Institut für Parallele und Verteilte Systeme Abteilung Verteilte Systeme
Universität Stuttgart

Universitätsstraße 38
D-70569 Stuttgart

Master Thesis

Design of Autonomous Algorithms for Location Privacy

Murat Ünal

Studiengang:	INFOTECH
Prüfer:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Betreuer:	M. Sc. Zohaib Riaz
begonnen am:	10th April 2017
beendet am:	10th October 2017

Abstract

Over the past years, the number of location-based applications have been increasing due to the advancement of mobile computing technology and the sensors built in the mobile devices. These applications grant individuals to locate one another, and benefit from services and information those are delivered according to individuals' location. However, despite the popularity of location-based applications, they are increasingly criticised for putting user privacy at risk by disclosing location data mistakenly to undesired parties. Adapting and predicting changes of user's sharing attitude over time in order to overcome possible leakages, presents challenging problem for location-based applications and location privacy approaches.

In this thesis, we extended an existing work on detecting routine and out-of-routine location events of a user based on entropy estimation, to present a control mechanism that can discover correlation between sharing behaviour and routineness of a location for individuals. It is designed as supportive mechanism that aims to prevent possible leakage induced by location privacy approaches. We additionally extended an existing policy generation algorithm to evaluate proposed control mechanism by identifying optimal privacy preferences for users.

The proposed approaches were evaluated and implemented in Python environment. We ran simulations with different metrics to test the overall performance of the system.

Acknowledgements

I would like to express my sincere appreciation for my thesis supervisor, M. Sc. Zohaib Riaz, for his valuable time and advices during supervision of my thesis. I would also like to thank Prof. Dr. Kurt Rothermel for giving me a chance to do my thesis in the Department of Distributed Systems.

I am grateful to my family members for their continuous support, and thankful to all my friends for their encouragement.

Table of Contents

List of Figures

List of Tables

1. Introduction

Due to the increasing demand for mobility, mobile devices have started to play a key role in people's daily life by allowing them to store and exchange their personal information with no time and location barrier. In 2016, global revenues from mobile application has reached $44.8 billion [3]. The capability of mobile devices to pinpoint users' location with the help of the embedded GPS sensor, cell towers, wireless positioning and IP location has promoted development of location-sharing applications. Moreover, adoption of location-sharing technologies retains its importance for mobile industry as they want to offer advertisements and services designated by users' recent or most visited locations.

Approximately one hundred location-based applications (LBAs) have developed [1] and become prevalent with the fast adoption of location-enabled devices such as smart phones. A typical example of these applications is geo-social networks such as Facebook, Google+ and Swarm where users share their location data, along with other content, with their family and friends. However, LBAs disclose users' contextual and location information, and sharing personal information with non-trusted service providers as well as malicious acquaintances can be detrimental to a user's privacy. A malicious entity with access to the user's location data may infer their interests, political affiliations, and religious inclination etc., resulting in serious privacy concerns.

1.1 Privacy Concerns & User Burden

Despite the popularity of location-based applications, they are increasingly criticised for putting user privacy at risk [4]. Further, privacy concern of the users is often observed as a reason of their slow adoption [1,5,6]. Accordingly, LBAs have started to implement and offer different privacy mechanisms so concerns toward location privacy may shift over time.

The most commonly used privacy settings by popularly adopted methods which are based on location-sharing rules [6, 8] to avoid location privacy concerns, are the following:

- Blacklist: One of the least complex setting type enables users to restrict specific individuals from viewing their location updates at all times. They are easy to define and user friendly and require only single rule.

- Whitelist: Other least complex setting type allows users to grant specific individuals viewing their location updates at all times. Similar to blacklists, they are easy to define and user friendly.

- Location: This privacy setting allows users to reveal only specific locations with target individuals. Compared to whitelists and blacklists, they require a rule for single location which user is comfortable to share.

- Granularity: This more advanced form of location setting allows users to define at which level of detail they would reveal their location updates to specific individuals.

- Time: Users can define; when and for which period of time in a day, they would like to disclose their location updates with individuals. Similar to location setting, time setting is more complex and may require multiple rules for every different time and period of time set (i.e. different start and end time).

- Group: Users are able to define groups of individuals and policies explicitly generated for those groups.

Users can often indicate several rules described by one or union of multiple privacy setting which also increases the required number of rules for decent privacy. These rules define a geographical region or a time-span (or both) inside which a particular LBA may be given access to the location data of a user. Since these rules are defined by the user, they have various disadvantages. First, the privacy-preserving definition of rules requires the users to foresee complex situations where their location privacy may be breached. It has already been shown in [9] that user-defined rules are usually insufficient to preserve user privacy as they do not cover all scenarios under which privacy breaches may occur. Secondly, the definition of location-sharing rules burdens the users as they have to invest considerable time, concentration, and effort.

As an example, let's assume a user who wishes to share his location only when he is at his home and, that also, only in the evening time. If the user defines a time-based rule to allow access to his location in the evening time, this rule alone may be insufficient to meet his sharing preferences as the user may also be present at other locations (than just home) during evening times. While adding another location-based rule to specify sharing only when at the home location may overcome this problem, it burdens the user with definition of an additional rule. Since users typically find rule-definitions burdensome [6], they define a limited number of location-sharing rules which typically may result in false-positives (i.e. incorrect sharing decisions) where the user actually does not want to share the current location event.

1.2 Out-of-routine Events

In order to reduce the number of false-positives in sharing decisions, an interesting direction is to augment location-sharing rules with additional algorithms. The basic idea behind such algorithms will be to address the expected short-comings of location-sharing rules, i.e., to identify those location events for which these rules generate false-positive decisions. In this regard, one promising methodology is to differentiate between routine location events (representing typical user-movements) and out-of-routine location events (representing a-typical movements) of the user. Here, the underlying assumption is that users define their location-sharing rules based on the knowledge of their routine movements. For instance, in the above given example, the user expected himself to be at home in the evening times and therefore defined a time-based rule as per this expected knowledge. By defining these rules based on expected/routine movements, users typically ignore their out-of-routine movement behaviour, which may be one of the major reasons of the resulting false positive sharing decisions.

This thesis therefore investigated whether there is a correlation between location events of the user which are out-of-routine and those for which the location-sharing rules generate false positives. Our results of evaluation show that learning subject's correlation between routineness and sharing behaviour for specific locations can provide detection of probable privacy leaks in general.

1.3 Thesis Outline and Contributions

In detail, this thesis consists of the following tasks:

- Design of an adaptive algorithm for defining limited location-sharing rules given the actual sharing-behaviour of the users.

- Design of an adaptive algorithm for detecting out-of-routine events.

- Preparation and processing of large-scale dataset in order to implement and evaluate above algorithms.

- Analysis and evaluation of implemented approach.

At the remaining part of the thesis; we present background information and related works on location privacy mechanisms and mobility analysis mechanisms in Chapter 2. Later in Chapter 3, we introduce our system model and provide a formal specification of the problem statement. Chapter 4 gives detailed description on an algorithm for identification of best policy for a user [6], and the optimizations applied on the algorithm. In Chapter 5, we discuss a study for out-of-routine detection based on conditional trajectory entropy estimation, and define adaptive algorithm for real-time trajectory entropy estimation. Chapter 6 describes preliminary data processing applied on the dataset before the evaluations. Then, we present the simulation setup, and a thorough analysis of the location-dataset for studying the correlation between out-of-routine events and false-positives generated by location-sharing rules in Chapter 7. Finally, Chapter 8 concludes this thesis and proposes possible future works.

2. Related Work

This chapter is intended to provide a basic understanding of the key aspects and concepts that form the basis of this thesis, while giving overview of the previous work done in the area. In the following sections, we will first discuss location privacy mechanisms including different approaches and researches done relevant to the topics. Later, we will give brief overview on calculation of trajectory entropy with and without conditions, and an approach that enables detection of waypoints from mobility of the users utilizing entropy of trajectories.

2.1 Location Privacy mechanisms

Location privacy mechanisms are the systems those intend to reveal location events of a user as accurate as possible by adopting user's privacy preferences, and autonomously resolving user's intention towards sharing. This section discusses two common approaches utilized by the location privacy mechanisms in order to prevent probable privacy leaks.

2.1.1 Static Mechanisms

Systems using static mechanisms commonly adopt location sharing rules [6, 8] to represent users' privacy preferences or policies. Since rules are defined by the user once, system cannot react to changes of users' sharing attitude over time. The rules should be updated by the user repeatedly to prevent potential privacy leaks. Because most users are not good at expressing location-sharing rules, and their privacy preferences are not stable over long periods of time, privacy concerns are inevitable. Another approach [5] which built upon [8] has shown that giving daily feedback of location disclosure history to users, has reduced privacy concerns. Knowing log of information disclosure every day enables users to act before vast leaks, and give them more control. However, it increases user-burden.

2.1.1.1 Approaches

The authors in [6] have investigated the trade-offs between accuracy and user-burden consisting different privacy setting types with 27 participants on over 7.500 hours of data. In contrast to utilizing end system, they aimed to analyse required number of rules in order to achieve decent accuracy (i.e. how much scope is covered correctly in history of participants' location sharing behaviour) for each privacy setting they have introduced. In additional to whitelist, location and time; the following privacy setting types were conducted:

- Time with weekends (Time+): This setting is similar to time, but they additionally allow users to indicate defined time duration in either weekdays or weekend, as a flag label.

- Location and Time (Loc/Time): Union of location and time setting. Users are granted to specify location additionally.

- Location and Time with weekends (Loc/Time+): Union of location and time with weekends setting.

Participants have been asked if they are comfortable revealing their location events to four different social target groups (e.g. friends & family, Facebook friends, university community and advertisers) instead of individuals. In short, the common privacy setting 'group' had been implemented in all other policy settings evaluated.

For the experiment setup; firstly, location-tracking software has been installed on participants' smart phones. It ran continuously in the background and receives updates from GPS and Wi-Fi module. Locations were afterwards filtered if user not remained stationary for at least 15 minutes in that location. Secondly, web-application has been developed where participants required to visit every day to audit the locations they visited that day. Web-application asks whether or not the subject would have been comfortable sharing presented location with the target social group, and during which period of time.

For the evaluation, representation of rules for each privacy setting types were generated from the feedback given in web-application. Later, these rules were compressed if they had shared same location, same time, same group or etc. Then, from the pool of compressed rules, A* search algorithm is utilized to find best policy (i.e. user's privacy preference or set of rules) with k rules. The results show that with 5 or more rules it is possible to reach 70% accuracy using loc/time+ setting.

On the other hand, PeopleFinder [8] is an end-user application which can be used on smart phones and laptops by users to share their locations with others. In PeopleFinder, a user can be both a target user who is queried for location information by others, and requester who query location information from others. The system is utilized with centralized approach to store location information on their server. Although central server is a potential privacy vulnerability (those may require k-anonymity, location obfuscation etc. which are not covered in this thesis), application is able to provide requesters to enquire last stored location of targets when the target is not available, and more demanding processes can be executed more efficiently than on distributed and mobile platform.

Each user has their agent on the server which lets users request for location information, and every agent owns Policy Enforcing Agents (PEA) that handles queries and operates according to a policy preferences of the user. When requester's agent invokes target's agent, PEA checks whether the location disclosure is allowed by the privacy preferences. If it is, it forwards location information to requester.

The privacy setting types included in definition of rules are location and time with days. Instead of only weekend flag used in previously presented approach, time with days covers every day in a week.

Relatively important point is; during their initial lab experiments, they have found that the number of the rules and the time spend by participants defining and tinkering the rules has slight correlation between policy accuracy. Most users are not able to define highly accurate policies, and their decisions often have conflicting rationales.

2.1.2 Dynamic Mechanisms

In order to overcome these users' ineffectiveness and evolving privacy preferences, dynamic mechanisms have been applied to improve the decision-making process. Dynamic mechanisms based on machine learning or similar algorithms, target to disclose information as possible and prevent privacy leaks by adapting to user's sharing behaviour and sometimes predicting possible alterations on it, using different number of features. In those systems, users don't require to define detailed privacy preferences with complex privacy settings. Semi-supervisedly trained (semi-autonomous) systems [10,12] rather needs to learn from either from audit of user (e.g. feedback, survey information) or only user's manual decisions without bothering them, in order to label the data. Number of different required input features can be taken from mobile device of the user during labelling process. For some systems to being able to make accurate decisions, more features might be required compared to static mechanisms. On the other hand, unsupervised (autonomous) methods [11] pursue extracting privacy preferences automatically from social context of users by grouping them, without needing to labelled data. It can also be addressed as using crowdsourcing to gather information from the public [28].

2.1.2.1 Approaches

SPISM [10] was presented as one of the privacy-aware information-sharing system that are employed with machine-learning techniques, including cost-sensitive classifiers based on support vector machines and naïve Bayes. Similar to PeopleFinder [8], a user can be both a target user and requester. Different than previously introduced approaches; it is able to decide different levels of detail of disclosed information (i.e. granularity) at each sharing decision, allows users additionally to share time-schedule availability, and grants third-party services to request information.

The platform consists of the Information Sharing Directory (ISD) and the subscribers of the service. As would be expected, ISD is centralized server where subscribers connect to and credentials are stored on. But, classifiers are implemented in mobile devices with the help of WEKA [13] library.

SPISM predicts whether or not to reveal location and the granularity, based on personal and contextual features and past decisions of users. These features are;

- Familiarity of requester (if requester is user)
- Social tie of requester (if requester is user)
- User ID of requester (if requester is user)
- Service category of requester (if requester is third-party online service)
- Service ID of requester (if requester is third-party online service)
- Information type
- Details of information
- Latitude
- Longitude
- Semantic

- Time
- Weekday
- Daytime
- Activity
- Neighbours of requester
- Neighbours type
- Time since last request
- Details of last request

These 18 features are extracted from the request forwarded to the target user's device, and fed to the decision core. If SPISM is able to make the decision with enough confidence, the request is processed automatically. Otherwise, the target user needs to decide manually. After every manual input, the decision along with the request is added into ground-truth data. Events in ground-truth data are weighted as descending fashion from newest to oldest, and later fed to a classifier in order to train decision core. Weighted events allow classifier to dynamically adapt to the changes in the users' sharing attitude by prioritizing latest decisions.

Performance of the SPISM were measured in the binary case (without level of detail i.e. granularity) with and without active learning. Without active learning; classifier is trained from different percentages of the dataset, and is tested on remaining part. The results show that when classifier has learned from forty and higher percentage of the dataset, median of correct decisions made remained still at approximately 75%. Percentage drops from under-share and over-share were evenly distributed when no cost classifier was used. Over-share caused by classifier were less likely to happen while cost factor was increasing, but correct decisions have dropped to 60% because of under-share. With active learning; the classifier was initialized with some percentage of the dataset and dynamically learn from remaining while in testing state using confidence interval. When classifier has not enough confidence on its decision, it requires manual input from the user and actively train itself. When fifty percent of the dataset was used to initialize classifier, correct decisions were made almost 85% times and over-share cases were encountered in significantly less rate.

Another interesting experiment was training one-size-fits-all classifier from all but one user data, and testing it on remaining user. Median of 67% of correct decisions were preserved.

The approach in [12] that aimed high correct prediction rate and a low privacy leak rate, has evaluated simple classifiers with different methods, and machine learning classifiers with different algorithms at once in the paper.

The default privacy preferences defined by the users resulted in correct predictions 68% and privacy leaks 11% of time. Simple classifiers were able to reach %76 correct decision rate and 11% privacy leak rate. On the other hand, machine learning classifier with Rotation Forest[1] algorithm has obtained 85% correct decision rate and 3% privacy leak rate. The considered dataset attributes were;

- User ID
- User feedback
- Time of day

[1] Rotation Forest algorithm is improved form of J48 algorithm. It sacrifices computational efficiency for the sake of increased performance.

- Type of location
- A Facebook friend-list ID (a social group which user wants to share or not)
- Flag for the decision
- Co-location
- Four other attributes from the pre-briefing data

Along those features, according to Weka's attribute selection classifiers, the most important attributes were Facebook friend-list ID and size of the list. When only these two attributes were used, Rotation Forest has achieved 76% correct predictions and 12% privacy leaks.

Similar to previously presented approach, they have additionally included Cost Sensitive option in machine learning classifiers, where over-share decisions are weighted as ten times more than under-share. With the option, maximum achievable correct prediction and privacy leak rates were 67% and 1% respectively.

Considering all presented static or dynamic approaches, the generic conclusion reached was; privacy policies defined by users never match the actual user preferences. Using different number of privacy setting types allows more accurate policies but causes more complexity and user-burden in return. However, active learning algorithms those are able to adapt changes in users' attitude with the help of methods such as confidence interval has obtained better correct decision rates than static mechanisms.

This thesis aimed to investigate and develop a scheme that can be used as confidence interval for static mechanisms. Accordingly, we first need to examine entropy of trajectories.

2.2 Mobility Analysis mechanisms

Mobility of the users is one of the key aspects that enables us to discover people's behavioral data. Extracting meaningful locations (waypoints) from their mobility trajectories is crucial for location-based applications in order to provide context-aware services, since those reveal more information about the person and their lifestyle. Waypoints are the places people try to reach and spend their time (e.g. restaurant, café, shop, cinema etc.), and they move on intermediate points (e.g. highway, street, train station, airport etc.) to reach their destination as efficiently and quickly as possible. These important locations can contribute to predicting person or crowd next location [20, 21] or whole trajectory [16], revealing popular locations in a geospatial region, travel recommendation etc.

2.2.1 Approaches

The interface model in [18] has been developed based on Hypertext Induced Topic Search (HITS); a link analysis algorithm that rates Web pages, was used to rate and extract interesting waypoints from GPS trajectories. The authors have first constructed a tree-based Hierarchical graph (TBHG) from the stay points obtained from GPS logs, using a density-based clustering algorithm. They have used HITS on

TBHG to estimate users' travel experiences and location interests in a region. Later, these parameters are used to extract interesting locations in a region.

On the other hand, entropy is the most fundamental quantity estimating the randomness characterizing a time series, hence it can also allow us to extract the meaningful locations and predict waypoints or trajectories of users. For example, by using the entropy, Song et al. [21] have studied mobility pattern dataset of 45,000 mobile phone users and shown that average user mobility predictability rate was 93% independent of the significant differences in the travel patterns and the distance of the trajectories. They have used three different entropies to predict user's next location; the random entropy which estimates the predictability of the user's actual location, the temporal uncorrelated entropy which characterizes the heterogeneity of visitation patterns, and the actual entropy of the trajectory.

Similar to [21], an algorithm has been proposed in [7] to calculate instantaneous entropy of an individual by using real-time estimator based on Lempel-Ziv measure for uncovering the behaviour and predictability of the users. They have showed by applying their algorithm on Nokia Lausanne dataset, that when instantaneous entropy of user was high, mobility patterns were more alternating and had low predictability. They have also found that behaviour of mobile application usage is correlated with instantaneous entropy of the users, which proves that users behave differently in unfamiliar situations or out-of-routine events.

The authors in [16] went beyond the research of predictability of mobility in [21] or individuals in [7] and were able to measure the predictability of the whole trajectory using the entropy of conditional Markov trajectories that allows them to compute the extent to which an event is different from a normal event.

Users often reveal their trajectory information through a sequence of check-ins via services (e.g. Swarm, Facebook or Twitter etc.), or from geo-tagged photos. But those check-ins do not necessarily include information of time spent by the user, making the discovery of waypoints tougher. Waypoints play important role in individuals' lives since they spend more time at those locations; those therefore reveals more information about the person and their lifestyle. The work [16] focuses on extracting waypoints and segmentation of trajectories on user's mobility trajectory data only from spatial information, not requiring timing information on location events.

The model is built based on the work [15]. They implied that given two waypoints (a starting location and final destination), the intermediate location that most increases –conditional– trajectory entropy is presumably another waypoint between those two, and the user would spend more time at this waypoint than the average time spent at other intermediate locations. They have tested their model with randomly 90%-10% divided into train-test dataset, and achieved improvement on baseline methods. Namely, 43% more accurate than geo-stretch and 20% more accurate than path-stretch approaches.

3. System Model and Problem Statement

In this chapter, we will first discuss system model for location-sharing system. In the second part, we will define the problem statement for our modified decision-making system.

3.1 System Model

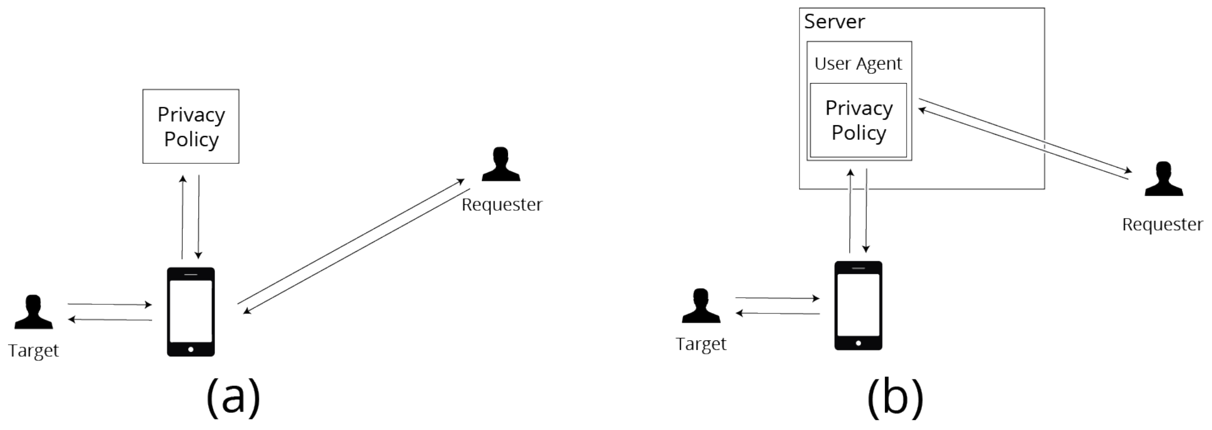

Figure 3.1 - System Models. Distributed approach (left) and centralized approach (right)

The privacy systems, whether they have been developed based on static or dynamic mechanisms, have two common schemes as shown in Figure 3.1. Each scheme has their own benefits and disadvantages. While centralized approach in Figure 3.1(b) is offering computationally light-weight clients on mobile devices and logging of user's location events so that requesters are able to fetch last actual location; distributed (peer-to-peer) approach in Figure 3.1(a) is easier to secure since it has not central server as potential privacy vulnerability [8,19]. Centralized scheme is often preferred for complex dynamic systems because of energy efficiency of mobile devices while providing required computation power. Furthermore, a hybrid scheme has been proposed by several approaches [25,26] which relies on both. In all schemes, users directly or indirectly generate privacy preferences which disclose information only when certain conditions are met.

Mobile device in the Figure 3.1, is the key instrument of the system model used to track user, capture user's location events and allow interaction of user to define privacy policies. In Figure 3.1(a), the privacy policy and location updates are stored in the mobile device of the user. When requester queries target location, he communicates with target's mobile device and reply is sent according to the policy of the user. On the other hand, in Figure 3.1(b), mobile device captures user's location events and sends them to the trusted location server where they along with privacy policy are stored. Requester queries target's location from the server and reply is sent if privacy preferences allow it. When server or mobile device cannot decide itself, they often notify user to acquire manual decision.

In the system model, we can ponder over many privacy aspects such as security of communication channel; reliability of parties, systems or servers etc. However, these aspects are beyond the scope of our study and can be addressed in other works. In this thesis, we will focus on extending decision making competence of the privacy policy.

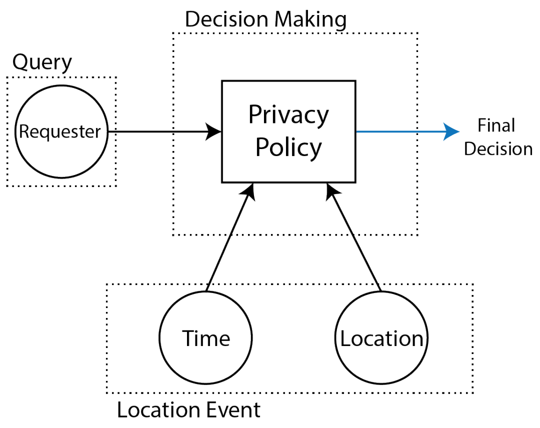

Figure 3.2 - Common model for Decision Making Process

Typically, decision making process of the LBA consists of only privacy policy which trained either by machine-learning algorithms or rule-based approach. When requester queries location, privacy policy collects attributes depending on privacy setting type; such as requester-ID (or social group he is in), time information (time, weekend, day, month etc.) and location information (spatial or granularity) as shown in Figure 3.2. If those attributes match with the rules, or decided as sharable by classifiers; private location information is sent to requester.

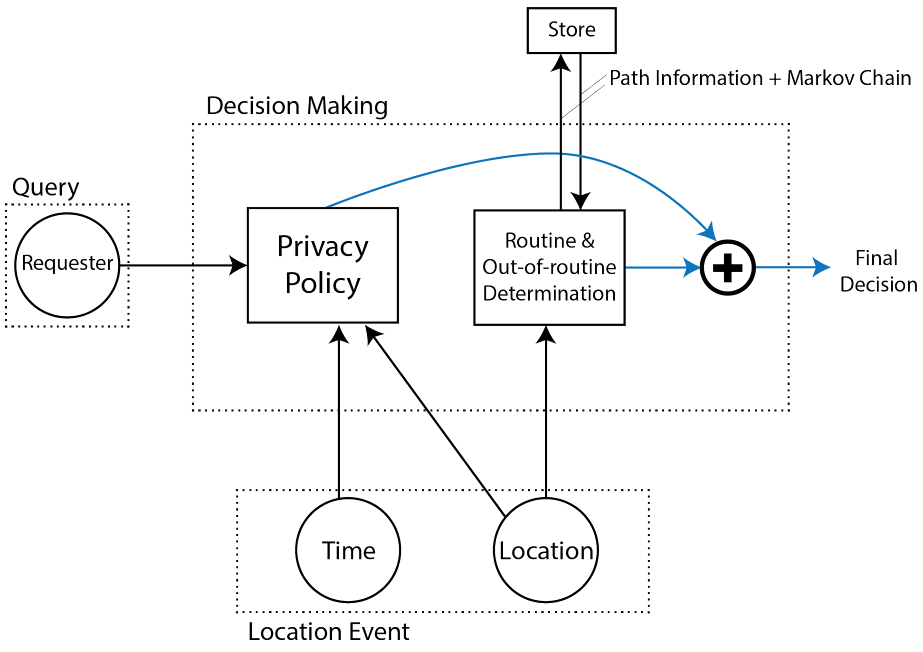

Figure 3.3 - Offered model for Decision Making Process

Built upon typical decision-making strategy, we propose new system includes routine aware control mechanism as shown in Figure 3.3. Similar to the system in Figure 3.2, privacy policy collects attributes such as location, time, and requester-ID, and predict decision depending on those. Conversely, in order to generate final decision, proposed system also requires confirmation from the routine and out-of-routine control mechanism. By storing history of location events, the mechanism is able to resolve if recent location event was occurred from routine or out-of-routine series of events. If it is a routine event, mechanism grants permission and system replies back to the requester. If it is an out-of-routine event, then mechanism may notify user in order to get manual decision.

