Institute of Parallel and Distributed Systems (IPVS)
Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart

Master thesis

Location-history Partitioning Algorithms for Privacy in non-trusted Geo-Social Networks

Qi Zhang

Tutor:	M. Sc. Zohaib Riaz
Examiner:	Prof. Dr. Kurt Rothermel
Begin Time:	13.12.2016
End Time:	12.06.2017

Abstract

Due to the rapid development of mobile device technology in the past couple of decades, mobile devices are playing a more and more important part in our daily life. Many mobile services along with mobile devices have integrated into our activities or even reshaped our lifestyle. Location services are one of the main mobile services being widely used. One can share ones location to get to know information nearby, and it could be shared with friends in social media. New mobile applications are showing up at an amazing speed, together with that, the usage of location data is a privacy threat. If too much information is shared, the user's movements could be predicted; highly privacy sensitive locations, such as home location of user, could be leaked.

Many location based applications, such as geo-social networks (GSN), use Location Servers to store user position information. However, since GSN providers may not be fully trustworthy or may not be able to protect user data, users may not want to store all of their privacy-sensitive location information with a single provider. Therefore, this thesis focuses on developing and evaluating methods to partition location data among multiple servers as similarly attempted in other approaches.

In this thesis, we try to partition location data to achieve privacy protection. We have studied a range of mobility modeling methods that consider the different fundamental dimensions of the location data, i.e., spatial, temporal, and semantic, as well as their combinations. Inspired by those methods, we have proposed partitioning methods to increase privacy protection. Furthermore, a couple of other partition methods, which are combinations of spatial, temporal and semantic, are implemented. Eventually all the partition methods are evaluated with our data.

Acknowledgement

I am appreciated that Mr. Zohaib Riaz could give me a chance to work on this topic. Also I am very thankful for all the instructions and suggestions he gave to me during the past several months.

Moreover, I would like to express my appreciation for all my family members and friends for their supports and encouragements.

Table of content

List of Figures

List of Tables

Acronyms

GSN	Geo-Social Networks
PCA	Principle Components Analysis
LS	Location Server
SC	Social Circle
MD	Mobile Device
GPS	Global Positioning System
PC	Personal Computer
SMS	Short Message Service
PMM	Periodic Mobility Model
PSMM	Periodic & Social Mobility Model
LTP	Long-term Privacy

1 Introduction

1.1 Mobile development

In the past few decades, mobile technology has been developing at a rapid pace. From the point of mobile phones only supporting calls and SMS to smart phones which are already a mini computer, new generations of mobile devices keep coming out and it has affected our life patterns in many ways. Moreover, different from traditional PCs, mobile devices have more embedded functions, such as a GPS or a camera, which enables them to be capable for more life services.

Along with the development of mobile devices, mobile applications are growing in an incredible speed as well. Smartphones are steadily replacing personal computers as they become the first choice of connection with the internet. In software industry, mobile application has become a huge market and keeps growing daily without an end in sight [1] [2].

There are many mobile services that lie behind those applications, which make our life more convenient and digitized. Due to the GPS module, mobile devices have become a main provider for location data. Location services offer the possibility to gain information of the area, to find friends nearby and to share a location in social media. However, at the same time privacy is being threatened.

1.2 Location sharing services and privacy concerns

Privacy is an individual right to have a selective expression about oneself. One of the key privacy's we really care about is the location information. Under the context of location history, we mean to have the ability to have a selection of third parties, to whom the location information will be given, based on the user's preference. Currently Geosocial Networks (GSNs) such as Facebook, Twitter, Instagram, etc. are extremely popular. These apps keep asking users to share location information; some even have 'check-in' services which publish the location accessible for the public. Based on a significant amount of location data, it is not hard to get an image of the user's activities or to predict the user's location. Locations like banks, home address and hospital visits are quite privacy sensitive. They are in danger of being leaked.

GSN functionality is implemented using background location servers to store the location information. This raises further privacy concerns as data stored on these servers may not be safe due to the possibility of data breaches 错误!未找到引用源。. Thus the servers are non-trusted servers.

Besides, there is a risk of single point failure. If the server is attacked, everything stored in the server will be leaked and the user's privacy will be compromised. For instance, details of personal user information, such resident address, can be acquired. Unwanted advertisements could easily reach user in any possible way. Someone can easily get to know where the user lives and what kind of sickness

he is suffering now. It is universally considered as unpleasant experience and could bring extra troubles in daily life.

Finally, we share location information with other people in GSN. And there are different users groups such as friends, family and colleague, which we call it social circles. The privacy regarding to social circles [3] is within our concerns too. A certain social circle should not acquire more information than those the user intends to share. For example, the user might not want to his or her family to know about those nightlife check-ins. And such unintended sharing is defined as oversharing.

1.3 Contributions and thesis structure

Certainly protecting mechanisms are needed to fight against all kinds of privacy leaking. Here we focus on protecting user privacy with respect to location servers and the social circles of users the location sharing privacy and will try to partition location data to achieve the goal.

To avoid privacy loss, there are many possible ways to do in a location sharing system. In this thesis, we consider the problem of online partitioning of location information. To do this, we develop algorithms that train themselves on location history information before becoming usable for online partitioning. Data will be partitioned to enhance privacy protection.

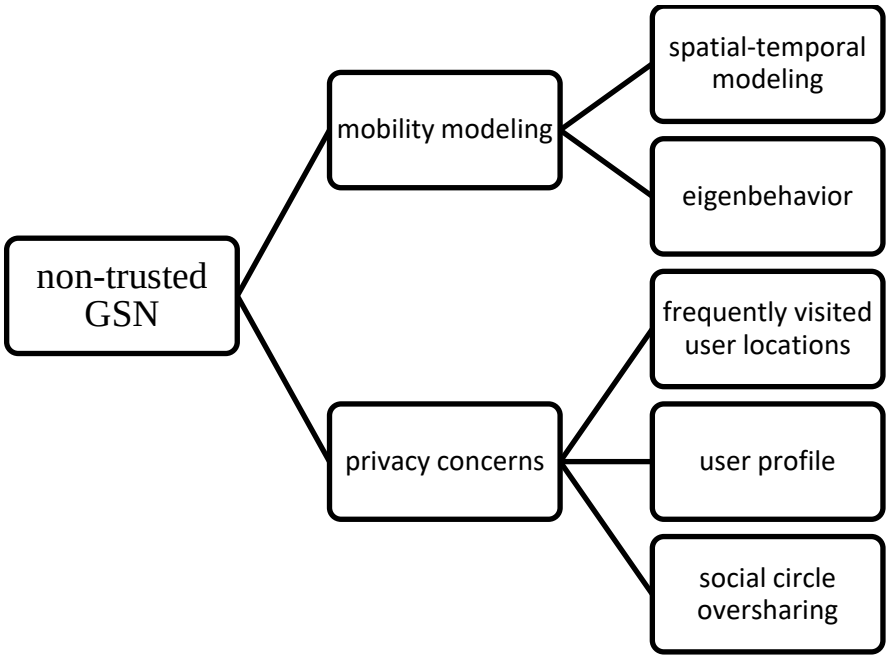
We have studied some typical mobility modeling methods and proposed a variety of corresponding partitioning methods as countermeasures. After that, we do evaluation and comparison for all the methods we have proposed.

In this thesis, we illustrate background and related works at first. Secondly, the system model and problem statement is made. Then all the partitioning methods we use for protecting privacy are introduced and evaluated. In the end, we draw a conclusion of this study and propose some ideas for future work.

2 Background and Related work

This chapter is aiming to give an introduction of related works which have been done. There are existing works addressing “non-trusted servers”. An optimized location update protocols for position sharing is introduced [5]. A set of location update protocols are proposed to achieve efficient and privacy guaranteed location sharing. For the sake of protecting privacy of frequent visits, an approach is proposed to distribute location data to different servers [6].

There are also preceding works of mobility modeling. To be able to predict user’s movements, a mobility modeling method is proposed [7]. In this approach, the researchers study human movements and combine spatial, temporal and social relational information to make location prediction. Another mobility modeling method called eigenbehavior is introduced [8]. This method uses principal component analysis to get typical human movement patterns.



2.1 Spatial-temporal mobility modeling

A friendship and mobility modeling approach has been proposed [7]. While in reality human’s movements always follow a certain pattern, especially periodic in time. One would expect based on spatial and temporal dimensions of data, it is feasible to find the inner patterns of a person’s movements.

Cho et al. have studied the properties and features of human geographical movements and found out temporal and geographic periodicity of human movement.

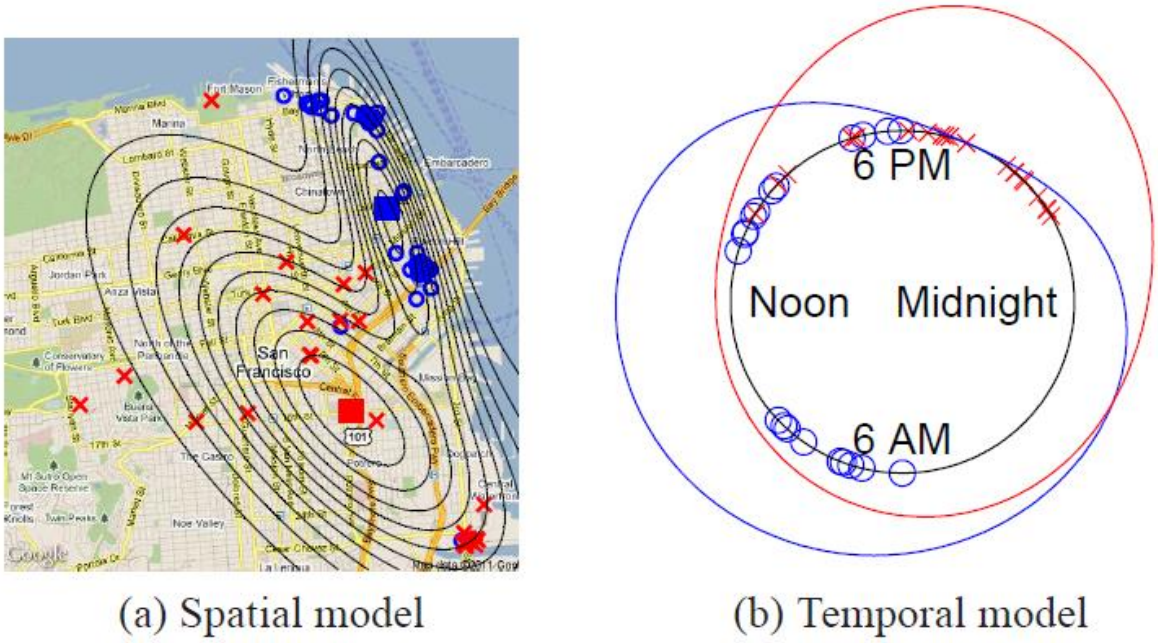


Figure 1. Spatial and temporal model [7]

Figure (a) shows a user's movements in San Francisco, who lives in the south-east of the city (red cross) and works along the coast in north-east (blue circles). After collecting those data, Gaussian distributions for both home and work check-ins could be formed. This figure shows a jointed distribution.

Figure (b) shows the temporal distribution in a period of a day, with red cross standing for home and blue circle for work. It is obvious during the day more check-ins go to work and in the evening go to home.

In this paper one possible approach to predict user's location is proposed. Based on the conclusion human movement is periodic in both temporal and geographic range, a Periodic Mobility Model (**PMM**) has been proposed. PMM considers both time and locations to fit Gaussian distributions. Besides, the social relation impact on human movement has also been studied. Another model Periodic & Social Mobility Model (**PSMM**) has been introduced too, which adds a social component to PMM.

After evaluation with real data, those models have been proven to have high accuracy and low error rate.

2.2 Eigenbehavior

Another mobility modeling method 'eigenbehavior' is proposed [8]. Inspired by eigenvectors and principal component analysis (**PCA**), the method is trying to find the typical movement pattern inside human movements.

This method is also based on the assumption that human movement is periodic. But different from ‘friendship and mobility’, this method does not consider spatial information. It uses semantic information instead. In this paper, researchers give an example of 5 states of user movements, respectively home, work, no signal, else and off. After getting data for a number of days, all the data will be transferred to a matrix with each row standing for a daily pattern. PCA will be applied to the matrix to get the principle components named as eigenbehaviors. According to this paper, this method is proven to be working well. Top eigenbehaviors refers to some top daily movement pattern, like going working during the day and staying at home in the evening.

Furthermore, this paper discussed the application of this method to recognize the social relationship between people.

2.3 Location update protocol

Location sharing is a widely used service in all kinds of Geo-social Networks. For our concerns, privacy is not guaranteed in the traditional way of location updating. Thus an approach distributing location in different servers was proposed to increase privacy [10]. However, because location data will be sent to several servers instead of one in this model, a communication overhead occurs. In order to solve this issue, a novel location update protocol was proposed [5].

The system model used in this approach contains three components, namely mobile device, location servers and location based applications. The location data is generated by mobile device. Then data will be distributed to location servers and location based applications will have access to some servers.

Between mobile device and location servers, the location update protocols will be applied. Three novel update protocols, namely Dead Reckoning, Selective Update and Selective Dead Reckoning haven been proposed.

Dead Reckoning protocol enables location server to predict real-time location of the moving object. When the deviation between prediction and actual location exceeds a threshold, new updates will be required.

In Selective Update protocol, only a minimal number instead of a fixed number of visits which are seen as refinement shares will be updated.

After carrying out experiment with the two protocols above, advantages and disadvantages were found out for each protocol. Dead Reckoning has low overhead but performs poorly at jerky patches. Selective Update has higher overhead but stable performance in terms of sharp turns. So a protocol, Selective Dead Reckoning, which is a mix of both, was proposed.

According to evaluation results, Selective Update and Selective Dead Reckoning protocols provide more efficient position sharing as well as privacy guarantee.

2.4 Critical information protection

A frequently visited location could represent personal characters or behaviors. The court ruled: “A person who knows all of another’s travels can deduce whether he is a weekly church goer, a heavy drinker, ... and not just one such fact about a person, but all such facts.” [6] Revealing of frequent semantic locations is seen as a threat of privacy.

In order to protect user’s privacy in frequent semantic locations, an approach was proposed [6].

Similar to the system model in the location update protocol, the system used here also consists three components. And there is a predefined persona for each location based application. Based on the persona, location based application only gain accesses of selected location servers.

The basic idea of archiving critical location protection is that each location server only stores a limited number of location data, so that it can only infer a small number of critical locations. Moreover, another location server $LS_0$is used to store completely safe data, which means there will be no critical locations being revealed. The following figure illustrates the idea.

