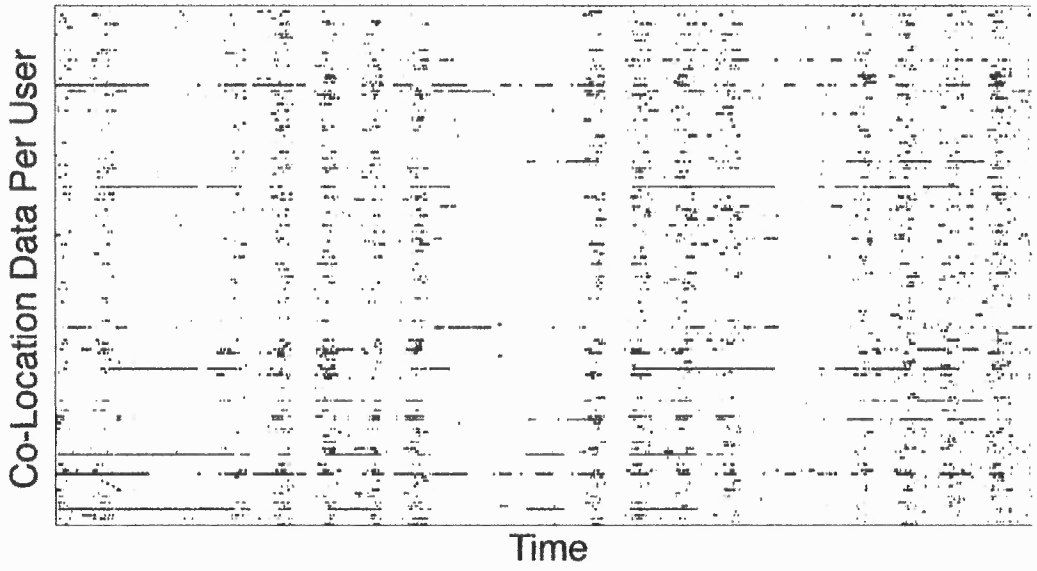


Figure 4.1 Overall user activity over the entire study. Each horizontal row represents a user and each segment in a row represents a time period spent by a user with at least another user in the study.

Table 4.1 presents an overview of the collected Bluetooth co-location data. Out of 8,194 unique Bluetooth devices were found, only 181 were devices registered with the study (some of the subjects needed to swap phones while the study was in progress).

Figure 4.2 depicts the total number of hours recorded for each user. This is displayed through a cumulative density function. This figure shows that for 78 users, less than 24 hours worth of data was collected. The maximum amount of data that could have been collected was 672 hours, 24 hours for 28 days. This is not the total amount of time that the user was on campus, but rather the total time that the phone was on and collecting data. There were, however, some users were large amounts of data were collected. There were two users in particular who collected 325 and 333 hours worth of data, which

represents almost half of the maximum amount of data that could have possibly been collected.

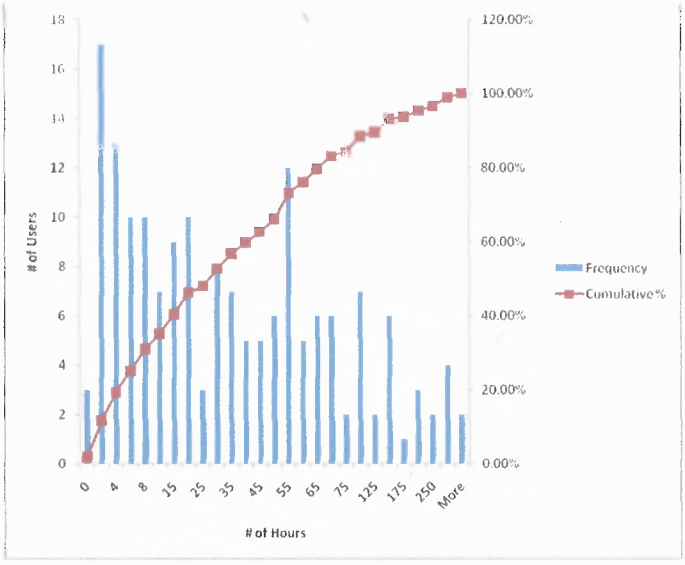


Figure 4.2 Cumulative density function of number of hours of data per user.

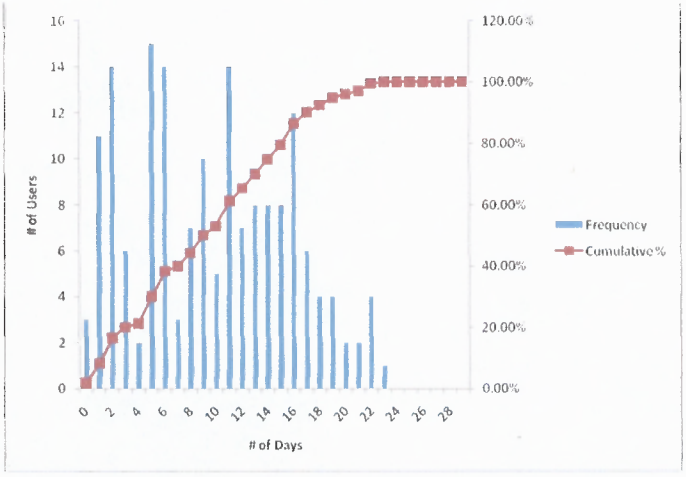


Figure 4.3 Cumulative density function of unique days with recorded data for each user.

Figure 4.3 illustrates the amount of unique days that each user contributed data while in the study. While the overall amount of hours of data per user is relatively small, users tended to provide this data over several different days. Fifty percent of the users in the study contributed data on at least nine different days. When viewed in conjunction with the data presented in Figure 4.2, one can presume that a large portion of the subjects came to campus often for short bursts of time, presumably for class.

4.2 Results Validation

After collecting the data, the algorithm 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 examine their effect on groups. The K-Clique algorithm was also run on the data, only including edges between pairs that spent more than 2,000s together throughout the experiment.

The second phase of the experiment consisted of verifying the output GDC by surveying the participating users of the study. A Facebook application was created that presents each user with a selection of groups generated from GDC and the K-Clique algorithm based on the data from the first phase of the experiment (the users were not aware of which group was produced by what algorithm). The user was prompted to rank these groups on a five point Likert scale, which is a scale from one to five where one represents a very bad group and five represents a very good group. Users were also encouraged to mark groups that contained other users that they did not know as "don't

know", rather than fabricate an answer. 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 that they were considered a member of.

Eighty-eight users responded to the Facebook questionnaire, 56 belonged to at least one group, from either algorithm, thus composing 482 different ratings for 265 different groups. Of these 482 ratings, approximately 60 percent were for groups only from the K-Clique algorithm and 32 percent were for groups from GDC. Each participating user rated 8.6 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 that all the subjects in a group were co-present for at least the minimum required parameters, but the subjects may not be aware of this co-presence. This may be because a group is not a group of friends, but rather a group of familiar strangers or acquaintances, who are not entirely aware of their continually co-presence. These familiar strangers are people who may frequently encounter each other during the course of their routines, but have never bridged the gap between stranger and acquaintance. Their faces may be recognizable, but users will typically not know each other's name, thus making it difficult for users to accurately self-report on these groups [29].

Furthermore, self-reported data on social interactions is historically troublesome as subjects are not always able to recall interactions or accurately report data on all their

relationships [10]. Because the algorithm discovers all groups of subjects, there is the potential to discover groups where all subjects are not aware of their frequent interaction together. Although these groups will most likely receive a poor rating by some users, they are still considered groups because they fit the criteria previously defined. It is clear that there is some unreliability inherent in this type of self-reported data, as users in the same group did not always report similar results for that group.