As for control mechanism, we have used similar technique from [16] for determining the out-of-routine nature of user events utilizing conditional entropy of trajectories [15]. The technique is based on information-theoretic algorithm for distinguishing routine and non-routine location events in a data-stream, which can be adapted into real-time entropy estimator. Fortunately, the authors in [7] found that mobile application usage was correlated with instantaneous (recent) entropy of the users. In other words, they were using applications differently when they were in out-of-routine nature. We can exploit that fact and approve the possible changes on location sharing behaviour in a similar way.

Additionally, in spite of dynamic mechanisms like SPISM [10] which require a number of attributes to train classifiers, entropy based decision making system only requires trajectory information to train mobility markov chain. Yet, constructing mobility Markov chain from location events is not an easy task and requires significant amount of user data, even for a low-order Markov chain. Consequently, we need to train and simulate privacy policy of users from a large dataset. The approach in [6] thereby is chosen to construct best privacy policy given location history dataset of each user.

3.2 Problem Statement

Static mechanisms offer rule-based privacy preferences defined by the users. However, they are lacking to detect changes of user attitudes that may lead to potential privacy leaks if the rules are not redefined over time. The more detailed privacy settings induce more complex interfaces and causes more user-burden. Further, most users are not able to define highly accurate and long-time stable policies even with complex settings [5,6,8,9,10].

Our objective is adding dynamic depth to typical rule-based approaches in order to prevent over-sharing while not requiring an extra user input, by using entropy for determining the out-of-routine nature of user events. This approach has never been addressed before. This thesis does not engage with developing novel end-system, but addresses following problems and evaluates the proposed approach accordingly.

- **Over-sharing** is mistakenly revealing private information which was not intended by the user (false-positives if policy shares, false-negatives if policy withholds information). The main reason over-sharing occurs with static privacy policies is that they cannot predict user behaviour change and redefined over time. Privacy would not be a problem if there were no risks or bad consequences of sharing location information. However, bad consequences can occur due to leak of location information to unscrupulous groups or an individual. As an example, PleaseRobMe [23] has demonstrated risks of undesirable privacy leaks by publishing the location of unoccupied homes using Foursquare and Twitter data.

Therefore, our main problem is to decrease over-sharing rate of a defined privacy policy. The over-sharing metric (as false-positives) of policy p is defined in the thesis as follows;

$$overshare(p) = \frac{incorrect\_hours(p)}{total\_predicted\_hours(p)} \tag{3.1}$$

- **Under-sharing** is preventing the sharing of location information when it was intended to be shared (false-negatives if policy shares, false-positives if policy withholds information). It may not pose the same privacy risk as over-sharing but higher under-sharing rate deprives purpose of LBAs. Therefore, our second problem is to keep under-sharing to the minimum. The under-sharing metric (as false-negatives) of policy p is defined in the thesis as follows;

$$undershare(p) = \frac{total\_hours\_to\_be\_revealed - correct\_hours(p)}{total\_hours\_to\_be\_revealed} \tag{3.2}$$

where total_hours_to_be_revealed is total hours intended to be shared by the user during test period.

- **Accuracy** metric is considered in the thesis with penalty term c_n in order to put weight on risks of accidently sharing information. In other words, we have merged above two problems and defined an accuracy metric accordingly.

$$accuracy(p) = \frac{correct\_hours(p) - c_n * incorrect\_hours(p)}{total\_hours\_to\_be\_revealed} \tag{3.3}$$

4. Simulating Privacy Policy

In this chapter, we will discuss in detail the approach [6] used to simulate best-user policy given set of user rules. Later, we explain the modifications we have done on this approach in order to make it usable on an extensive dataset.

4.1 Background

The system presented in [6] aimed to find the best user policy with k-rules that best fits their ground-truth gathered throughout 3-week user study. The main purpose of that study was to evaluate the different privacy settings on static mechanisms in terms of user-burden and possible reachable accuracy, rather than developing a novel location-sharing system.

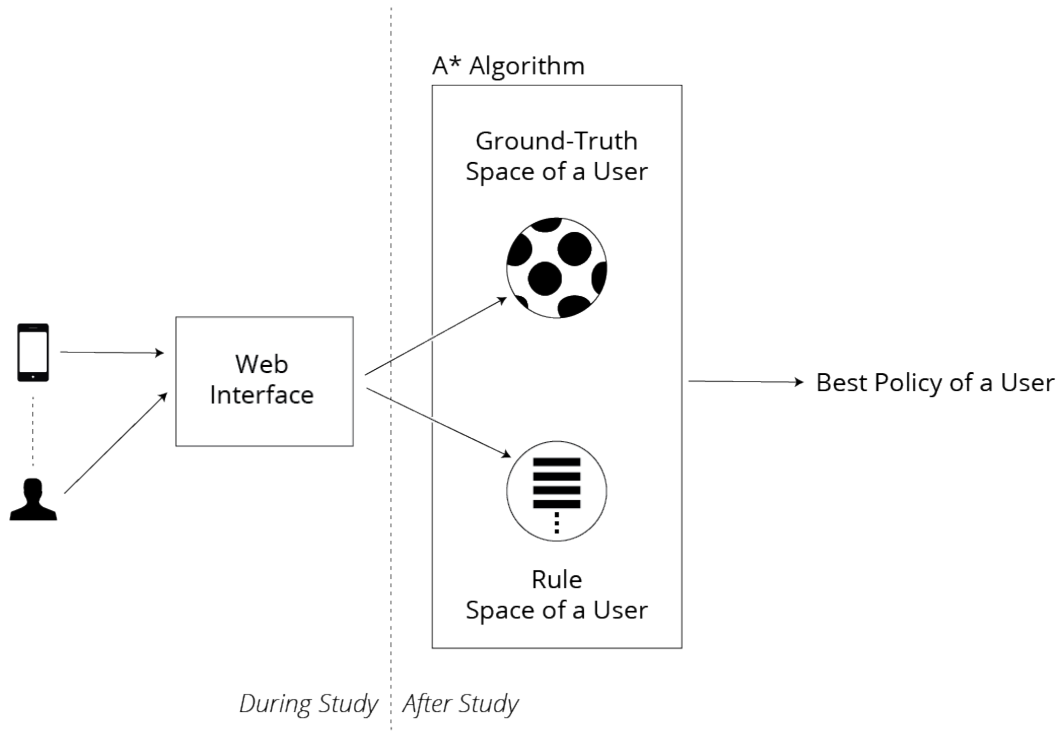

Figure 4.1 - Evaluation model for the system in [6]

Before the study, each of the phones was equipped with location-tracking program that recorded the phone's location and sent it to web servers at all times. During the study, all participants were required to visit web interface where they have been asked whether or not they would had been comfortable sharing presented location with the target social group, and during indicated period of time.

After the study, from all gathered participant behavioural data, ground-truth space and rule space of users were generated according to the privacy setting types wanted to be tested. For example, time was discretized into half-hour blocks (i.e. the size of time dimension of ground-truth space was 48) and rules were losslessly compressed by merging the location rules which are parts of same time interval and target group, or the time rules which have same locations and target group attributes. Later, best k-rules those together form best policy, are selected from rule space using combinatorial problem approach.

To explain these steps in more detail, we will now define some necessary terms as are used in the system [6], as well as in the implementation (Chapter 7.1) of this thesis.

4.1.1 Preliminary definitions

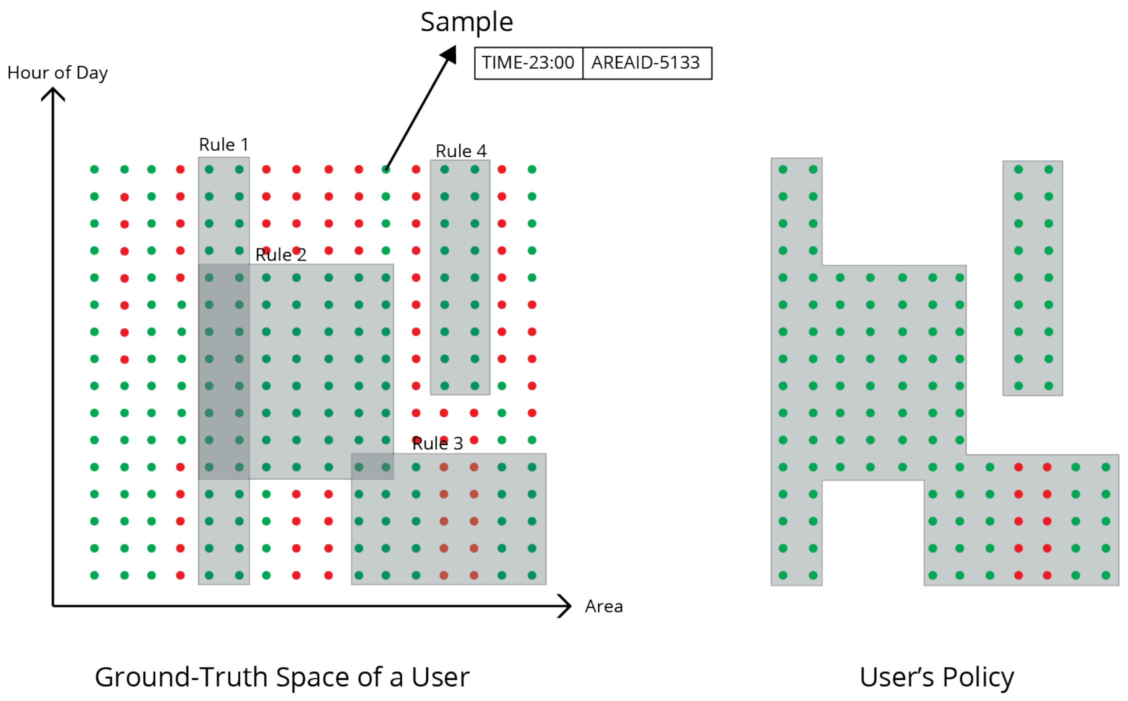

Figure 4.2 - Ground-Truth space and policy example

- **Ground-Truth of the User** is a multi-dimensional space whose dimensions are the attributes in considered by a privacy setting. For example in Figure 4.2 the privacy setting has so and so privacy attributes. A policy is set by the user by defining his sharing (Boolean) decision about whether he wishes to share or hide his location information for specific combinations of the attribute values. In other words, during the time of gathering location events, how many times user presented his location being sharable or not. The used weight function w(s) is;

$$w(s) = \frac{\sum s_{positivehit} - c * \sum s_{negativehit}}{\sum s_{hit}} \tag{4.1}$$

where *c* is the cost which is used to alter the balance between over-share and under-share cases. For example, in Figure 4.2, weight of red samples are below zero while the green samples are above zero.

Number of dimensions are directly proportional to number of privacy setting type, and the size of a dimension increases while more discretisation is applied in corresponding privacy setting type. As example, time is discretized into 16 slots in Figure 4.2 and 16 different locations are considered for the user. Different granularity level used by the privacy system or user visits to more distinctive locations would increase size of area dimension as well as the effort imposed on the user in specifying his sharing behaviour.

- **Rule** is the Boolean cluster of sample points on the ground-truth space. It may overlap positive, negative or both sample points. However, once the rule is appended into privacy policy, all overlapped sample points should be behaved by the location-sharing system as either positive or negative and not both. Changes in user attitudes may cause a rule to over-share. For example, Rule 3 in Figure 4.2 had been initially specified by the user as he wanted to share his seven locations at given timeslots, but he later decided to not closure two locations, leads Rule 3 to over-share.

- **Privacy Policy** is a collection of the rules. Often, they consist of only positive rules that share information or negative rules which hide it. Addition of the accuracies of multiple rules in a policy may not be correct, since sample points from overlapping rules are considered twice resulting in false accuracy estimation. For example in Figure 4.2, Rule 2 is overlapping with Rule 1 and Rule 3, naïve approach to sum accuracy of all rules can cause approximately 9% false accuracy improvement on policy in case each sample is considered as Boolean point.

- **A* Search Algorithm** is a method to find shortest find paths in graphs and offers a faster version of the Dijkstra's algorithm. Rather than expanding the search onto all possible nodes until goal node is reached, it starts from specific node and estimate cost to goal node d for all neighbouring nodes at each step. Then, the node that minimizes the cost to the destination most is selected to expand the path. Hence, it iterates faster compared to Dijkstra's algorithm. It uses cost function to estimate cost of each node. The cost function *f(n)* is;

$$f(n) = g(n) + h(n) \tag{4.2}$$

where n is the node, *g(n)* is the total cost so far acquired by the path and *h(n)* is the heuristic function. The heuristic function estimates the cost of the shortest path from node n to the final node *d*. It is problem-specific and must be admissible, i.e. it must not estimate the cost higher than the actual cost to reach a goal. Otherwise, A* algorithm may overlook the optimal solution.

4.1.2 Formal Definition

When there is no limit on the number of rules that form the privacy policy, finding the best policy that matches user behaviour is easy. The best policy for a user can be identified by greedily generating atomic rules for each sample in ground-truth space and adding them to the policy. However, a user himself may not able to define a large number of rules since rule definition cause user-burden. Hence it is not reasonable to use the greedy approach to represent an actual user. On the other hand, with limit on the total number of rules, say ‘k', identifying the best policy for a given privacy type is a combinatorial problem (e.g. traveling salesman problem, knapsack problem). In order to address this problem, authors [6] have developed a tree-search technique, based on the A* search algorithm.

Each level of the search tree represents a rule of the policy and each edge resembles a particular rule as can be seen in Figure 4.3. For example, when search algorithm expands the path until depth 3, the path corresponds a policy with 3 rules.

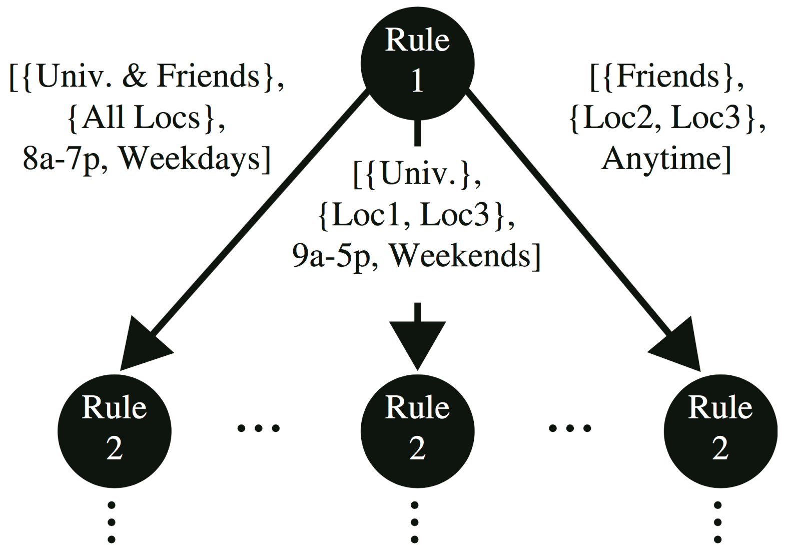
Rule 1
[{Univ. & Friends}, {All Locs}, 8a-7p, Weekdays]
[{Univ.}, {Loc1, Loc3}, 9a-5p, Weekends]
[{Friends}, {Loc2, Loc3}, Anytime]
Rule 2
Rule 2
Rule 2

Figure 4.3 - Part of a search tree for identifying a user’s most accurate privacy policy with Loc/Time+ settings [6]

The search starts at the root node (Rule 1 at depth 0 in Figure 4.3) and constructs child nodes for every remaining feasible rule at each depth. Then, heuristic function approximates the cost of the child nodes and they are added to a priority queue. When next node is popped off from the queue, the algorithm repeats the same procedure and find next rule to include in the policy. A rule is considered feasible if it does not overlap with any rule which is already included in the search path (already added to the incomplete-end-policy). Two rules overlap if they cover same sample points in the ground-truth space of the user.

The heuristic function approximates the accuracy of any policy with k rules originating from a node at depth level n as the total accuracy of rules included so far at the path (an incomplete-end-policy with n-1 rules), plus the accuracy of the policy generated from all remaining feasible rules with no rule limit. This strategy of heuristics ensures that any node contained in the search path has a lower or equal cost than any previously contained node[2], therefore guarantees the path first reached to the depth-k, is most accurate one possible.

[2] Using greedy solution on remaining nodes except the particular node j at the depth n, determines the total accuracy when rule j is missing. Hence, this technique guarantees that the node which has most impact on the accuracy is chosen at each depth

The accuracy function defined calculates correct and incorrect hours caused by the policy *p* of participant *i*, with target social group *G*. Later, it is normalized by each participant's optimal policy p^* which perfectly fits the participant's privacy-preferences.

$$acc(i,p,G,p^*) = \frac{correct\_hours(i,p,G) - c * incorrect\_hours(i,p,G)}{correct\_hours(i,p^*,G)} \tag{4.3}$$

where c is associated with the cost of accidently disclosing a location.

4.2 Optimizations

Converting construction of policy into combinatorial problem and solving it with A* algorithm, is definitely a favourable approach to simulate best possible policy with k-rules given the feedback from participant himself. The authors have achieved accuracies of policies defined using this algorithm in the range 60% up to 80% on a three-week long user-data. They have also explained that splitting rules into atomic ones (i.e. a rule for each sample in ground truth) greedily adding them into policy whenever they would result in positive accuracy, is an option to identify the most accurate policy, but causes policy to include excessive number of rules and is not reasonable for expressing user attitude.

Applying A* algorithm on three-week long data with participants' labelling oriented direct feedbacks, is adequate to reach decent accuracies with least number of rules since ground truth space of users are not massive and users' behaviours towards sharing would not show sharp changes in three weeks. However, participants would demonstrate more attitude changes in a dataset whose span is a year or more. Along with the excessive scale of the ground truth spaces and highly dispersed positive samples in the space, it is not feasible to reach decent accuracy with naïve compression of rule space based on their location or time, and with reasonable number of rules.

In order to address this, we have run A* algorithm twice; first for compressing rule space and second for generating policy.

4.2.1 Compressing the Rule Space

A ground-truth space compiled from a year-long feedback or sharing behaviour of users hosts pre-defined rules (clusters of samples of ground-truth space) with varying scopes. Over time, majority of these rules cause over-sharing if their coverage over the attribute space is broad. Otherwise when their coverage is narrow, the required number of those rules becomes very large for generating the best policy that may cover majority of the positive sample points in the ground-truth space of a user. This problem can be solved by defining anew rules from ground-truth space directly using combinatorial problem approach and A-star search algorithm. Our aim at compressing rule space phase is, identifying partitions as broad and as accurate as possible. In other words, generating as least as possible rules with decent accuracy from the ground-truth space.

Different than generating best policy from pre-defined rules, grouping sample points into clusters is approximating as closely as possible of partitions in ground-truth space. If these partitions have well-formed boundaries with each other, rules produced will be deterministic and the best, and not depends on the root node. Otherwise, if partitions are intersecting with each other and have not strong boundaries, rules generated will be non-deterministic and possibly the best depending on the root nodes and cost factor.

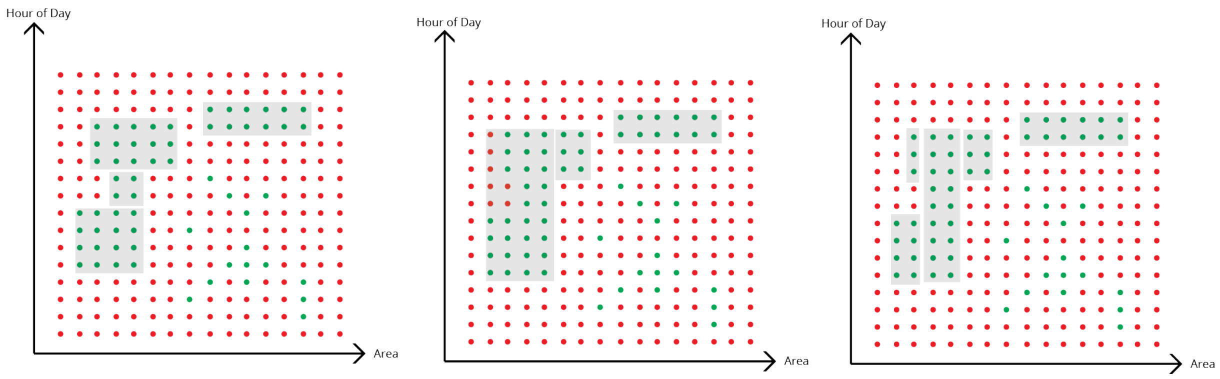

Figure 4.4 - Clustering of samples in ground-truth space. Left-middle cluster is non-deterministic because root node, cost factor and iteration

Figure 4.4 shows 2D ground-truth spaces as example. While the right-above group has not variations and would form same cluster not depending on the root node or iteration, left-middle group can form varying clusters depends on cost factor, root node and iteration, and therefore is non-deterministic.

Total coverage of a rule (number of sample points it covers) for Loc/Time+ settings (4D space) is given as;

$$totalcoverage = \Sigma_{uniqueareas} * \Sigma_{uniquetimes} * \Sigma_{uniquetargets} * \Sigma_{uniqueweekendflags} \tag{4.4}$$

4.2.1.1 Algorithm

Before running A* search algorithm, ground-truth space is constructed for each user given part of the dataset (training proportion). The following formula is used to score a sample *s* in dataset similar as previously discussed. Score function is different than weight function because we required to prioritize most popular rather than most accurate samples for covering. The total score is later normalized in cost function.

$$score(s) = c_p * \sum s_{positivehit} - c_n * \sum s_{negativehit} \tag{4.5}$$

where C_n and C_p can be tinkered in order to tamper over-share and under-share ratios caused by generated rule. We will later discuss this values in evaluation Chapter 7.

Since it is not possible for A* algorithm to reach destination (maximum achievable accuracy) with only one rule, we should define a maximum depth and/or maximum iteration that algorithm runs until.

Similar to original algorithm from [6], our search starts at the root node and builds a child node for each *feasible* positive sample in constructed ground-truth. A child node is a rule alone that covers the partition whose boundaries are properties of previously added samples + a child node itself. The nodes are added to a priority queue along with their cost from the heuristic function, and then are popped off the queue one at each step until maximum depth or iteration is reached. The best combination of boundaries is recorded at each step, in case the algorithm overestimates. A sample is considered *feasible* if it is not covered by the rule boundaries of parent node. A sample which has not received a hit (i.e. was not in ground-truth space of a user) is acknowledged as void point and has not influence on accuracy. Hence, they can be covered by the rule when required.

After each rule generation, the samples which are not covered previously created rules are considered *remaining* samples and further rules are searched through those samples. In other words, already covered samples whether their score was positive or negative, are considered void points and will not have positive or negative effect on generation of further rules. False accuracy estimation of overlapping rules is thus avoided, it results broad clusters at each generation step and affects positively identifying the policy at later phase.

Unfortunately, since generation of rule space phase has distributed processing of A* algorithms, their exclusive information exchange is set of remaining samples. The destination of each A* algorithm computation varies with root nodes, maximum iteration and maximum depth hence it is non-deterministic, so that the policy identified later is probably the best. However, our aim is to simulate a user policy with decent accuracy and reasonable number of rules, and not to identify best set of k rules. Therefore, following simple cost function have been used to allow search algorithm advancing.

$$f(n) = 1 - \frac{\sum score(s \in \mathrm{N})}{\sum score(s \in \mathfrak{p})} \tag{4.6}$$

where *N* is sample space covered by node *n*, and $\mathfrak{p}$ is sample space covering all samples with positive scores.

With this technique, cost of a node is guaranteed to decrease at each depth if a rule generated so far results in better accuracy.

On the other hand, because root node has perceivable influence on identifying best boundaries for partition, they are chosen randomly from *remaining* samples whose attributes are in the intersection space of five or less most occurred time and locations, at each rule generation. If the union is empty then root node is chosen randomly from all *remaining* samples.

$$\mathcal{S}_{rootnodes} = \mathcal{S}_{mostoccuredtimeslots} \cap \mathcal{S}_{mostoccuredlocations} \tag{4.7}$$

All created rules are later fed into A* algorithm for policy generation phase.

Algorithm 4.1: Generating Rules

```
    Input: dataset for a user, cost factor c, maximum depth k, maximum iteration i
    Output: set of rules ruleset

1:  begin
2:  initialize groundtruth with cost c from dataset
3:  ruleset = []
4:  remainingsamples = groundtruth
5:  while remainingsamples
6:          ST = five or less most occurred times in remainingsamples
7:          SL = five or less most occurred locations in remainingsamples
8:          RS = ST ∩ SL
9:          if RS not empty then
10:                 rootnode = randomly choose from RS
11:         else
12:                 rootnode = randomly choose from remainingsamples
13:         rule, remainingsamples = ASTAR(rootnode,remainingsamples,k,i)
14:         ruleset ← rule
15: end
16: return ruleset
17: end
```

4.2.2 Identifying Policy

Compared to compressing rule space, identifying policy from generated rules is more straightforward. Root node is opted empty, since one instance of algorithm will run during whole phase. At each step, child nodes are generated for all remaining nodes, they are added to priority queue along with their cost from heuristics, and later are popped off the queue until maximum depth is reached. In the end, an *optimal* policy (privacy preferences for a user, set of k-rules) is identified. The privacy is not the best but the optimal, because two A star algorithm runs separately. The rules generated from first A star algorithm aims to be best clusters, however, when identifying privacy policy, they may not result in best possible policy for a user with k-rules. The same cost function from Section 4.2.1 is used.

Algorithm 4.2: Identifying Policy

```
    Input: dataset for a user, cost factor c, set of rules ruleset, rule limit k
    Output: policy P

1:  begin
2:  initialize groundtruth with cost c from dataset
3:  rootnode = []
4:  P = ASTAR(rootnode,groundtruth,ruleset,k)
5:  return P
6:  end
```

5. Detecting out-of-routine Events

This chapter will discuss in detail the approach [16] which utilized conditional trajectory entropy estimation in order to distinguish waypoints on the user trajectory and segment the trajectories. Their research has shown that conditional trajectory entropy value of locations can be used as indicator of routine and out-of-routine events, and users are prone to expose behaviour changes (e.g. spend more time) at the locations with higher entropy values. Later in the chapter, we will explain necessary modifications we applied on their work in order to detect entropy values instantly, and adapt the system into decision making process.

5.1 Background

5.1.1 Computation of Entropy

The entropy in Information theory, is the measurement which used to estimate amount of information (i.e. amount of surprise) of the stochastic source of data, which is explicity written as;

$$H(X) = -\sum_{i} P(x_i) \log P(x_i) \tag{5.1}$$

where P is probability distribution matrix of a source of data.