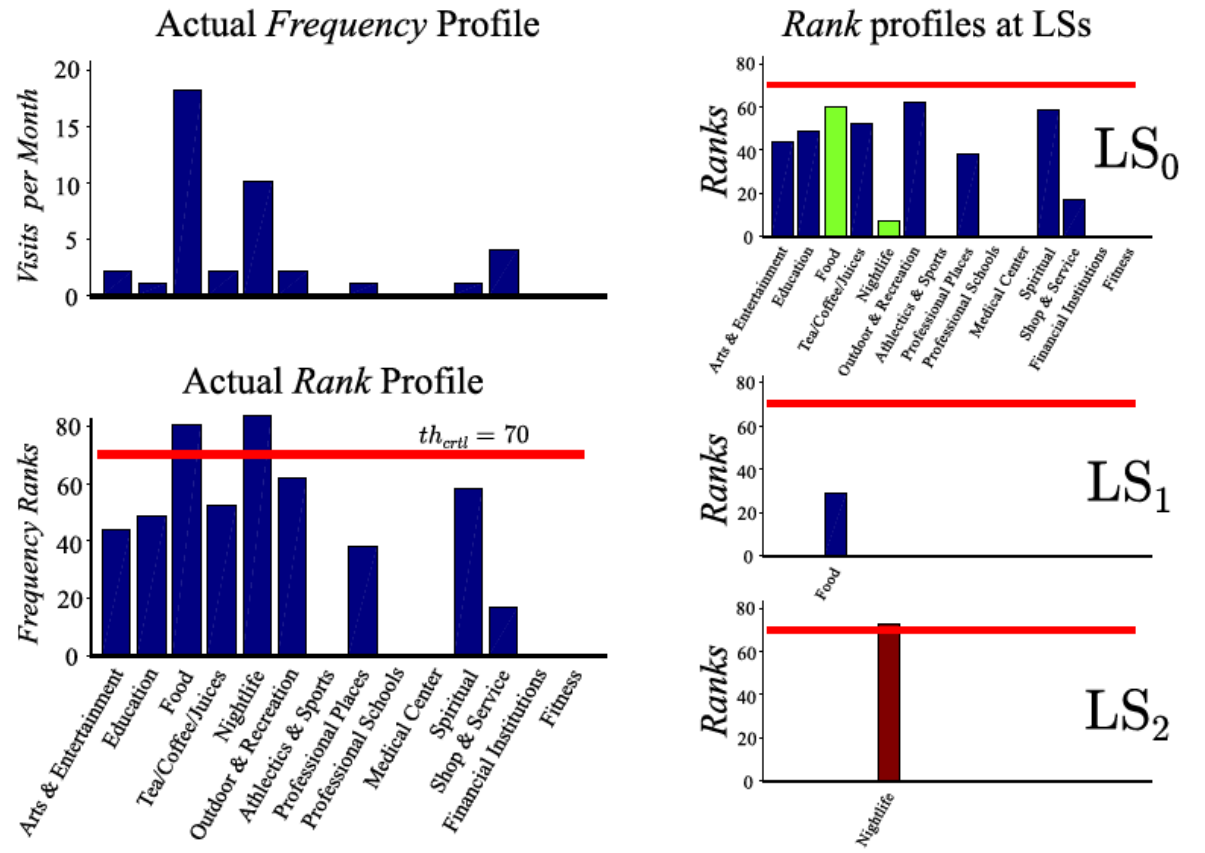


Figure 2. Actual frequency and rank profiles [6]

As it is shown in this figure, there is a threshold of 70, so food and nightlife are frequently visited locations. Some of their visits will be sent to LS_1 and LS_2.[10]

A basic and an advanced algorithm were proposed in the paper to determine which location server will store each visit. After evaluation, it is shown that the critical information is well protected and meanwhile sufficient information could be provided for location based applications.

3 System Model and Problem statement

In this chapter, the system model of this thesis will be introduced. Thereafter the problem statement of this thesis will be made.

3.1 System Model

The system comprises 3 components: mobile device, the Location Servers (LSs) and Social Circles (SCs).

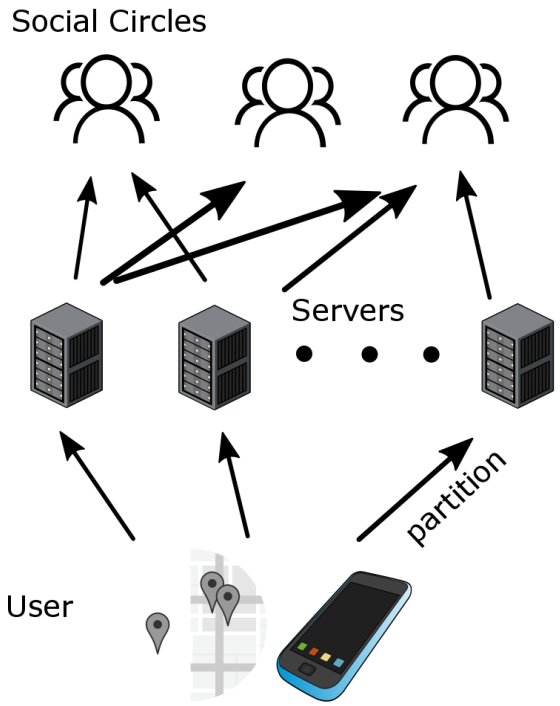


Figure 3. System model

This figure illustrates the relationship between components in the system model. In our system, users move freely with a mobile device. Check-ins are made with mobile devices and send to the internet. But before the data goes to servers, we apply our partitioning method to determine which server the data should go to. And from the servers, a particular social circle could have access to some of the servers to read data.

Mobile Device (MD) is used by user to produce user check-in c_i, and the complete check-in history is C.

$$C = \{c_1, c_2, \ldots, c_n\}$$

MD is seen as a trustable and thus partitioning algorithm is running on MD to determine which server each check-in should go to. Given a number of k Location Servers, through partitioning algorithm, the user data is partitioned into k subsets $CP_i = \{c_{a1}, c_{a2}, \ldots, c_{ak}\}$

$$CP_i \cap CP_j = \emptyset \ (i \neq j)$$

The ***Location Servers (LSs)*** belong to different providers such as Facebook, Twitter etc. LSs are seen as non-trustable. Moreover, having a single provider means a single point of failure for privacy. LSs are storing location updates from users. In traditional way, the LSs get all data directly from mobile devices. But in our system, the LSs are only receiving part of data from MD. The data stored in each LS is the partitioning result CP_i.

Social Circles (SCs) are the concepts of our actual social life, SCi can be family, friends, colleagues etc. We would like to share our location information with others, but not without limitation. Different social circles should acquire different kinds of location data, for example colleague can only can data about work. So based on that fact we have several LSs storing different subsets of data, it is possible for SCs to get certain types of data distributions by accessing several servers. And the accesses are determined by users. So each SCi is corresponding to a combination of LSs and only has access to these *LS*.

What kind of and how much data is accessed by each SC will be defined in later chapter.

3.2 Problem Statement

Our goal is to protect user's location privacy in each Location Server. However, the partitioning results' utility and privacy requirements on Social Circles are within our concerns too. This subchapter is aiming to introduce those requirements in details. Partitioning algorithms will need to be trained first on offline location history information. Then you can use them to partition in an online fashion.

The workflow of each partition method contains 2 parts: training and testing. The typical steps of applying and evaluating a partition method are to train the data at first and to apply the partition method using training results. After testing, we will evaluate the testing results.

3.2.1 Privacy requirement in Location Server

User profile is the personal data of a specific user. It is a digital representation of a user's identity. Much privacy sensitive information is associated with the user profile. It is an overview of all the user's behaviors, which is highly privacy sensitive. Location Server's Long-term Privacy (**LTP**) focuses complete user profile. To achieve user profile protection, ideally each individual venue distribution of all the data in CP_i should have a significant different shape comparing with the complete user data C's venue distribution is VD_c. The venue distribution is essentially a vector, with each element denoting a number of location visits belonging to each venue category. To measure the distance of venue distributions between CP_i and the complete data, we will use earth mover distance.

For two vectors v_i and v_j, at first they will be normalized: $v_i' = \frac{v_i}{|v_i|}$. And the earth mover distance between v_i and v_j is defined this way:

$$EM = \sum_{a=1}^{n} \left| v_i'(a) - v_j'(a) \right|, \qquad v_i', v_j' \in 1 \times n$$

3.2.2 Utility requirement in Social Circles

Privacy protection is our goal, but after partitioning the data should still be useful. For example in our system model, the SCs need to get a certain amount of information which defines their utility requirement. Our utility evaluation will focus on social circles. At first depending on user's preference, each social circle has its demanding numbers for each venue type. For the simplicity of evaluation, we will have a set of predefined requirement vectors for each social circle $UR = \{ur_{friend}, ur_{family}, ur_{colleague}\}$. Given a number of n venue types $u \in 1 \times n$. Values in a vector are proportions of complete data's venue distribution $u(i) \in [0,1]$. So the utility of social circle is

$u_i = ur_i \cdot VD_c$, i $\in \{friend, family, colleague\}$.

To measure the utility, each social circle will iterate all the possible combinations of LSs to have an as close as possible venue distribution to the utility $u_{friend/family/colleague}$, so that SC could have enough information. The utility will be measured by how much percentage of $u$are reached in SC.

3.2.3 Privacy requirement in Social Circles

When we evaluate utility, we focus on to what extent the requirements are reached without concerning a possible excess of shared information. This concept of oversharing should be evaluated as another privacy concern.

Similar to utility, we will also have a proportion referring to information being shared too much, defined as oversharing requirements: $OSR = \{osr_{friend}, osr_{family}, osr_{colleague}\}$. And the over sharing of social circle is

$os_i = osr_i \cdot VD_c$, i $\in \{friend, family, colleague\}$.

To measure oversharing, we use the same combination we used in utility evaluation. And calculate for the so far best possible combination, how much percentage the over shared data is taking up.

4 Partitioning Methods

In this chapter, we will introduce all the partitioning methods we proposed. For each method, we will discuss the motivation, basic idea and implementation. In the end of this chapter, an optimization, bottom up approach for all partitioning methods, will be introduced.

4.1 Overview of partition algorithms

For our system, user's location history is made of a sequence of check-ins as discussed in Chapter 3. Each check-in contains the information of location, check in time and the check-in spot information. These three types of information, respectively spatial, temporal and semantic data, could be used for the partitioning.

Location information contains the x and y coordinates values, where the base points were determined differently for different users. And we have a classification for check-in spot venue types, for example nightlife, food.

In our system, any algorithm will use one single dimension or combine 2 or 3 out of these three dimensions. The figure below helps to illustrate all the partitioning methods.

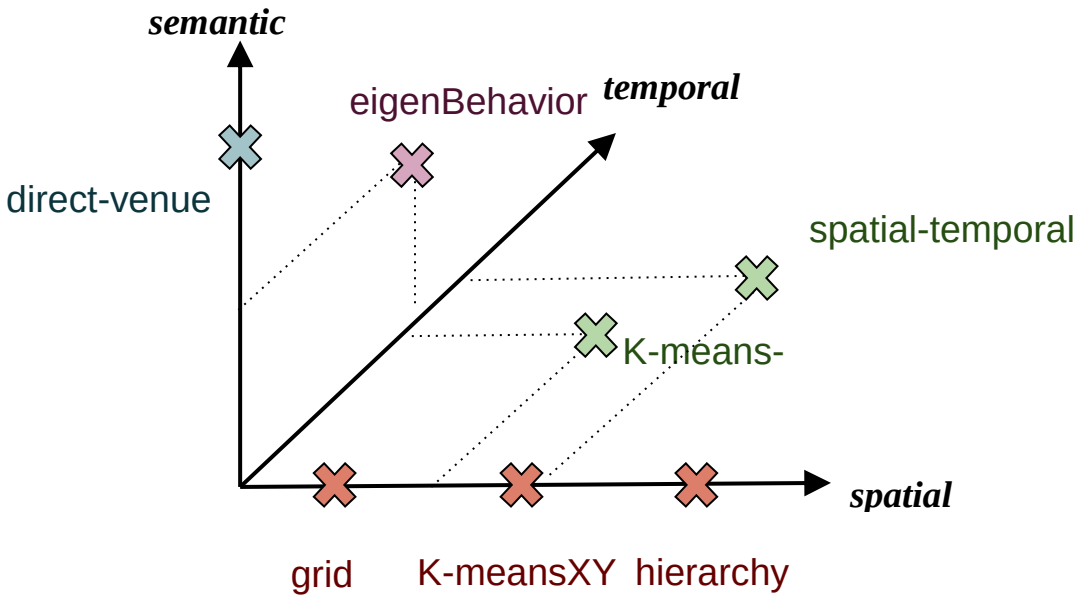


Figure 4. Partitioning methods overview

As one can see from this figure, 'grid', 'K-meansXY' and 'hierarchy' are listed in the spatial axis. They only partition data by doing clusters or divide space based on spatial information. Furthermore, if time is considered as well, we have 2 methods 'K-meansXYH' and 'spatial-tempo' to do the job.

Inspired by eigenvectors, ‘eigenbehavior’ was proposed as a possible way to describe human behavior, which uses semantic and temporal information. And we will make use of it to do partition. Finally, direct partitioning by venue types looks like a possible solution, and it only uses semantic information.

For all the partition methods, we divide the procedure into two parts. The first part is training and the second part is testing. So basically we use the training data to apply the partitioning method. With training results, according to certain predefined rules, each check-in will be assigned to a partition.

4.2 Spatial dimension

Since what we are dealing with is location data, t is natural that spatial dimension comes to our mind at first. The three pure spatial partitioning methods will be introduced here.

4.2.1 Grid

4.2.1.1 Basic idea

If we only consider spatial dimension, the simplest method is to divide the map evenly in space. This might turn out to be not optimal, as the check-in data is divided almost randomly. However, we could have a try and at least this method could serve as a comparison.

In this algorithm we will divide the whole map evenly into several grids.

4.2.1.2 Implementation

For simplicity, we just divide the whole map into a number of grids slightly bigger than the number of partitions we want. For example if we want 5 partitions, we just to divide the map into 9 grids. In the step following it, we will deal with it and finally we will get the number of partitions we want.

Before we start to partition the location data, it is important to ensure that data is in a more centralized range. Otherwise some outliers might affect the results. So our very first step is to remove outliers.

If a set of data is normally distributed, then 99.7% of the data will be distributed in the range of 3 times standard deviation to the mean [11]. This is one feature we can make use of to remove the outliers in our data set. There are two dimensions, X and Y respectively. We could remove the outliers for both X and Y coordinates. Because outliers in either X or Y dimension will impact on geographical distribution. For simplicity we just remove outliers in both X and Y dimensions.

Here is pseudocode:

Remove outliers	
1:	**Procedure** remove_outliers
2:	**While** num_outliers / num_all < 0.1
3:	[mean, deviation] = get_mean_and_dev()
4:	Indexes_outliers = get_outliers(mean, deviation)

5:	Num_outliers += length(Indexes_outliers)
6:	Remove_outliers(Indexes_outliers)
7:	**End**
8:	**End**

After removing outliers, partitioning could be applied to the dataset. Here I will use an example of 9 partitions to illustrate the implementation of the algorithm.

We assume the user's activity is randomly distributed in the space. So in both X and Y dimensions, there should be no significant difference in the aspect of the range. Since we want 9 grids from the map, it is reasonable to devise both X and Y axis into 3 even parts.