The reliability of user-reported data social interaction data has often been examined in the social sciences. One famous study conducted by Bernard, Killworth, and Sailer scrutinized the accuracy of user-reported data through the analysis of four different datasets [37]. These datasets contain the reports of all the subjects and their perceived interactions with ever other subjects in the study, on a five-point Likert scale, as well as a trained observer's perspective for a five-day period. Each dataset was collected using the same methodology for different communities of people, including a fraternity, a social science research firm, a graduate program, and a community of ham radio instructors. This research showed that group recollection is particularly unreliable as each member of the group will recall group meeting frequency differently, depending on the total number of interactions that each group member was involved in. Therefore, if a group member only had contact with a few groups, then they would rate a group higher than they would if they had contact with more groups [37]. Kashy and Kenny re-examined this collected data and determined that pair-wise data is more accurate than data that focuses on groups. As pairs spend more time together, the likelihood that both members of that pair declare that pair to be a frequent interaction is increased [38].

Because the validation phase of this study focuses on groups, not pairs, it is expected that the ratings are not as error-free as one would prefer. However, the variance of the group ratings within a group can offer valuable insight as to the accuracy of the ratings. If all members of a group provide the same ratings, then it can be assumed that this rating is accurate. Just the same, if there is a large degree of variance associated with these ratings, then it can be assumed that the ratings are not accurate. Because of the issues associated with co-presence data, familiar strangers, and user-reported data, groups that receive good ratings are an indicator of good groups, but badly rated groups may not be an indicator of poor groups. Because both algorithms are rated using the same methodology, it seems reasonable to compare them using self-reported user data.

4.3 Parameter Tuning

By applying the user's feedback for group detection accuracy, the initial parameters can be evaluated and better guidelines to select these parameters can be developed. Of course, the granularity cannot be evaluated through this feedback, but rather the effectiveness of the group meeting frequency parameter as well as the group minimum time parameter can be gauged.

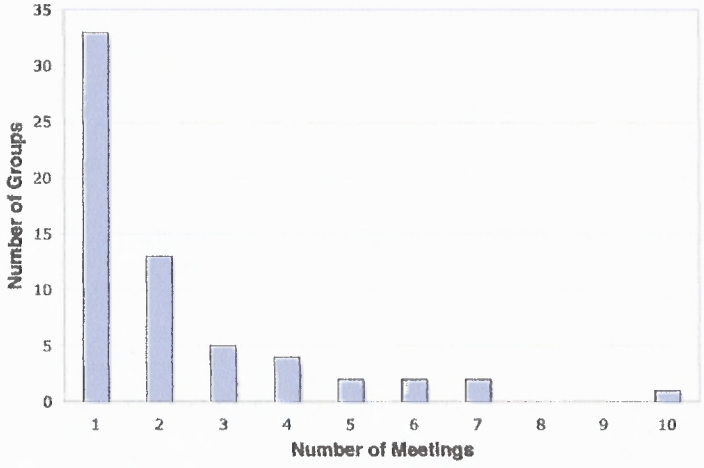


Figure 4.4 Group meeting frequency for GDC. The majority of the groups only met once.

4.3.1 Group Meeting Frequency (MGMF)

MGMF is the minimum number of times that a group must meet in order to be considered a final group. For the experiment, the parameter was initially set to one to include as many groups as possible. By examining the distribution of group frequency as function of the number of meetings (Figure 4.5), it can be seen that the vast majority of the groups, approximately 50 percent, met only once.

Figure 4.5 shows the distribution of ratings by group meeting frequency for the 61 groups from the results of GDC that were rated by the users. What is important about this figure is not that there are more poorly rated groups when the group meeting frequency is one, but that the ratio of well-received groups to poorly rated groups is low. As the group meeting frequency is increased, the poorly rated groups virtually disappear. This also makes sense, as groups that only met once for 2,000 seconds in the span of the month would not be expected to be considered as such by users. However, users that frequently

met for shorter intervals of time, may still consider themselves a group, even though they did not spend much cumulative time together overall.

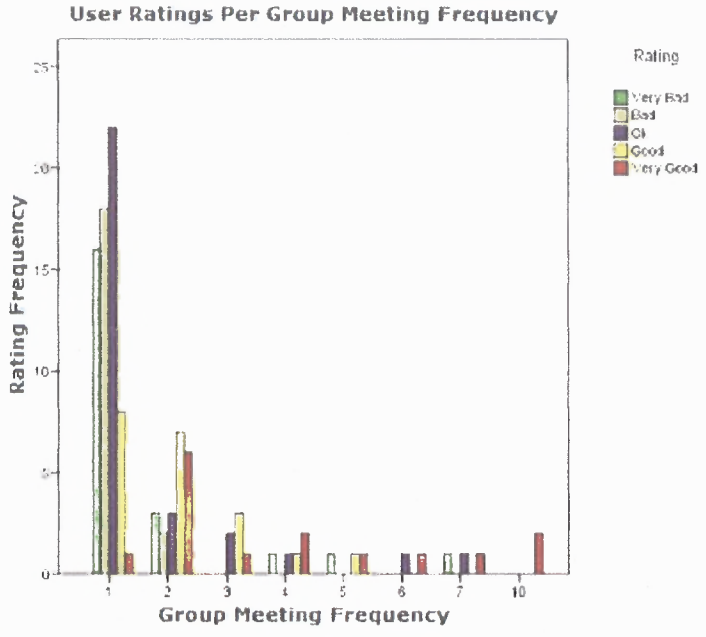


Figure 4.5 The distribution of ratings by group meeting frequency. This is based on the ratings provided by each user; hence, some groups are counted multiple times if multiple users from the same group rate.

4.3.2 Group Minimum Time (MGT)

The original minimum group time parameter was initially set at 2,000 seconds, approximately 33 minutes, which means that every group must spend at least 33 cumulative minutes together over the duration of the study to be considered a group. Examining the distribution of user rating by time shows that this parameter might have been too low to accurately capture real social groups. Figure 4.6 shows that although there are many groups that receive good ratings below 2,500 seconds, there are far more

poorly rated groups that fall within this timeframe. By changing our threshold to 2,500 seconds the majority of the poorly rated groups are removed and the majority of the well-received groups are still maintained.

Of course, by increasing the MGT, or the MGMF, the amount of total groups received as a result of the algorithm is decreased. This, however, is preferred, as the overall quality of these groups is significantly better. Conversely, by lowering these two parameters, there would be an increase in the amount of groups outputted by the algorithm, yet these groups would not adhere to the same standard as the current results.

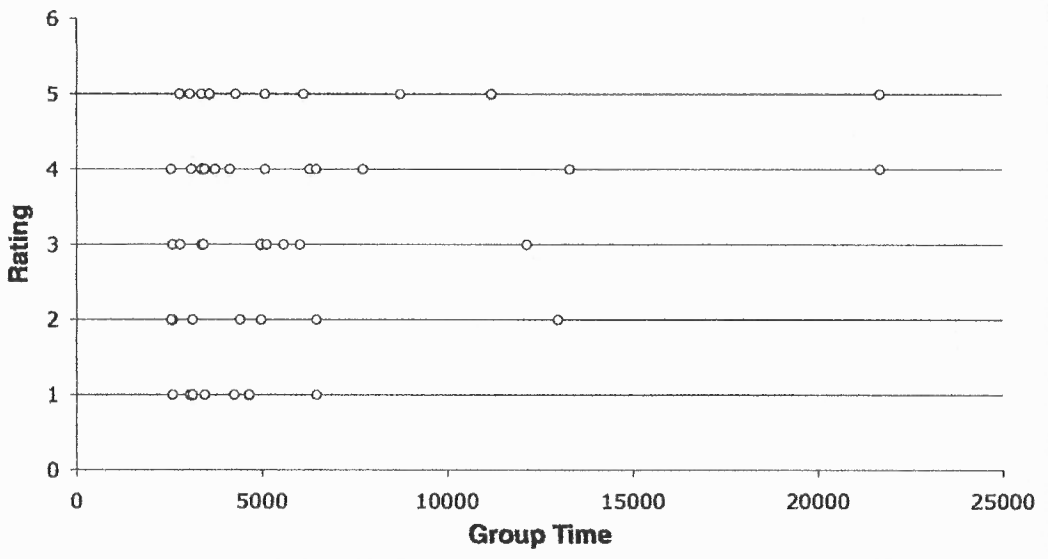


Figure 4.6 User ratings by group time for GDC. The parameters for this data are set at: MGT = 2,000, MGMF = 1, granularity = 15 minutes.