Alternatively, the entropy expression for the trajectories is used to quantify the randomness of trajectories of an irreducible finite state Markov chain with given initial state *s* and final state *d*. Instead of estimating entropy of one state, Ekroot et al. [14] has proposed a method which can estimate entropy value of transition H_{sd} between state *s* and *d* from probability distribution of every possible trajectories between each state.

$$H(\chi) = -\sum_{i,j} \mu_i P_{ij} \log P_{ij} \tag{5.2}$$

Number of applications in graph theory and in statistical physics require computation of the entropy of Markov trajectories. Yet, some of them demands calculation of the entropy of markov trajectories with specified intermediate states as studies of random walks on graphs do. Consequently, another expression for entropy of conditional markov trajectories was later proposed by Kafsi et al. [15] which builds on top of the definition from Ekroot et al. [14].

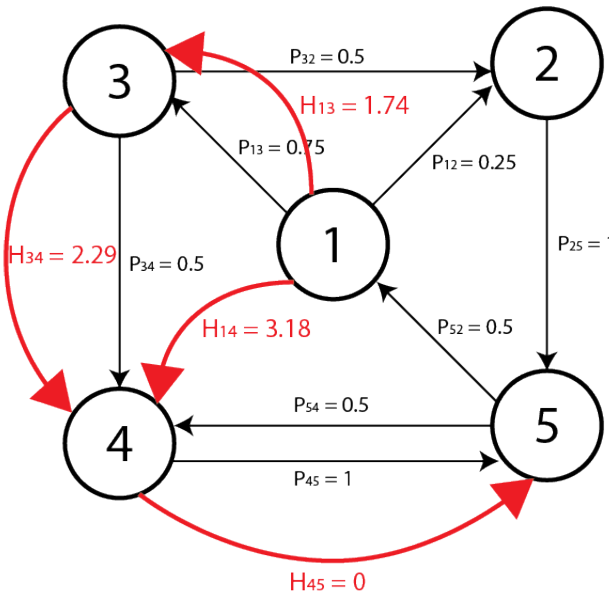

Figure 5.1 - An irreducible 5-state Markov Chain example as given in [15]

Unfortunately, conditional entropy $H_{sd|u}$ is not computationally easy since it is not the entropy of the random variable *sd* given random variable *u* but given realization of random variable *u*. Furthermore, simply using additivity property does not hold. When we consider the example from [15] in Figure 5.1, if we would like to compute entropy of trajectory 1-5 conditional on 4 ($H_{15|4}$), simply calculating $H_{13} + H_{34} + H_{45} = 4.03$ or $H_{14} + H_{45} = 3.18$ gives inaccurate results because $H_{15|4}$ should be zero since only one path exists supporting the condition, which is 1-3-4-5.

However, according to the authors, the conditional entropy of trajectory where *s* is source, *d* is destination and $u = u_1 u_2 \ldots u_l$ is sequence of intermediate states, can be measured using the chain rule for entropy as;

$$H_{sd|u} = H(T_{sd}|T_{sd} \in T_{sd}^{u}) = \sum_{k=0}^{l-1} H_{u_k u_{k+1}|\bar{d}} + H_{u_l d} \tag{5.3}$$

where $u_0 = s$. In order to compute $H_{u_k u_{k+1}|\bar{d}}$, P'_k matrix was defined as

$$(P'_k)_{ij} = \begin{cases} 0 & if\ i = u_{k+1}, d\ and\ i \neq j \\ 1 & if\ i = u_{k+1}, d\ and\ i = j \\ P_{ij} & if\ i \neq u_{k+1}, d\ and\ \alpha_{idu_{k+1}} = 1 \\ \dfrac{1-\alpha_{jdu_{k+1}}}{1-\alpha_{idu_{k+1}}} P_{ij} & if\ i \neq u_{k+1}, d\ and\ \alpha_{idu_{k+1}} < 1 \end{cases} \tag{5.4}$$

α_{sud} expresses the probability that the random trajectory T_{sd} goes through the state *u* at least once; and can be computed making states *u* and *d* absorbing and calculating the probability to be absorbed by state *u* from starting state *s*, as technique from [17] indicates.

Being absorbed by intermediate state u before destination state *d* guarantees that the trajectory goes state *u* before *d*, and it may afterwards finish up at *d* in finite time.

After the transformation (5.4) is applied on P, P'_k is not necessarily irreducible anymore and conflicts with closed-form expression from [14]. Therefore, new expression for the entropy of the trajectory has been proposed by [15];

$$P = \begin{pmatrix} Q_d & P_{1d} \\ P_{d1} & P_{dd} \end{pmatrix} \tag{5.5}$$

where P is the transition probability matrix of a finite state such that there exists a path from any state to *d*, and

$$H_{sd} = \sum_{k \neq d} ((I - Q_d)^{-1})_{sk} H(P_{k\cdot}) \tag{5.6}$$

where $H(P_{k\cdot})$ is the local entropy of state k.

In order to calculate $H_{u_k u_{k+1}|\bar{d}}$; we transform matrix P into P'_k at first. Let's assume *S₁* and *S₂* are two subsets of P'_k where $S_1 = \{i \in S : \alpha_{iu_{k+1}} > 0\}$ and $S_2 = \{i \in S : \alpha_{iu_{k+1}} = 0\}$. *S₁* and *S₂* does not intersect with each other and they are clearly a partition of P'_k, because $P'_{ij} = (1 - \alpha_{jdu_{k+1}}) \,/\, (1 - \alpha_{idu_{k+1}})\, P_{ij} = 0$ if $i \in S_1$ *and* $j \in S_2$. Then, Q_{u+1} which is extracted from *S₁*, can be used in (5.6) to calculate $H_{u_k u_{k+1}|\bar{d}}$.

Finally; $H_{u_l d}$ in (5.3), can be calculated directly from P using (5.5) and (5.6).

Algorithm 5.1: Conditional entropy calculation as given in [15]

Input: Transition probability matrix *P*, source state *s*, destination state *d*, sequence of intermediate states *u* = *u₁ . . . u_l*
Output: $H_{sd|u}$

```
1:  u_0 = s
2:  sum = 0
3:  for k = 0 to l – 1 do
4:          Compute $P'_k$ from P using (5.4)
5:          Compute $H_{u_k u_{k+1}|\bar{d}}$ from $P'_k$ using (5.5-5.6)
6:          sum = sum + $H_{u_k u_{k+1}|\bar{d}}$
7:  end
8:  Compute $H_{u_l d}$ from P using (5.5-5.6)
9:  $H_{sd|u} = sum + H_{u_l d}$
10: return $H_{sd|u}$
```

5.1.2 Related Work

An application [16] discussed previously, was successfully implemented a system that detects possible intermediate waypoints on subject’s trajectories. However, to being able to train feasible mobility Markov chain that captures the patterns of mobility, and evaluate the system; large-scale dataset is one prerequisite. Therefore, they have used GPS trajectories within the city of Beijing, China in the Geolife dataset [18] which collected mobility traces of 182 subjects for five-year period and includes approximately 18,000 trajectories, as can be seen in Figure 5.2.

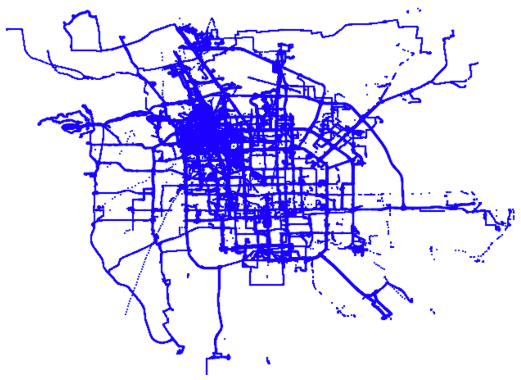

Figure 5.2 - GPS trajectories within the city of Beijing, China [16]

In order to train mobility Markov chain; they first have discretized the GPS records into squares whose side lengths are 1km and encloses set of locations as graph abstraction of the world. Examples of representation of squares were shown in Figure 5.4. Secondly, a weighted graph *G(V,E)* was formed whose vertices represent discretized areas previously, edges represent transition between areas and weight of the edge denotes count of the transition occurred. Jumps due to loss of GPS signal and self-transitions were excluded from the graph. Finally, the first order Markov chain that captures the patterns of population mobility, was constructed from *G(V,E)* using a maximum likelihood estimator because available train data was not sufficient to train high order high order Markov chain.

Before evaluation phase, they have distinguished intermediate waypoints from intermediate points in the dataset using the residence time a subject spent. Secondly, they conducted an experiment, where average time $\hat{\mu}_\alpha$ spent at intermediate locations which satisfy the inequality $H_{sd|u} > \alpha H_{sd}$ is calculated with different values of α. $s \rightarrow d$ traces were frequently associated with home to work, however, shopping and sightseeing were also included as starting or destination points at times.

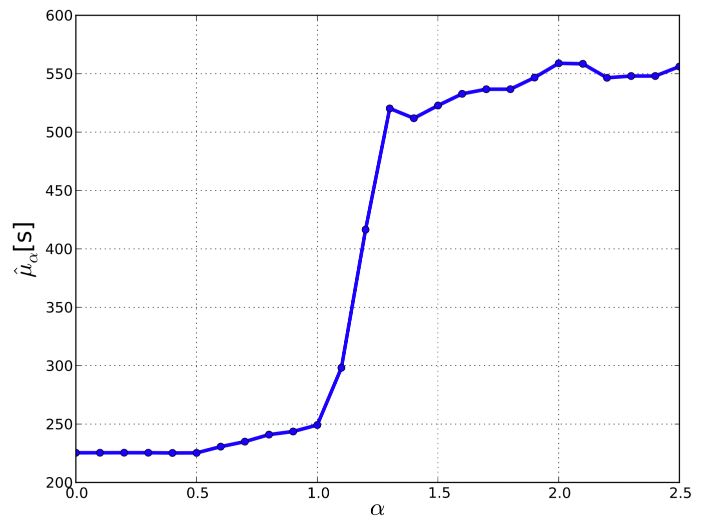

Figure 5.3 - Average residence time $\hat{\mu}_\alpha$ of remaining locations increases with a filter factor α [16]

The results in Figure 5.3 showed that average time $\hat{\mu}_\alpha$ increases with α, and there was sharp transition in the value of $\hat{\mu}_\alpha$ as soon as α is considered as greater than one (i.e. the intermediate locations where conditional trajectory entropy is less than trajectory entropy, are filtered out). Since residence time quantifies the importance of the location, the intermediate locations withheld with high values of α were considered most likely waypoints.

For trajectory segmentation by extracting intermediate waypoints, they have proposed a recursive algorithm. The algorithm first tries to find a waypoint (line 9) that may segment the trajectory, and if it is succeed, other possible waypoints are searched through trajectories split (line 12-17). In every step, the waypoint that most increases the conditional entropy were taken into consideration (line 21). A sensitivity parameter α is used to make algorithm more selective at extracting waypoints (line 23). Average time $\hat{\mu}_\alpha$ spent at intermediate locations which satisfy the inequality $H_{sd|u} > \alpha H_{sd}$, increases with α. Higher α values ensure selection of more significant waypoints.

Algorithm 5.2: Trajectory segmentation as given in [16]

```
     Input: trajectory traj, transition probabilities matrix P, sensitivity α
     Output: indices of waypoints U

1:   begin
2:   U = []
3:   if len(traj) > 2 then
4:           segment(traj, 0, len(traj) - 1)
5:   end
6:   return U
7:   end

8:   Function segment(traj, i, j)
9:   k = partition(traj, i, j)
10:  if k >= 0 then
11:          append(U, k)
12:          if i + 1 < k then
```

```
                segment(traj, i, k)
        end
        if k + 1 < j then
                segment(traj, k, j)
        end
end

Function partition(traj, i, j)
s = traj[i], d = traj[j]
k = argmax$_{i<k<j}$ $H_{sd|traj[k]}$
u = traj[k]
if $H_{sd|u} > \alpha H_{sd}$ then
        return k
else
        return 1
end
```

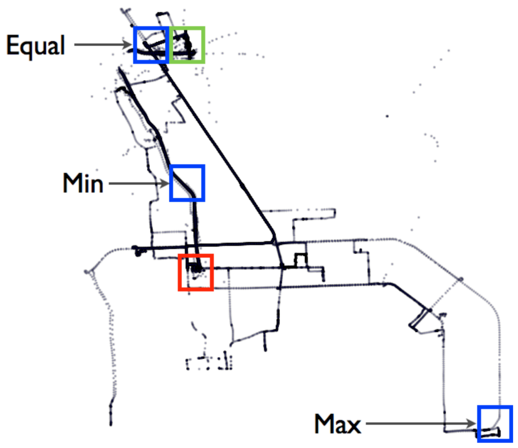

Figure 5.4 - Intermediate location near to the starting location has almost no effect on the conditional entropy. Revealing an intermediate point along popular trajectories decreases conditional entropy at high amount. However, revealing the intermediate location in the lower-right corner increases the conditional entropy most [16]

In Figure 5.4, all raw trajectories between initial waypoint (green square) and destination (red square) were shown. The conditional entropy of intermediate points (blue squares) near to starting point were equal to non-conditional entropy H_{sd}. While approaching to the destination, conditional entropy value of intermediate points was decreasing and lower than H_{sd}; meaning they significantly increased trajectory predictability and were less likely intermediate waypoints. However, the intermediate point in the lower-right corner maximized conditional entropy by increasing randomness of the trajectory; meaning this point was not on the most popular paths between source and destination and possibly were not just an intermediate point on route to the destination but intermediate waypoint instead.

5.2 Optimizations

The routine aware control mechanism presented in Chapter 3 should be able to instantly calculate entropy of each location, using only available information which is recent location, history of location activity and Markov chain generated based on location activity. However, the presented work from Kafsi et al. [16] is aimed to extract waypoints and segment trajectories given start and final destinations, which is not appropriate for real-time estimating because final destination information would not be known by the system.

In order to address this, we have modified the original algorithm where each recent location is considered as a destination waypoint and previously visited locations are either intermediate points or source location. If a location *loc* visited had been logged before in the location history *traj* at position *i* or as a source at position 0, intermediate point u_{i+1} is assigned as a new source location, and new path information is constructed as *newtraj* = u_{i+2} u_{i+3} … u_n where u_n is previously visited location before *loc*. The example is given in Figure 5.5. Once user visits a location, the entropy can be calculated instantly given the trajectory history. Other difference from the original algorithm is, intermediate locations are not fed into entropy calculation one by one but as a whole trajectory, since our aim is to estimate entropy of destination utilizing whole location history log.

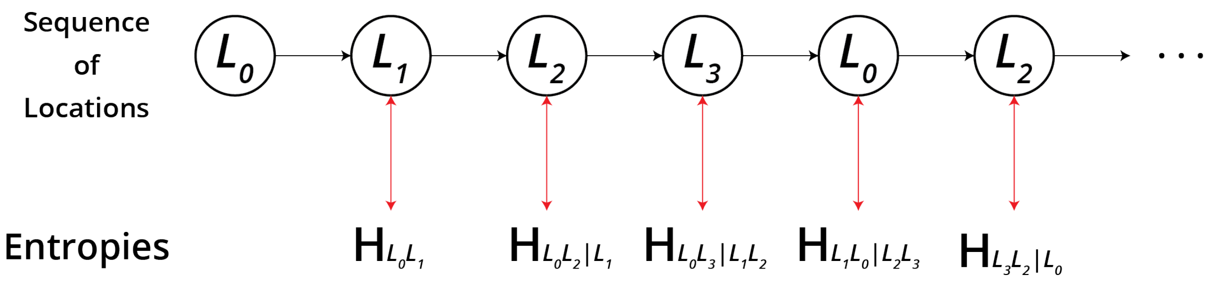

Figure 5.5 - Entropy is calculated at each location update of a user

Since series of locations are not generated randomly but from user mobility behaviour, and holds information about the routineness of the trajectory towards destination, this approach guarantees to find if recent location is reached by using the routine or out-of-routine trajectories. However, because system considers the trajectory routineness, it cannot differentiate a routineness specific to one location. In other words, once a location which emits high entropy is visited, all further locations will emit high entropy as well, until path information is overwritten. This effect would be avoided, if we could precisely predict subject's destination location at a specific time, and estimate entropy between initial and destination location given condition on the recent location.

In the optimized Algorithm 5.2, time tolerance *tol* is additionally designated in order to distinguish occurrence of missing location updates, and specify the entropy value *missent* of location update after the missing update.

Algorithm 5.2: Entropy of Locations

```
    Input: trajectory traj, transition probabilities matrix P, recent location L, time tolerance tol,
    missing entropy value missent
    Output: entropy of recent location ent, updated trajectory newtraj
1:
2:  begin
3:  destination = L
4:  prevlocupdatetimestamp ← traj
5:  range = currenttime – prevlocupdatetimestamp
6:  if range < tol then
7:          if destination in traj then
8:                  get location index i from traj
9:                  source ← traj[i+1]
10:                 path ← traj[i+2:]
11:                 ent = compute_cond_entropy(source,destination,path)
12:                 newtraj ← traj[i+1:]
13:         end
14:         else
15:                 source ← traj[0]
16:                 path ← traj[1:]
17:                 ent = compute_cond_entropy(source,destination,path)
18:                 newtraj ← traj
19:         end
20:         newtraj ← destination
21: else
22:         source ← L
23:         ent = missent
24:         newtraj = ø
25:         newtraj ← source
26: return ent, newtraj
27: end
```

6. Data processing

As indicated in [16], constructing mobility Markov chain from location events requires large-scale dataset to accurately cover all possible trajectories and probabilities. Thus, we have used anonymized version of Reality mining dataset [24] from MIT Human Dynamics Lab, generated by interoperating of 94 participants (90% student and 10% staff) over a ten-month period between 2004 and 2005. The dataset contains phone logs, Bluetooth scans, application usage, location events including cell tower IDs and cell tower area-IDs, phone status and survey data of the users. However, we will require location event and phone (call) log attributes for the implementation and evaluation of the system, as discussed next sections.

Data continuity (i.e. having less missing data) plays an important role in the training and testing phase. The continuity of dataset for users is shown in the Figure 6.2 where the black line stands for a location event, green line for outgoing or incoming communication and red line for declined or missed communication. Furthermore, distribution of time interval (hours) between each location event in the dataset for users is given in Figure 6.1. Overall, majority of participants can provide important insights in our implementation and evaluation.

Following sections, we will discuss required pre-processing and processing on the dataset for the evaluation.

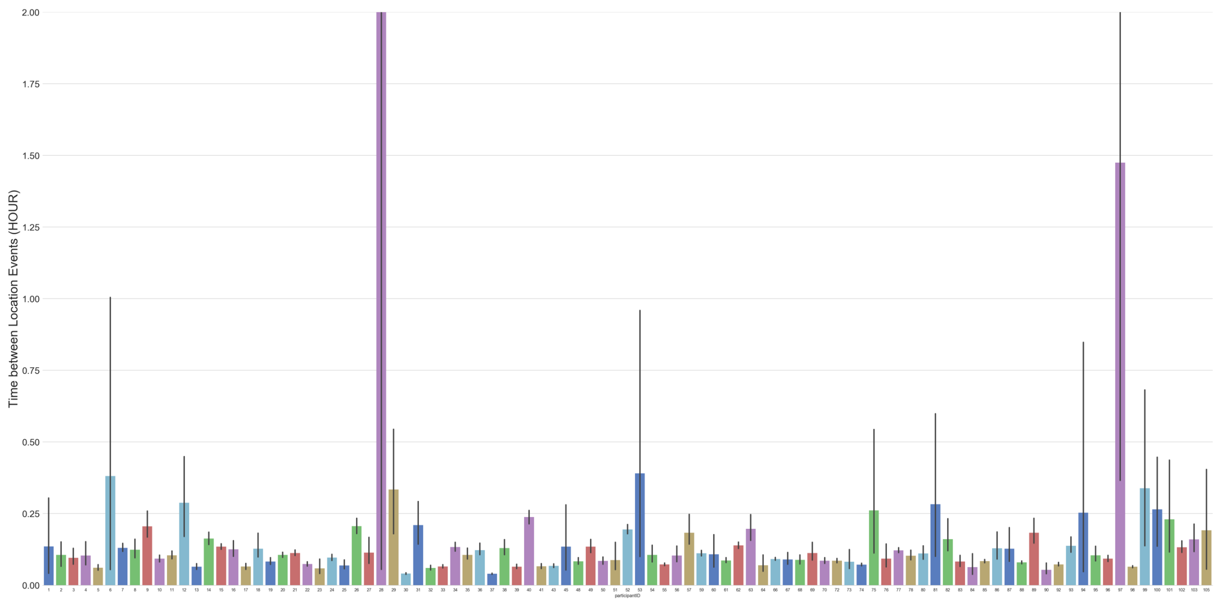

Figure 6.1 - Distribution of inter-arrival time (IAT) of location updates (samples) per participant

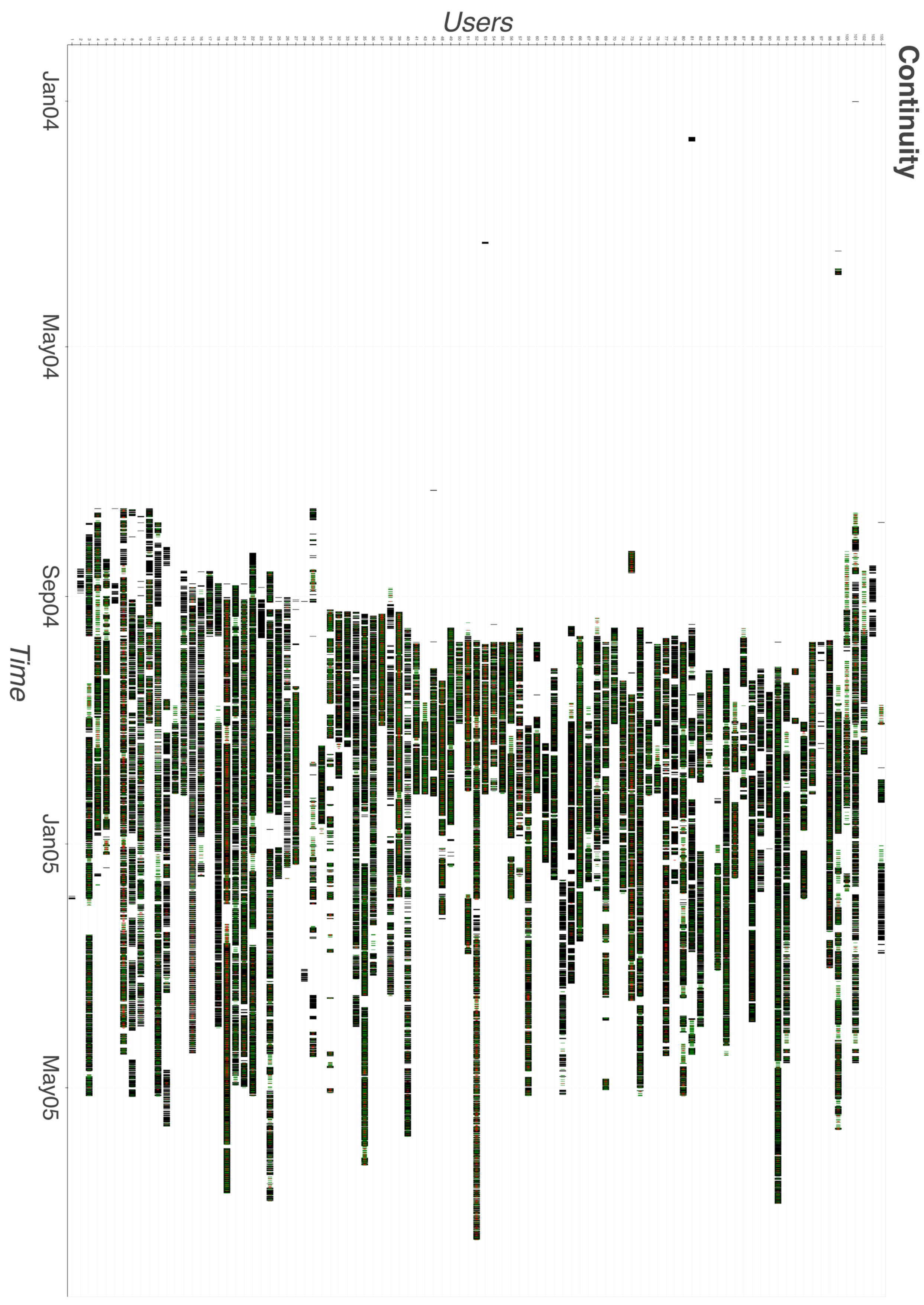

Figure 6.2 - Continuity chart of Reality Mining dataset. Blacks are for location updates (samples), greens are for incoming/outgoing call events, reds are for missed/declined calls

6.1 Pre-processing Dataset

In this section, we will discuss the required attributes and pre-processing of the dataset before we could utilize it for our evaluations, and how we have applied them.

6.1.1 Required Attributes

Concerning evaluation and training of the system, we will need extensive series of location events of users including the timestamp and spatial information to construct mobility Markov chain, and their sharing behaviour or feedbacks (i.e. labelled data) with detailed properties according to privacy setting types used (i.e. target social group, time, day, location etc.) in order to extract the ground truth and generate best possible policy. Unfortunately, most of large-scale datasets does not come with labelled data that shows participants' sharing behaviour and Reality mining dataset is hardly different. However, we can take advantage of phone log data of each user and imitate the ground truth as explained in next sub-section.

6.1.2 Pre-processing Location Samples

participantID	date	areaID	cellID
3	2005-01-27T17:39:54	5188	40332
32	2004-09-10T02:49:51	24127	132
86	2004-11-09T18:37:57	11212	56901
70	2004-10-09T13:22:07	100	11557
43	2004-11-22T22:16:46	5188	40353

Table 6.1 - Sample from location updates (samples) dataset

Reality mining dataset consists of series of location samples (Table 6.1) as data-stream whose average estimated inter-arrival time for participants are shown in Figure 6.1. Because the dataset is anonymized, area ID of cell towers and cell tower IDs are covered instead of spatial reference system. Even though it would not be sufficient for detailed study or analysis, considering area IDs as collection of spatial coordinates at low granularity level is to our advantage considering size of the dataset, and hinders the requirement of running accurate classification algorithm. Using low granularity level lowers the size of the ground truth making it more authentic to simulate user's best policies, and eases construction of Markov chain. Therefore, we have avoided using cell tower IDs and processed area IDs directly.

In order to process, evaluate the system and be able to label the dataset, we require the locations with residence time information instead of series of location samples. Hence, we have pre-processed sequence of location samples and construct location events with residence time feature using Algorithm 6.1. The input *tol* allows us to define tolerance interval for missing samples between sequence of same location samples. Sample table from pre-processed location dataset with tolerance time of thirty minutes is shown Table 6.2.