At first we will divide the map into 9 partitions, and then for each grid we will calculate the center of the grid. And the X and Y coordinates of each center will be saved as training results. The following figure is a visual description of this algorithm.

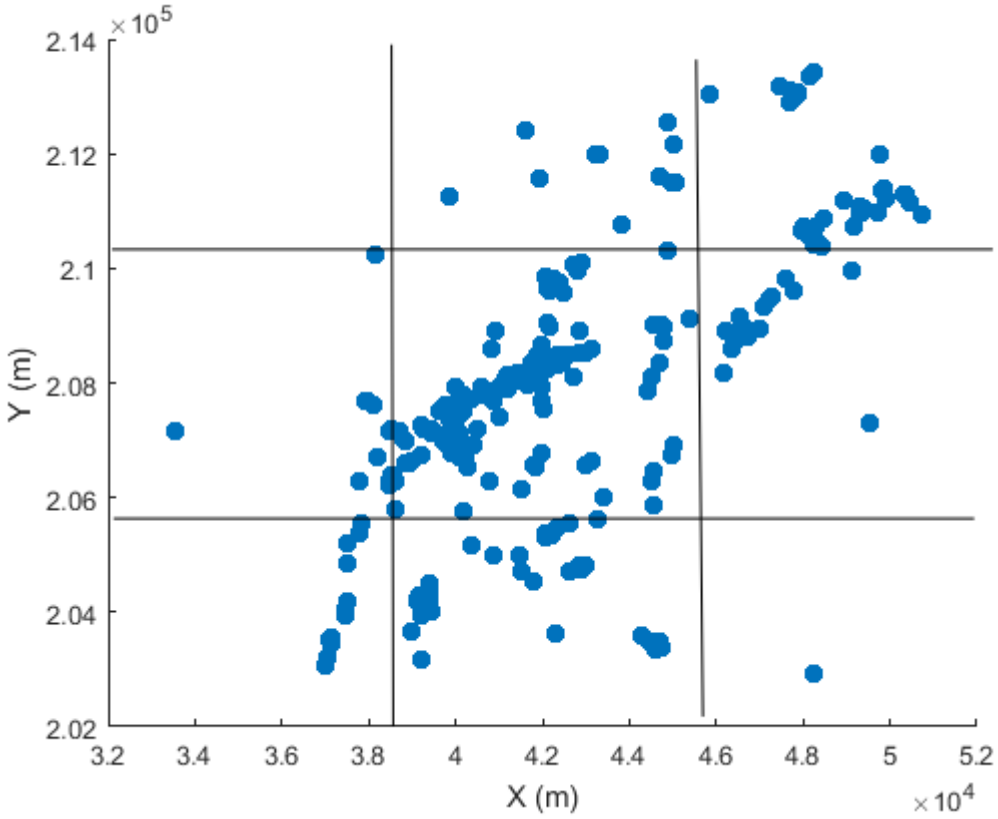


Figure 5. Grid partitions method illustration

As training results, a couple of centers will be used in testing part to do the partition. For each check-in we will calculate the distance to all the centers we got. And the check-in point will be assigned to the closest center. If there is more than one closest center, the cluster will be randomly decided among those.

The following figure is one example result of applying the training results to test data.

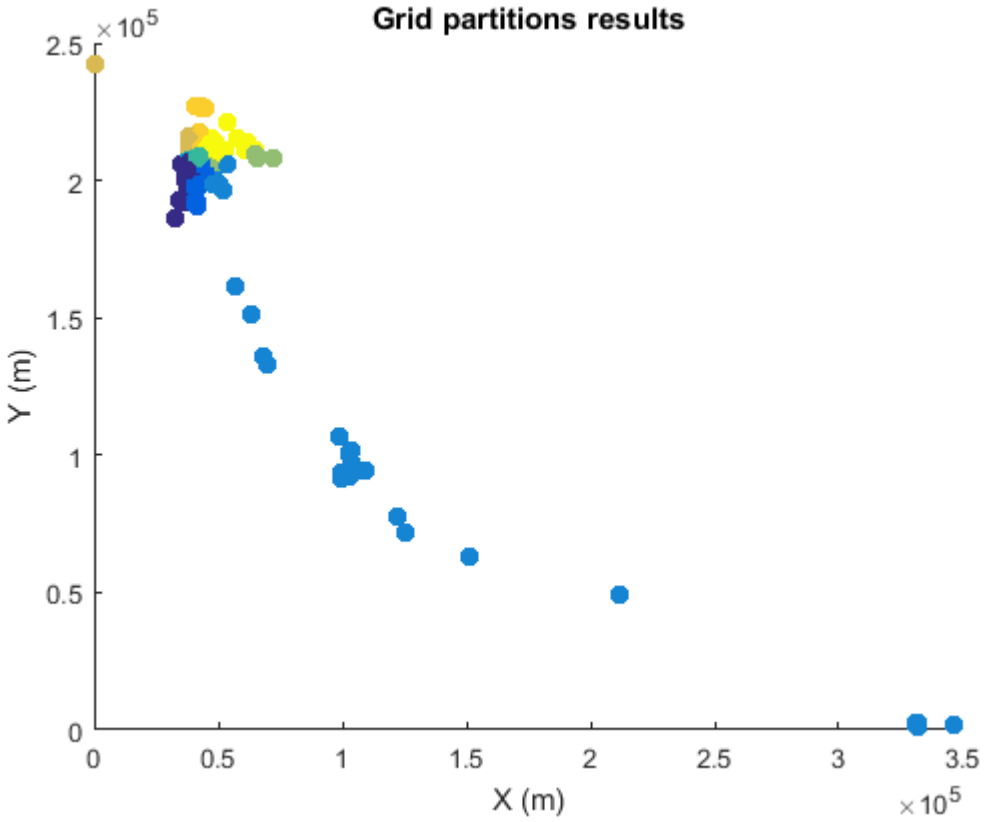


Figure 6. Grid partition result example

Since we removed outliers at first, one can see the centers are all located in the left upper part in this figure. So the sparsely distributed check-in points in the right bottom part are all assigned to one cluster. If outliers were not removed, it is obvious that there will be some grids without any data in it and some grids get many check-ins.

Here is pseudocode for grid method.

Partition - grid
1: **Procedure** grid_partition
2: Divide_map_into_grids()
3: centers = Get_centers_of_each_grid()
4: **For** check-in in check-ins
5: Assign_to_closest_center(centers, check-in)
6: **End**
7: **End**

4.2.2 K-means-XY

4.2.2.1 Basic idea

The method of grid might be not perfect because the check-ins might not evenly distribute throughout the geographical space. And often it is the case that there are some stay points more closed to each

other and some others gathered together. So it will be nice if we can have a method which can find the proper clusters based on the closeness of a group of data points. Then K-means comes to our mind.

K-means is a cluster method which is widely used for clustering. Given a dataset $(x_1, x_2, \ldots, x_n)$ and a cluster number k, this method is aiming to classify all the data points to a number of k clusters S_i, so that the following equation could be satisfied.

$$arg\ min \sum_{i=1}^{k} \sum_{X \in S_i} || X - \mu_i ||^2$$

In K-means, a number of $k$initial points will be chosen randomly at beginning. And step by step it will update the results to approach the best results and finally it will converge. However, a global optimal solution can not be guaranteed. The result is only a local optimal solution.

Here we only consider location data which means we will only use X and Y coordinate data.

As mentioned above, K-means can only have a local optimization results, for different starting points, the result might turn out to be different. In order to get more optimized results, we will try to apply k means more than one time.

In addition, it is required to specify the number of clusters before using K-means. But in our case, even though we have a desired number of partitions, for the dataset the number might be not proper for clustering. So it is not clear how many clusters should be applied. In order to find a proper number of clusters, we will try out different numbers of clusters. After running for all chosen numbers of clusters, we will have them evaluated and then choose the best number of clusters.

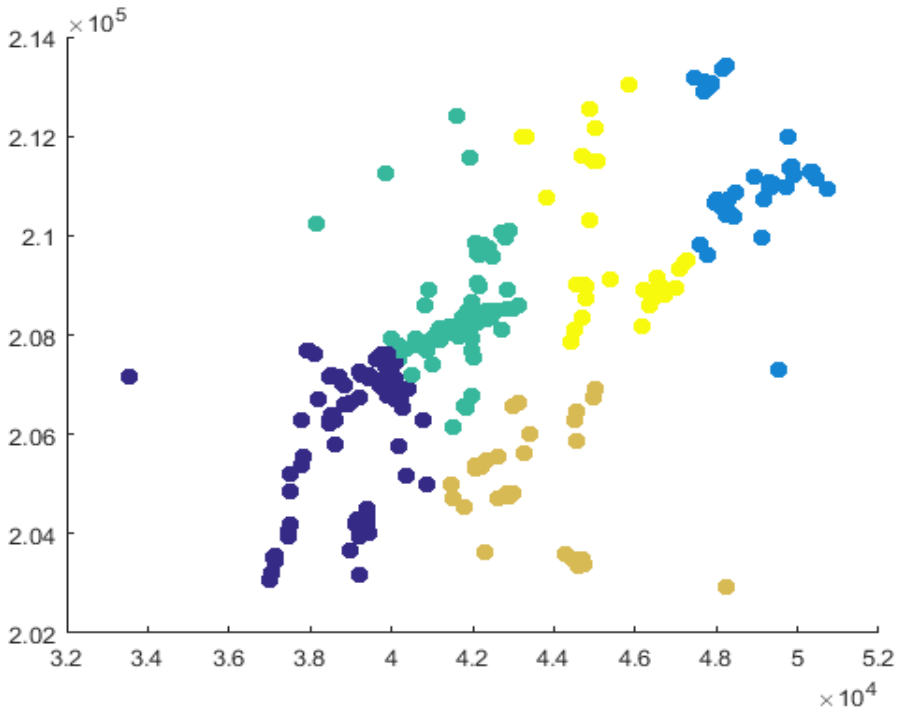


Figure 7. K-means partition result example

This figure gives an example of K-means result.

4.2.2.2 Implementation

Similar to the partitioning method 'grid', K-means might be also affected by outliers. So it is necessary that before applying this partitioning method to remove outliers at first. The removing outlier procedure is identical to the one we used for 'grid'.

As we discussed above, we will have a couple of number k prepared. And K-means will be running through all the numbers in order to find the best one. Depending on the final desired number of partitions, we use numbers from which goes up to 15. Besides, the results might be different for each time running K-means. In order to get a more reasonable result, we will run k means 5 times for each k value.

Among the 5 times of running K-means for each k value, we will judge the result by the smallest cluster size. The one with the largest size of smallest cluster will be chosen. In our system we wish all the servers can have a close number of check-ins.

And among different values of k, we will judge the results by the silhouette coefficient [12]. Silhouette coefficient considers both the cohesion and separation of the clusters and is often used as an evaluation of clustering. Silhouette coefficient ranges from -1 to 1, and a higher value corresponds to a better result.

For an element xi in the whole dataset, let $a(i)$ be the average distance between xi and all the elements in the same cluster. Find another cluster and calculate the average distance between xi and all the elements in it. Then iterating through all other clusters, let b(i) denote the lowest average distance. Silhouette coefficient is defined by the following equation.

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

And the mean silhouette coefficient of all elements is the coefficient for this cluster result.

So we will choose the number of clusters with the highest silhouette coefficient.

Here is the pseudo code for this procedure:

Partition - K-means	
1:	**Procedure** K-means_partition
2:	Cluster_using_K-means()
3:	centers = Get_centers_of_each_cluster()
4:	**For** check-in in check-ins
5:	Assign_to_closest_center(centers, check-in)
6:	**End**
7:	**End**

4.2.3 Hierarchy-XY

4.2.3.1 Basic idea

Hierarchy clustering is another method which we can use to generate clusters. As we discussed above K-means has its shortcomings, hierarchy clustering is introduced as an alternative approach to cluster spatial check-ins.

Different from K-means, hierarchy cluster will generate a tree structure for all the data [13]. In the tree every leaf is a data point, and the leaves are combined together depending on the distance between each other. The figure below shows an example of data being structured into a tree structure.

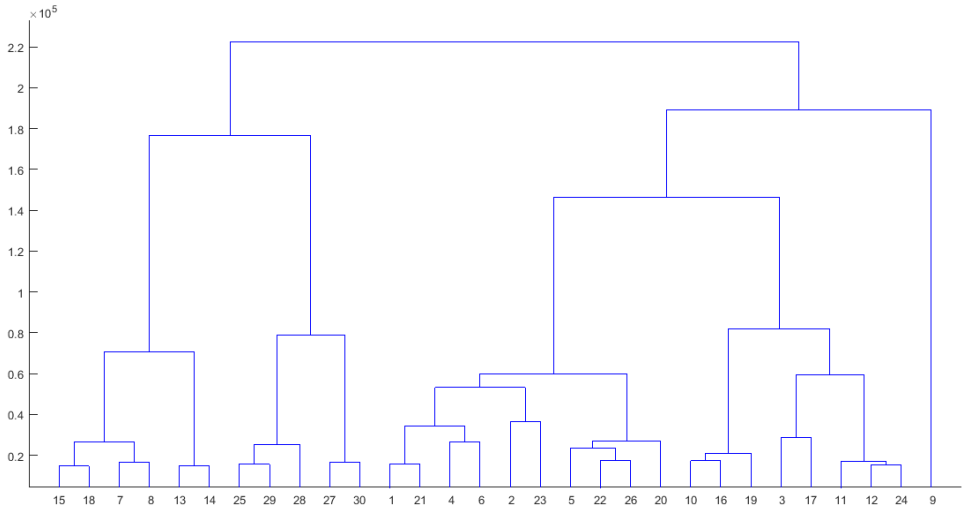


Figure 8. Hierarchy clustering tree graph

Instead of the random results from K-means, hierarchy clustering can always stably generate the same results for each running.

4.2.3.2 Implementation

The implementation is pretty similar to K-means, except the step of clustering. In MATLAB there many predefined functions which makes the approached much easier to implement.

Here is the code for generating hierarchy clusters:

```
D = pdist(data, 'euclid');
T = linkage(D, 'ward');
IDX = cluster(T, 'maxclust',NUM);
```

4.3 Combination of spatial and temporal dimensions

Time is one important factor for check-ins. Based on time, usually humans are moving in a certain pattern. For example someone always go to cafe in the afternoon and go to work during the day in weekdays. Time might be helpful for us to shape the user's movement pattern and we could do partition on the basis of it.

K-means XYH and spatial-tempo methods will be introduces in this subchapter.

4.3.1 K-means XYH

4.3.1.1 Basic idea

Since we could do clustering for spatial dimension, somehow combining time and spatial information is also one possible way to do clustering. The most natural way is to use k means to cluster based on the three dimensions X, Y and Time. And for the time we will use the time during a day which means the hour and minutes.

Since we have successfully applied K-means for X and Y dimensions before, the procedure should remain the same. But we have one issue here, time and spatial distances are totally different units. And when k-means is applied, every dimension will be considered equally.