4.3.3 Rating Results

Table 4.2 shows the results if the base parameters of the GDC algorithm are changed to the ones described above (MGMF = 2, MGT = 2,500). Out of the 50 groups produced with these parameters, there are 39 total ratings, representing 29 different groups.

Roughly 65 percent of the groups GDC found are considered good by the users compared to 15 percent that are considered bad.

Figure 4.7 displays the resulting relationship between the group meeting frequency and the total group time when the algorithm is rerun with the ideal parameters specified above. The majority of these groups meet four or less times, for about two and a half hours or less, but there are a handful of groups that meet up to ten times, for up to five hours in total.

Table 4.2 Ratings for GDC With Parameters Set at MGMF = 2, MGT = 2,500, Granularity = 15

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	1 Very Bad	4	10.3	10.3	10.3
	2 Bad	2	5.1	5.1	15.4
	3 Ok	7	17.9	17.9	33.3
	4 Good	12	30.8	30.8	64.1
	5 Very Good	14	35.9	35.9	100.0
	Total	39	100.0	100.0	

4.4 K-Clique Comparison

The groups that the users rated in the Facebook application were based on a group minimum time of 2,000 seconds for both GDC and K-Clique. Therefore when the algorithms it is necessary to use the 2,000 second minimum group time parameter, even though it has been determined that this is not the ideal parameter GDC. Otherwise, the parameters for our algorithm are the same as above, with a required minimum group meeting frequency of two and MG=15min.

Table 4.3 shows a basic comparison between the ratings from GDC compared to K-Clique for all the records. Not only does GDC perform better than K-Clique in every category, but a chi-square test has a result of .0002 proving that this difference is statistically significant. By treating the Likert data as a continuous the means of the ratings can be compared. K-Clique has a mean of 2.73 and GDC has a mean of 3.62, which a t-test proves significant at the .000 level.

Because there is an uneven distribution of group size and not all users responded to the survey, groups that had many members rate them could possibly skew the results of this analysis. For example, a large group that is rated very poorly could effectively cancel out the effects of several smaller groups that performed very well. Therefore it is necessary to use weights, so that each group counts once regardless of the number of ratings that it received. In order to do this, the mean of all the user's ratings in a group is taken and then a comparison is formed. The average rating per groups for GDC is 3.41 compared to K-Clique at 2.62. Again, these findings are significant with .004 significance. Table 4.4 illustrates the difference between the ratings for each category of the Likert scale.

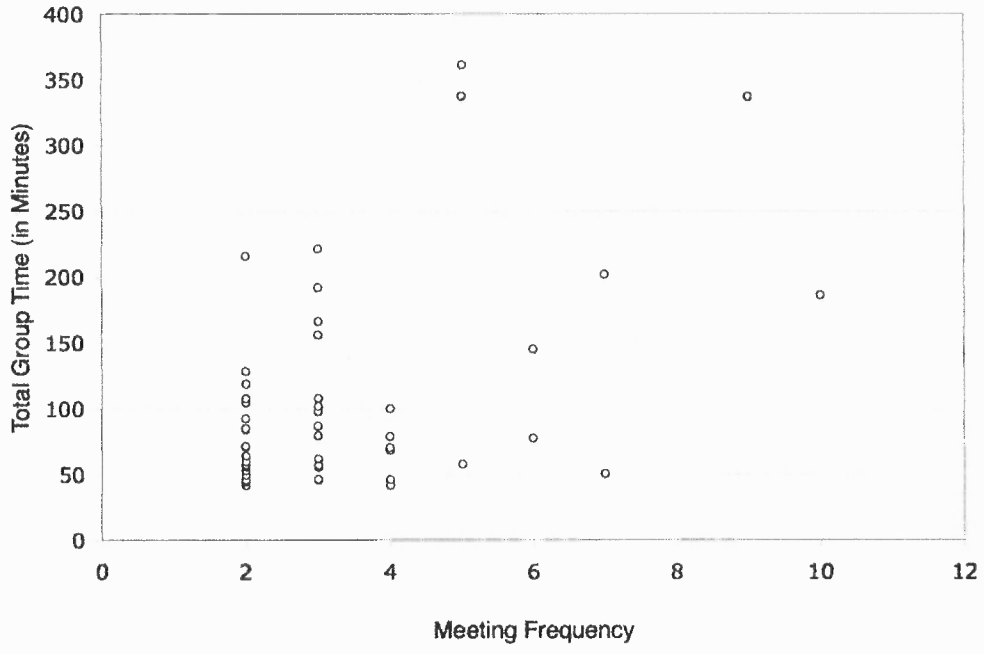


Figure 4.7 Group distribution as function of group meeting frequency and total group time for GDC. The parameters are set at MGT = 2,500, MGMF = 2, granularity = 15 minutes.

Out of the 292 groups discovered by K-Clique, 21 of these groups were also found by GDC. These groups were excluded from the above analysis because they fall into both categories. Within these 271 remaining K-Clique groups, 51 of them are considered groups by the definition of this thesis, which is that all members of the group spent at least some time together throughout the study. However, all of these groups spent less than 2,000 seconds together overall. Furthermore, there were no subgroups of our algorithm that overlapped with K-Clique. The difference between ratings between K-Clique groups that met for some amount of time compared to those that never did is examined. Out of the 218 ratings for K-Clique, 53 of them were for groups that met and the remaining 165 were for groups that never did. This emphasizes the problem of

community detection graph algorithms when working with co-location data (they tend to "discover" many inexistent groups).

Table 4.3 Comparison Between Ratings for GDC and K-Clique

	Rating										
	1 Very Bad		2 Bad		3 Ok		4 Good		5 Very Good		Total
GDC Algorithm	6	14%	2	5%	8	19%	12	29%	14	33%	42
K-Clique	48	22%	49	22%	59	27%	38	17%	24	11%	218
Total	54		51		67		50		38		260

Table 4.4 Comparison Between Groups for GDC and K-Clique

	Ratings										
	1 Very Bad		2 Bad		3 OK		4 Good		5 Very Good		Total
GDC Algorithm	4.83	17%	1.50	5%	7.33	25%	7.50	26%	7.83	27%	29
K-Clique	34.33	24%	34.50	24%	40.50	28%	24.50	17%	12.17	8%	146
Total	39.17		36.00		47.83		32.00		20.00		175

Because K-Clique discovers groups based on transitivity, it has the potential to identify non-existent groups when used on co-presence data. When used on data consisting of social ties, the use of transitivity may be valid to construct communities and groups, however this process does not convert well to co-presence data. This is because each edge in the graph only represents that two users spent more than a certain time in proximity of each other and does not represent friendship or even a social tie. If two users live and work in the same communities, than we may see similar results, as these two users could frequently encounter the same people throughout the course of their routines. However, if two users live in different neighborhoods, we wouldn't expect them to have the same set of acquaintances from their different neighborhoods.

4.5 Group Statistics

Continuing the analysis with the same parameters as input (MGMF = 2, MGT = 2,000) the groups and the differences between GDC and K-Clique is examined.

The first striking difference is the sheer number of groups and their relative size. GDC results in 65 total groups, with the vast majority of them composed of only three members. There are several groups of four and only one group of five whereas K-Clique has 292 total groups, with many groups of three, four, and five and several larger groups up to fourteen members in total. This is an expected result as K-Clique is finding communities in a graph, regardless of whether all users had co-presence or not. This group size frequency is shown in Figure 4.8.

The reason why there were so few large groups is due to the density of the NJIT data and the amount of time that users spent with other users in the study. Unfortunately, the sample was not large enough to cover many overlapping groups of friends. Our university has a population of approximately 8,000 students and this sample represents less than 2 percent of the student body. Furthermore, only 27 percent of all undergraduates live on campus, making it difficult to collect continuous data from users on campus.