Algorithm 6.1: Constructing location events with residence time

```
     Input: unlabelled location samples dataset ULSD of user, tolerance time tol
     Output: unlabelled location events dataset ULD of user

1:   begin
2:   sort(ULSD)
3:   initialize prevarea,RS,RE
4:   for sample in ULSD
5:       if sample['areaID'] == prevarea then
6:           if sample['date'] – RE > tol then
7:               ULD ← areaID, RS, RE
8:               RS = sample['date']
9:               RE = sample['date']
10:          end
11:          else
12:              RE = sample['date']
13:          end
14:      else
15:          ULD ← areaID, RS, RE
16:          RS = sample['date']
17:          RE = sample['date']
18:          prevarea = sample['areaID']
19:      end
20:  end
21:  return ULD
22:  end
```

areaID	dateperiod	participantID
24124	2005-05-02T11:17:13_2005-05-02T11:17:22	19
24127	2004-10-27T06:47:26_2004-10-27T07:41:18	49
24123	2004-10-10T17:56:49_2004-10-10T18:05:09	62
5119	2004-10-08T17:09:57_2004-10-08T17:11:10	40
11027	2004-12-28T09:03:35_2004-12-28T09:23:18	66

Table 6.2 - Sample from location events dataset

6.1.3 Labelling Location Events

The authors in [27] have presented a work including 42 participants that explores which possible features of interpersonal relationships influence sharing and at which extent. Their results show that closeness of participant with an individual is the strongest indicator of willingness to share his location, and frequency of communication approximates closeness and willingness to share. Along similar lines, another research [29] has found that the most important factors were the requester's identity and the reason behind his query. By referencing these findings, we have correlated communication events of subject with their intention of sharing.

6.1.3.1 Pre-processing Phone Logs

participantID	date	event	numberhash	contact	type	direction	duration
95	2004-12-20T11:36:53	973	NaN	-1	Packet Data	Outgoing	34
21	2004-11-14T19:04:28	1420	2533.0	-1	Short message	Incoming	0
52	2004-12-25T18:56:20	1975	5737.0	31	Voice call	Outgoing	28
66	2004-11-22T15:21:41	2494	6611.0	245	Voice call	Outgoing	2
83	2004-10-22T19:03:23	488	48.0	28	Voice call	Outgoing	0

Table 6.3 - Sample from phone logs dataset

A communication event in dataset consists of date, hash of phone number, contact ID, type, direction and duration as can be seen in Table 6.3. There are 5 unique types which are packet data, short message, voice call, data call and MMS, and 3 unique directions which are outgoing, incoming and missed. We have first filtered some events in dataset considering following steps;

- The events whose contact is -1 (i.e. the number was not in participant contact list), are thought as they represent negative influence of closeness and filtered.
- Data calls are filtered because they were not towards actual individuals and occurrence rate was 0.01% in whole dataset.
- MMSs are filtered because direction information was missing and occurrence rate was 0.01% in whole dataset.
- Packet data is filtered because target information was missing. Moreover, internet messaging over mobile devices was not popular among users throughout the collecting period of dataset (2004-2005).
- Short messages are filtered. There are several reasons of this action; first, only 0.04% of events in whole short message event set had contact information. Second, we cannot extract neither closeness nor accurate sharing behaviour since declining or missing incoming messages are not an option.

After application of above steps, only voice call events were remained. Yet, we had 470 unique contacts (average 45 contacts per participant), and defining accurate policy with least number of rules was not feasible since 'target' dimension of ground truth would have long length, and dataset is not sufficient to reveal detailed requester profile for each contact. Moreover, users often have more generic social groups with different levels of closeness.

6.1.3.2 Clustering contacts

Defining and identifying community structure in networks is an important issue and addressed by algorithmic methods such as spectral graph partitioning or hierarchical clustering [30]. Although, in our study, we aim to cluster contacts of a participant in terms of closeness level by exploiting communication frequency. Hence, knowledge of whole network was not necessary, and we could apply simple clustering algorithms based on K-Means, DBSCAN, Mean-shift, Affinity Propagation and such, by using features in Table 6.4 for a classifier.

Contact	**Major features**		**Minor features**		
	Communication frequency	Total duration of calls	Morning communication ratio	Afternoon communication ratio	Evening communication ratio

Table 6.4 - Major and minor features used for the classifier. The major features are weighted more than minor features

Before training classifier, major features are transformed individually by using Formula 6.1 (minimum-maximum scaler) so that they will be in range between zero and one, and they are then optionally weighted more than minor features. The algorithms thereby prioritize clustering using major features, but if samples are closely distributed on major features space, then minor features will help to further categorize of contacts. Aledavood et al. [31] has proposed a research shows that daily call patterns of an individual in mobile phone communication differs towards different groups of people by dividing day into 6-hour time spans. We have used similar approach to cluster individuals further by dividing day into 8-hour time spans.

$$X = \left(\frac{X - \min(X)}{\max(X) - \min(X)} \right) \tag{6.1}$$

Including declined or missed calls (i.e. negative behaviour) in our classifier training dataset would not be fair at evaluation of the main system, hence they are not included in the training data. Additionally, the participants who are missing adequate communication data for clustering as well as evaluation of the system, were filtered ahead.

Discovering number of groups (determining number of clusters) for each participant is another issue in actually solving the clustering problem. Hierarchical clustering algorithms avoid this issue by building a hierarchy of clusters in greedy manner, and determine number of cluster itself. However, each participant has their own unique contact network structure resulting varying number of groups with default classifier. It would be also an option which may require additional study but we decided to abide common pattern adopted by the introduced papers in Chapter 2. Preferred number of social groups by these papers are commonly between three and five, and they are addressed as ‘Family’, ‘Close Friends’, ’Friends’, ’Colleagues’ and occasionally ’Advertisers’. Therefore, K-Means classifier with four clusters were applied on the dataset and contacts of participants were grouped into four different social groups. In Figure 6.3, contacts of participants 7 and 56 are shown as example before classifier are applied on them. Then, application of classifier with major features weighted two times more than minor features, is shown in Figure 6.4.

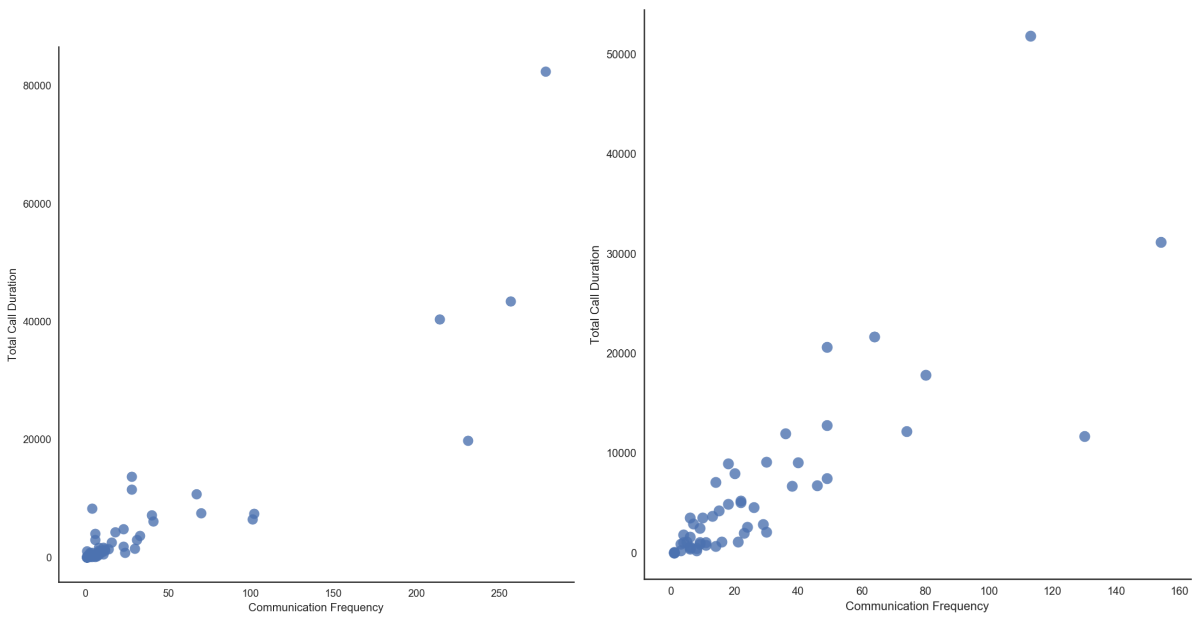

Figure 6.3 - Distribution of contacts for participant 7 (left) and 56 (right)

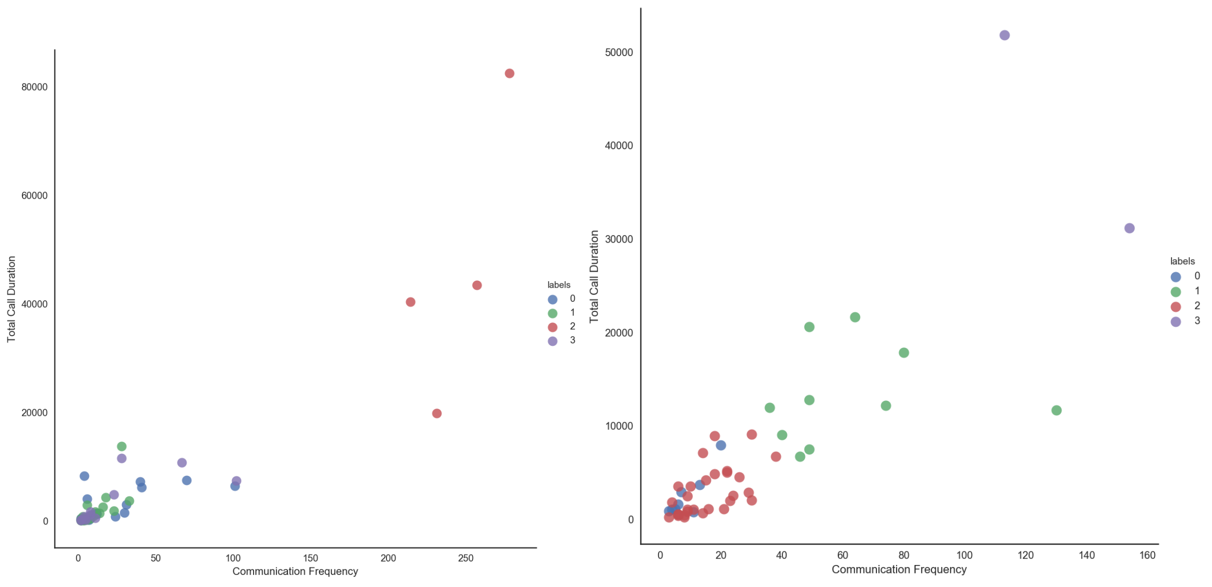

Figure 6.4 - Distribution of contacts after classifier was applied, for participant 7 (left) and 56 (right). Labels (colours) indicate different social groups

6.1.3.3 Labelling

Labelling is done by merging phone logs dataset and location events dataset utilizing Algorithm 6.2. The phone logs dataset has data samples with three main event types which are call, missed call and declined call. Call samples are considered as fair indicator of positive sharing behaviour. Missed calls on the other hand, are treated as negative attitude only if participant has not called back to or took other calls from same social group throughout his stay, in order to depreciate the influence of involuntary missing calls. Yet, categorization of declined calls is discretionary because people declines call either to let requester know their unavailability (i.e. positive behaviour), or to demonstrate their unwillingness towards talking (i.e. negative behaviour). Ultimately, optimistic approach has been taken towards declined calls and they are labelled as positive behaviour.

It is probable participant would not have received or forwarded a voice call during his stay. That cases are treated as 'non-behaviour' and will be discussed next chapter.

Algorithm 6.2: Labelling location events

```
Input: pre-processed phone logs dataset CD of user, unlabelled location events dataset ULD
of user
Output: labelled location events dataset LD of user

begin
LD = ULD
for sample in ULD
        S = sample['arrive date']
        E = sample['leave date']
        actions = get_actions(S,E,CD)
        if len(actions) > 0 then
                LD[sample]['actions'] = actions
        end
        else
                LD[sample]['actions'] = 'Non'
        end
end
return LD
end

Function get_actions(S, E, CD)
U = []
samples = CD[(date > S) & (date < E)]     #check communication events between given dates
for sample in samples
        actiontype = check_action_type(sample)
        target = sample['target']
        U.append([actiontype,target])
end
remove duplicates in U
remove 'Neg' action for a target who has both 'Neg' and 'Pos' actiontype in U
return U

Function check_action_type(sample)
A = sample['type']
if A == 'outgoing' or 'incoming' or 'declined' then
        return 'Pos'     #positive behaviour
if A == 'missed' then
        return 'Neg'     #negative behaviour
```

6.2 Initializing Ground-truth Space

Ground-truth space of a user is the key instrument to evaluate accuracy, under-share and over-share rates of the decision-making system. It is generated from the partition (train set) of labelled locations events dataset of a user, given choice of the discretisation being used in the evaluation. Loc/Time+ privacy settings type from [6] is considered to extract features and form dimensions of the ground-truth. These features include time, location, target social group and weekend flag.

In order to initialize ground-truth space; first, the arrive time and leave time of an event are rounded to the nearest discretised time slot to perfectly distribute event into time dimension of ground-truth. Later, event is discretised into several samples based on time feature, and each sample is labelled with weekend indicator, target social group, area ID and the behaviour of user.

Even after pre-processing phase of location samples, some location events with immensely narrow or zero residence time remained if they were intermediate point in the trajectory or covered by missing data. Therefore, filtering is applied on these events before the final step.

Finally, all distributed samples with same attributes are grouped and simplified into one sample point with additional attributes such as how many times it has received positive, negative and neutral hits. Therefore, these extra attributes can later be used in A* algorithm

participantID	areaID	time	weekend	target	positive	negative	neutral	total
39	11027	00:00	0	0	4	0	15	19
39	11027	00:00	0	1	3	0	16	19
39	11027	00:00	0	2	0	0	19	19
39	11027	00:00	0	3	3	0	16	19
39	24127	00:00	0	0	40	0	10	50
39	24127	00:00	0	1	10	0	40	50
39	24127	00:00	0	2	12	0	38	50
39	24127	00:00	0	3	41	0	9	50

Table 6.5 - Sample from ground-truth space of participant 39. Positive is for number sharing behaviour, negative is for number of intentionally withholding information, and neutral is for number of unintentionally withholding information

In Table 6.5, sample from generated ground-truth space of participant 39 is given. Total represents the overall hits the location received given time and weekend indicator. Positive and negative shows how many times a target received hits at location and given time. Neutral hits are where participants haven't shown any attitude towards target social group.

Algorithm 6.3: Generation of Ground-truth

```
     Input: labelled location events dataset *LD* of user, tolerance time *tol*, time discretization
     factor *Td*
     Output: Ground-truth GT of a user

1:   begin
2:   filteredLD = []
3:   for event in LD
4:       if event['leave time'] – event['arrive time'] > tol then
5:           filteredLD ← event
6:   end
7:   round boundaries of events in filteredLD according to Td
8:   GT ← discretize events in filteredLD according to Td
9:   extract totalhits from GT for each location,time,weekend
10:  extract positivehits and negativehits from GT for each target,location,time,weekend
11:  GT ← totalhits,positivehits,negativehits
12:  return GT
13:  end
```

6.3 Constructing Mobility Markov Chain

Mobility Markov chain (MC) is required for our routine aware control mechanism to estimate entropy of recent locations.

After pre-processing location dataset as explained in previous section, we construct a weighted graph G(V,E) from given partition of the dataset *TS* (train set) of a user. Vertices of the graph represents area IDs, edges represent transition between those areas and weight of an edge is number of transitions occurred between vertices throughout span of *TS*. However, we have excluded following transitions;

- **Missing state**. If the interarrival time between sequential location events is higher than given tolerance time difference *t*, then the transition is ignored. However, because entropy estimation requires irreducible Markov chain (i.e. MC without partitions) and we required high order Markov chain; degraded transition is added to the weight of the edge in order to prevent partitioning, and the end effect on overall weighted graph is insignificant. For most of the users, the distribution of interarrival time between location events is approximately 1 hour as can be seen in Figure 6.5.

- **Loops**. Self-transitions of areas are ignored; additionally, because of pre-process phase of location dataset, sequential occurrences of same locations are rare. The evaluation using non-processed location dataset shown that using self-transitions increases conditional entropy of trajectory approximately 0.1%, hence, the effect is minimal. On the other hand, we aim to estimate trajectory entropy without including the influence of residence time of locations.

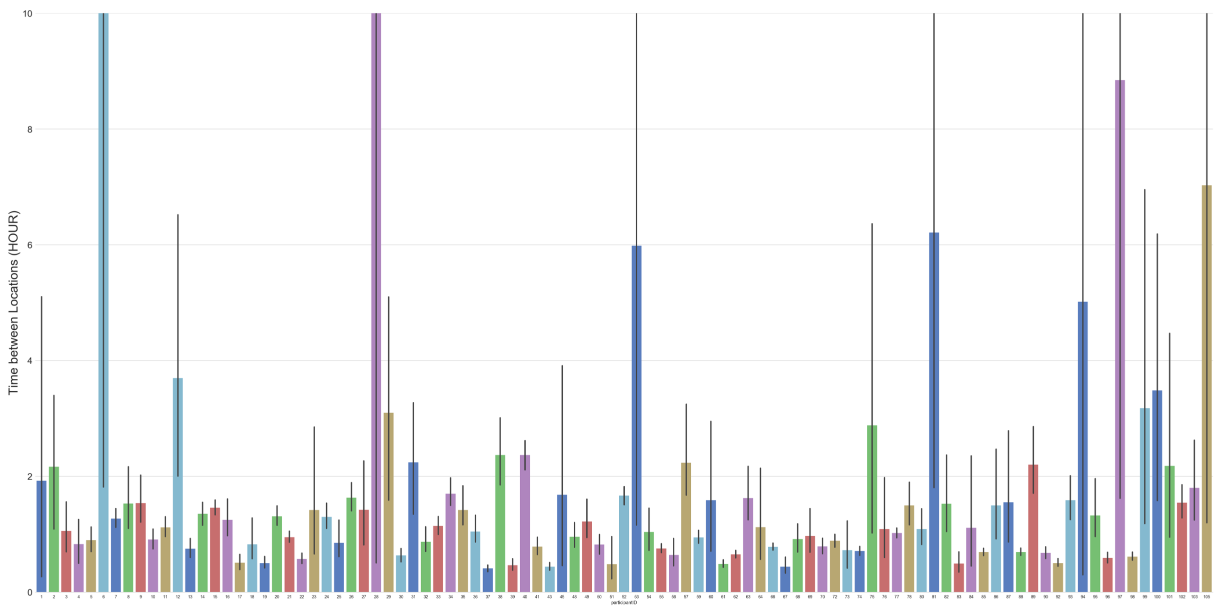

Figure 6.5 - Distribution of inter-arrival time (IAT) of location events per participant

Later from the weighted graph, transition matrix and indexes of areas are obtained. Laplace smoothing (additive smoothing) is not applied beforehand on weighted graph since it consumes valuable trajectory information and is observed as it greatly affects entropy calculation of trajectories. Rather than low-order MC (that captures all user's mobility pattern), we have obtained high-order MC that explicit to a user, because granularity level of dataset was low and we required a mobility pattern that explicit to a user.

Obtained sample MC for participant 7 using whole dataset is shown in Figure 6.6 as example. States represent area IDs and size of states indicates number of incoming and outgoing edges. Edges represent transition between states and value of edges is equal to the transition probability between two states.

Algorithm 6.4: Constructing MC

Input: labelled location events dataset *LD* of user, tolerance time *tol*, degraded transition score *scr*
Output: transition probability matrix P, area indexes *IND*

```
1:  begin
2:  initialize prevarea, prevtime, G(V,E)
3:  for event in LD
4:          area = event['areaID']
5:          time = event['leave time']
6:          if prevarea != area then
7:                  if time – prevtime < tol then
8:                          G(V,E) ← E(area, prevarea) += 1
9:                  else
10:                         G(V,E) ← E(area, prevarea) += scr
11:         prevarea = area
12:         prevtime = time
13:         end
14: end
15: P, IND = maximum_likelihood_estimation(G(V,E))
```

```
16:  return P, IND
17:  end
```

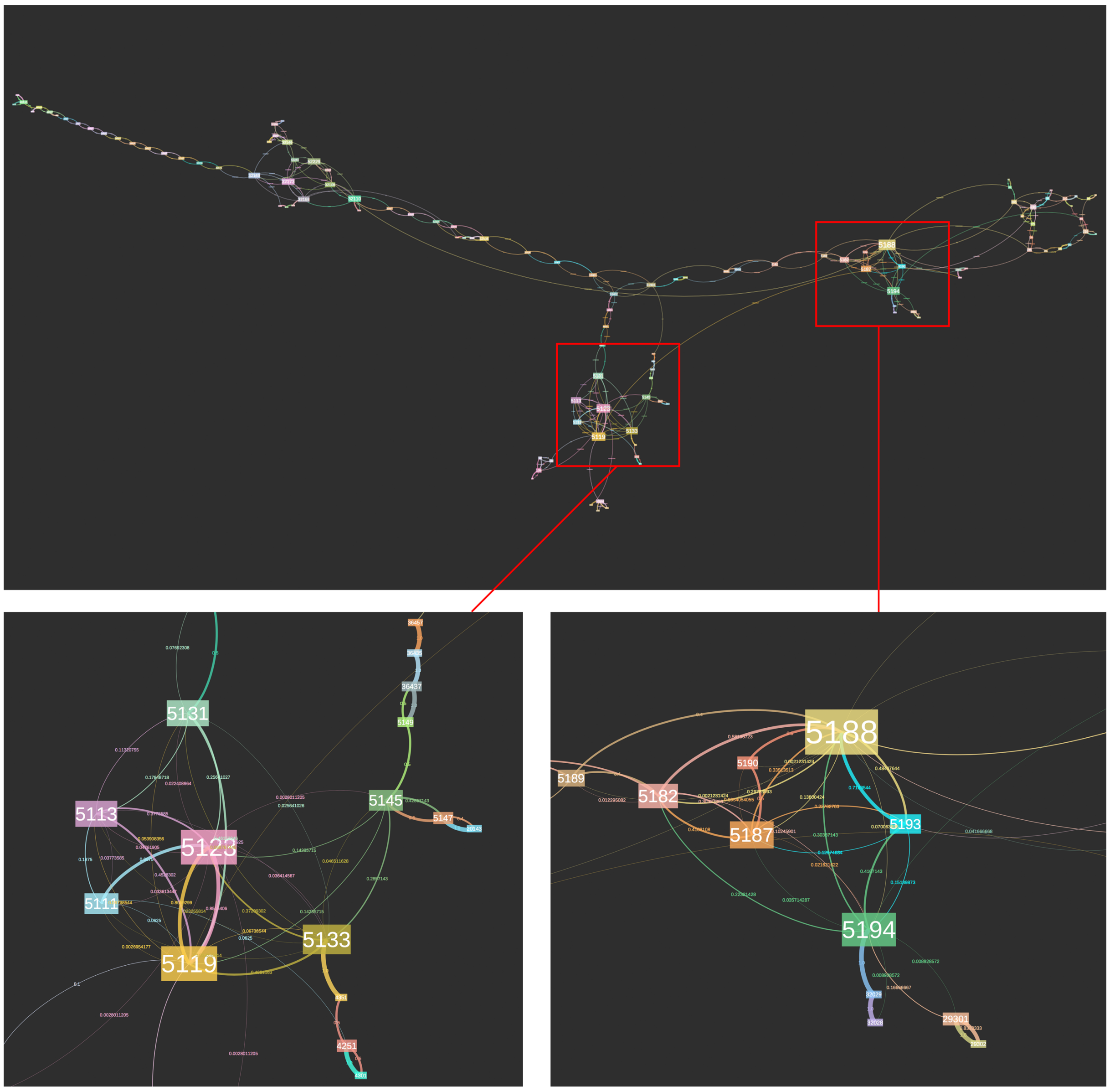

Figure 6.6 - Markov Chain for participant 7. Vertexes represent areas, edges show direct transition between areas and edge weights are equal to direct transition probabilities

7. Implementation and Evaluation

In this chapter, we first present the simulation setup and implementation environment. Then, types and results of evaluations are discussed.

7.1 Implementation

All implementations, evaluation environment and evaluations are done using Python 3 and Jupyter Notebook [22]. The evaluation environment seen in Figure 7.1 is generated to establish several tests. The diagram shows the steps applied onto dataset according to the previously explained pre-processing and processing steps. With the introduced evaluation environment, we can evaluate the cases with different variables.

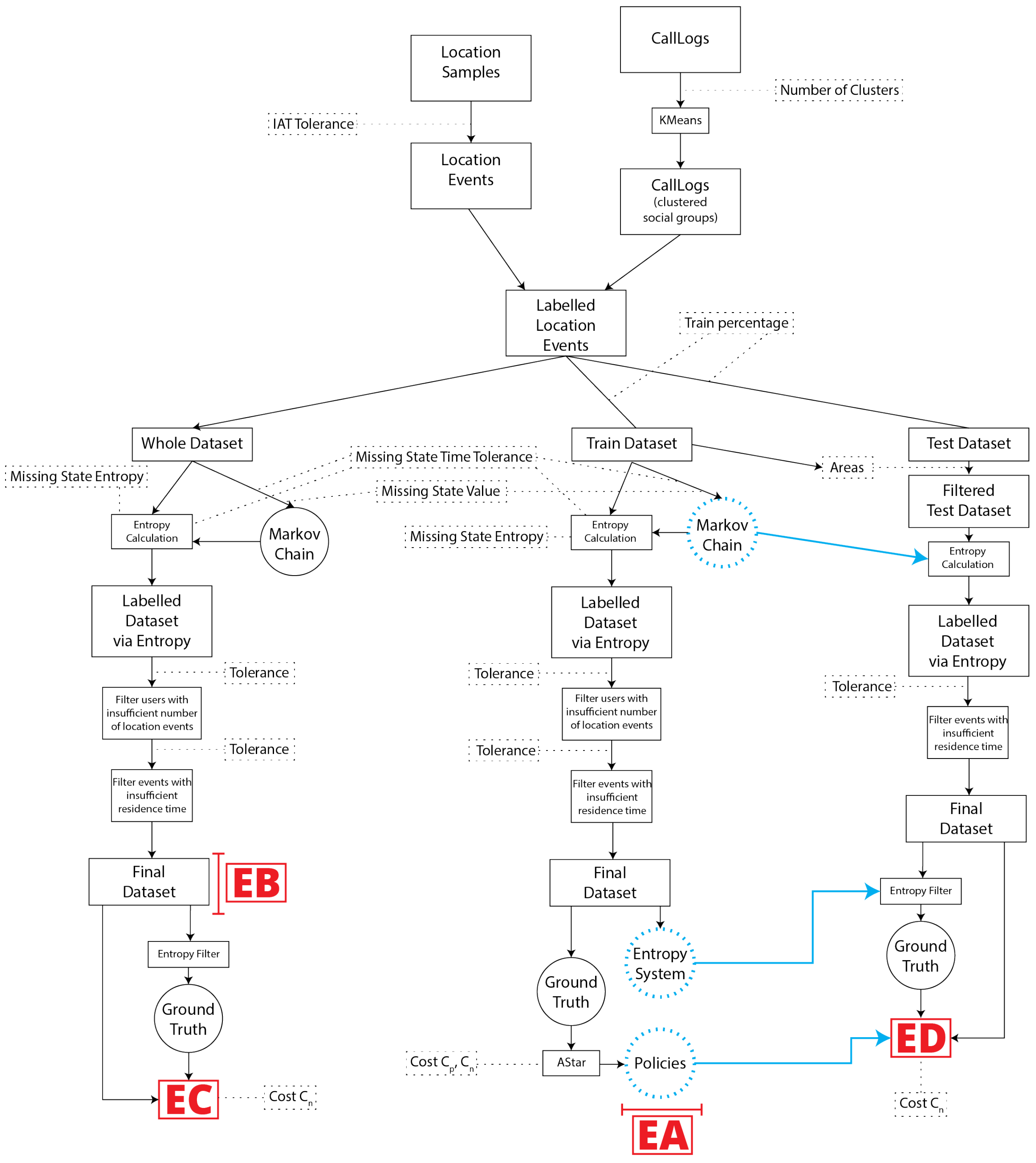

Figure 7.1 - Implementation of evaluation environment

7.2 Evaluations

This chapter divided into several sections for different evaluations. We will discuss the aim of each evaluation and the results received. The test cases, and at which point they have been applied are shown in Figure 7.1. For the Evaluation B and C, whole dataset is used in order to show entropy distribution of users and influence of possible entropy system designs on detecting non-sharing behaviour. In Evaluation A, we have utilized part of the dataset in order to identify optimal policy for subjects, and same part is also used in order to train different entropy systems. Later in Evaluation D, the policies and entropy systems are applied together on the remaining part of the dataset.