So if we just use hour and XY as the input of the clustering method. Since X and Y are in the range of $10^4 - 10^5$ and time is just 0 - 24, the impact of time data will be almost ignored.

In order to solve this issue, we will bring another weight parameter to multiply with H to make it comparable with XY.

The figure below is an example of applying K-means for XY and time dimensions. Here we use a weight parameter of 500.

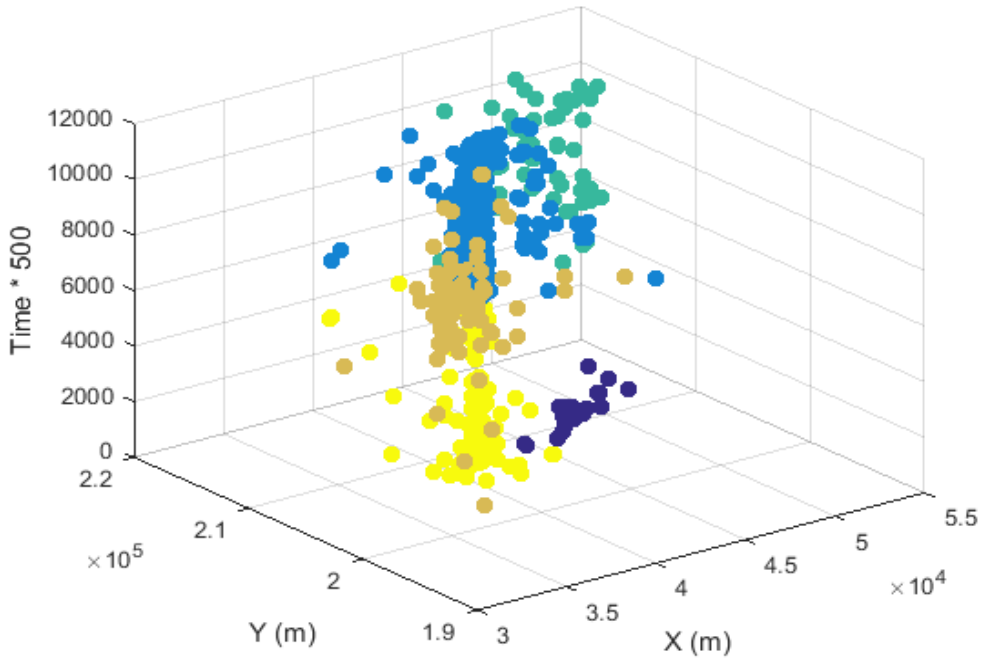


Figure 9. K-means XYH partition result

4.3.1.2 Implementation

So for this method we will reuse the code we implement for K-means X and Y. The only thing we need to change is to replace X and Y with X Y and time.

Here is the pseudo code:

Partition – K-meansXYH	
1:	**Procedure** K-meansXYH_partition
2:	Cluster_using_K-meansXYH()
3:	centers = Get_centers_of_each_cluster()
4:	**For** check-in in check-ins
5:	Assign_to_closest_center(centers, check-in)
6:	**End**
7:	**End**

4.3.2 Spatial-temporal

4.3.2.1 Basic idea

In reality, human movements usually form a certain pattern; especially the behaviors are periodic in term of time. Eunjoon Cho, Seth A. Myers and Jure Leskovec have proposed a method to predict user location based on geographic and temporal information [7].

In their paper, a 'Periodic Mobility Model' was proposed [7]. In this model, each check-in is labeled as one of the two state 'home' and 'work'. It is based on the assumption that a person's typical life pattern is mainly between work and home. For a certain time, one can predict the user is likely to be more near work or home.

Inspired by this model, we proposed a partition method. Similar to this model, the idea also has spatial and temporal components.

For spatial component, we will do clustering for the geographical data. And we try to fit a 2-D Gaussian distribution for each cluster.

For temporal component, time of check-ins in each cluster is used to form a temporal Gaussian distribution.

After getting a number of clusters and their spatial and temporal Gaussian distributions, we will calculate the probability of falling into a cluster for each particular check-in. And check-in will be assigned to the cluster, for which it has the largest probability.

4.3.2.2 Implementation

Clustering and Gaussian distributions will be done in the training parts.

For clustering, the training data we will at first cluster it using spatial information.

After clustering, mean of XY and the covariance matrix along with the mean and standard deviation of time for each cluster could be calculated, which will be used later to form spatial and temporal Gaussian distribution.

Because time we use here is 24-hour, which is a circular quantity. To find the mean of a group of time, we will convert 24-hour to 2π and use the following equation [14].

$$\underline{\alpha} = atan2(\sum_{j=1}^{n} \sin \alpha_j \ , \sum_{j=1}^{n} cos\ \alpha_j)$$

For the testing part we will use the parameters we got from training part to shape the Gaussian distribution.

For the calculation of the probability, a check-in belongs to a cluster, we will need 2 components. The first one is probability getting from spatial Gaussian distribution. And the second one is probability from temporal Gaussian distribution. For the probability of falling into a Gaussian distribution we will use cumulative distribution probability.

In the end we will combine those two parts together by multiplying. After comparing all the probability for each available cluster, the largest one is considered as the most likely cluster this check-in belongs to. And we will assign it to this cluster.

Here is the pseudo code:

```
Partition – spatial-temporal
1:    Procedure spatial_temporal_partition
2:      Cluster_using_hierarchy_clustering()
3:      Foreach cluster
4:        (XYmean, XYcov) = get_mean_cov_spatial()
5:        (Hmean, Hdev) = get_mean_deviation_temporal()
6:      End
7:      For check-in in check-ins
8:        Ps = Cdf(XYmean,XYcov,x,y)* Cdf(Hmean,Hdev,h)      //for each cluster
9:        Assign_to_biggest_P_in_Ps()
10:     End
11:   End
```

4.4 Combination of temporal and semantic dimensions: Eigenbehavior

As we mentioned in previous subchapter, usually human's activities follow a certain pattern. For example one might goes to work during the day in week days and goes to Cafe in the evening. And usually this pattern is periodic in time. In last subchapter, we have discussed the possible way of partition data based on spatial and temporal information. If we replace spatial info with semantic information, we could make new partition method as well.

So if we take temporal and semantic dimensions out of the three dimensions into consideration, we could try to find the semantic patterns in time inside the human activities. This might be better approach comparing with spatial-temporal. Because here instead of assuming a cluster of geographical locations represents a particular state, like home or office, here we really make use of the actual venues.

There was already a research in this field being carried out. Nathan Eagle and Alex Sandy Pentland have proposed a concept called Eigenbehaviors [8]. As the name indicates, probability inspired by eigenvector and eigenvalue in linear algebra, the concept of Eigenbehaviors is created to represent typical patterns in human's behaviors.

In machine learning we have a method principal components analysis. In principal component analysis (**PCA**), given a number of different data, **PCA** will try to find the principal components shared by different data.

So here we have the data for each day's activities for many users. And we got the daily check-ins and the check-in time and semantic information. So based on the information we have got, we can have a daily patterns for many different days for each user. If we apply principal component analysis to those different days' daily patterns, we will get the principal behavior pattern for 24 hours. Like one pattern could be: from 9 a.m. to 5 p.m. in the office and from 0 to 6 a.m. at home, which represents a typical weekday activity pattern.

Figure 10. Top 3 eigenbehaviors

This figure is one result after applying eigenbehaviors method. Here lists the top three eigenbehaviors. Each eigenbehaviors has 10 vectors and each vector is a venue's value distribution in 24 hours. In this example, color more close to yellow means it has higher probability. So the 1st eigenbehavior shows a pattern from 6:00 to 15:00 at work. 2nd eigenbehavior shows a pattern 15:00 to 24:00 at home. And 3rd one shows more relaxing pattern with 3:00 - 5:00 outdoor and 8:00 - 16:00 in shops. It is quite typical for daily activities.

4.4.1 Implementation of eigenbehavior

To be able to apply eigenbehavior method, we need check-in venues for each timeslot in a day. However, it is not possible in reality. User cannot have check-ins for each hour. For our data, in some previous work we applied Hidden Markov Models (HMMs) to predict the venue for each timeslot. Suppose a user has in total of d days of checkins. The data for each user is a $d \times 24$ matrix, with each row a 1×24 vector meaning the venues for 24 hours. Values in this matrix range from 1-10, which means the venue ID.

Since the numbers 1-10 are just IDs, 1 and 2 is not less different than 1 and 9. We will transfer the matrix into another $d \times 240$ matrix$B$. Each element $B(i,j)$ will be transferred to a 1×10 vector, for example $2 \Rightarrow [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$.

Now, Principal component analysis (PCA) will be applied to B, and a set of eigenbehaviors (EB) is available to use.

4.4.2 Eigen-behavior - linear combination

4.4.2.1 Basic idea

What we could get from this method is actually a set of eigenvectors and eigenvalues from our behavior matrix. Since they are eigenvectors and eigenvalues, to use them to determine the partitions, it is natural to do linear combinations.

For a number of k eigenbehaviors and user's actually behavior in the same length period of time, we will solve the following equation:

$$Behavior = \sum_{i=1}^{k} w_i \, EB_i$$

And the eigenbehavior with the largest weight w is considered as best representation of user's movements in this period of time. All the check-ins in this period of time will be grouped with this eigenbehavior.

However, since we have a lot of check-ins for each day, if we group the whole day into a group, it might cause inaccuracy. So our idea is cut a day into small segments, like 6 hours. Along with it we also generate the eigenbehaviors according to the segment size. Later in the evaluation part, we will try out for different segment size.

4.4.2.2 Implementation

In training part, similar to the implementation we introduce above but depending on the segment size, a set of eigenbehaviors will be generated.

Testing

Segment size could be chosen from $\{4,6,8,12,24\}$for the sake of easy calculation. For the simplicity of illustration, I will use segment size of 6 as an example in the following paragraphs.

For each segment, from the actual data we have some check-ins. However there are 2 issues causing troubles for direct applying the linear combination. Firstly, not every hour is filled by check-ins. Secondly; an hour might have more than one check-ins.

In case of only part of the segment has the data, we will remove the time slots without any data in it. In case of there are more than one check-ins in time slot, we will choose the one which appears most often as a representative venue. If some venue types appear the same amount of times, one of them will be chosen randomly.

As we discussed above, the eigenbehavior with largest weight will be chosen as the principal component for this segment. And the segment with the same principal eigenbehavior will be clustered together.

Here is the pseudo code:

```
Partition –EB_linear_combination
1:      Procedure EB_linear_combination_partition
2:          Get_eigenbehavior()
3:          Foreach segment
4:              Calculate_eigenvectors_weights()
5:              Assign_to_ev_with_biggest_weight()
6:          End
7:      End
```

4.4.3 Eigen-behavior - lookup table

4.4.3.1 Basic idea

In the previous method linear combination, sometimes we have more than one check-ins in a time slot. In some cases it is hard to select proper venue to represent semantics information for the time slot. Imagine the case for 3 time slots out of a segment we have 2 check-ins each. And in linear combination approach we have to randomly select one for the 3 time slots. Thus there are $2^3 = 8$ possibilities of outcomes, which bring unignorably uncertainty. Furthermore, in linear combination method we have to put everything in a certain range of period time into one partition. This might bring issues too.

As an alternative approach, we have another method called lookup table to partition all the data based on eigenbehaviors. In order to get rid of the 2 troubles, we will focus on single check-in instead of using linear combination for a block of check-ins.

Since each eigenbehavior is a 1×240 vector, containing value for all venues in all 24 hours. And the value in eigenbehavior is highly associated with probability. Firstly all the eigenbehaviors will be labeled with a partition ID.

Based on eigenbehaviors, we could generate a 10×24 lookup table LT for 24 hours and for all the 10 venues. For the particular time and venue, we will go through all the eigenbehavior we have, to find the eigenbehavior with the largest value. The labeled partition ID will be recorded in cells of lookup table. It means given a check-in with hour i and venue type j, the partition it should go to is $LT(i, j)$.

This figure shows one example of possible look up table.

In addition, we introduce the parameter segment size as well when we label the eigenbehaviors. The original 24 hours eigenbehavior could be further divided into, for example, 4 ×eigenbehaviors, each with 6 hours' information. We will go through different segment size in evaluation part.

4.4.3.2 Implementation

Training

We will use the same procedure as in the beginning of this chapter to generate eigenbehaviors. And then we will select a number (depending on number of partitions) of top eigenbehaviors.

Having a set of n Eigenbehaviors Γ:

$$\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_n\}, \Gamma_i \in 1 \times 240$$

For venue v and hour h, we will look the kth ($k = h \times 10 + v$) element in all eigenbehaviors in this time period to find $max(\Gamma_{m1}(k), \Gamma_{m2}(k), \ldots, \Gamma_{ms}(k)) \Rightarrow \Gamma_a$. Then record partition ID in lookup table: $LT(v, h) = a$.

Lookup table will be saved as training results.

Testing

Each check-in data has a check-in venue v and check-in time h. We will simply look up the value in the lookup table: $LT(v, h)$ and get the partition ID, to which this check-in should go to. So this way lookup table is used to accomplish partition.

Partition –EB_lookup_table	
1:	**Procedure** EB_ lookup_table _partition
2:	Get_eigenbehavior()
3:	Table = get_lookupTable()
4:	**Foreach** check-in in checkins
5:	Get_partition_ID_from_table()
6:	Assign_to_this_partition()
7:	**End**
8:	**End**

4.5 Semantic dimension

4.5.1 Direct semantic partition

4.5.1.1 Basic idea

As what we discussed in problem statement, we will evaluate of partition results based on the semantic information. So it will be a possible way to partition data based on purely semantic information. And it should be able to archive a very high long-term privacy.

Because we have a limited number of venues, we could directly partition by venue type. Check-ins with same venue type should be assigned to one partition. And then depending on how many partitions we want, combine partitions to reduce number of partitions.

4.5.1.2 Implementation

At first we cluster check-ins with same venue type together. Different partitions are equally different from each other. We will only consider the size of partitions and combine partitions with small size.

The combination is done in training part, for all the data in training dataset we go through the partitioning method as how we will do for testing data. Then compare the partition sizes and do combination. The combination is fixed then for testing procedure to use.

4.6 Optimization of partition algorithms

4.6.1 Motivation

For many methods we have implemented, it is not easy to directly generate the number of partitions we want. So it is necessary to have another procedure to process the intermediate results. Our idea is for those partition methods, which cannot directly produce the desired number of partitions, we generate more partitions than the desired number. And combine some of them together according to a certain standard to form new partitions. In this way arbitrary number of partitions could be achieved.

For example if we have 5 partitions and the desired number of partitions is 3. The partitions ID are

$\{[1], [2], [3], [4], [5]\}$.

And after optimization, it could look like this:

$\{[1,4], [3], [2,5]\}$.

4.6.2 Basic idea and implementation

For privacy our main concern is the long-term privacy of user profile. It requires each partition has a as different as possible venue distribution from the user's actual movements' venue distribution. So the principle of combining different partitions is based on the similarity of the venues distribution.