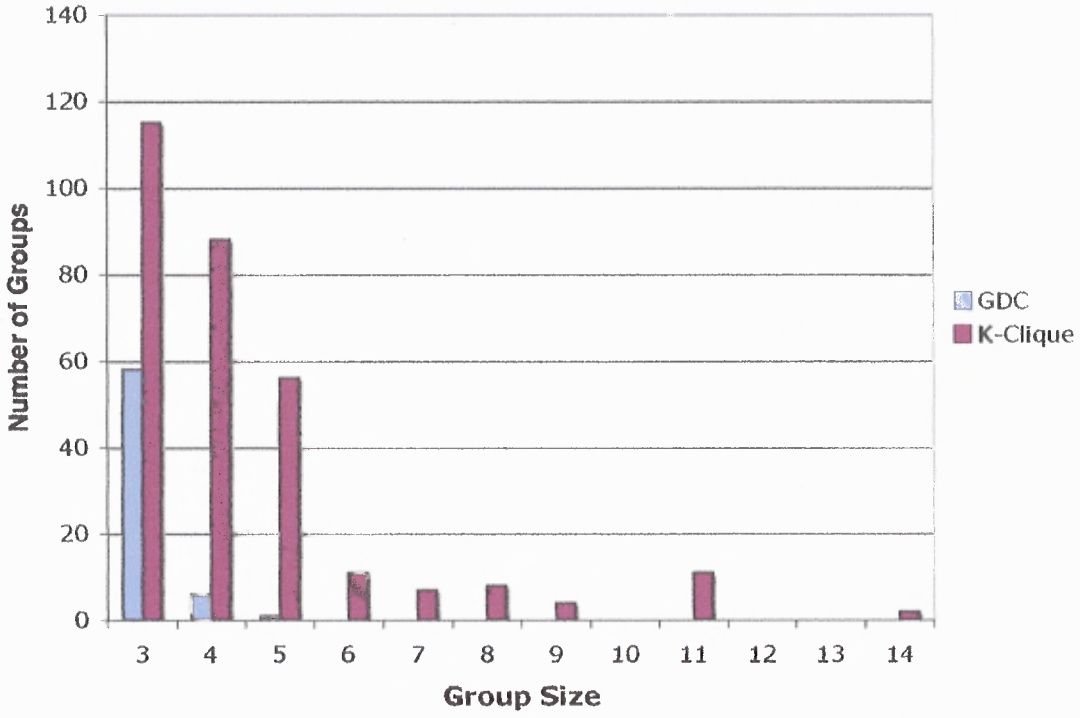


Figure 4.8 A comparison of the group size frequency between the results GDC and K-Clique. Not only does K-Clique have more groups, but their groups do not experience such a rapid decline in group size as the groups from GDC.

Figure 4.9 shows the number of users that each user is grouped with (i.e., members in one or multiple groups). Out of the 141 total active users in the study, GDC finds that only 76 of them are in groups that met the minimum required parameters. Within these 76 users, the average user is grouped with five other users. The minimum number of users that a user is grouped with is 2 and the maximum number is 12. On the other hand, K-Clique finds 127 users that have at least one group. Within these 127 users, the average user is grouped with 12.2 other users, the minimum is grouped with two and the maximum is grouped with 34. This figure shows that the social graphs generated by the K-Clique algorithm are much more connected than the social graphs generated by GDC. This is an expected result as much of the groups that are generated by K-Clique

have never actually spent time together; rather they are communities because they share the same friends. Every group generated by GDC has spent significant time together, over the course of multiple meetings; therefore it can be shown that every group does in fact represent a true social group.

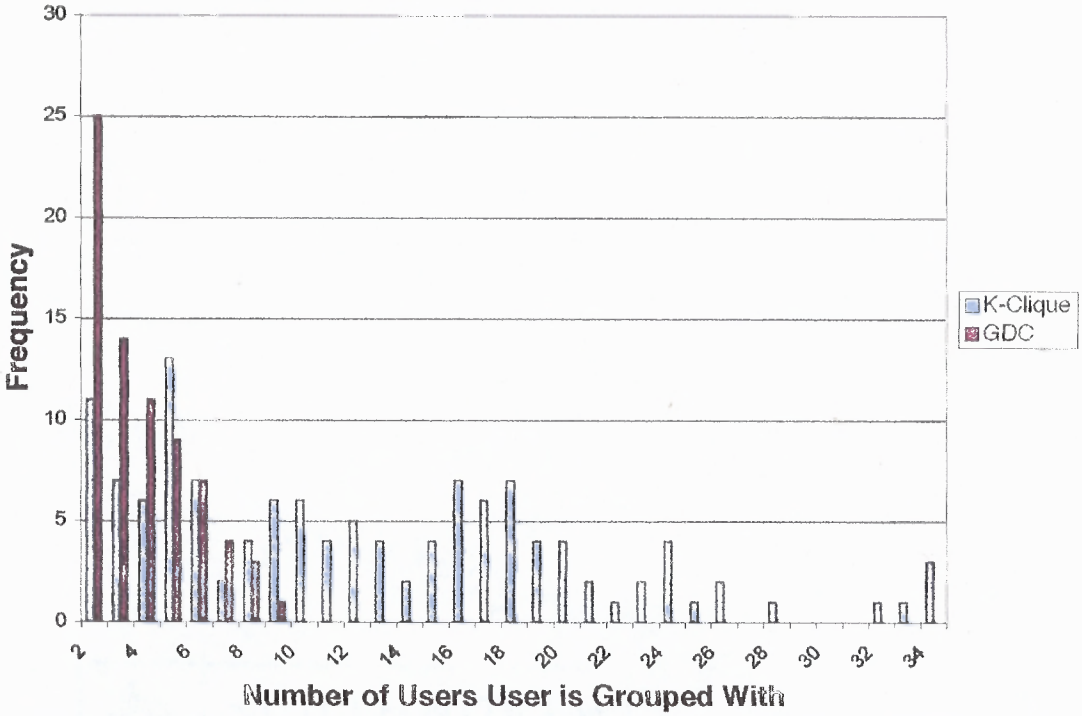


Figure 4.9 A comparison between the number of users each user is grouped with for the results of GDC and K-Clique. For the results of GDC, we see a smooth decline for the number of users that each user is grouped with, whereas the results of K-Clique display more variance.

4.6 Results Using the Reality Mining Dataset

The Reality Mining dataset is a perfect complement to the NJIT dataset for social group detection algorithm based on co-location as it represents a much larger time and includes more subjects who spend more time together. The Reality Mining dataset was created by the MIT Media Lab in 2009 and includes vast variety of data including, location and co-

location traces for 96 subjects over the course of nine months. The subjects chosen for this study are either members of their university's business school or work at the same building on campus [9] [10].

4.6.1 Methodology

Rather than capturing co-presence data as each Bluetooth record, the Reality Mining dataset lists the start and end of the intervals that two users spent together. However, the dataset reflects what each user saw instead of a global view. Therefore, in order to create a global view of the data (phase 1 of our algorithm), both perspectives of every pair-wise co-presence interval must be combined. For example, if user A sees user B from 1 to 3 and user B sees user A from 2 to 4, then the global view would have user A seeing B from 1 to 4 and vice versa. With the exception of this modified first phase, the algorithm is run as described.

4.6.2 Parameter Tuning

Because the data from the Reality Mining dataset is so different from the NJIT data, GDC was run with the most inclusive parameters possible, thus as an exploratory analysis in adapting parameters for a new, denser dataset. The initial parameters are a granularity of 15, a group meeting frequency of 18, and a minimum group time of 22,500 seconds. These parameters are the same parameters that were found to be ideal for the NJIT dataset, but adapted for the difference in time (Reality Mining covers nine months compared to NJIT's single month). Figure 4.10 shows the distribution of groups by the total number of meetings and the total time spent together. As you can see, these parameters are too low for this dataset as a large proportion of the users work in the same

building and see each other each day. Therefore the parameters can be raised to find a better fit.