However, before test cases are applied, analysis of positive and negative sharing behaviour of subjects are done in Figure 7.2 considering calls and declined calls as positive sharing behaviour and missed calls as negative sharing behaviour. Unfortunately, the dataset was not sufficient to test accurately and fairly our system model. Maximum count of negative sharing behaviour was 34, and this would give invariably affirmative results in every test case we applied.

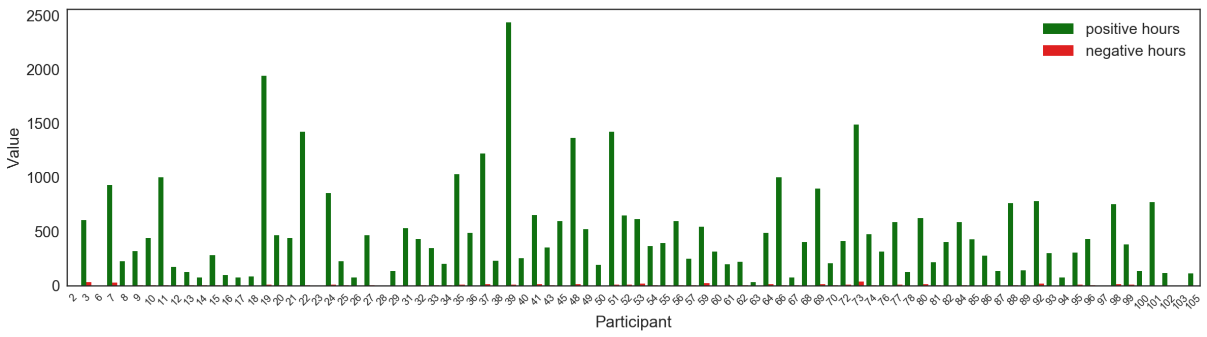

Figure 7.2 - Total shared and non-shared hours per subject in the whole dataset

In order to address this, we additionally consider an event where subject has not shown any communication behaviour (i.e. neutral hit) as negative sharing behaviour towards all social groups. For this setting, further filtering based on residence time for evaluations is introduced to not include events where user had not sufficient time to expose sharing behaviour. Although it is highly biased approach, it can allow us to estimate influence of routine aware decision-making system. After filtering events with narrow residence time (less than 2 minutes), new occurrences of positive and negative sharing behaviours are shown in Figure 7.3.

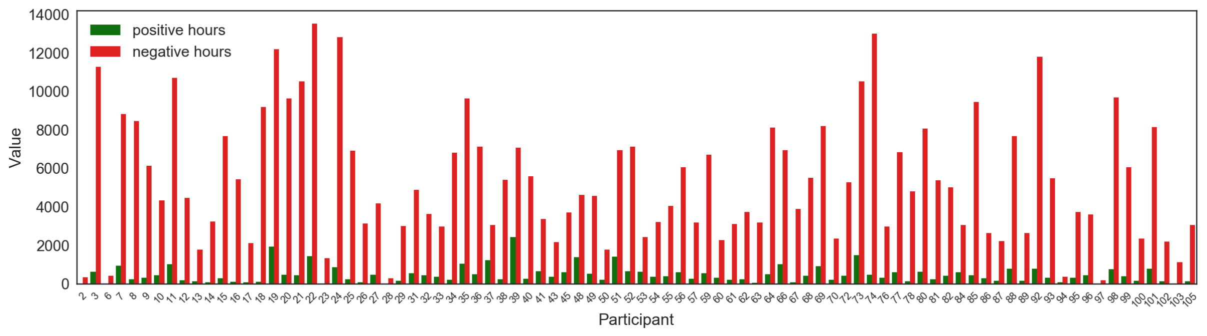

Figure 7.3 - Total shared and non-shared hours per subject in the whole dataset with biased approach

Despite occurrences of negative hours are relatively higher than the positive, we can still identify optimal policy and test it together with the entropy system. Therefore, throughout the evaluation section, we consider answered incoming calls, outgoing calls and declined calls as positive sharing behaviour towards social groups as mentioned in Section 6.1.3.3. If there was no communication event during subject's stay at location towards a social group, or if he intentionally missed a call from a social group (i.e. he did not respond or call back later during his stay); we consider these actions as negative behaviour towards social groups.

Occasionally, one drawback of biased approach is that maximum achievable accuracies for participants are considerably low when there is overlapping between positive and negative samples. To illustrate this, maximum achievable accuracies with most accurate greedy policies those covers every sample with positive accuracy in ground-truth space, with loc/time+ privacy type and mistakenly revealing cost of one are shown in Figure 7.4 as example using Formula 3.3 given in Chapter 3. The ratios are generated using whole dataset without including out-of-routine events. For the participants which are not shown, it is not possible to construct a sharing policy because their rate of withholding information for every location and time is more than their sharing. Simply using blacklists would give higher accuracies than positive sharing rules for most of the participants with biased approach.

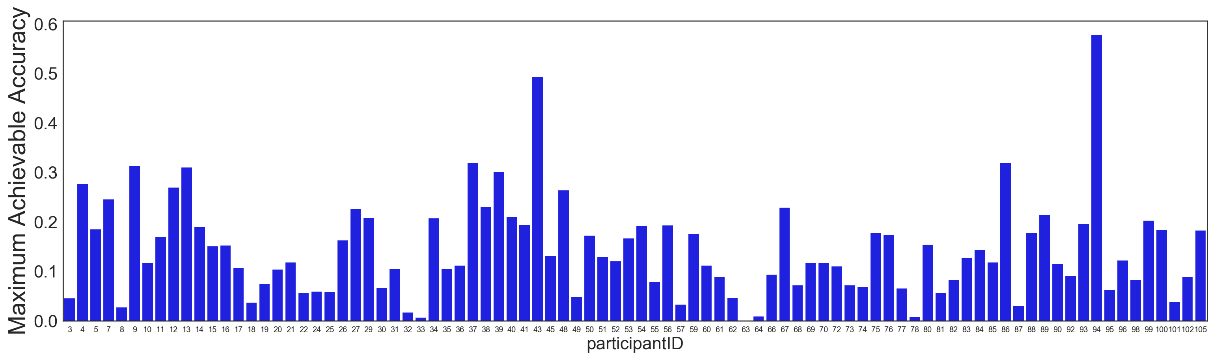

Figure 7.4 - Maximum achievable accuracies (with c_n = 1) per participant using greedy policy, with biased approach

Furthermore, following attributes and values are used in our implementation and evaluation;

- **IAT Tolerance** is needed to define missing samples and form more accurate location events with residence time, as previously discussed in Section 6.1.2. As can be seen in Figure 6.1, average inter-arrival time of samples are approximately thirty minutes for most of the users. Hence, we have defined thirty minutes tolerance, so that, if there was no sample for more than thirty minutes during participant's stay at a location, we close previous event and start new event.

- **Number of Clusters** is defined as four as previously explained in Section 6.1.3.1.

- **Missing State Tolerance** is minimum required time between location events in order to add degraded transition to an edge of weighted graph as discussed in Section 6.2.2. We have used 1 hour to indicate if there was possibly a missing state between two states. On the other hand, the same value is also used to label location updates with entropy as described in Section 5.2. If time difference between two location updates is higher than given tolerance value, then recent location overwrite stored trajectory information and entropy of recent location is recorded as **Missing State Entropy**, which can be defined at below zero to identify these events later (entropy value should be not below zero).

- **Missing State Value** is degraded transition value added to the edge weight in case of missing states, aims to prevent partitions in transition probability matrix.

- **Tolerance for insufficient residence time** is used to filter location events those have zero or narrow residence time where participant would not be able to share his location, as discussed in Section 6.2.1. The important aspect is, filtering applied after all location updates are used to generate markov chain and label updates with entropy values. In other words, whole location updates are used as trajectory data but is not considered as shareable locations, hence they are not used to train user policy or evaluate accuracy of the policies. To justify this, system would be allowed to share participant recent location only after his stay time exceeds tolerance time. In evaluations, two minutes is used as tolerance.

- **Tolerance for insufficient number of location events** is used to filter participants who does not have sufficient variety and number of location events to present informative evaluations. Participants whose number of location events are less than 20, are filtered in all evaluations.

- Two different **Cost** values are used to grade samples in ground-truth in order to tinker balance between over-share rate and under-share rate of the generated rules and policies. One is the penalty of mistakenly revealing information, and the other is reward of correctly sharing information. Reward factor is only used in generating rules and identifying policy step, and not included in accuracy calculations.

- **Train percentage** is used to divide whole dataset into separate train and test sets. The important point is dataset is not shuffled beforehand, in order to correctly estimate entropy values and transitions. In other words, dataset is divided from its timespan into two sets such as past and future. In Evaluation A, we have used 70% of the dataset in order to identify optimal policy for the participants, and train entropy systems. Later, in Evaluation D, we have used remaining 30% of the dataset to test our system.

7.2.1 Evaluation A – Identifying Participants' Policy

In the first evaluation section, we will discuss and identify policies for participants utilizing the optimized algorithm as proposed in Chapter 5. As privacy setting type, Loc/Time+ is used with area, time, weekend and target social group dimension. The algorithm first approximates best clusters of sample points based on given score function. Later, it identifies optimal policy from generated rules. The score function used in both heuristic function is;

$$score(s) = c_p * \sum s_{positivehit} - c_n * \sum s_{negativehit} \tag{7.1}$$

as given in Chapter 5. Because maximum achievable accuracy for large-scale dataset is low and users' attitude change is eminent, we introduced a variable cost C_p in order to simulate varying realistic policies those users might have defined by themselves. With both cost factors, we can tinker end-policy to share more in exchange for over-share rate, or leak less in exchange for under-share rate. This grants us to identify more strict or loose policies on accuracy. In this evaluation, we have identified policies with four different setting types.

Before the evaluation, dataset has been divided into separate train and test sets. Train set consists of 70% of the whole dataset and the location updates are not shuffled beforehand, therefore, accurate entropy calculation for both sets can be applied. If a covered sample point by rule or policy, has been never occurred in train dataset, it is considered as void point and does not affect under-share, over-share or accuracy.

In our first A star algorithm run for cluster generation, we have used C_p:1 and C_n:5 as cost factors. The iteration phase for first and second shaping rules for participant 7, are given in Figure 7.5. Both rules are achieved their maximum score before heuristic function over-estimates, at sixth and seventh iteration correspondingly; where they are recorded to be later used in identification phase of the policy. Because the A star is strict on accuracy factor given cost setting, while over-share rate is remained still at lowest level, under-share rate decreases slightly until maximum score is achieved. Therefore, algorithm defines boundaries of rules cautiously, and number of covered samples at each iteration shows slight increment. In these setting, reaching maximum achievable accuracies given in Figure 7.4 is possible.

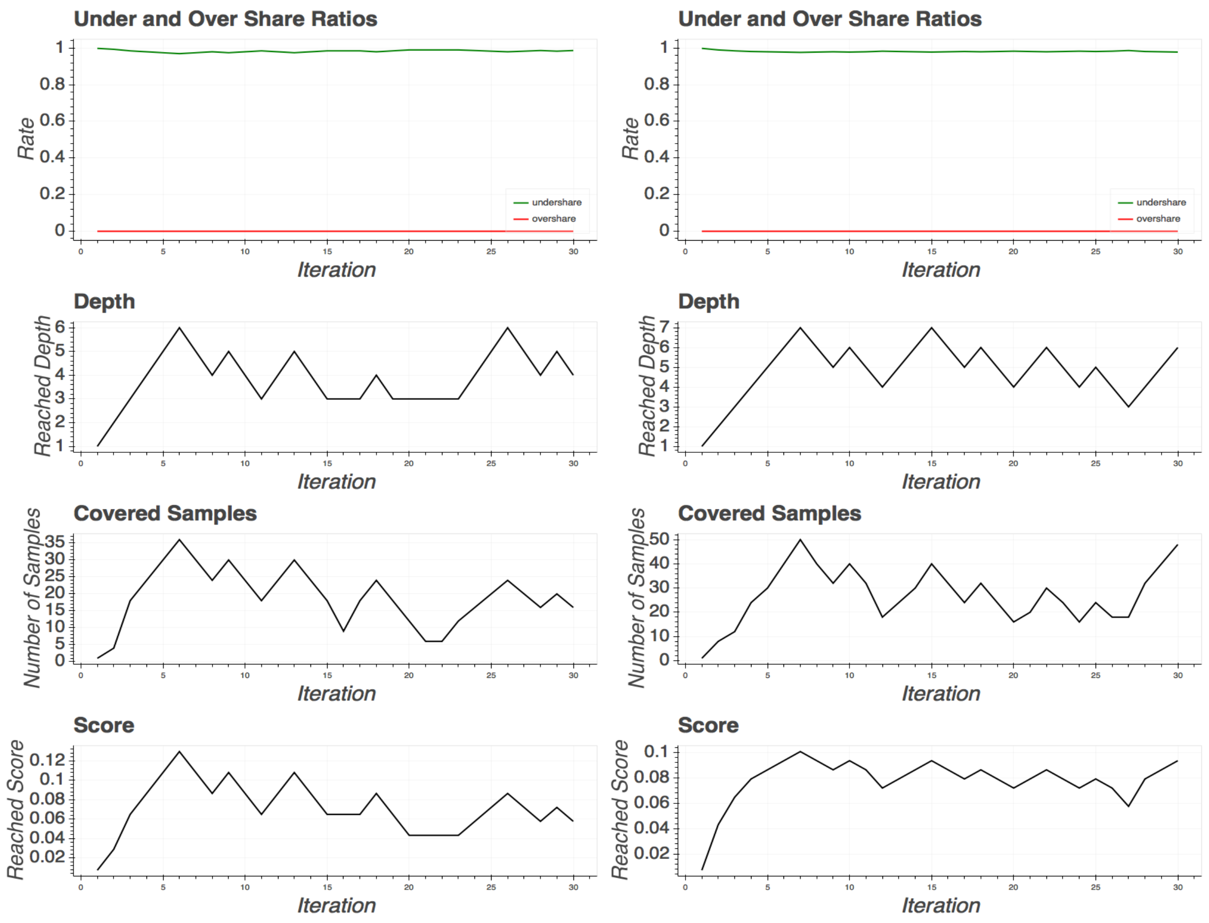

Figure 7.5 - From left to right: First rule generation, second rule generation for participant 7 with C_p = 1 and C_n = 5

In the second setting, C_p:1 and C_n:1 cost factors are used. Hence, algorithm grades each sample in ground-truth space equally. With relaxed constraint, first shaping rule in Figure 7.6 have achieved its maximum achievable score at higher iteration levels compared to previous setting. In exchange for over-share rate, rules are able to reach slightly better under-share rate. Over-share ratio for the shaping rules are observed to reach approximately 0.2, meaning the rule generated reveals location information mistakenly at 20% rate, and 80% of times it shares correctly. While over-share ratio is remained under 0.5 rate, accuracy (with C_n:1) caused by rule will always higher than zero, but lower than previous A star setting which tries to avoid over-share as much as possible.

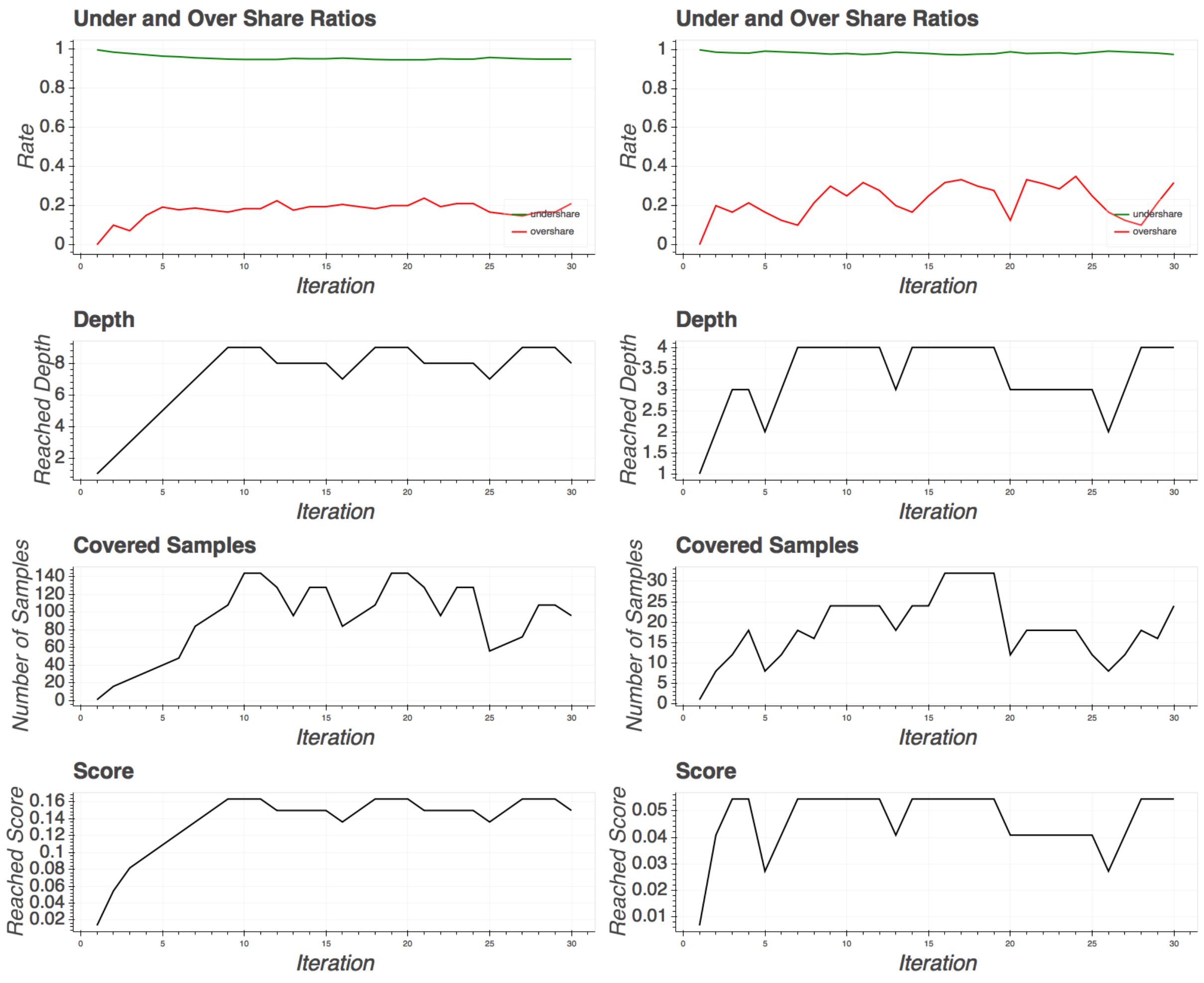

Figure 7.6 - From left to right: First rule generation, second rule generation for participant 7 with C_p = 1 and C_n = 1

On the next setting type, we have configured cost function of A star with C_p:5 and C_n:1. When higher C_p is used, A star bias shifted towards extracting samples if they have some amount of positive sharing behaviour. Algorithm tries to avoid missing samples which contains positive behaviour than omitting samples with negative behaviour. Since cost ratio is not equal, shaping rules causes low or negative accuracy.

In Figure 7.7, first two shaping rules of A star algorithm with given cost factors are shown. Approximately 80% of shared events with the rules are mistake. However, because rules are more audacious to share events, under-share rate chart presents sharper reductions for both generation phases. The other observed outcome is that, the first phase of rule generation is prone to covers all possible samples and reach greater scores, which leaves fewer remained samples for other rules. This outcome is not observed in our first setting with C_p:1 and C_n:5 because boundaries of every possible clusters in ground-truth space were more distinct. When more relaxed constraint is applied, new comprised clusters start to intersect with each other, that cause first generated rule to cover samples at maximum level.

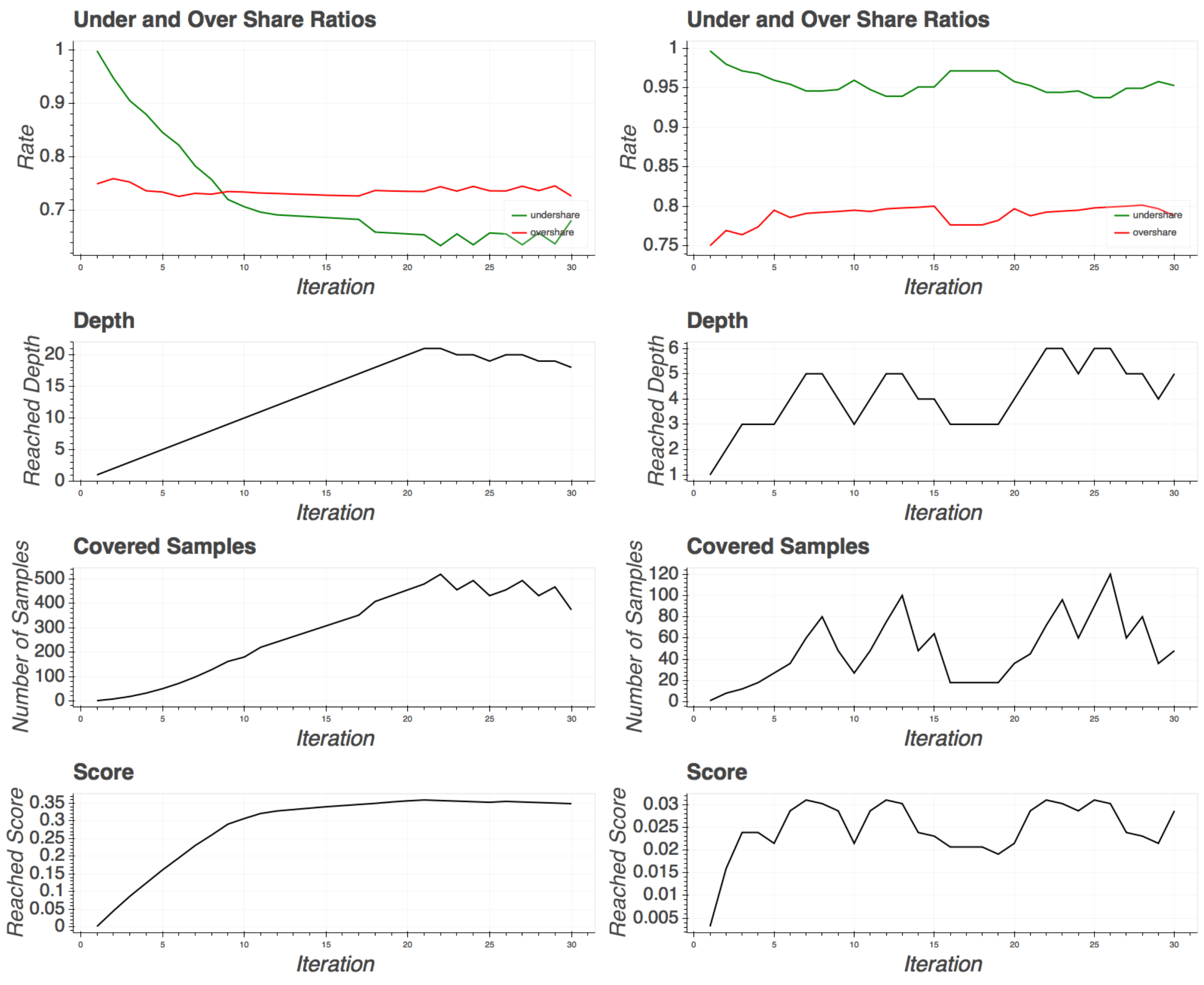

Figure 7.7 - From left to right: First rule generation, second rule generation for participant 7 with C_p = 5 and C_n = 1

In the last setting, C_p:10 and C_n:1 is used in Figure 7.8. At first rule generation, almost 60% of positive samples are covered by the shaping rule at twenty-fifth iteration and approximately 50% of total score points reached, which caused second shaping rule to reach only 4% of total score points. This means that the largest possible cluster is seized by the first rule. Further, the more relaxed constraint is applied, the more iteration it takes for a rule to reach its maximum achievable score.

Later, all generated rules for users per setting type, are fed into another A star algorithm to identify optimal settings with each setting type. As for an example, we have provided policy generation phase for participant 7 and 34 in Figure 7.9 and Figure 7.10. For the accuracy presentation chart, C_n:1 cost is used.