With 10 venue types, our venue distribution of each partition is a 1×10 vector. To compare how different it is for any pairs of vectors among all the partitions, we use cosine similarity to compare those vectors.

Cosine similarity is a measurement of the angle between a pair of vectors. For a pair of vector A and B. Cosine similarity is defined this way:

$$similarity = cos(\theta) = \frac{A \cdot B}{||A||||B||}$$

Besides, we have to take partition size into considerations. Ideally we wish check-ins could be separated relatively evenly in different servers. Therefore all servers could have balanced amount of check-ins stored. However, although it is hard to achieve, at least the situation that some servers have very few amount of check-ins should be avoided.

Therefore, another parameter small limit $sl, sl \in [0,1]$ is introduced. By multiplying with the total amount of check-ins c, a size limit is set. Any partition with smaller size than this size limit will be considered as too small and therefore should be merged with other partitions at first.

In this procedure, small size is taken as higher priority than cosine similarity.

This procedure is carried out before testing. The result from this optimization procedure is a set of partitions combinations. The testing data will still be assigned with partition ID before merging at first. And then with the combinations we get from this procedure, partition IDs will be rearranged.

Here is the pseudo code:

Bottom up approach

```
1:  Procedure Bottom up approach
2:      Loop (number_partitions – desired_num_partitions)
3:      If size(smallest_partition) < limit
4:          Merge_it_with_most_similar_partition()
5:      Else
6:          merge_2_most_similar_partitions()
7:      End
8:      return combination
9:  End
```

5 Evaluation

In this chapter, the evaluation of all partition methods we have proposed is introduced. At the beginning of this chapter, we demonstrate the social circle evaluation standards. Later on, each method will be evaluated in three aspects, namely, user profile privacy from LSs, privacy from social circle (oversharing), and utility with respect to social circles. Finally, an overall comparison of all methods is brought up. Moreover, discussion summarizing the partitioning quality will be made.

5.1 Earth mover distance

We have proposed earth mover distance as a measurement of user profile. And according to its definition, the value ranges from 0 to 2. Unlike utility and oversharing for social circles, it is a quantity one cannot have a direct judgement. Since we are about to carry out evaluations with this important quantity, we would like to give a visual impression of the earth mover distance.

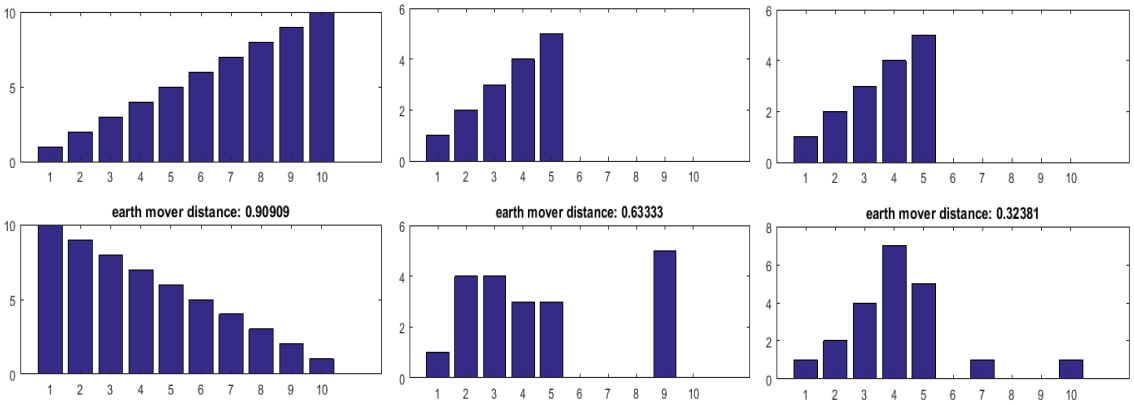


Figure 11. Earth mover distance examples

This figure shows three pairs of vectors with earth mover distance of 0.9, 0.6 and 0.3 from left to right. The 1st one is seen as totally different. The middle pair looks different but shares some similarities. And the last pair looks pretty similar to each other. Therefore, we could say earth mover distance of 0.3 is still considered as close, and above 0.6 could already be seen as different.

5.2 Social circle definitions

As we discussed in problem statement, besides long-term user profile privacy, utility should also be evaluated. And our evaluation is focusing on social circles.

Here the social circle we are referring to is typically the user's friends group in social media network. For different social circles, different types of check-ins should be shared. The utility here we mainly focus on the quantity of shared check-ins. The idea to evaluate is simple; we have a minimal requirement of sharing percentage for each venue type. And we will judge the actual sharing by comparing it with the requirements.

Since we have a minimal requirement for each venue, we also need to think about the case for oversharing. The information should be shared, but not too much. So another evaluation of oversharing is proposed as well. Similar to utility, we set a maximal percentage for each venue. The actual sharing will be evaluated in this aspect by comparing with the quantity of total shared data.

To set the minimal requirements and maximal sharing limits, we need to define the value for each venue type. And the definition could vary from person to person, which is quite subjective. Thus, we have searched for some related work [15] [16]. An experiment was carried out and how people react with sharing location with different acquaintances is shown [15]. We will use the data from this paper as an evaluation standard to go on with our work.

Here are the data for utility and oversharing in details:

Table 1 social circle utility requirements (%)

	Friend	Family	Colleague
leisure	75	58	55
profession	92	83	100
food	75	58	55
outdoor	75	58	55
medical	0	50	0
spiritual	0	50	0
residence	100	100	68
shop	75	58	55
transport	75	58	55
nightlife	75	58	55

Table 2 social circle oversharing requirements (%)

	Friend	Family	Colleague
leisure	100	84	77
profession	100	100	100
food	100	84	77
outdoor	100	84	77
medical	20	70	20
spiritual	20	100	20

residence	100	100	77
shop	100	84	77
transport	100	84	77
nightlife	100	84	77

In the paper there are two categories for working relationship: colleague and boss. For simplicity we will classify these two categories into one as colleague. So we have processed the data using the mean from boss and colleague.

Moreover, in the paper they classify everything except work and home into ‘other place’. Here we use the data for the rest of venue types as well, except for 2 more privacy sensitive venues medical and spiritual. The actual definitions for both venues could be found in the table.

Here is an example for colleague social circle:

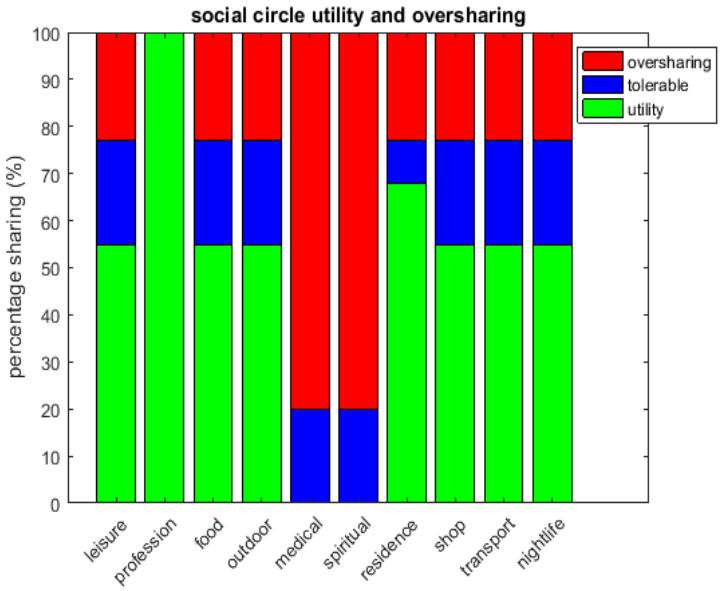


Figure 12. Utility and oversharing in social circles

This figure illustrate oversharing and utility for colleague social circle.

To judge the utility and oversharing, we will try to iterate all kinds of combination of servers and find one with the minimal distance to the perfect requirements distribution vector. And it will be used in oversharing evaluation as well.

5.3 Grid

5.3.1 User profile: earth mover distance

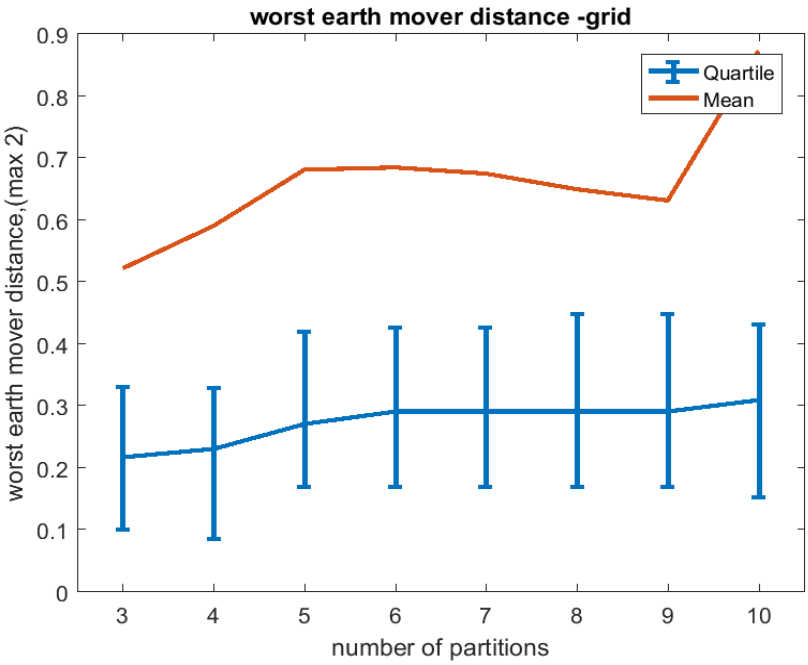


Figure 13. Worst earth mover distance – grid

Figure shows the long-term privacy evaluation for grid partitioning. One can see the tendency of either mean of worst case privacy is increasing along with the number of partitions growing. More number of partitions brings more privacy also meets our expectation.

The value of worst case privacy does not very so much difference between 0.2 and 0.3, which is generally not a good privacy protection result. So as a nearly random partition method, it is shown as bad as our expectation.

If we compare the two evaluation curve, difference is quite clear. The mean privacy has a slight going down from 5 to 9. It is due to our partitioning method. Because we always use a number of an integer's square to cutting the map. Therefore, we always use 9 grids for number of partitions from 5 to 9. And the later steps we combine the grids into the desired number of partitions. This explains the going down in the figure: the original grids data are same, but along with combination, some partitions with closer venue distribution shape are combined together. It leads to the reducing of average privacy.

But this does not have a strong impact on worst case privacy. Only if the closer distributed partitions happen to contain the worst case.

5.3.2 Social circle

5.3.2.1 Utility

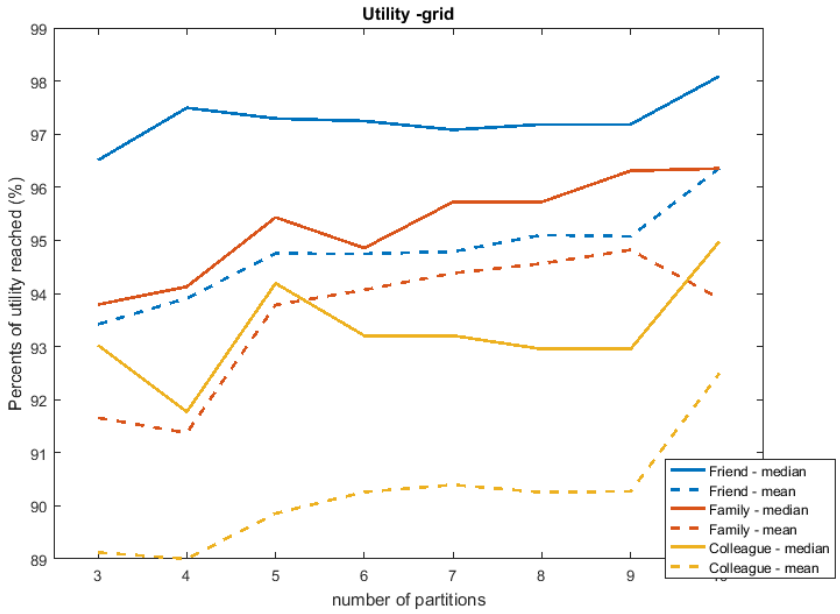


Figure 14. Utility evaluation – grid

This figure shows utility for social circles. And for each social circle we have drawn mean and median utility. Generally the utility is growing along with the number of partitions, which is reasonable because there are more possibilities of combinations for more partitions. Also friend generally has a better utility result, which is because we have higher sharing requirements for friend which makes it easier to find a proper combination. In contrast, colleague has relatively lower utility. We have a stricter rule for sharing with colleague, which makes it harder to find a combine close to requirements.

However, the results do look good for all the social circles. Even the lowest value for colleague, which is about 89%, is still acceptable.

5.3.2.2 Privacy: oversharing

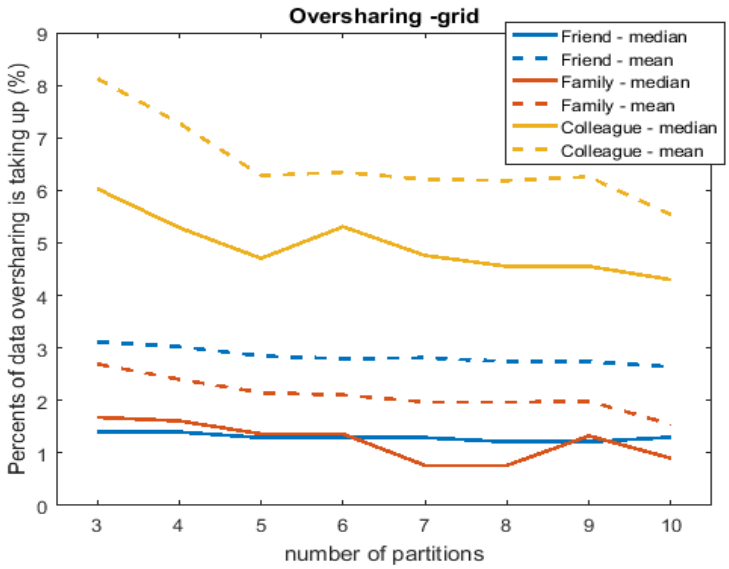


Figure 15. Oversharing evaluation - grid

This figure shows oversharing results. Similar to utility, generally more number of partitions leads to better results. And colleague has the worst behavior among those.

Still the results look acceptable. Only oversharing of colleague for small number of partitions (3 and 4) is a little bit high.