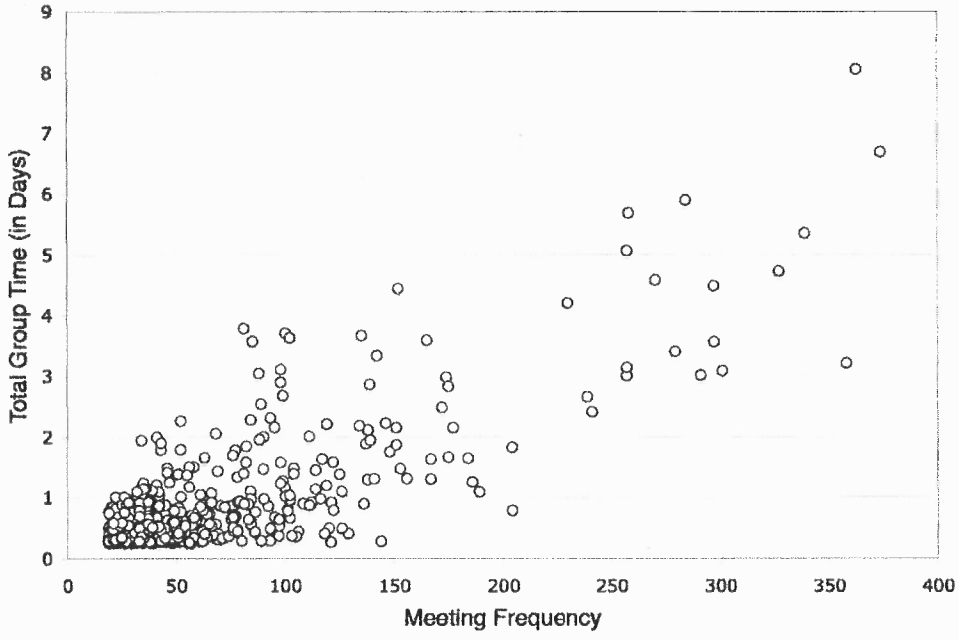


Figure 4.10 Group distribution as function of meeting frequency and total time spent together. for the results of the initial run of GDC on the Reality Mining dataset. The majority of the groups met for a cumulative time of less than 100,000 seconds and less than 80 meetings. The parameters are MGT = 22,500, MGMF = 18, granularity = 15.

If the parameters are raised to require a group meeting frequency of 90 and a minimum group time of 27,000, which equates to 10 meetings and 3,000 seconds per month, there are new results. Figure 4.11 shows these new results. The results of the algorithm are 424 total groups, 73% of which have three members, 22% have four members, 4% have five members, and there are also two groups of six members.

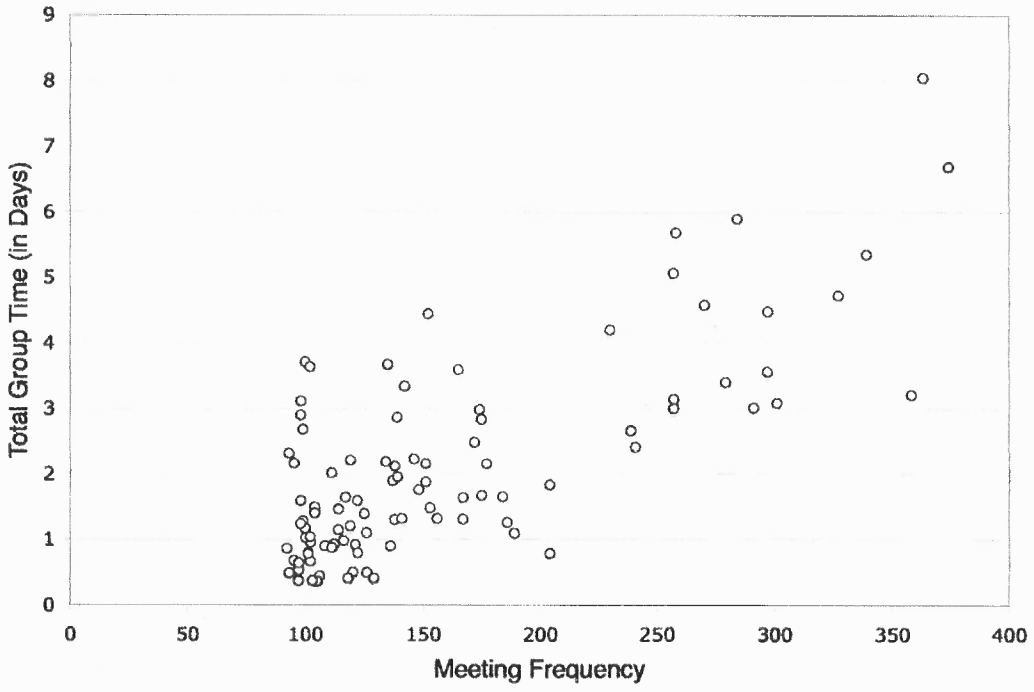


Figure 4.11 The total group meetings by the total group time for the results of GDC with adjusted parameters on the Reality Mining dataset. The parameters are MGT = 27,000, MGMF = 90, granularity = 15.

4.6.3 Pair-wise Statistics to Describe Data

Before running GDC, the statistics of all pairs who were co-present can be examined in order to determine some basic characteristics of the datasets. These characteristics can be used to guide the parameters for the algorithm. Table 4.5 shows the basic statistics for our dataset and the Reality Mining dataset and table 4.6 shows a pair-wise comparison between the two datasets.

Table 4.5 Basic Statistics for the NJIT Dataset Compared to the Reality Mining Dataset

	NJIT Dataset	Reality Mining
Number Different Users	141	96
Number Different Pairs	3510	6219
Total Possible Pair-wise User Combinations (Non-Directed)	13203	9120
Percentage of Pairs to Total Pairs	26.58%	68.19%

Table 4.6 NJIT Dataset Compared to the Reality Mining Dataset In Terms of Pair-wise Statistics

	NJIT Dataset Adjusted for 9 Months	Reality Mining	NJIT Dataset	Reality Mining Adjusted for 1 Month
Average Number of Meetings Per Pair	19.51	30.44	2.17	3.38
Std Dev of Number of Meetings Per Pair	29.57	66.58	3.29	7.40
Average Total Seconds Together Per Pair	41030.00	55053.00	4558.89	6117.00
Average Total Minutes Together Per Pair	683.83	917.55	75.98	101.95
Std Dev of Total Minutes Together Per Pair	4090.23	4116.55	454.47	457.39

Even though Reality Mining has fewer users than the NJIT dataset, there are more pairs. This suggests that most subjects in Reality Mining come into contact with more subjects in the study. After scaling the Reality Mining dataset to represent the equivalent of a month's worth of data, the average time and meetings each pair met for as well as the standard deviations of each can be calculated. Reality Mining pairs spent a third more time together overall and met 3.38 times on average compared to NJIT's 2.17. Both dataset have very similar standard deviations for the total time per pair, which suggests that users in the Reality Mining Dataset generally spend more time together, rather than just a few pairs skewing the data. This corresponds to the meetings statistics, as the standard deviation of the number of meetings is much larger than NJIT dataset's. Figure

4.12 shows the group meeting frequency of both datasets as a percentage of the total number of meetings. The Reality Mining dataset is normalized for one month in this figure, where each pair-wise group meeting is divided by nine and rounded. This figure shows that the Reality Mining pairs meeting frequency is more evenly distributed than the NJIT dataset.