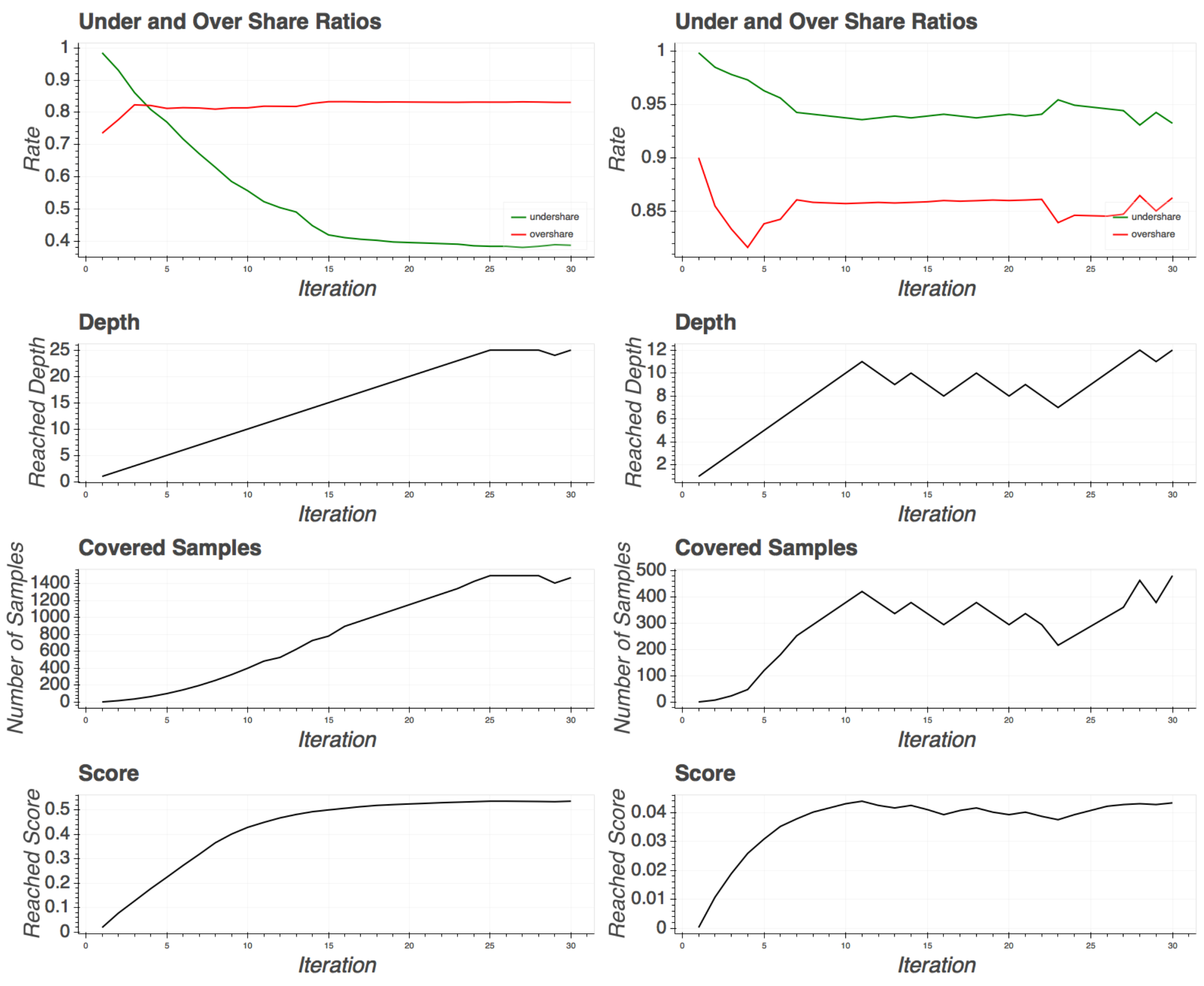

Figure 7.8 - From left to right: First rule generation, second rule generation for participant 7 with C_p = 10 and C_n = 1

As expected for both examples, reached accuracy drops while C_p is increasing and C_n is decreasing. The strict setting with C_n:5 causes most accurate policies possible, in contrast, under-share rate is increasing with accuracy. Because it is biased approach as discussed at the beginning of the chapter, they are anticipated results. One interesting outcome is, number of rules required by the policies to reach maximum achievable results differs from user to user. It shows us that the distinction of clusters' boundaries in ground-truth space differs greatly from subject to subject. In other words, while some subjects (e.g. participant 34) have more predictability towards sharing preferences (i.e. common patterns), others (e.g. participant 7) have more complex sharing behaviour patterns because they require more distinct rules to correctly define their sharing preferences and cover all samples.

All identified policies with different setting types are tested later on test set in Evaluation D.

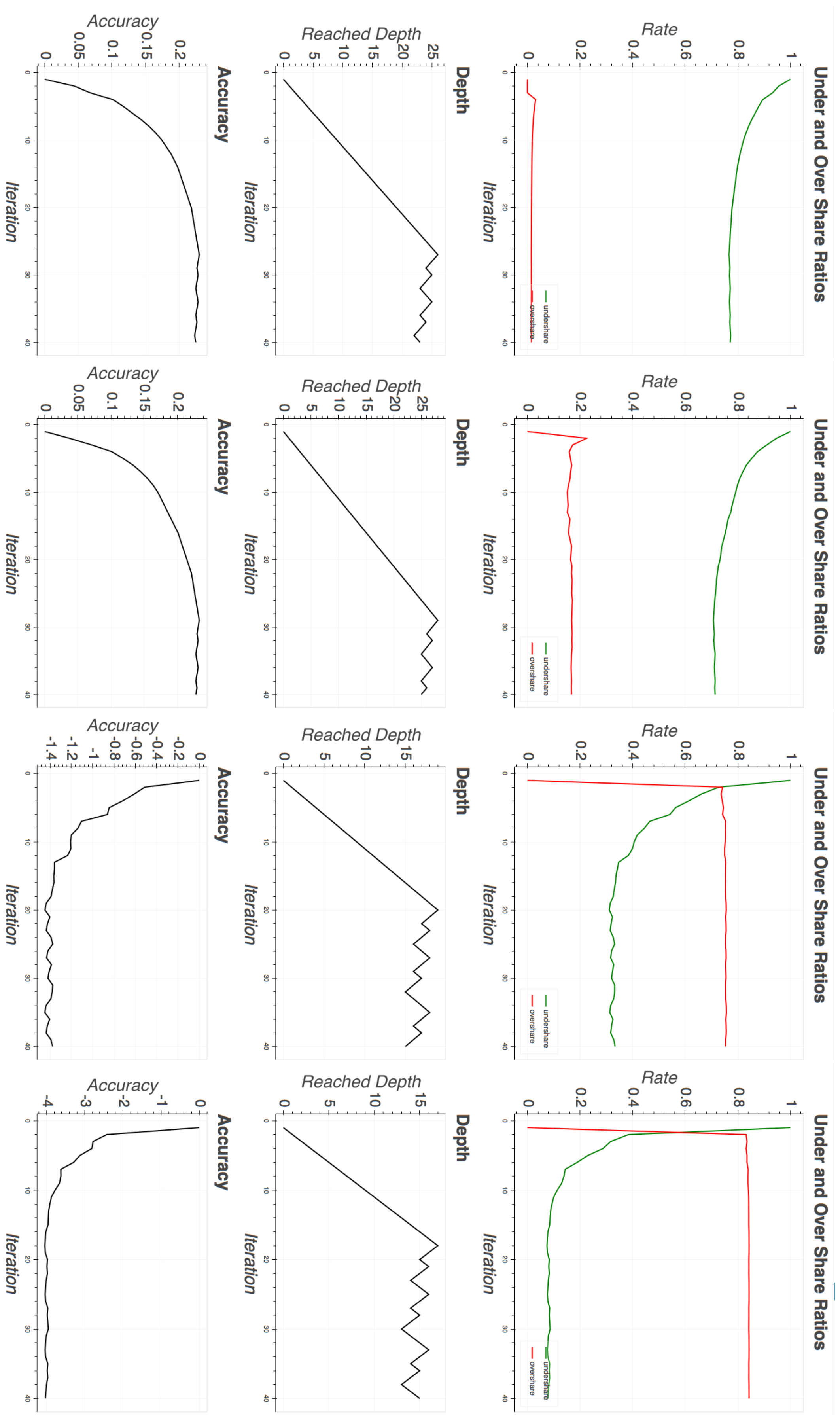

Figure 7.9 - Policy generation phase for participant 7. Clockwise: Policy with C_p:1-C_n:5, C_p:1-C_n:1, C_p:5-C_n:1 and C_p:10-C_n:1

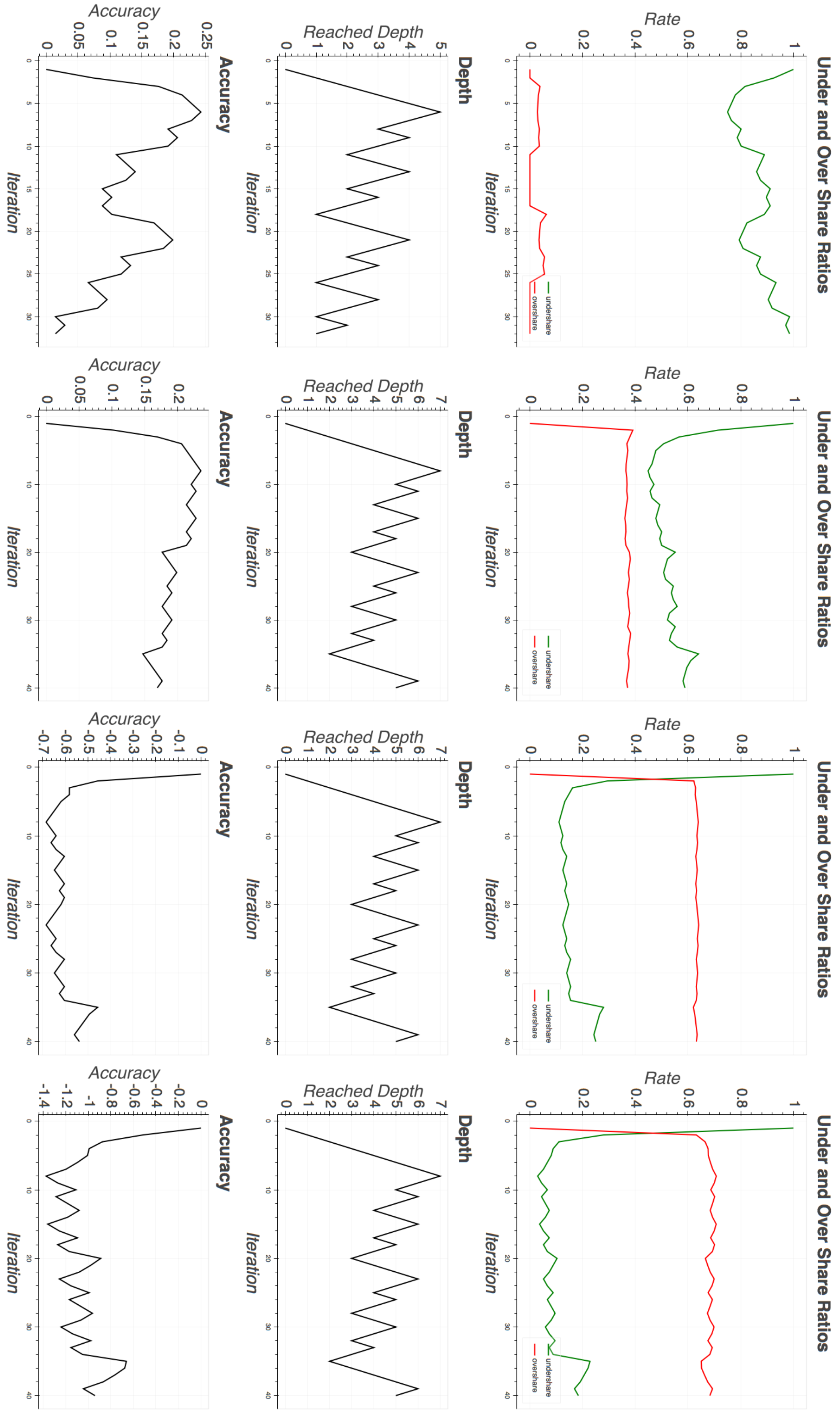

Figure 7.10 - Policy generation phase for participant 34. Clockwise: Policy with C_p:1-C_n:5, C_p:1-C_n:1, C_p:5-C_n:1 and C_p:10-C_n:1

7.2.2 Evaluation B – Correlation between Entropy and Sharing Behaviour

The evaluation is applied at lifespan EB at evaluation environment, using whole dataset which labelled with entropy values beforehand. Our aim at this point is to discover correlation between entropy value and participants' sharing behaviour. The location updates which are labelled with *missing state entropy*, are not included in the evaluation. In the figures, green circles (value=1) indicates that call event (except missed calls) occurred during participant's stay at location and at estimated entropy value, and red circles (value = 0) represents that missed calls or no call event occurred during the stay at estimated entropy; all without discretization of target dimension. Radius of the circles shows the total number of events occurred at same entropy value. The residence time filtering is applied before the evaluation; therefore, events where user had no sufficient time to expose accurate sharing behaviour, are filtered and not included in the tests.

The first chart with lowest degree of detail (for all population and all areas) for different entropy ranges is given in Figure 7.11. The first noticeable detail would be that most of the sharing and non-sharing behaviours are gathered at lowest entropy levels. The Reality Mining dataset includes students and faculty members from MIT, and the results show that students often expose routine events or rather use routine trajectories between home and school/work. Furthermore, another reason can be given as that even if a user would have visited a location once throughout 1 year period or more than once but using same trajectory, entropy of that location will appear as zero in the dataset because of only one known/appeared trajectory.

At the lowest degree of detail, we can think of defining routine aware system that will not allow sharing location updates if they are above 30,000 or 150,000 entropy value.

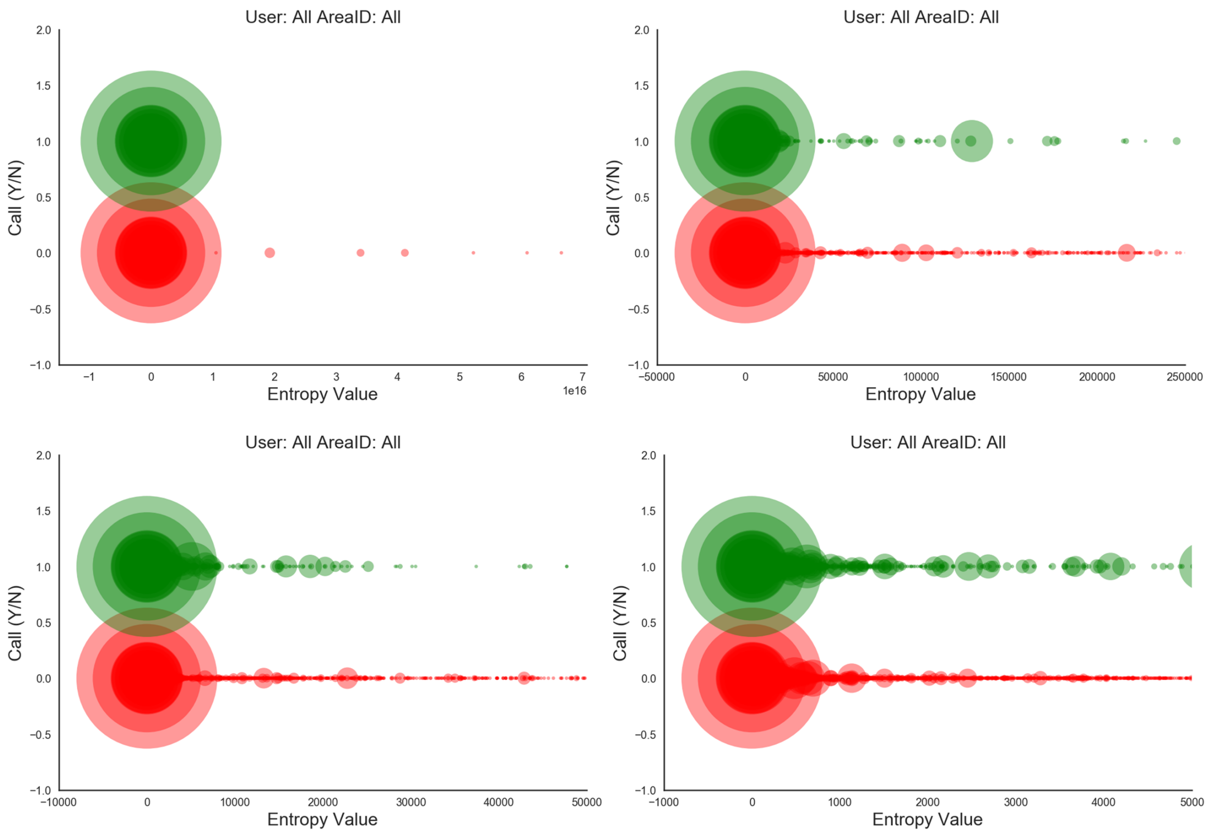

Figure 7.11 - Call indicator vs entropy value for all users and areas, for different entropy ranges. Range decreases from left to right and above to bottom. Radius of circle indicates total number of events at same entropy value

At one step higher degree (exclusively for users but includes all areas), maximum entropy values reached by users are varying as can be seen in Figure 7.12. Hence, previously proposed naïve entropy filtering approach would not work on all participants and each participant has different mobility pattern. However, most of the participants has similar pattern as the population that their majority of location events are gathered in lower entropy values. In other words, they are prone to sharing their locations at lower entropy values. At this degree, identifying entropy filtering levels exclusively for users would results in lower over-share rates than a system based on entropy of the population. Additionally, since maximum entropy value for a user varies, normalization and defining common 'the high value' is hardly applicable.

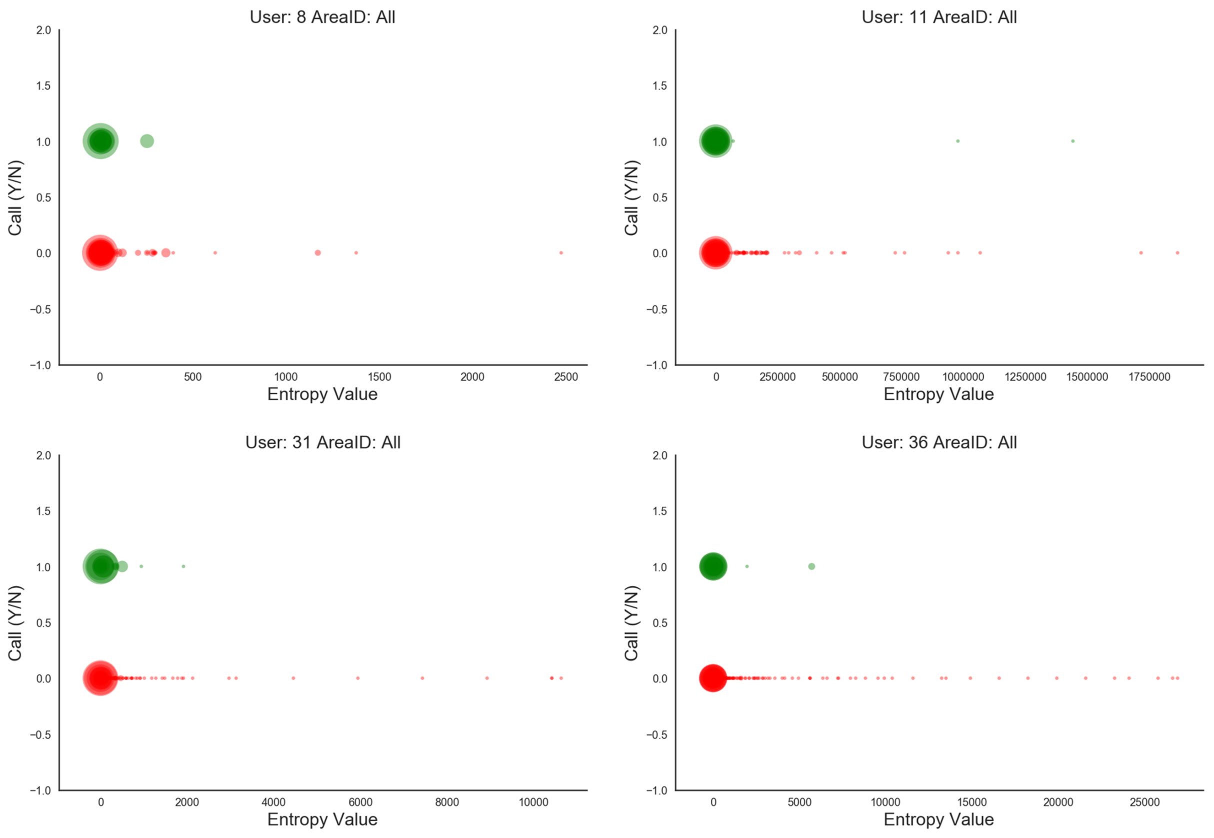

Figure 7.12 - Call indicator vs entropy value for randomly selected participants 8, 11, 31, 36 and all areas. Radius of circle indicates total number of events at same entropy value

Another chart is rendered for participant 8 in different entropy ranges in Figure 7.13. Despite its lower zoom levels where filtering higher entropy values seems reasonable; at higher zoom levels, it can be observable that not only filtering higher entropy values but also specific entropy ranges would give better results on accuracy.

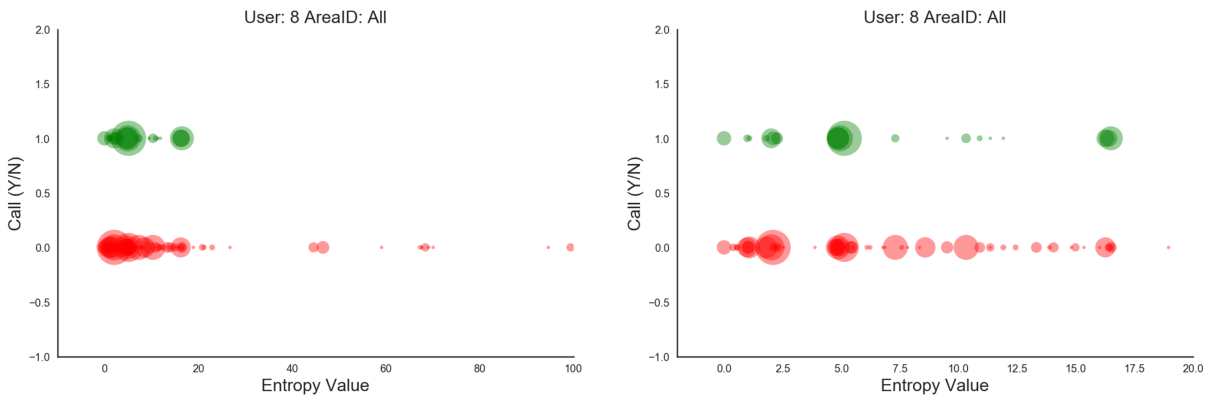

Figure 7.13 - Call indicator vs entropy value for participant 8 and all areas, for different entropy ranges. Range decreases from left to right. Radius of circle indicates total number of events at same entropy value

At higher degree of detail where entropy values and calling indicator are distributed on exclusively a user and an area; more diverse patterns are appeared.

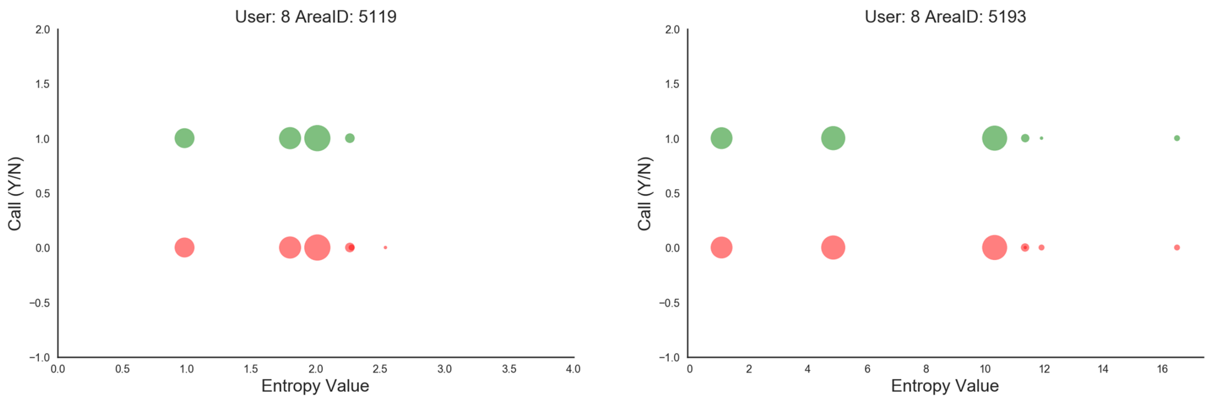

Figure 7.14 - Call indicator vs entropy value for participant 8 and area 5119-5193 pairs. Radius of circle indicates total number of events at same entropy value

In Figure 7.14, where chart is rendered from participant 8 and area 5119-5193 pairs, sharing and non-sharing behaviours are appeared almost symmetrical to each other. Filtering which is applied according to exclusively users and locations, is hardly applicable for those and similar pairs, and would not result in better accuracy.

However, not all areas from participant 8 have similar patterns. For example, sharing behaviours in area 5113 and 5187 are not distributed symmetrically as can be seen in Figure 7.15. While positive behaviour is gathered at narrow entropy ranges, negative behaviour spread broad entropy range. Corresponding entropy filtering may induce increase in accuracy and decrease in over-sharing rates. Another interesting aspect is that some areas including the ones given in the Figure 7.15, do not share expected pattern which is previously observed in population or user entropy charts. Positive sharing behaviours were occurred relatively high entropy values. It may show us that, subjects are more willing to share certain locations, when their visit to that location was not a routine event.

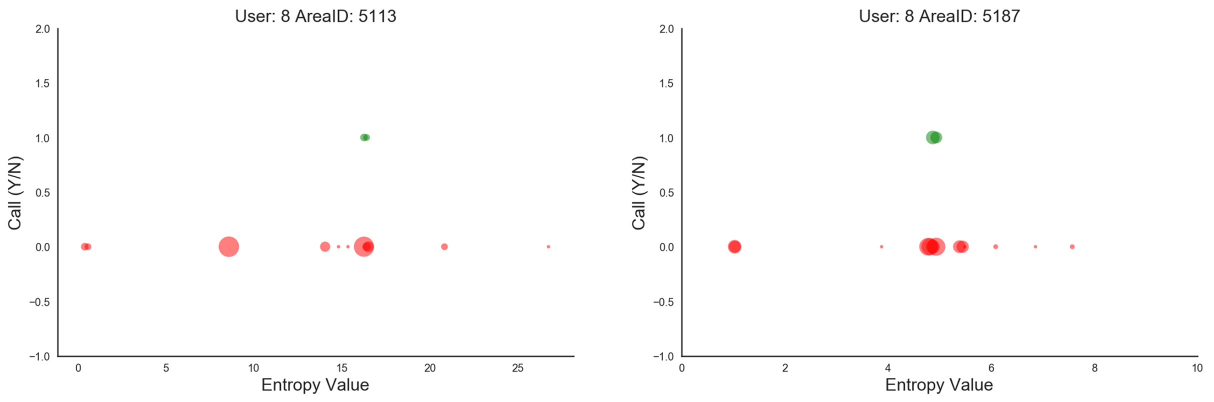

Figure 7.15 - Call indicator vs entropy value for participant 8 and area 5113-5187 pairs. Radius of circle indicates total number of events at same entropy value

On the other hand, it is not uncommon to detect participant-area pairs those share similar patterns as in population entropy chart. For example, participant 36, area 5113 and 5193 pairs are given in Figure 7.16, where majority of positive sharing behaviour are occurred at lower entropy values compared to negative behaviour.