5.4 K-means

5.4.1 User profile: earth mover distance

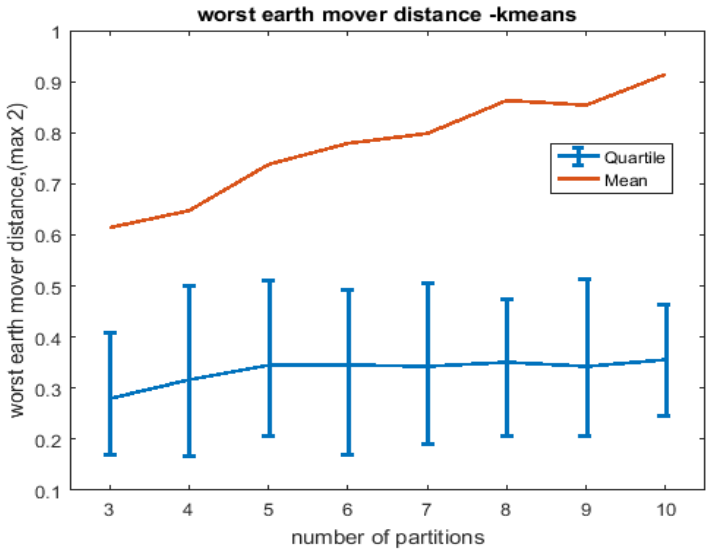


Figure 16. Worst earth mover distance - K-means

This figure shows long-term privacy evaluation for k-means method. Generally the result is getting better along with the number of partitions grows. The worst case earth mover distance does not change as much as mean earth mover distance.

The values here has slight enhancement comparing with grid method. But with an approximate range of [0.3, 0.35], the privacy is still under threat.

We cluster in geographical space; however, the venues could distribute not in the same manner as geographical locations. When we do clustering, a mixed group of different venues could be partitioned into a partition. This leads a low privacy result.

5.4.2 Social circle

5.4.2.1 Utility

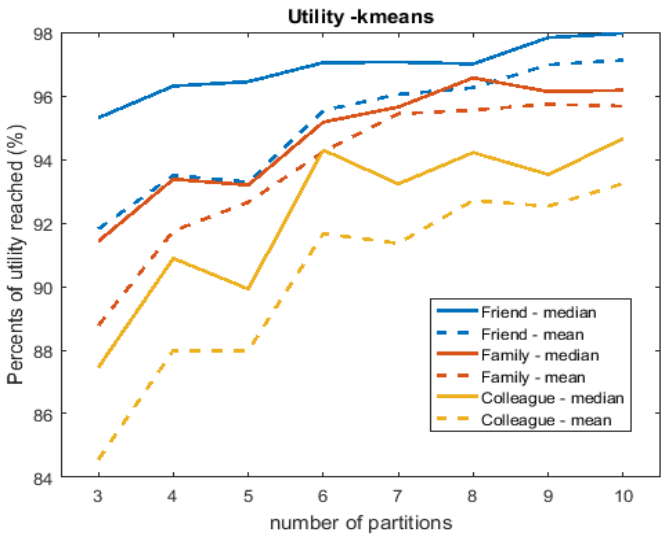


Figure 17. Utility evaluation - k-means

This figure shows utility for K-means method. The tendency is quite obvious, along with the growth of number of partitions, utility of K-means is increasing. And similar to previous method, friend performs the best and colleague the least best. Starting from number 6, the results look good.

5.4.2.2 Privacy: oversharing

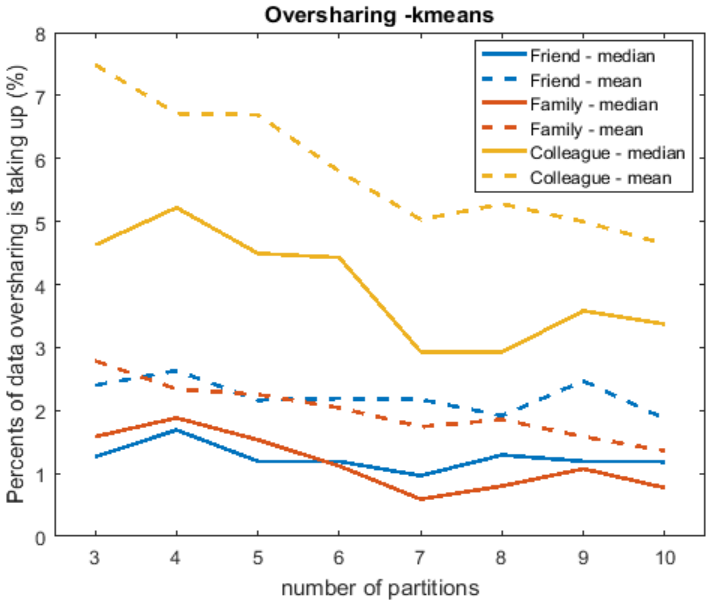


Figure 18. Oversharing evaluation - k-means

This figure shows oversharing for different number of partitions in K-means method. The highest oversharing is less than 8%. So we could say there is no problem of oversharing. Especially for friend and family, oversharing is always in a very low level (below 3%).

Oversharing is also decreasing along with the increasing of number of partitions.

5.5 Hierarchy

5.5.1 User profile: earth mover distance

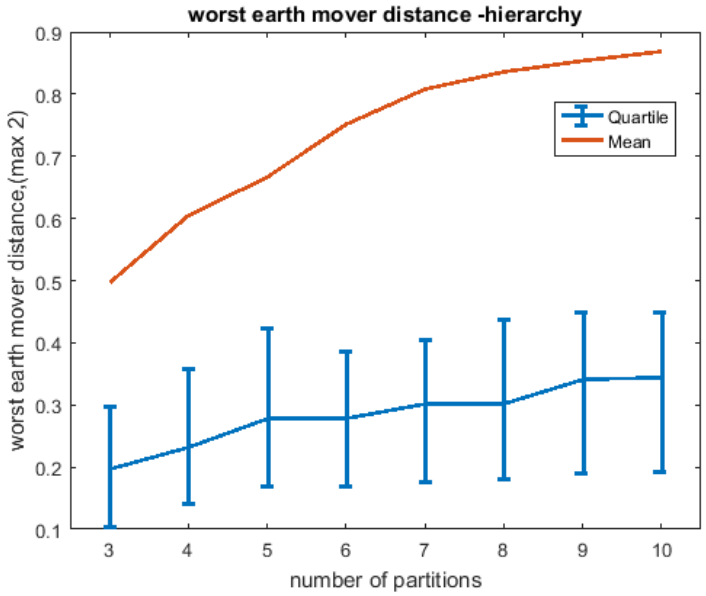


Figure 19. Worst earth mover distance – hierarchy

This figure shows earth mover distance for different number of partitions in method hierarchy cluster.

Worst case earth mover distance increases along with number of partitions growing from 3 to 10. And the worst case earth mover distance values are all in a rather low range (0.2 - 0.35), which means in the worst case privacy could be lost.

Average earth mover distance keeps increasing as more number of partitions applied. It is shown that our partitioning method is working to get better results.

5.5.2 Social circle

5.5.2.1 Utility

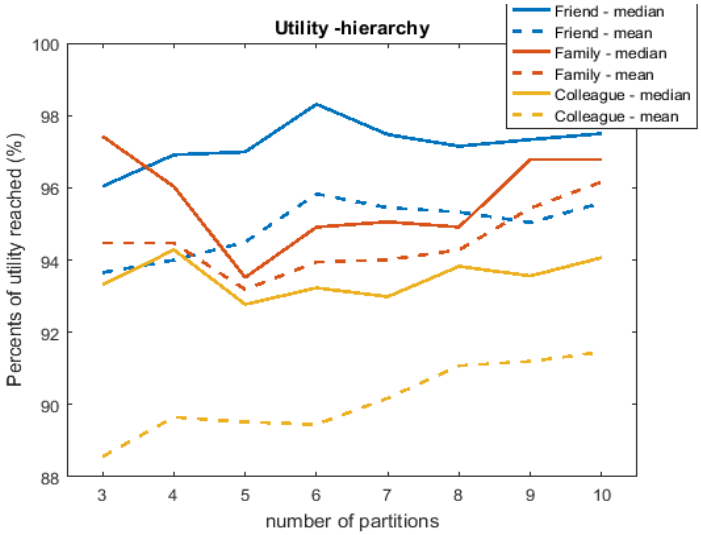


Figure 20. Utility evaluation - hierarchy

This figure shows the utility evaluation of hierarchy clustering method. Although the there are some drop downs, all the curves are in general increasing as the number of partitions grow. Because for our approach the user profile privacy is the most important, social circle evaluation is judged after the partitioning. So for some cases the median or average utility varies. For the social circles we try to find a combination of partitions which has the closest shape with utility requirements, it is individual for each evaluation. So the drop down is normal and acceptable.

What we really care is the actual value of utility. And in this case, except median colleague has relative low value, the rest are good. Medians utility of colleague is below 90% with 3-6 partitions. It is not sufficient, but later with the growth of partitions number, it reached near 92%, which is good. Another thing one can see is, average value is always higher than median value, which means there are some users getting a very high value in utility.

In summary, the utility requirements are met if partitions number is above 6.

5.5.2.2 Privacy: oversharing

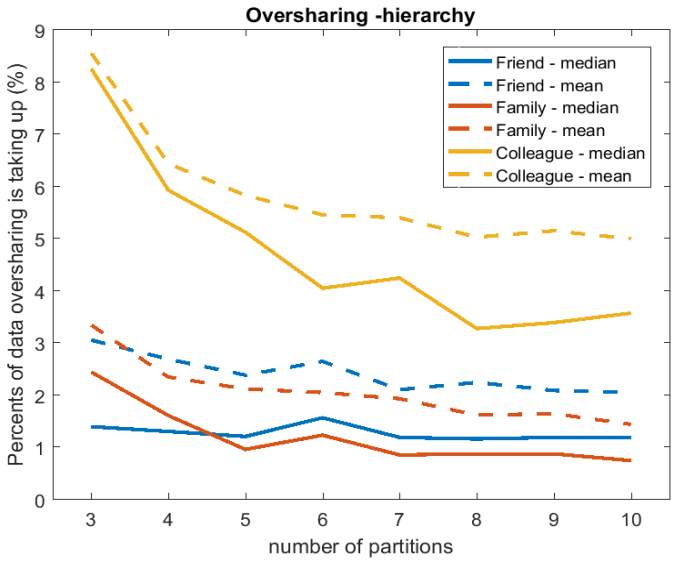


Figure 21. Oversharing evaluation – hierarchy

This figure shows the oversharing evaluation of hierarchy clustering method. It reflects the tendency that as partitions number grows, there will be less oversharing.

Friend and family social circles' oversharing is obviously better than colleague's. It is due to our stricter colleague sharing requirements. It is shown that friend and family's oversharing values are all below 3.5%, which is an excellent result. However, case of colleague is worse. When we have 3 partitions, the oversharing could reach close to 9%, which means there will be one oversharing for nearly every 10 location sharing. As the partition grows, if we have 4 partitions, both average and median are between 6% and 6.5%. It is a fair result. And when the number goes even more, colleague's oversharing is reducing and it is acceptable.

So in a word, the requirements of oversharing are met for 4 partitions and above.

5.6 K-meansXYH

K-meansXYH is pretty similar to K-meansXY, except for another added dimension of hours. As we discussed in introduction of this partitioning method, we will have to evaluate the weight parameters which is multiplying with hour at first. Because the original values in hour cannot compare with X and Y value in geographic space. We use 10 partitions to do this first evaluation.

After the evaluation of weight parameter, we will choose the one with best performance and use it to go on the evaluation for different number of partitions.

5.6.1 Weight parameter of time

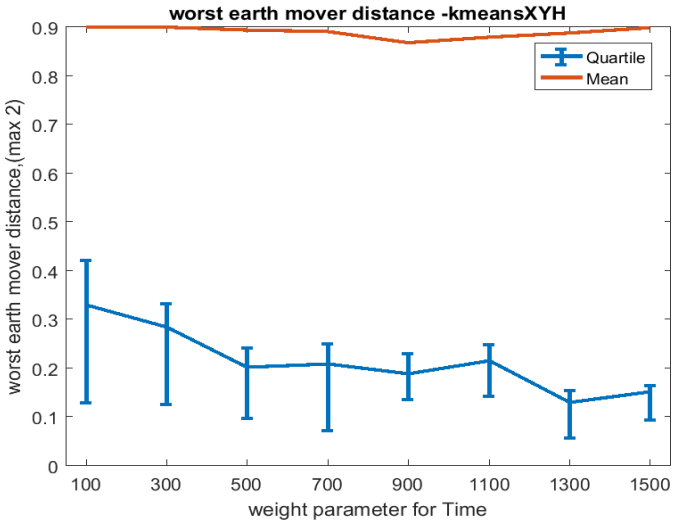


Figure 22. Worst earth mover distance - weight parameter - k-meansXYH

This figure shows earth mover distance for different weight parameters. The red curve is the median of average earth mover distance. Since we use 10 partitions, which is already a value with good performance in mean earth mover distance. It is proven in previous three partition methods' evaluation. So in this figure, the mean value does not change much and fluctuates around 0.9.

However, the worst case earth mover distance is decreasing along with the weight parameter's growth. So in the aspect of worst case earth mover distance, 100 turn out to be the best among those numbers.

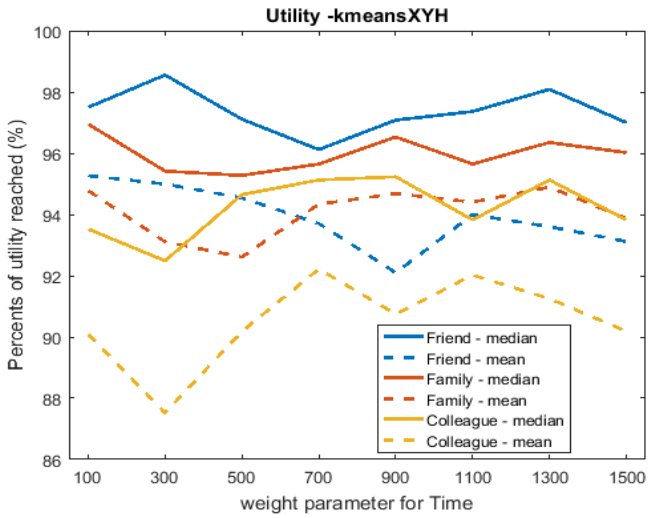


Figure 23. Utility evaluation - weight parameter - k-meansXYH

This figure shows utility requirements evaluation for different weight parameters. If we look it from a larger scale, the utility's value does not fluctuate much among different parameters. For the case 100, median colleague utility is about 90% and the rest are above or close to 94%. So we could conclude weight parameter with value 100 passes utility evaluation.

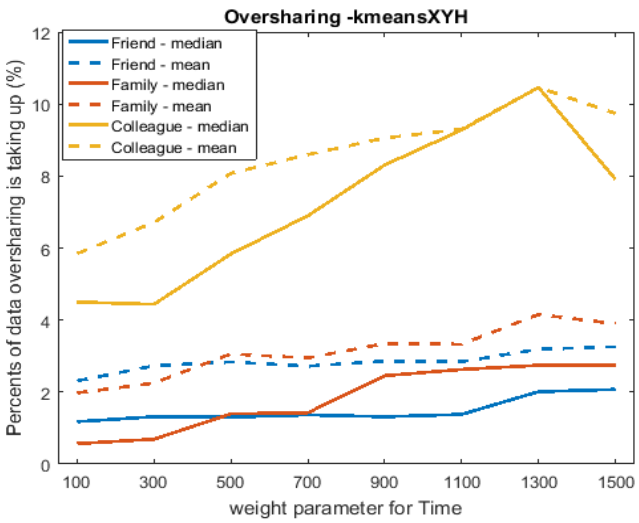


Figure 24. Oversharing evaluation - weight parameter - k-meansXYH

This figure shows the oversharing for different weight parameters. The oversharing is increasing with weight parameter growing. Since for oversharing lower value corresponds to better performance, the tendency here reveals smaller weight is better.