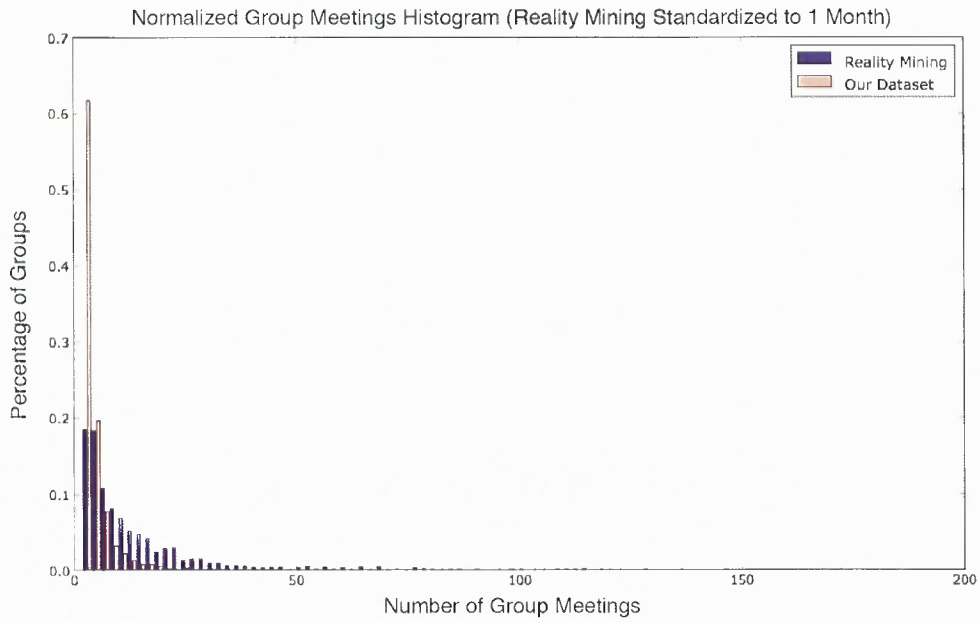


Figure 4.12 A comparison of the group meeting frequency in terms of percentage of total pairs for the results from GDC run on the NJIT dataset and the Reality Mining dataset.

4.7 General Parameter Estimation

Of course, the parameters for the algorithm will change not just on the datasets but also on the researcher's preference. However, a few guidelines can be offered as the NJIT dataset is a good representation of a sparse data and Reality Mining is a good representation of dense data. It is recommended to use approximately half the average total minutes together per pair as the minimum group time parameter and the average meeting per pair-wise for the group meeting frequency parameter. The researcher should

also do a simple gauge of the pair-wise group meeting frequency in order to access what a good parameter would be. With more datasets representing the varying degrees of data, it is anticipated that there will be a more robust parameter selection measure.

CHAPTER 5

TOWARD A DISTRIBUTED VERSION OF THE ALGORITHM

Currently, GDC has been discussed in centralized terms, meaning that the algorithm runs on data that has already been collected at a central location. By slightly modifying the design, the algorithm can run in a distributed manner, such that each mobile phone can store co-location traces and calculate groups for its owner without relying on a centralized server. The phones can exchange data with other phones of interest using the locally available Internet connectivity. There are many benefits to this refinement, especially concerning the privacy of users. This section discusses the feasibility and the necessary changes for a distributed version of our algorithm.

5.1 Benefits of a Distributed Version

The main benefit of a distributed version of this algorithm is better privacy for the user. In this version, the user is protected from sharing data with a "big brother". The user has to share data with other users, but these are only the users that 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 group information. Depending on the application, final social groups might be pooled together, however co-presence traces still remain local, so specific details, such as time of user encounters remain private. Interactions with users who are not in any final social groups will remain confidential as well.

Other benefits of a distributed version of this algorithm are resiliency and flexibility. Distributed GDC is more resilient as there will be no central point of failure: both the data and the 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 social groups are of a very dynamic nature and are constantly changing; however, we expect users to have different rates for evolving social groups dependent on a variety of factors.

5.2 Necessary Changes to the Algorithm

In order to transform our algorithm into a distributed one, several important changes must be made. The first of these changes is significant and involves the first phase of the algorithm. For the normal, centralized version of the algorithm, the first step involves taking the union of all Bluetooth records, such that each user has the same perspective for each pair-wise view. This is an important step because it removes many of the errors associated with recording co-location via Bluetooth.

If the Bluetooth devices were accurate enough to record all encounters without much error, than by removing this step, we would only expect to see tiny differences between each user's perspectives of the same encounter. These differences can be attributed to the tail-end of each encounter, as the Bluetooth trace would not mark the exact second that two users encountered each other. For example, if user A saw user B at 12:30 until 1:00, than we could reasonably expect user B to see user A at 12:28 and 12:58. For large encounters, this would not be much of an issue but shorter encounters that are closer to the size of the granularity cause more of a problem.

This phase is necessary for the NJIT dataset as not every recorded interaction was present for each pair-wise user. Reality Mining seems to have a similar problem were both users do not seem to record the same encounters. By taking the union of all traces in the first phase, we can eliminate a large portion of the errors and have a dataset that is closer to reality. Without taking the union of the data, a significant portion of the data would be incomplete, leading to inaccurate groups. Therefore, a distributed GDC will have to exchange pair-wise record summaries in order to determine the missing data, and then retrieve these data from the pair.

The second phase of the algorithm is only modified slightly compared to the first phase. Rather than iterating through all records in a single pass, GDC running at the phones will only run through its local records. Each phone will thus have every cluster in which its owner is a member as long as those clusters meet the required MGT threshold.

However, the members of a cluster as determined by the centralized GDC might end-up with different sub-clusters in the distributed GDC. This is certainly the case when two users who are not co-present with each other are chained together by a third user, who can see both of the other users. This problem is solved in the centralized GDC by the third phase; however, in the distributed GDC, there is no centralized collection of users. Therefore, in order to mimic the third phase in the distributed GDC, each phone must query each other phone in all of its clusters to determine whether the cluster should be kept or removed. Therefore, in order to mimic the results of the centralized GDC, a number of messages must be passed between phones determined by the number of groups a user belongs to and the number of other users in that group. If this modified third phase is not utilized, then the results will be completely different. To imitate these results on a

centralized GDC, the third phase of the algorithm should be changed so that the most frequent member is the determining factor of a group rather than the least frequent member.

The fourth phase of the algorithm is not changed much, however it is run locally on a single user's perspective rather than globally. However, if users do not have the same total cumulative time spent together for each group, than some users will consider a specific group to be a subgroup whereas others will consider that same group to be a final group included in the output. If a group of users spend large amounts of time together, and there is no chaining of two users together through a third party, then these errors will be minimized. However, as groups spend less time together, then these subtle changes becomes more drastic and contribute to bigger differences between final groups between users.

5.3 Experiments

In order to compare the differences between the centralized version of the algorithm and a distributed version, several tests were performed using data from the NJIT dataset. Four different versions of the algorithm were run, where each version represents a different intermediary step from fully centralized to fully distributed (i.e., localized without communication with other phones). Each of these versions is run with the same parameters, which are as follows: MG=15min, MGT=2,000 seconds, and MGMF=2 meetings. The four versions are described below.

Normal: This is the standard version of GDC algorithm. It is to be used as the baseline to compare to the other results. The union of all meetings is completed in the first phase and a global view is created of all clusters based on the least frequent group member.

Max Group: This permutation is similar to the normal version except that while comparing each user's perspective of a group (during the third phase), we take the maximum time that a group member sees the group instead of the minimum phase. Consequently, clusters where only one member sees the entire cluster are considered clusters, whereas previously they were not.

Union Local: This permutation uses the union of all meetings in the first phase of the algorithm, however a global list of clusters is not maintained. Rather each user is treated as a separate entity and has their own output. Thus there is no third phase of the algorithm. The output is pooled together after the last phase in order to compare to the other permutations of the algorithm

Nonunion Local: This permutation does not use the union of all meetings in the first phase of the algorithm, rather it uses the traces seen by each user, but is otherwise similar to the union local phase. This is the true and final version of the local algorithm as it replicates each user running the algorithm on their own data.

Figure 5.1 depicts the total number of unique groups outputted by each version of the algorithm. It is fairly straightforward why the normal version of the algorithm finds the least number of groups and why the max version finds more. The normal version of the algorithm relies upon the least frequent group member whereas the max version relies upon the most frequent member. If each user had the same perspective of all groups, then the union local version would have the same results as the max groups. This is not the case, so the additional groups that are found by the union local version are due to previously designated subgroups being defined as final groups. This is due to the fact that a subgroup spends significantly more time together than the least frequent user sees. The differences between the nonunion local version of the algorithm and the union local version are due to accumulated differences between each user's perspectives of

encounters. Clearly some of these encounters are only being recorded by one user instead of two.