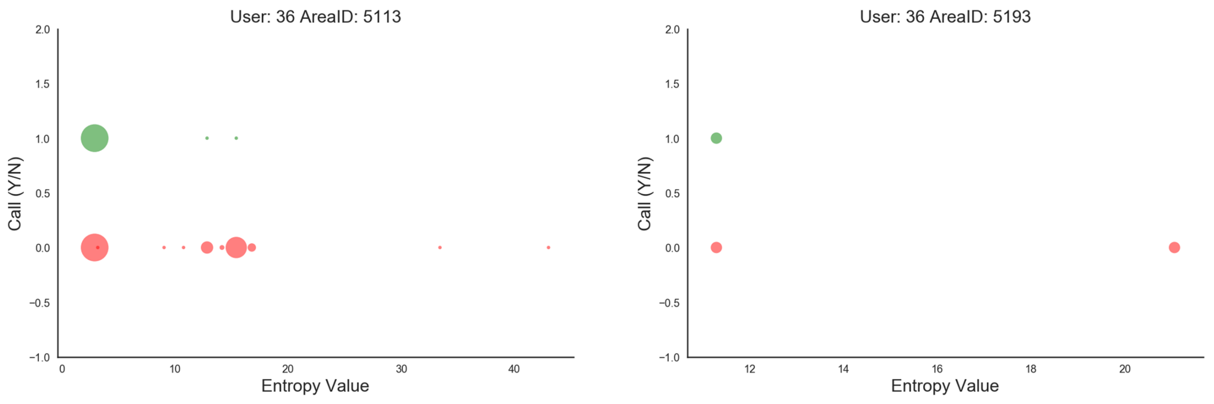

Figure 7.16 - Call indicator vs entropy value for participant 36 and area 5113-5193 pairs. Radius of circle indicates total number of events at same entropy value

After inspecting charts for participant-area pairs, we can indicate that entropy value ranges of sharing behaviour of a user vary with different areas which user often visits, and likely to vary with higher degree of level (i.e. with weekend indicator or time dimension). Users are not always prone to share their locations at lower entropy levels but at specific entropy ranges relevant to areas. Unfortunately, the dataset does not allow us to analyse entropy values at higher degrees of level because of insufficiency of data samples. However, current degree grants us an insight to develop routine aware control system with various probable settings which are tested in next evaluation.

7.2.3 Evaluation C – Impacts of Different Entropy Systems on Correct and Incorrect hours of a Policy

In this section, we have evaluated different entropy systems with naively generated policies, those cover all correct hours and cause 0% under-share rate. With the evaluation, we can simulate how policy would have revealed location information if a proposed entropy system type existed. As we mentioned earlier, because participants may intend to show great amount of sharing attitude changes through one year, the naively generated policy results in high overshare rates, and total negative hours caused by the policy is higher than total positive hours for majority of participants as well. In Figure 7.17, total positive and negative hours caused by the policy is given. Our aim is to decrease negative hours and protect positive hours as much as possible. However, the location updates whose entropy values are equal to *missing state entropy*, are not included neither at training nor filtering phases. Hence, it is not possible to achieve 100% drop rate on positive or negative hours if the participant had missing location update between events. Maximum achievable average drop rate of negative hours was thereby 48.58%, and average positive drop rate was 45.11%. This signifies that majority of location events are dropped in *missing state entropy* value. The events whose residence time were less than tolerance time are filtered out before the evaluation.

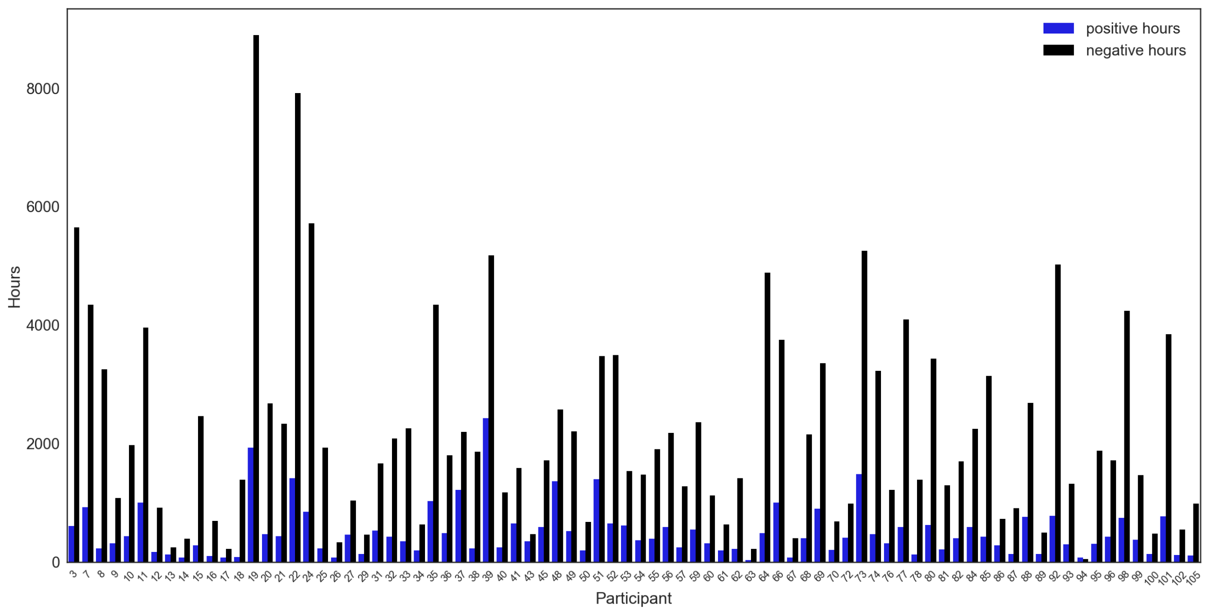

Figure 7.17 - Total positive and negative hours caused by the policy for each participant

The first two entropy system we have tested is based on population entropy values as mentioned in previous evaluation. The Figure 7.18 (ES1) shows the positive and negative hour change ratios if an entropy system which filters events above 30,000 entropy value for all subjects, would be applied. Even though it implies suitable based on Figure 7.11, this setting type affected only one user positively, others have lost more positive hours than the negative. The result implies that majority of negative sharing behaviour occurrences above 30,000 value belongs to one specific participant, and reached entropy values by other participants throughout one-year period were lower.

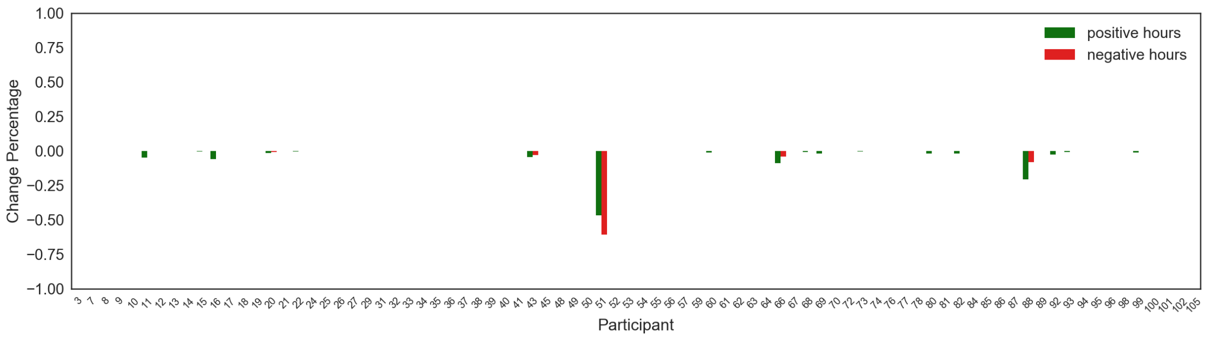

Figure 7.18 – Change rates of positive and negative hours on naive policy when events above 30,000 entropy value, are filtered (ES1)

Another entropy system type which is applied in Figure 7.19 (ES2), learns the population mean entropy value of positive sharing behaviour and filter location events above the mean. However, it has not resulted in better results for any subjects. The result implies that the difference between high and low entropy values is excessive and users are prone to share their locations at higher entropy values on occasions. Similar results can be observed in Figure 7.20 (ES3) where similar entropy system is applied exclusively for participants based on participants' mean entropy value of positive sharing behaviour. For majority of subjects, it caused more positive hours drop than the negative. While the average positive hours drop rate was 7.42%, average negative hours drop rate was equal to 4.32%. Hence, the presented systems so far were not reasonable.

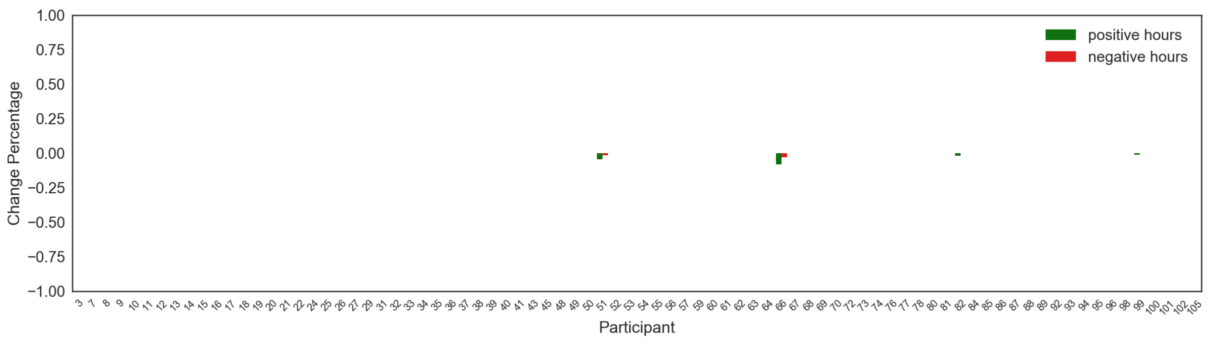

Figure 7.19 - Change rates of positive and negative hours on naive policy when events above mean entropy value of population, are filtered (ES2)

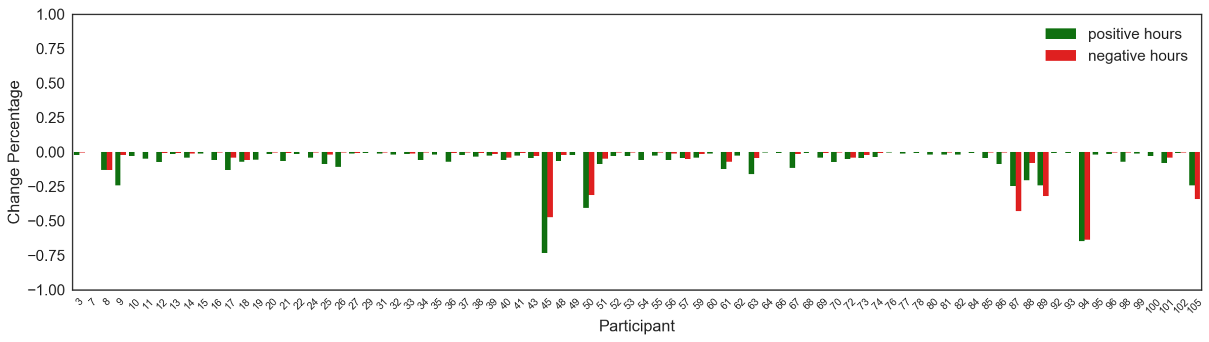

Figure 7.20 - Change rates of positive and negative hours on naive policy when events above mean entropy value exclusive to participants, are filtered (ES3)

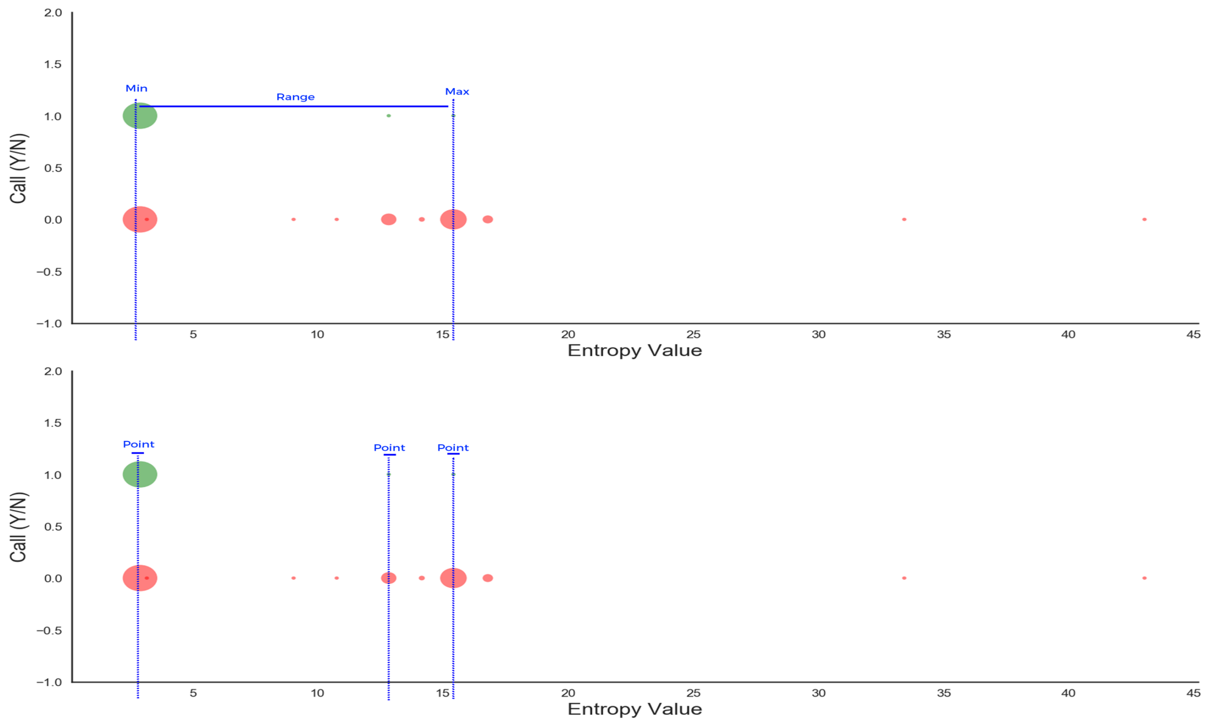

Figure 7.21 - Above; range based entropy system. Below; point based entropy system

Before introducing other possible entropy systems, we define two preliminary terms; range based entropy systems and point based entropy systems. In Figure 7.21, examples for both cases are given. When the system consists of entropy ranges, it will filter other events which are out of boundaries of the range. On the other hand, if entropy system consists of entropy points, events which do not have exact entropy value recorded on the system, are filtered.

Another user exclusive entropy system based on ranges, is applied in Figure 7.22 (ES4) where events between a user's maximum and minimum sharing entropy values are remained and others are filtered. As expected, drop rates are only occurred in negative events. Average drop rate was 0.15% and maximum occurred drop rate was 4.11%, which was positive but not sufficient.

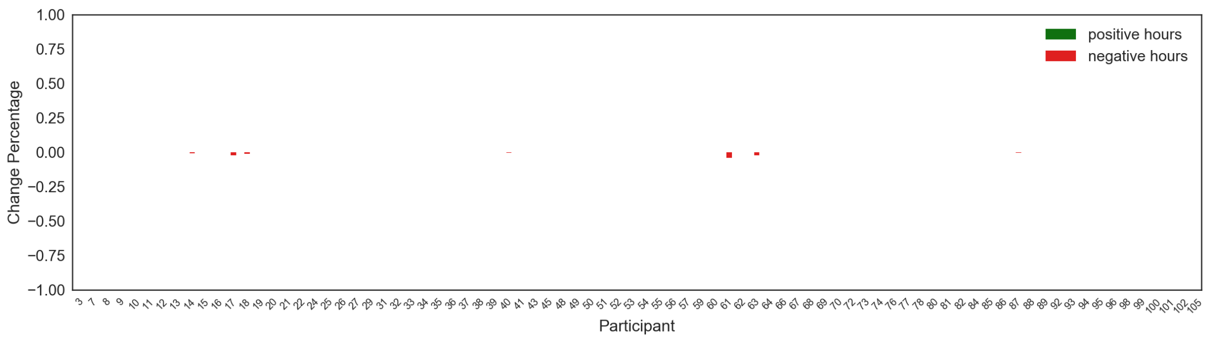

Figure 7.22 - Change rates of positive and negative hours on naive policy when events between maximum and minimum entropy value of positive sharing behaviour exclusive to participants, are remained (ES4)

As a next step, we applied similar min-max entropy system based on participant and area exclusively in Figure 7.23 (ES5). First, we have fetched minimum and maximum entropy values for each location of a participant as training, and then filtered events which are out of boundaries. The results were better compared to lower degree system in Figure 7.22, which implies that a subject may have different entropy sharing boundaries for different areas. Because the system covers all positive sharing behaviours, in the end we got 0% drop rates on positive hours and average 2.31% drop rates on negative hours. Maximum drop rate occurred was 19.82%.

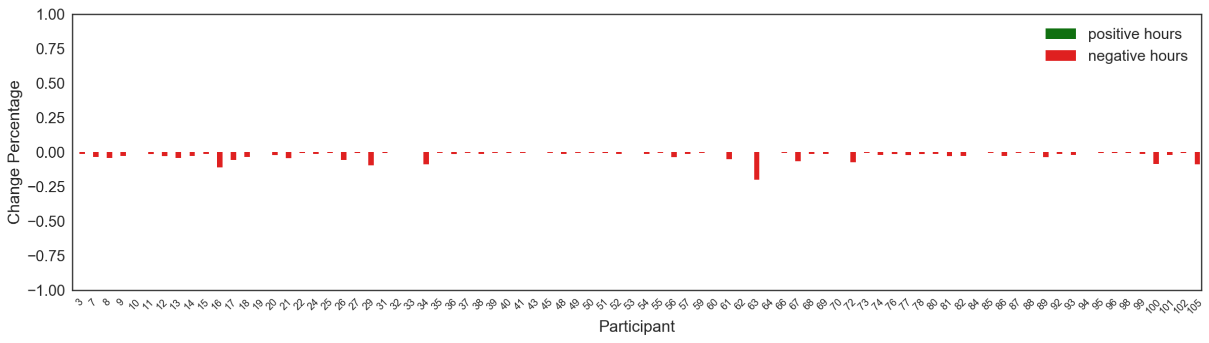

Figure 7.23 - Change rates of positive and negative hours on naive policy when events between maximum and minimum entropy value of positive sharing behaviour exclusive to participants and areas, are remained (ES5)

As we have seen in Evaluation B section, min-max entropy value range of sharing behaviour of some areas can be divided into more sections where more events with negative sharing behaviours can be filtered out. Hence, we have trained another entropy system in Figure 7.24 (ES6) based on points, that records every entropy value those resulted in positive sharing behaviour for each area and participant. The location updates are filtered out if their entropy value has never caused positive sharing behaviour given a participant and an area. We got 0% positive hour drop rate as expected, and average 3.62% drop rate on negative hours. Maximum achieved drop rate was 21.62%.

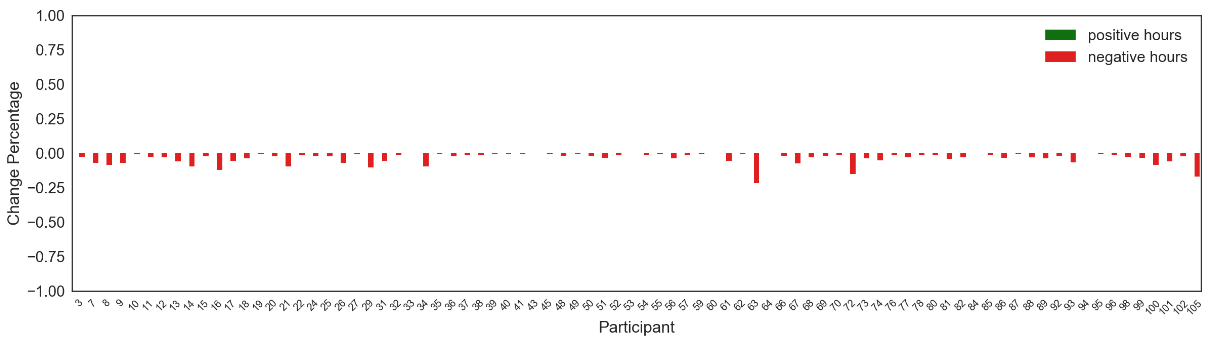

Figure 7.24 - Change rates of positive and negative hours on naive policy when events those were at entropy value of positive sharing behaviour exclusive to participants and areas, are remained (ES6)

In the last step, we have developed similar entropy system but promoted with the same score function used in training user's policy. The system (ES7) first discovers entropy values which results in positive sharing behaviour, however, it only records an entropy value if its score is higher than zero. With score function, we can tinker cost for positive (C_p) and negative (C_n) hit in order to approximate intended negative hours drop rate and to sacrifice positive hours.

$$score(e) = c_p * \sum positivehit - c_n * \sum negativehit \tag{7.2}$$

In table 7.1, the results of the entropy system with different cost factors are given. The most efficient range of C_p appears between 5 and 10, where difference between average positive and negative hours drop rate reached approximately 9%. However, because naïve policies have higher negative hours than the positive, first option with C_p:1 and C_n:1 factors would give better accuracy rates in exchange for under-share rate.

C_p	C_n	**Average Positive Hours Drop**	Maximum Positive Hours Drop	**Average Negative Hours Drop**	Maximum Negative Hours Drop
1	1	**41.27%**	94.24%	**47.23%**	89.83%
5	1	**18.53%**	86.83%	**27.62%**	86.83%
10	1	**8.12%**	65.84%	**16.00%**	74.33%
15	1	**3.83%**	63.85%	**9.92%**	74.33%
20	1	**2.70%**	63.85%	**8.00%**	74.33%

Table 7.1 - Results of entropy system (ES7) with given cost variables

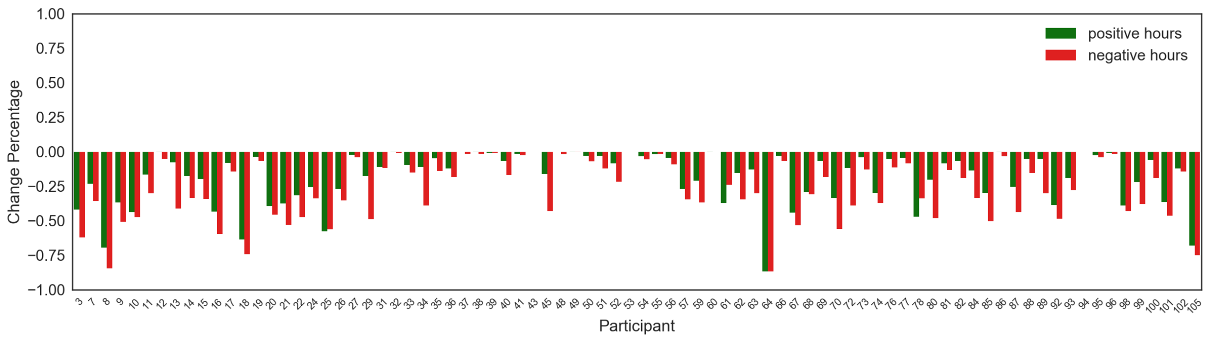

Figure 7.25 - Change rates of positive and negative hours on naive policy when entropy system with score function (ES7) given C_p = 5 and C_n = 1 is applied

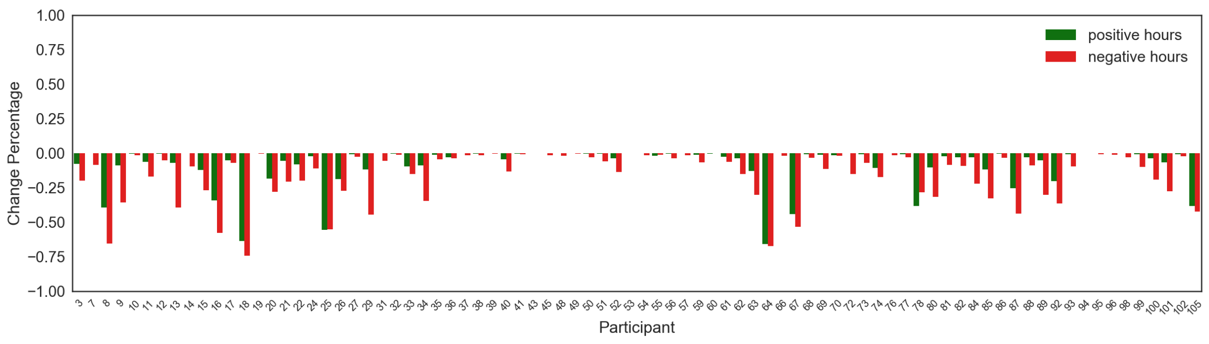

Figure 7.26 - Change rates of positive and negative hours on naive policy when entropy system with score function (ES7) given C_p = 10 and C_n = 1 is applied

Unfortunately, because dataset has several missing location updates, and is not sufficiently extensive; tests with higher degree of detail could not be applied (e.g. entropy systems exclusive for participant, area, and weekend or time).

7.2.3.1 Results of Entropy Systems without Missing State Entropy

In this section, we have applied all previously discussed entropy systems with naively generated user policies, without using *missing state tolerance* time. Therefore, each location update in the dataset has received entropy value. Even though the destination location were missing a path information, entropy calculation of the location would be still accurate if not precise. In Table 7.2 and Table 7.3, the results of previously mentioned entropy systems are given. This setting grants us the ability to apply filtering on all location events, independent from their inter-arrival times.