After reviewing the evaluation for 3 aspects, weight parameter of 100 is shown as the best parameter value among those. We will use it to multiply with hour and do the rest of evaluation in K-meansXYH.

5.6.2 User profile: earth mover distance

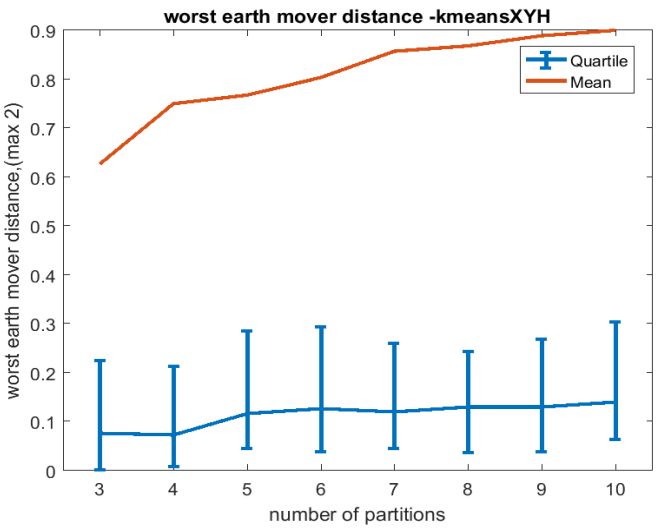


Figure 25. Worst earth mover distance - k-meansXYH

This figure shows the earth mover distance for different number of partitions. The trend is like previous method shows, more privacy are protected as more partitions are applied.

Apart from the trend, if we look into the worst case earth mover distance values, they are all in a rather low range. The highest worst case earth mover distance (with 10 partitions) is still just below 0.15. Which means this method is doing very badly at privacy protecting.

A possible reason for the bad behavior is: Time is a periodical variable. However, the way we use time in our clustering method does not reflect the periodicity of time. In reality between 11 p.m. and 1 a.m. there's only two-hour difference. But in our clustering method the time difference between 11 p.m. and 1 a.m. would be 22 hours. So the check-ins around midnight will have lower chance to be clustered together. And if that check-ins is close to each other in reality they should be clustered together.

5.6.3 Social circle

5.6.3.1 Utility

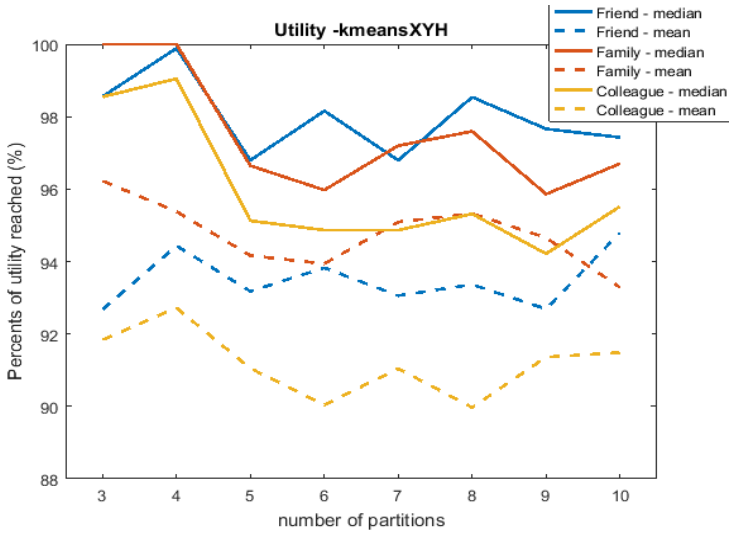


Figure 26. Utility evaluation - k-meansXYH

This figure shows utility evaluation results for different number of partitions. One can see the actual values fluctuate in a small range. Even in the lowest points (colleague's mean at 6 and 8 partitions) could reach 90%. So for utility evaluation, all the cases here meet the requirements.

5.6.3.2 Privacy: oversharing

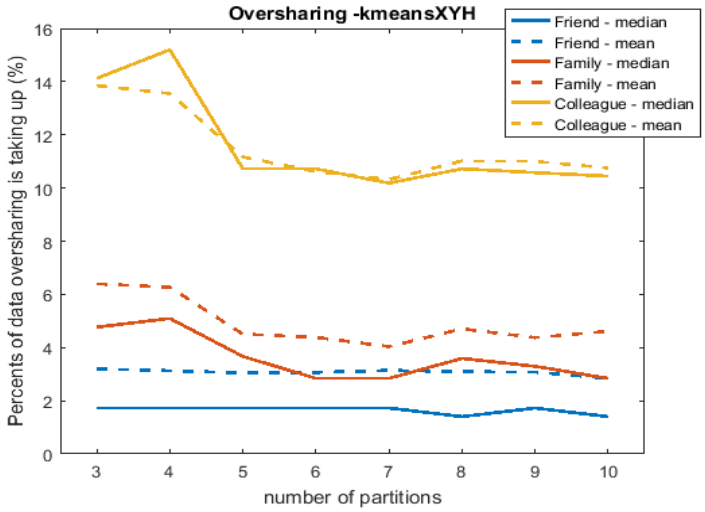


Figure 27. Oversharing evaluation - k-meansXYH

This figure shows the oversharing evaluation for K-meansXYH. The oversharing rate is going down if more partitions are available for all three social circles, which is expected.

Friend and family's oversharings are always in a rather low range. With 10 partitions, mean rate could reach 3% and 5%, and median could reach less than 2% and 3% respectively. This is a good performance.

However, for colleague social circle, even in best case oversharing rate is still above 10%, and in worst median of colleague's oversharing is close to 16%. It is an unacceptable result for all cases (1-10 partitions).

The requirements for oversharing are not fulfilled in all the cases for K-meansXYH method.

5.7 Spatial-temporal

5.7.1 User profile: earth mover distance

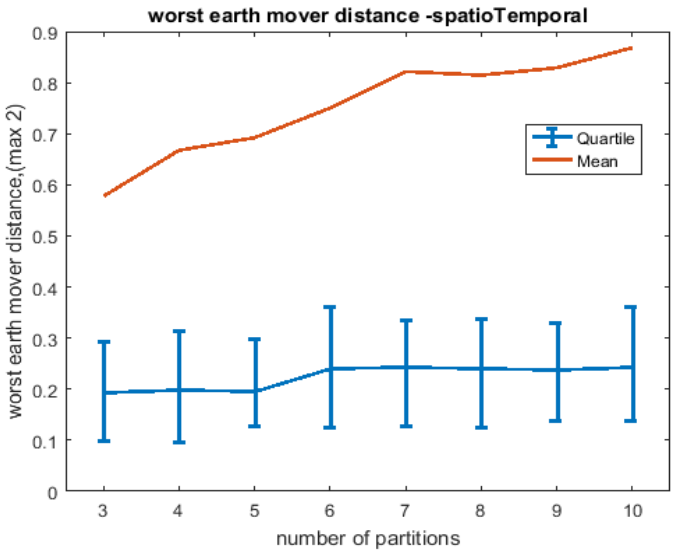


Figure 28. Worst earth mover distance - spatial-temporal

This figure shows earth mover distance evaluation results for spatial-temporal method. This figure shows a tendency of more privacy archived as number of partitions grows too. But the worst case earth mover distance does not have obvious incensement from 3 to 5 and 6 to 10 partitions. And all the values are in a very low level. The highest value of worst case earth mover distance is still below 0.25. In another word, privacy is not well protected.

We design this method by the inspiration of paper 'Friendship and Mobility' [7]. They have inferred 2 states home and work in their model. But in our case, depending on the number of partitions, we infer the same number of clusters. And we did it by hierarchy clustering. However, one can see from the evaluation results from hierarchy clustering method; this method cannot cluster user's movements of similar venue type properly. If so, the earth mover distance would be already in a very high level. Therefore, we do not have proper Gaussian distributions' centers as what they archived in [7]. Since check-ins with different venue types are clustered together, in temporal aspect, the distribution of time just reflects a mix of different venues instead of a pure venue's time distribution like home or work.

Another reason for this bad behavior could be our evaluation standard. We only consider venue distributions, and not care the geographic distribution for each partition. And in our data, there are 10 venues instead of just home and work. For example a check-in in food, transportation and nightlife category could also be at a place near home, and when they are classified into one partition, according to our evaluation standard, it is not a good partition result because it contains too much data.

5.7.2 Social circle

5.7.2.1 Utility

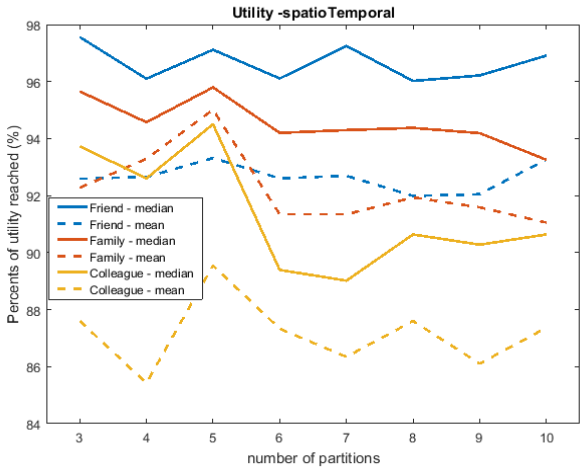


Figure 29. Utility evaluation - spatial-temporal

This figure shows evaluation results of utility for spatial-temporal method. Colleague's mean utility rate is under 88% most of the time, which is not really a good performance. For friend and family the results look good, because we have softer requirements for these two social circles comparing with colleague.

It is hard to say if the utility requirement is fulfilled. It will depend on how many percentage of utility is considered as sufficient. But we could at least conclude this method does not have a excellent performance in utility.

5.7.2.2 Privacy: oversharing

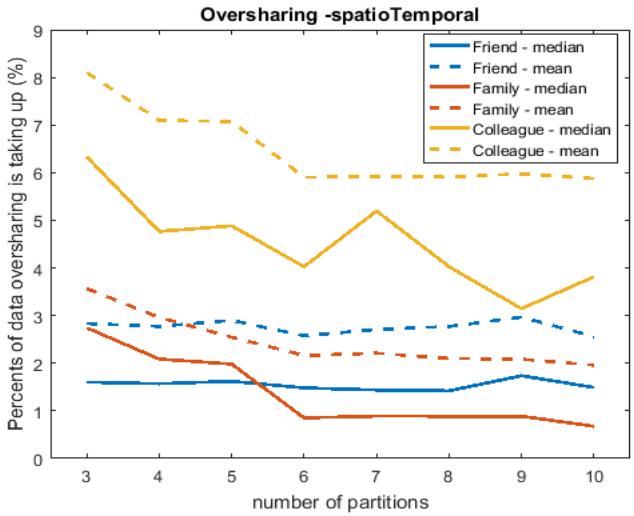


Figure 30. Oversharing evaluation - spatial-temporal

This figure shows the oversharing evaluation results for spatial-temporal method. The oversharing is going down as more partitions are used. Friend and family are both in a low level for all different

numbers. For colleague, the mean rate is about 8% when number of partitions is 3, which is high. But starting from 6 partitions, it is about 6% which is acceptable.

Overall, for spatial-temporal partitioning, because worst case earth mover distance has a very small value for all cases, it cannot provide good privacy protection. And its performance is not so good in utility as well. This method is not feasible.

5.8 Eigen-behavior - linear combination

In eigenbehavior evaluation, we will at first evaluate the performance for different segment sizes. In our implementation, a day could be cut into several segments like 4 hour or 6 hours. They might have impact on the performance. So we will do the evaluation and select the one with best results and use this value for the rest of evaluation. Here we use 5 numbers of partitions in segment evaluation.

5.8.1 Segment size

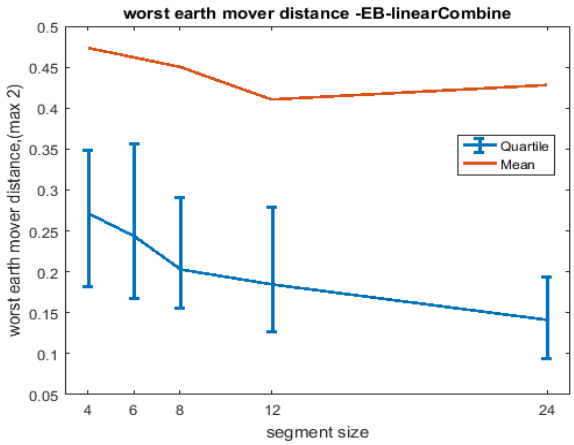


Figure 31. Worst earth mover distance - segment - EB-linearCombination

This figure shows segment evaluation in worst case earth mover distance. We have chosen 4, 6,8,12 and 24 for evaluating. As the figure shows, the worst earth mover distance is decreasing as the segment size increases. And so do mean earth mover distance, except there is a slight climbing up from 12 to 24. So according to this figure, we should use a segment size as small as possible.

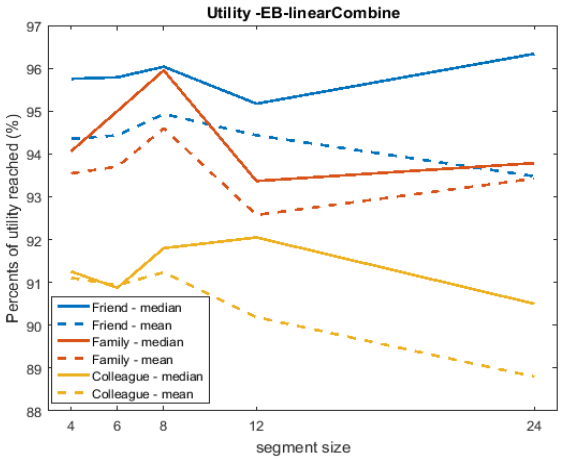


Figure 32. Utility evaluation - segment - - EB-linearCombination

This figure is the evaluation results for segment size in eigenbehavior linear combination. There is not really a tendency of performance for different segment size. If we only focus on segment size 4, as the earth mover distance evaluation suggests we should pick the smallest possible value. The utility for all three social circles are above 91%. And this is a good result. So segment size 4 passes utility requirements.