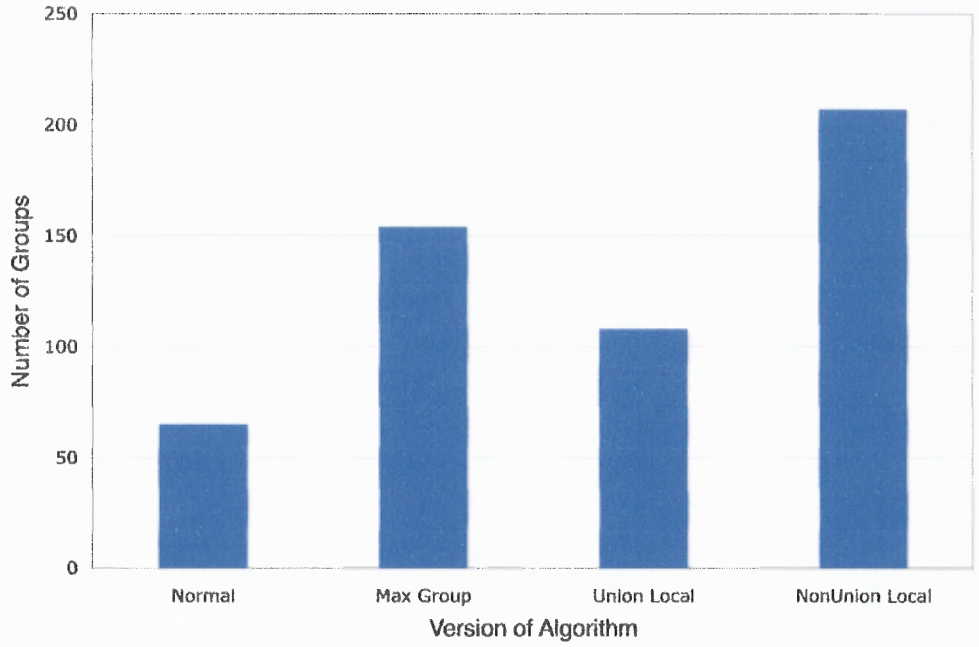


Figure 5.1 A comparison of the total number of groups for the final output for each version of the algorithm.

Figure 5.2 demonstrates the frequency of group size for each of the versions of the algorithm. The biggest difference comes from the normal to the max group versions of the algorithm. This is due from the conceptual change of from all users needing to see a group to be considered a group, to only one user needing to see the group. The local versions have higher counts of smaller sized groups because these smaller sized groups are subgroups in the global versions of the algorithm.

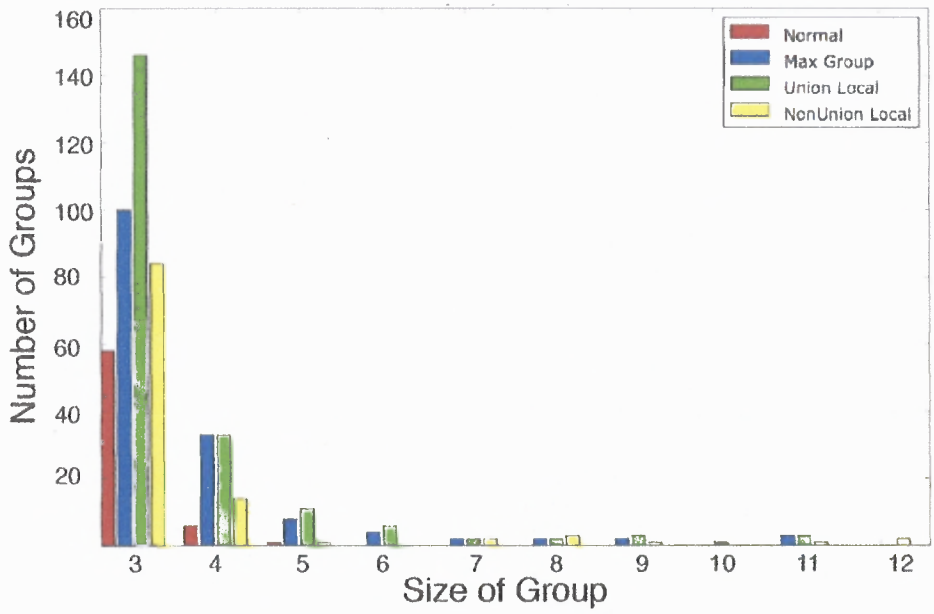


Figure 5.2 A comparison of the frequency of group size for the four different permutations of our algorithm.

Based on the results of the distributed experiment, the most important phase of the algorithm that is lost by switching to a distributed localized version is the first phase. It can also be determined that message passing between the different users is certainly essential as the results of the algorithm change. Although these results are easily explainable and all of the additional groups are subgroups simply get included in the final output, these are major conceptual changes that should not be taken lightly. It is difficult to quantify the expected differences between a distributed version of the algorithm and a centralized version as the differences depend on the data. If the total cumulative time for a group varies greatly for each member, then there will be large differences between the distributed and centralized version.

To currently convert the algorithm into a distributed version, the number of messages that must be passed it governed by the number of users in each group. For each

group, a message must be passed to each user in the group validating that each user recognizes the same group. Once a user who does not recognize the group is reached, then that user must tell all other users in the group that the group does not exist. Similarly, once all users verify that the group is a valid one, then they must be informed by the last member of that group. The simplest way to pass these messages is over the Internet. An alternative is to pass them using ad hoc opportunistic communication. If they are passed in an opportunistic manner, then the problem of not knowing the next meeting of a group becomes an issue. Therefore, if the messages are passed through the Internet, then they can be passed as soon as the algorithm has reached the third phase.

CHAPTER 6

CONCLUSION

This thesis presented a new algorithm, Group Discovery through Co-location (GDC), which can discover social groups from a dataset of co-presence traces. These groups can then be used in other applications, such as standalone programs, middleware, ad-hoc networking, or for social science research. These groups are different from the groups created through other algorithms as it can be proven that every member of every groups spent significant time with each other. Furthermore, these groups maintain data, specifically the number of times that each group met and the total cumulative time of those meetings. This data can be used in further analysis to compare or rank the groups.

These new groups are discovered through four distinct phases, each one using the output from the previous phase as its input. The first phase creates the pair-wise meeting records. The second phase creates user clusters. The third phase deals with clusters that are viewed differently by its members. The fourth and final phase selects the user groups. GDC's results are function of two main parameters, meeting frequency and total meeting time, whose values depend on pair-wise statistics of the collected data.

Applying the algorithm to two very different types of datasets has validated its use on datasets of varying density and degree. This has been further examined through the user's self-reported data, which ranked the groups on a five point Likert scale. These results show that the algorithm finds accurate groups, and that this algorithm outperforms the K-Clique algorithm by 30%.

REFERENCES

[1] http://www.brightkite.com, 06 2009.

[2] http://www.hitwise.com/us/datacenter/main/dashboard- 10133.html, 06 2009.

[3] http://www.loopt.com, 06 2009.

[4] M. S. Bargh and R. de Groote. Indoor localization based on response rate of Bluetooth inquiries. In MELT '08: Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments, pp. 49-54, New York, NY, USA, 2008. ACM.

[5] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. Wireless Communications and Mobile Computing, 2(5): pp. 483-502, 2002.

[6] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Pocket switched networks: Real-world mobility and its consequences for opportunistic forwarding. University of Cambridge Computer Laboratory, Tech. Rep. UCAM-CLTR- 617, 2005.

[7] I. Derenyi, G. Palla, and T. Vicsek. Clique percolation in random networks. Physical review letters, 94(16):160202, 2005.

[8] I. Dhillon, Y. Guan, and B. Kulis. A unified view of kernel k-means, spectral clustering and graph cuts. The University of Texas at Austin, Department of Computer Sciences. Technical Report TR-04, 25, 2005.