Entropy System	**Average Positive Hours Drop**	Maximum Positive Hours Drop	**Average Negative Hours Drop**	Maximum Negative Hours Drop
ES1	**3.47%**	83.90%	**1.79%**	84.00%
ES2	**0.37%**	18.31%	**0.16%**	8.72%
ES3	**13.04%**	97.74%	**8.50%**	92.64%
ES4	**0%**	0%	**0.19%**	6.30%
ES5	**0%**	0%	**1.97%**	10.79%
ES6	**0%**	0%	**4.41%**	23.90%

Table 7.2 - Results of entropy systems previously discussed, when applied on location events without missing state entropy

C_p	C_n	**Average Positive Hours Drop**	Maximum Positive Hours Drop	**Average Negative Hours Drop**	Maximum Negative Hours Drop
1	1	**77.68%**	98.50%	**83.53%**	99.59%
5	1	**35.97%**	98.50%	**47.24%**	96.76%
10	1	**12.74%**	87.11%	**23.32%**	96.76%
15	1	**7.04%**	83.13%	**15.93%**	96.76%
20	1	**4.42%**	83.13%	**11.70%**	96.76%

Table 7.3 - Results of entropy system (ES7) with score function previously discussed, when applied on location events without missing state entropy

7.2.4 Evaluation D – Results of Proposed Decision-Making System

In this last evaluation section, identified policies and entropy systems from the train set, are applied on the test set, first alone and then together. Throughout this section, penalty of accidently revealing information cost in accuracy function is determined as one. We define abbreviations for previously discussed policies and entropy systems to be used in this section;

- **Policy A**: Set of policies identified with C_p:1-C_n:5 score setting type.
- **Policy B**: Set of policies identified with C_p:1-C_n:1 score setting type.
- **Policy C**: Set of policies identified with C_p:5-C_n:1 score setting type.
- **Policy D**: Set of policies identified with C_p:10-C_n:1 score setting type.
- **ES5**: Entropy system based on range that is trained according to the minimum and maximum entropy values of shared events, explicitly for a user and an area.
- **ES7-11**: Entropy system based on points that is trained according to exact entropy values of shared events utilizing score function with C_p:1 and C_n:1, explicitly for a user and an area.
- **ES7-51**: Entropy system based on points that is trained according to exact entropy values of shared events utilizing score function with C_p:5 and C_n:1, explicitly for a user and an area.
- **ES7-101**: Entropy system based on points that is trained according to exact entropy values of shared events utilizing score function with C_p:10 and C_n:1, explicitly for a user and an area.

One detail required to be highlighted is, the locations visited only in test set of subjects are not reckoned in evaluations and entropy calculations.

7.2.4.1 Testing Identified User Policies

First, identified policies with different types are applied on the test set without using entropy system. The results are given in separate charts whose y-axis is limited between 1 and -2 for better comparison, in Figure 7.27. The chart axis is limited because accuracy values may drop notably low values if the user withheld information at any time.

According to the charts, Policy B leads most accurate results among other policies, and Policy D is the policy with least under-share but highest over-share rates. Though it is expected from Policy C and D to deliver lower accuracy values than zero considering their sharing nature (i.e. share as much as possible), it is unexpected for Policy A and B to yield lower accuracy values than zero. The policies those have received zero under-share and accuracy values, imply that they couldn't manage to hit any samples in ground-truth space, and this situation occurred in Policy A the most. In other words, subject never visits some of his popular locations at his usual time in the test set, but he has exposed utterly opposite behaviour in train set.

The policies in Policy A were identified strictly on the samples which hold highest scores after high penalty of mistakenly sharing is applied. They have reached their maximum achievable accuracies in the train set, yet, they are not able to predict subject's sharing attitude in test set. Four of the possible sources of this situation are given below.

- Call events have more unpredictable pattern than actual location sharing behaviour, which is more difficult to predict precisely with static mechanisms.
- Because the dataset is highly (negatively) biased as previously discussed, it influences the evaluation.
- Considering the train and test sets are separated without shuffling the dataset beforehand, subjects might have relocated or had semester holidays (majority of the subjects are students).
- Subjects might have been prone to drastic sharing attitude changes.

Bearing all probable reasons in mind, next we discuss how different entropy systems affects the test set.

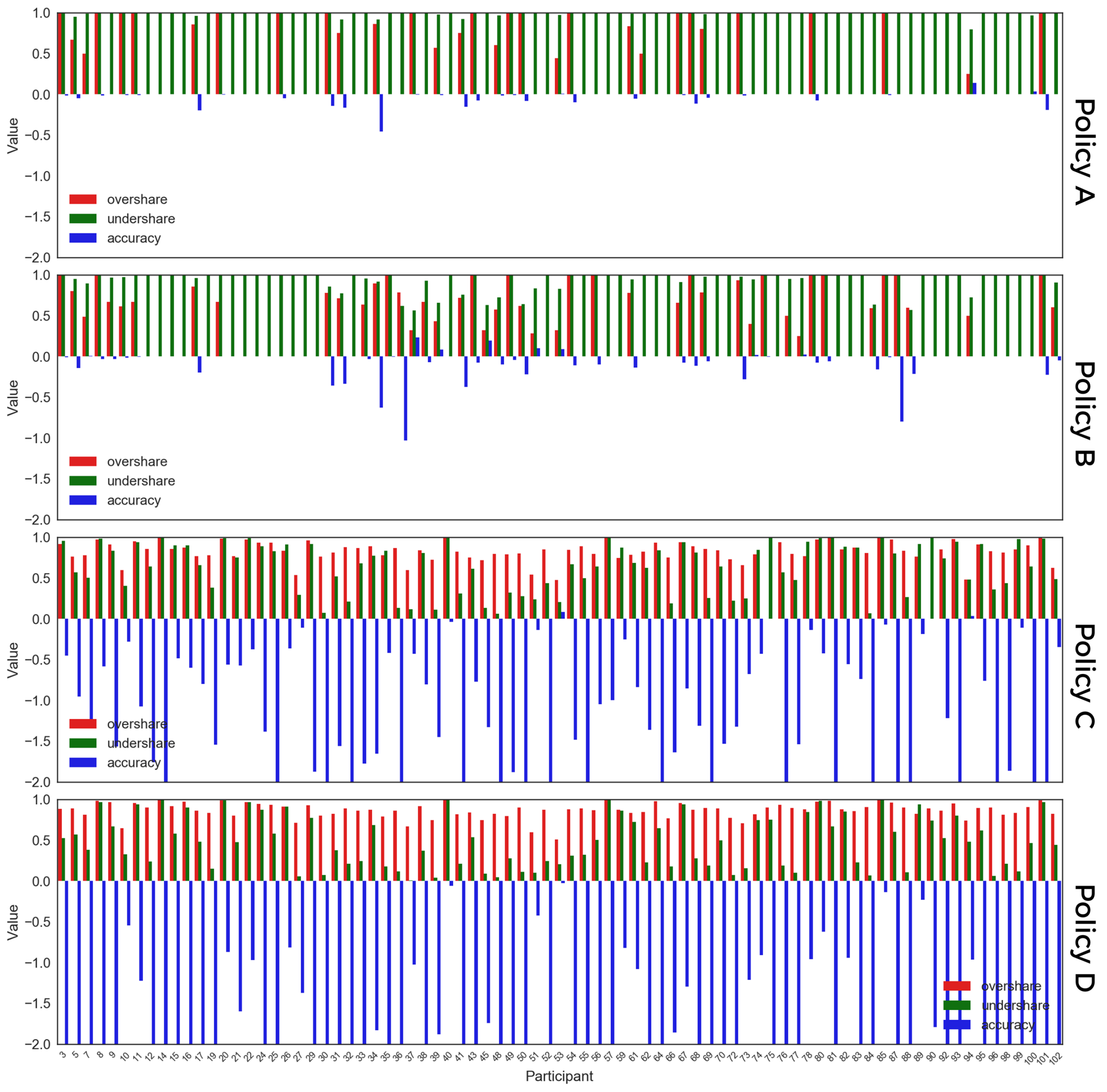

Figure 7.27 - Results of policies with different setting types applied on test set

7.2.4.2 Testing Trained Entropy Systems

In this section, the proposed entropy systems which are trained using train set, are applied on the test set and percentage of drop on overall positive and negative hours of subjects are recorded similar to Evaluation C. However, we define an additional term *success rate* (*SR)* used to show percentage of users benefitted from the system.

$$SR(ES) = \frac{P(ES)}{N(ES) + P(ES)} \tag{7.3}$$

where *P(ES)* is total number of participants who benefitted from the entropy system *ES* more (i.e. they received higher negative hour drop rate than the positive) and *N(ES)* is total number of participants who suffered from the entropy system more (i.e. they received higher positive hour drop rate than the negative)

Entropy System	**Average Positive Hours Drop**	Maximum Positive Hours Drop	**Average Negative Hours Drop**	Maximum Negative Hours Drop	**SR**
ES5	**13.85%**	81.81%	**19.08%**	87.50%	**61.53%**
ES7-11	**45.21%**	100%	**49.99%**	90.75%	**53.66%**
ES7-51	**28.86%**	100%	**35.96%**	89.52%	**61.25%**
ES7-101	**21.58%**	100%	**29.11%**	87.50%	**63.75%**

Table 7.4 - Results of different entropy systems applied on the test set, when missing state tolerance is considered

For the first scenario, we have labelled sequence of locations utilizing missing state entropy. Meaning that, if sequential location events have less time than pre-defined *missing state tolerance* time, then recent location receives *missing state entropy* and path information is cleared for further entropy calculations. The location events which received *missing state entropy* is not considered at training phase, and not filtered in any case at test phase.

For the evaluation, the entropy systems are applied on the test set alone and drop rates on positive and negative hours are recorded for each subject. The average drop ratios, maximum drop ratios and SR values caused by the entropy system for first scenario is given in Table 7.4. Success rate for each entropy system is above 50% and approximately at 60% level, which demonstrates that more than half of the subjects benefitted from the entropy systems. The average drop ratios show similarities the rates presented utilizing whole dataset in Evaluation C. Average and maximum drop ratios present increments compared to previous evaluation, average negative hours drop rates are still higher than the positive. However, in point based entropy systems, 100% maximum positive drops are observed in some subjects. The reasons may be similar to the ones discussed in evaluation of the policies section. If subjects rarely expose sharing behaviour in test set, or if they have changed their usual paths towards locations, those may cause high positive hour drop rates on point based system because they require precise entropy value to avoid withholding information.

Because newly introduced locations those are visited by subjects only in test set, are not considered in evaluation and entropy estimation, this situation may cause high amount of missing state occurrences between two sequential location events. Therefore, we additionally trained entropy systems which estimate trajectory entropies independent from inter-arrival times as discussed in

Section 7.2.3.1. This setting grants us the ability to apply filtering on all location events. For the second scenario, we applied those systems on the test set.

Entropy System	**Average Positive Hours Drop**	Maximum Positive Hours Drop	**Average Negative Hours Drop**	Maximum Negative Hours Drop	**SR**
ES5	**22.86%**	96.25%	**28.90%**	98.98%	**60.25%**
ES7-11	**76.56%**	100%	**82.34%**	99.87%	**32.50%**
ES7-51	**48.64%**	100%	**57.31%**	99.46%	**57.69%**
ES7-101	**35.97%**	100%	**45.30%**	99.46%	**59.49%**

Table 7.5 - Results of different entropy systems applied on the test set, when missing state tolerance is not considered

In Table 7.5, the results of entropy systems independent from inter-arrival times are given. The influence of limitless filtering capabilities granted to the system, is observed as rate increments on each drop rate. However, except for ES7-11, success rates are slightly decreased. Because nature of ES7-11 is similar to strictness of Policy B, it avoids revealing location information if its confidence for estimated entropy value was not sufficiently high during training phase.

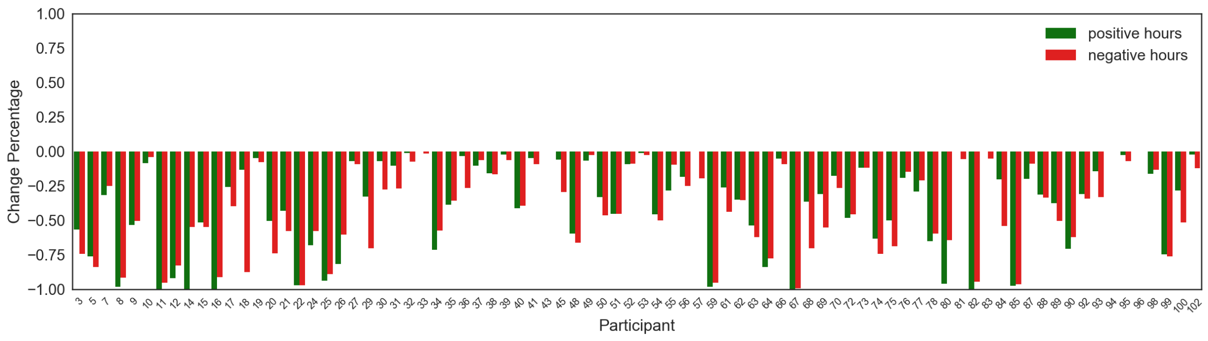

Figure 7.28 - Example drop rate chart for ES7-101 independent from inter-arrival times

Compared to static rule mechanism, trained entropy systems could adapt subjects' overall sharing behaviour more successfully in given dataset, and manage to lower false-positives more than true-positives successfully at approximate 60% of times. Another important argument is; because of the biased dataset, the non-normalized value of negative drops is relatively higher than the positive drops for every subject.

In the following section, we applied both systems together on test set.

7.2.4.3 Testing Decision-Making System

In this section, we test the proposed entropy systems together with identified policies for participants. Because total negative hours revealed by majority of policies are drastically dominant on total positive hours, normalization of even slightest change occurred in positive hours after entropy system applied can be unfair to compare with normalized change occurred in negative hours. Therefore, we have defined new metric for fair comparison of influence of entropy system on identified policies with different setting types.

$$Rate(x) = -\frac{x_{old} - x_{new}}{P_{old_{total}}} \tag{7.4}$$

where x can be either negative or positive hours, x_{old} is total negative or positive hours revealed by the policy before entropy system is applied, x_{new} is total negative or positive hours revealed by the policy after entropy system is applied, and $P_{oldtotal}$ is total number of hours revealed by the policy before entropy system is applied.

By utilizing proposed normalization function, it is certain that non-normalized change values of both negative and positive hours will be equal, and we can deliver fair comparison.

In Figure 7.29, caused change ratios which are normalized using given Formula (7.4) for each participant are given for different kind of policies and entropy systems. Entropy systems independent from inter-arrival time are used in performance test, in order to avoid areas recently introduced in test set specifically. Furthermore, the plots are divided with a line ($y=x$) and each scattered point represents policy for a subject. The region ($y<x$) is considered poor where entropy system restrains more positive sharing than the negative of a policy, and reduces overall accuracy of a policy. However, points in the region ($y>x$) receive accuracy improvement compared to standalone evaluation done previously, by being prevented to leak location information more than share correctly.

Because Policy A and B are able to rarely reveal location information, the influence of entropy system is not observed significantly. However, majority of the policies which were able to hit samples in test set, are enhanced with all entropy system. The points are farther dispersing in ES7-11, and gathering towards origin while strictness of entropy system is decreasing with higher C_p. According to the results, in general ES7-51 had highest number of policies those were affected in positive manner.

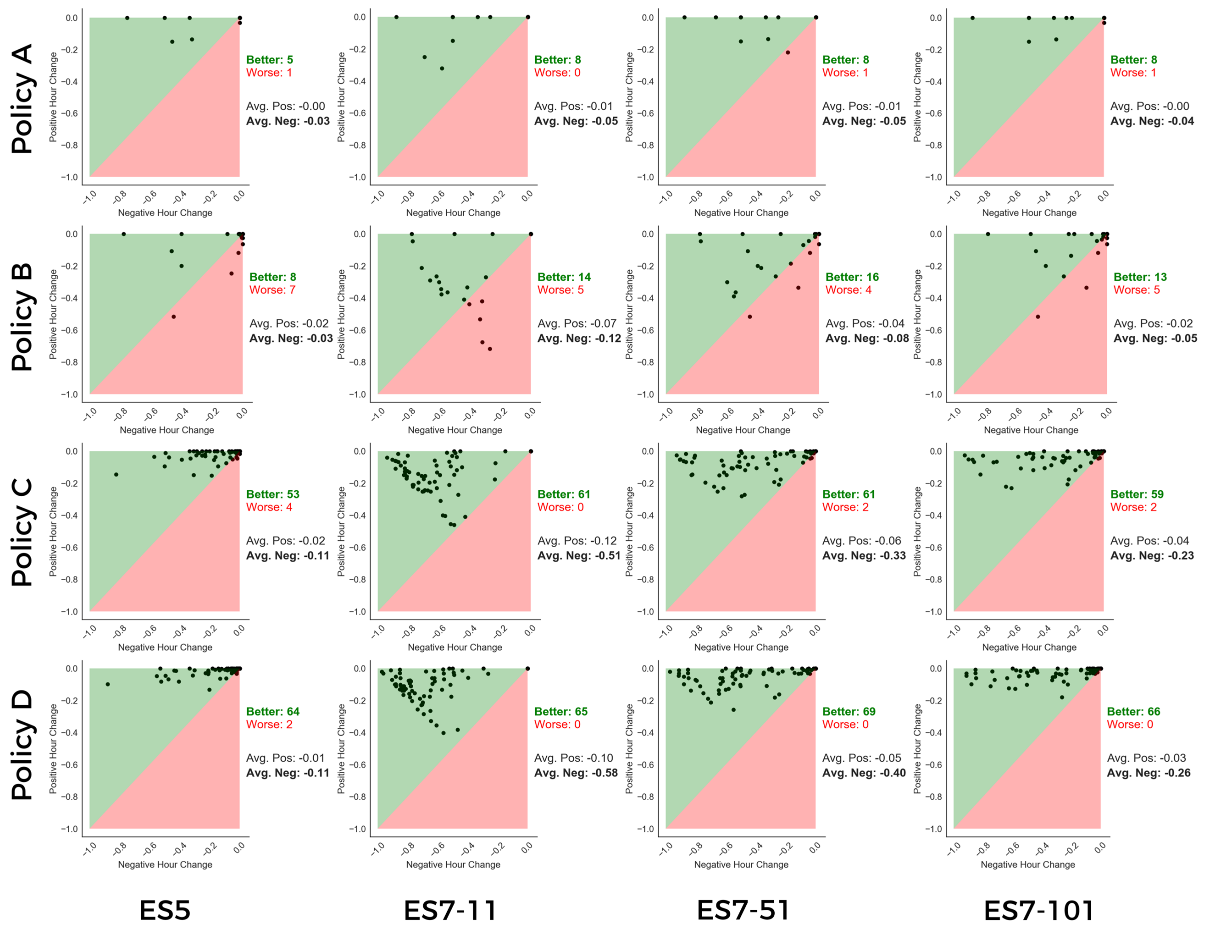

Figure 7.29 - Normalized drop rates using Formula (7.4) for each kind of Policy and Entropy System combination

8. Summary and Future Work

8.1 Conclusion

The change of individuals' sharing attitude over time presents a challenge of adaptation for location-based applications, especially for the ones which use static mechanisms. In this thesis, we first presented an optimization on algorithm in [6] that identifies optimal policy for a subject utilizing long-period training set. Four policies per subject is defined in different levels of strictness in order to simulate sharing preferences of individuals with diverse confidence intervals. Second, we presented routine aware control mechanism at decision-making process of location privacy applications, that detects probable false-positives caused by user-defined privacy preferences. The system is based on estimation of conditional trajectory entropy, and similar to approach proposed in [16]. In order to evaluate our approaches, we have used Reality Mining dataset [24] on which communication events are used as reference for sharing behaviour.

During the evaluation phase, it was discovered that even policies those are trained aiming higher accuracy value rather than the most sharing, was not able to cover subjects' sharing attitude in test set successfully. However, overall performance of routine aware control mechanism together with rules which have static nature, has implied improvement on prevention against false-positives. Further, in standalone evaluation, entropy systems had positive influence on approximately 60% of subjects. The results indicate that considering given dataset, entropy value of locations was more informative to distinguish false-positives and true-positives than the rules in policies, and entropy systems can play supportive role in location privacy mechanisms.

8.2 Future Work

As discussed in Chapter 4, using two A star algorithms with separate heuristic functions results in identifying not the best but the optimal policy for a subject. The reason is, heuristic function of first A star algorithm is able to measure cost in its own domain, but unable to measure in second algorithm's domain. This implies that first ran tries to form clusters in ground-truth space as broad as possible, but do not consider if generated clusters would form the best policy given cluster limit in second algorithm. The future work on heuristic function may provide identification of best policy given rule limit using large-scale datasets.

Further, estimating entropy of a location by utilizing trajectory entropy of previously visited locations has a drawback which is, once trajectory is affected by surprise location or sequence of locations, system cannot accurately differentiate farther locations whether they are routine or out-of-routine events until path information is overwritten. To overcome this situation, system should able to recognize initial and future destination locations. Further research on probabilistic system which can predict destination location given time slot, and construction of a joint system together with entropy mechanism can restrain this influence.

Finally, another implementation may be considered together with routineness aware control mechanism and dynamic location privacy system adopts machine-learning algorithms. That being said, dynamic learning was not in the scope of this thesis. Further evaluations can be applied using dynamic learning approach with real-time markov chain updates, which can adapt mobility pattern changes (e.g. relocation or holiday situations) of subjects. On the other hand, the entropy systems with higher degree of detail (e.g. entropy systems exclusive for participant, area, and weekend or time) can be trained and tested mining larger datasets. Because individuals are prone to use different mobility patterns on weekdays and at weekends, more detailed conclusion can be reached.

9. Bibliography

[1] J. Y. Tsai, P. G. Kelley, L. F. Cranor, and N. Sadeh, “Location-Sharing Technologies : Privacy Risks and Controls,” *A J. Law Policy Inf. Soc.*, vol. 6, no. 2, pp. 119–151, 2010.

[2] N. Sadeh, *M-commerce : technologies, services, and business models*. John Wiley & Sons, 2002.

[3] https://newzoo.com/insights/articles/global-mobile-market-report-app-market-to-gross-44-8bn-this-year/

[4] L. Church, J. Anderson, J. Bonneau, and F. Stajano, “Privacy stories: confidence in privacy behaviors through end user programming.,” *Soups*, pp. 6–7, 2009.

[5] J. Y. Tsai, P. Kelley, P. Drielsma, L. F. Cranor, J. Hong, and N. Sadeh, “Who Is Viewed You The Impact Of Feedback In A Mobile location Sharing Application,” *Proc. 27th Int. Conf. Hum. factors Comput. Syst. - CHI 09*, p. 2003, 2009.

[6] M. Benisch, P. G. Kelley, N. Sadeh, and L. F. Cranor, “Capturing location-privacy preferences: Quantifying accuracy and user-burden tradeoffs,” *Pers. Ubiquitous Comput.*, vol. 15, no. 7, pp. 679–694, 2011.

[7] J. McInerney, S. Stein, A. Rogers, and N. Jennings, “Exploring periods of low predictability in daily life mobility,” *Nokia Mob. Data Chall.*, 2012.

[8] N. Sadeh *et al.*, “Understanding and capturing people’s privacy policies in a mobile social networking application,” in *Personal and Ubiquitous Computing*, 2009, vol. 13, no. 6, pp. 401–412.

[9] K. Connelly, A. Khalil, and Y. Liu, “Do i do what i say?: observed versus stated privacy preferences,” *Proc. 11th IFIP TC 13 Int. Conf. Human-computer Interact.*, pp. 620–623, 2007.

[10] I. Bilogrevic, K. Huguenin, B. Agir, M. Jadliwala, M. Gazaki, and J. P. Hubaux, “A machine-learning based approach to privacy-aware information-sharing in mobile social networks,” in *Pervasive and Mobile Computing*, 2016, vol. 25, pp. 125–142.

[11] G. Danezis, “Inferring privacy policies for social networking services,” *Context*, pp. 5–10, 2009.

[12] G. Bigwood, F. Abdesslem, and T. Henderson, “Predicting Location-Sharing Privacy Preferences in Social Network Applications,” *Proc. First Work. Recent Adv. Behav. Predict. pro-active pervasive Comput. (June 2012)*, pp. 1–12, 2012.

[13] I. H. Witten, E. Frank, and M. a. Hall, *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition*, vol. 54, no. 2. 2011.

[14] L. Ekroot and T. M. Cover, “The Entropy of Markov Trajectories,” *IEEE Trans. Inf. Theory*, vol. 39, no. 4, pp. 1418–1421, 1993.

[15] M. Kafsi, M. Grossglauser, and P. Thiran, “The entropy of conditional Markov trajectories,” *IEEE Trans. Inf. Theory*, vol. 59, no. 9, pp. 5577–5583, 2013.

[16] M. Kafsi, M. Grossglauser, and P. Thiran, “Traveling salesman in reverse: Conditional Markov entropy for trajectory segmentation,” in *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2016, vol. 2016–Janua, pp. 201–210.

[17] W. Stewart, *Probability, Markov chains, queues, and simulation: the mathematical basis of performance modeling*. 2009.

[18] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, “Mining interesting locations and travel sequences from GPS trajectories,” in *Proceedings of the 18th international conference on World wide web - WWW ’09*, 2009, p. 791.

[19] J. L. King, “Centralized versus decentralized computing: organizational considerations and management options,” *ACM Comput. Surv.*, vol. 15, no. 4, pp. 319–349, 1983.

[20] S. B. Cho, “Exploiting machine learning techniques for location recognition and prediction with smartphone logs,” *Neurocomputing*, vol. 176, pp. 98–106, 2016.

[21] C. Song, Z. Qu, N. Blumm, and A.-L. Barabasi, “Limits of Predictability in Human Mobility,” *Science (80-.).*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[22] http://www.jupyter.org/

[23] http://www.PleaseRobMe.com/

[24] N. Eagle and A. Pentland, “Reality mining: Sensing complex social systems,” *Pers. Ubiquitous Comput.*, vol. 10, no. 4, pp. 255–268, 2006.

[25] Z. Riaz, F. Durr, and K. Rothermel, “Optimized location update protocols for secure and efficient position sharing,” in *Proceedings - International Conference on Networked Systems, NetSys 2015*, 2015.

[26] Z. Riaz, F. Durr, and K. Rothermel, “On the privacy of frequently visited user locations,” *Proc. - IEEE Int. Conf. Mob. Data Manag.*, vol. 2016–July, pp. 282–291, 2016.

[27] J. Wiese, P. G. Kelley, L. F. Cranor, L. Dabbish, J. I. Hong, and J. Zimmerman, “Are you close with me? are you nearby?: investigating social groups, closeness, and willingness to share,” *UbiComp 2011 Proc.*, p. 197, 2011.

[28] E. Toch, “Crowdsourcing privacy preferences in context-aware applications,” in *Personal and Ubiquitous Computing*, 2014, vol. 18, no. 1, pp. 129–141.

[29] S. Consolvo, I. E. Smith, T. Matthews, A. LaMarca, J. Tabert, and P. Powledge, “Location Disclosure to Social Relations: Why, When, & What People Want to Share,” *CHI 2005 Conf. Hum. Factors Comput. Syst.*, pp. 81–90, 2005.

[30] J. Scott, *Social network analysis: A handbook*, vol. 3, no. 5. 2000.

[31] T. Aledavood *et al.*, “Daily rhythms in mobile telephone communication,” *PLoS One*, vol. 10, no. 9, 2015.

Declaration

All the work contained within this thesis except where other acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

Murat Ünal