[9] N. Eagle and A. Pentland. Reality mining: sensing complex social systems. Personal and Ubiquitous Computing, 10(4): pp. 255-268, 2006.

[10] N. Eagle, A. Pentland, and D. Lazer. Inferring social network structure using mobile phone data. PNAS, 2007.

[11] M. Girvan and M. Newman. Community structure in social and biological networks. Proceedings of the National Academy of Sciences, 99(12):7821, 2002.

[12] J. Golbeck. The dynamics of web-based social networks: Membership, relationships, and change. First monday, 12(11), 2007.

[13] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. Computer Networks, 52(14): pp. 2690-2712, 2008.

[14] L. Holmquist, J. Falk, and J. Wigstr "om. Supporting group collaboration with interpersonal awareness devices. Personal and Ubiquitous Computing, 3(1): pp. 13-21, 1999.

[15] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, pp. 244 -251. ACM New York, NY, USA, 2005.

[16] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay tolerant networks. In Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing, pp. 241-250. ACM New York, NY, USA, 2008.

[17] F. Li and J. Wu. Localcom: A community-based epidemic forwarding scheme in disruption-tolerant networks. In Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on, pp. 1-9, June 2009.

[18] A. Lindgren, A. Doria, and O. Scheln. Probabilistic Routing in Intermittently Connected Networks. In Proc. Workshop on Service Assurance with Partial and Intermittent Resources, August 2004.

[19] A. Madhavapeddy and A. Tse. A study of bluetooth propagation using accurate indoor location mapping. Seventh International Conference on Ubiquitous Computing, (UbiComp '05), pp. 105-122, August 2005.

[20] M. McNett and G. Voelker. Access and mobility of wireless PDA users. ACM SIGMOBILE Mobile Computing and Communications Review, 9(2): pp. 40-55, 2005.

[21] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, pp. 29-42, 2007.

[22 M. Musolesi and C. Mascolo. Designing mobility models based on social network theory. ACM SIGMOBILE Mobile Computing and Communication Review, 11(3), July 2007.

[23] M. Newman. Analysis of weighted networks. Physical Review E, 70(5):56131, 2004.

[24] M. Newman. Finding community structure in networks using the eigenvectors of matrices. Physical Review E, 74(3):36104, 2006.

[25] M. Newman and J. Park. Why social networks are different from other types of networks. Physical Review E, 68(3):36122, 2003.

[26] T. Nicolai, E. Yoneki, N. Behrens, and H. Kenn. Exploring social context with the wireless rope. 1st International Workshop on Mobile and Networking Technologies for social applications (MONET'06), Montpellier, France.

[27] S. Olofsson, V. Carlsson, and J. Sj "olander. The friend locator: supporting visitors at largescale events. Personal and Ubiquitous Computing, 10(2): pp. 84-89, 2006.

[28] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. Nature, 435(7043): pp. 814-818, 2005.

[29] E. Paulos and E. Goodman. The familiar stranger: anxiety, comfort, and play in public places. Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 223-230, 2004.

[30] J. Pouwelse, P. Garbacki, J.Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. Van Steen, and H. Sips. TRIBLER: a social-based peer-to-peer system. Concurrency and Computation: Practice and Experience, 20(2): pp.127-138, 2008.

[31] V. Srinivasan, M. Motani, andW. Ooi. Analysis and implications of student contact patterns derived from campus schedules. In Proceedings of the 12th annual international conference on Mobile computing and networking, pp. 86-97. ACM New York, NY, USA, 2006.

[32] U. von Luxburg. A tutorial on spectral clustering. Statistics and Computing, 17(4): pp. 395-416, 2007.

[33] S. Wasserman and K. Faust. Social network analysis: Methods and applications. Cambridge University Press, 1994.

[34] Borcea, C. and Iamnitchi, A. 2008. P2P Systems Meet Mobile Computing: A Community-Oriented Software Infrastructure for Mobile Social Applications. In *Proceedings of the 2008 Second IEEE international Conference on Self-Adaptive and Self-Organizing Systems Workshops - Volume 00* (October 20 - 24, 2008). SASOW. IEEE Computer Society, Washington, DC, pp. 242-247.

[35] Gupta, A., Paul, S., Jones, Q., and Borcea, C. 2007. Automatic identification of informal social groups and places for geo-social recommendations. *Int. J. Mob. Netw. Des. Innov.* 2, 3/4 (Feb. 2007), pp. 159-171.

[36] Borcea, C., Gupta, A., Kalra, A., Jones, Q., and Iftode, L. 2007. The MobiSoC middleware for mobile social computing: challenges, design, and early experiences. In Proceedings of the 1st international Conference on Mobile Wireless Middleware, Operating Systems, and Applications (Innsbruck, Austria, February 13 - 15, 2008). MOBILWARE, vol. 278. ICST (Institute for Computer Sciences Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, pp. 1-8.

[37] Bernard, H. Russell, Peter D. Killworth, and Lee Sailer. 1980. "Informant Accuracy in Social Network Data IV: A Comparison of Clique-Level Structure in Behavioral and Cognitive Network Data." Social Networks 2: pp. 191-218.

[38] Kashy, D.A. and Kenny, D.A. "Do you know whom you were with a week ago Friday? A re-analysis of the Bernard, Killworth, and Sailer studies. Social Psychology Quarterly (53)1: pp. 55-61, 1990.

[39] Abderrahmen Mtibaa , Augustin Chaintreau , Jason LeBrun , Earl Oliver , Anna Kaisa Pietilainen , Christophe Diot, Are you moved by your social network application, Proceedings of the first workshop on Online social networks, August 18, 2008, Seattle, WA, USA

[40] Bigwood, G., Rehunathan, D., Bateman, M., Henderson, T., and Bhatti, S. Exploiting self-reported social networks for routing in ubiquitous computing environments. In Proceedings of the 1st International Workshop on Social Aspects of Ubiquitous Computing Environments (SAUCE), Avignon, France, Oct. 2008.

[41] Newman, M. E. J. Fast algorithm for detecting community structure in networks. Phys. Rev. E 69, 066133, 2004.

[42] Y. Iwatani, "Love: Japanese Style," *Wired News*, 11 Jun 1998; http://www.wired.com/culture/lifestyle/news/1998/06/12899.

[43] M. Terry et al., "Social Net: Using Patterns of Physical Proximity over Time to Infer Shared Interests," *Proc. Human Factors in Computing Systems* (CHI 2002), ACM Press, 2002, pp. 816-817.

[44] N. Eagle and A. Pentland (2005), "Social Serendipity: Mobilizing Social Software", IEEE Pervasive Computing, Special Issue: The Smart Phone. April-June 2005. pp 28-34.

[45] http://www.dodgeball.com 10 2009.

[46] N. Li and G. Chen. Multi-layer friendship modeling for location-based mobile social networks. In Proceedings of the Sixth Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous), Toronto, Canada, July 2009.

[47] http://www.myspace.com

[48] http://www.facebook.com

[49] http://www.linkedin.com

[50] Baik Hoh, Marco Gruteser, Hui Xiong, Ansaf Alrabady. "Enhancing Security and Privacy in Traffic-Monitoring Systems", IEEE Pervasive Computing Magazine (Special Issue on Intelligent Transportation Systems), 5(4), 2006.

[51] http://www.youtube.com

[52] http://www.flickr.com

[53] N. Ravi, P. Stern, N. Desai, L. Iftode, Accessing Ubiquitous Services using Smart Phones, Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom), Kauai Island, Hawaii, USA, March 2005.

[54] Dragoş Niculescu, Positioning in Ad Hoc Sensor Networks, IEEE Network Magazine, July/August 2004. pp 24 - 29.

[55] Gupta, A., Kalra, A., Boston, D., and Borcea, C. 2009. MobiSoC: a middleware for mobile social computing applications. Mob. Netw. Appl. 14, 1 (Feb. 2009), pp. 35-52.

[56] http://www.google.com/latitude/intro.html