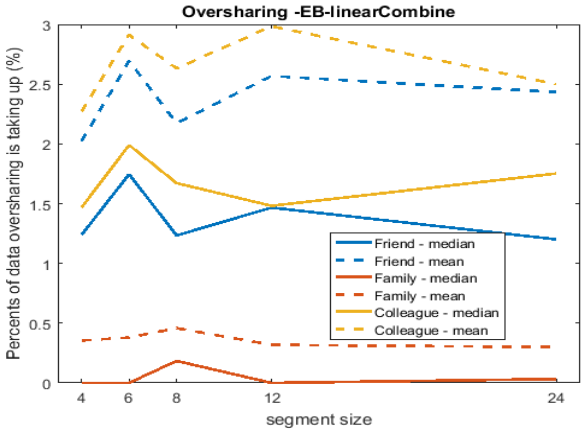


Figure 33. Oversharing evaluation - segment - EB-linearCombination

This figure shows oversharing evaluation results for different segment size. We could see a very good result for all possible situations. Especially for the case segment size 4, oversharing is under 2.5%.

After evaluation in three aspects, segment size 4 has the best privacy protection among all values and fulfills utility and oversharing requirements. We will use this value for the rest of evaluation.

5.8.2 User profile: earth mover distance

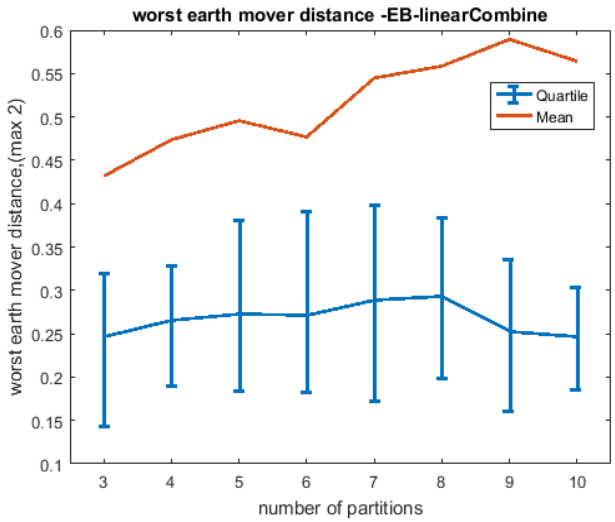


Figure 34. Worst earth mover distance - EB-linearCombination

This figure shows the earth mover distance evaluation results for different number of partitions. The privacy protection is growing with number of partitions until 8 partitions. And then it goes down from 8 partitions to 10 partitions.

Furthermore, the value of worst earth mover distance is fluctuating between 0.25 and 0.3, which is an unacceptably low range for earth mover distance. Besides, the average is also significant low comparing with previous methods we have evaluated.

In term of privacy protection, this eigenbehavior-linear combination is proven to be a very bad method.

After looking deep into the implementation and actual partition processes, we have found out three possible reasons for this bad behavior.

Firstly, there are some time slots with more than one check-ins. When we do the partitioning, we will select one from them in this case. We select based on the frequency, but often the venue types are of equal frequency. Thus we have to randomly select one from them as a representative. Imagine the situation that f we have 3 time slot in one segment with more than one check-ins, and each of them has 2 check-ins with different venues. We will have in total $2^3 = 8$ possible selections depending purely on randomness. And this is a common issue happening on many segments. So the partition results could be very random.

Secondly, after getting the representative venues for a segment, we solve an equation to find the eigenvector with largest weights. It happens very often that there is no unique solution for this equation because eigenvectors matrix' rank is deficient. So in the end we could only randomly assign them into one partition.

Finally, we always assign all check-ins in a segment together into one partition, which might bring issues too. Because different venues appear in a segment, when we do the partition based the equation's solution, they could only stand for those check-ins, whose venue type is selected as representative. Those check-ins without being selected should not but in our implementation are actually assigned to the partition too.

5.8.3 Social circle

5.8.3.1 Utility

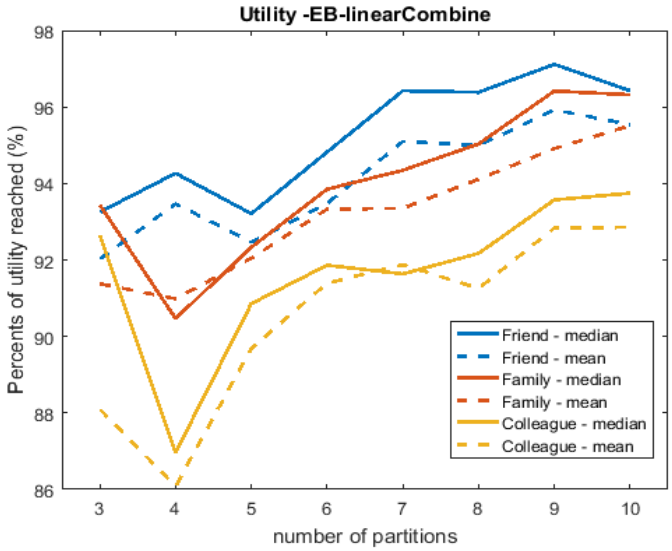


Figure 35. Utility evaluation - EB-linearCombination

This figure shows utility evaluation for eigenbehavior linear combination method. All the social circles' utilities grow along with the number of partitions' growth. Only exceptions are for 3 and 4 partitions, colleague has low rate of utility.

5.8.3.2 Privacy: oversharing

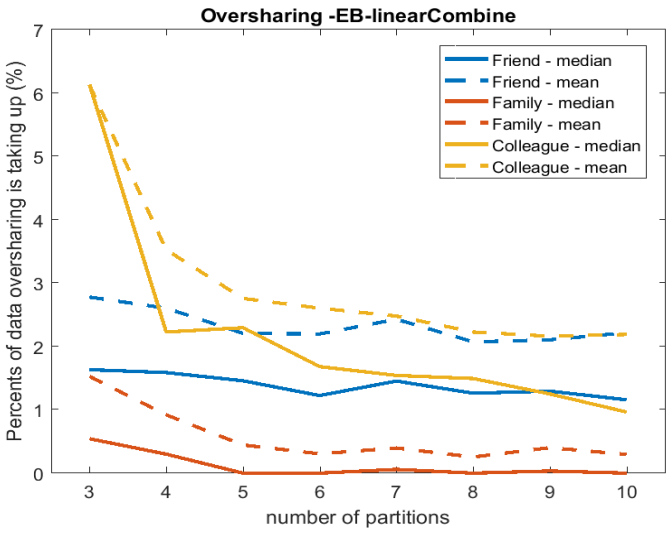


Figure 36. Oversharing evaluation - EB-linearCombination

This figure shows oversharing evaluation results for eigenbehavior linear combination. Oversharing is in general in a tendency of going down as more partitions are available. When number of partitions equals 3, colleague has 6% oversharing in both mean of median oversharing. But it is still acceptable. For the rest of rates, they are all below 4%. So there is no issue for this method in the aspect of oversharing.

Overall, this method has poor performance in user profile privacy protecting. But there is no issue with utility or oversharing.

5.9 Eigen-behavior - lookup table

Similar to eigenbehavior linear combination, we have a segment size parameter in this method. After the evaluation of segment, the best value will be selected to carry out number of partitions evaluation. After a number of evaluations for previous methods, number of partition of 5 does not have any obvious drawback and not too extreme as well. So this value will be used during the evaluation for different size of segment.

5.9.1 Segment size

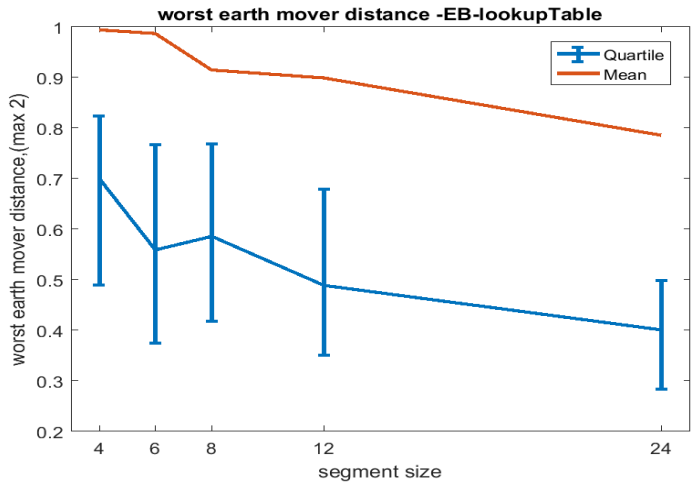


Figure 37. Worst earth mover distance - segment - EB-lookupTable

This figure shows the earth mover distance evaluation for different segment size. In general the performance of user profile protecting is decreasing as larger segment size is used. Size of 4 has the best value among those.

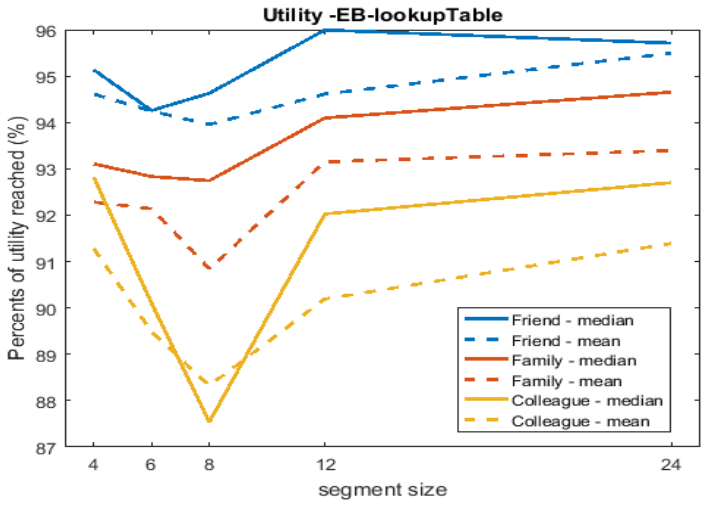


Figure 38. Utility evaluation - segment - EB-lookupTable

This figure shows the utility evaluation results for different segment sizes. There is a drop down for segment size 8. The lowest value median utility for colleague reaches below 88%. But except this, utility values for other segment size are all above 90%. Especially for segment 4 three social circles have 95%, 93% and 92% respectively. Utility requirements are fulfilled.

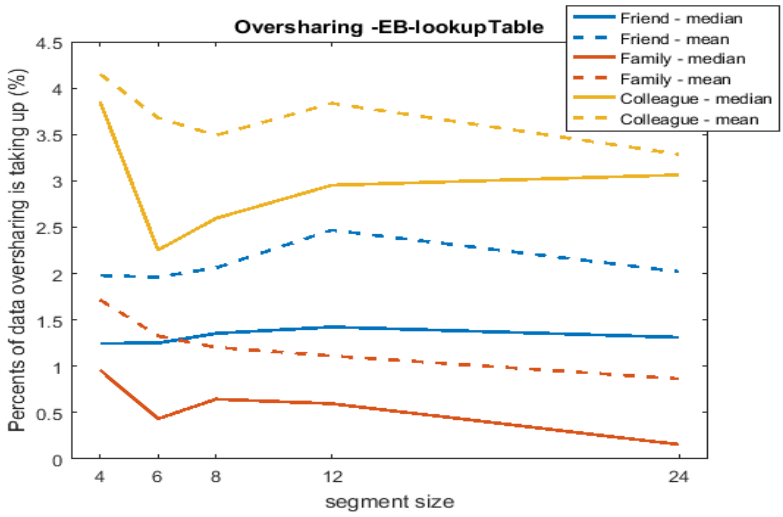


Figure 39. Oversharing evaluation - segment - EB-lookupTable

This figure shows the oversharing evaluation for different segment size in eigenbehavior lookup table method. All the rates are relatively small. The highest oversharing is just 4%. If we look at them from a larger scale, they also do not have much fluctuation. So for oversharing, all the segment sizes fulfill the requirements.

As a conclusion, in earth mover distance segment size 4 has an excellent result, and it also does not have any violations in utility and oversharing evaluations. We will use this value for segment size in following evaluation.

5.9.2 User profile: earth mover distance

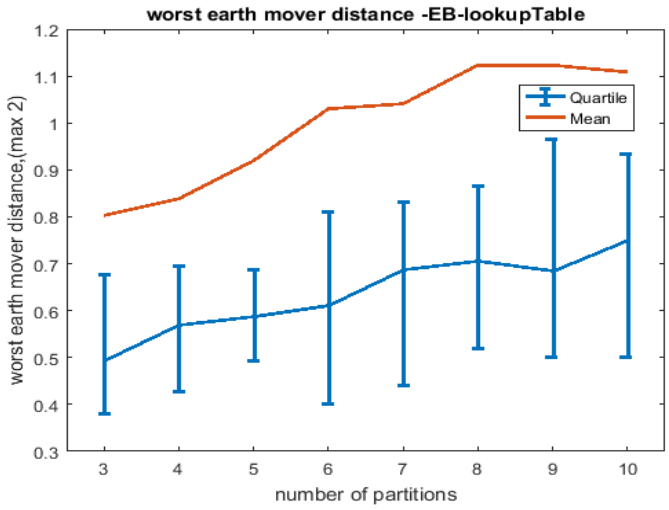


Figure 40. Worst earth mover distance - EB-lookupTable

This figure shows worst earth mover distance for different number of partitions. It also reflects the tendency that more privacy protection is archived along with more partitions is used. Notably, the earth mover distance values are significantly high here. With 3 partitions it is already 0.5 and when 10

partitions are used it is about 0.75. Like the examples we gave in the beginning of this chapter, if we say 0.6 is different and 0.9 is totally different, the worst earth mover distance is already something between ‘different’ and ‘totally different’. The mean earth mover distance can even reach 1.1.

In term of user profile privacy, this method has a very good performance.

5.9.3 Social circle

5.9.3.1 Utility

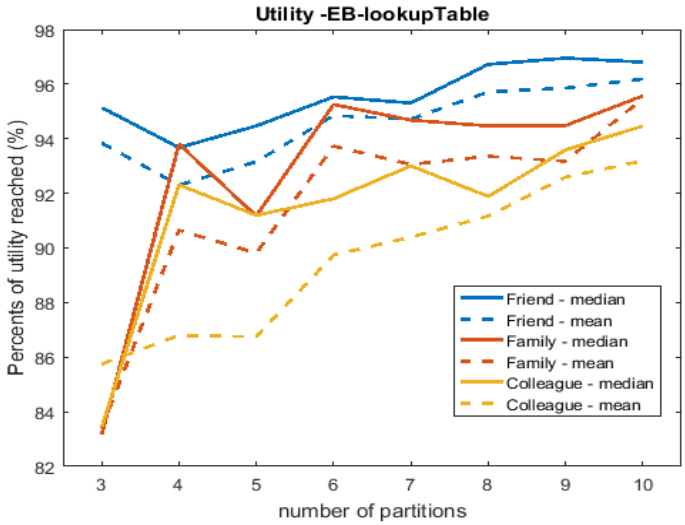


Figure 41. Utility evaluation - EB-lookupTable

This figure shows utility evaluation of eigenbehavior lookup table method for different number of partitions. The utility is increasing along with the incensement of partitions number. Friend’s utility always remains at a high level for different numbers of partitions. However, it is not the case for family and colleague. In the beginning when there are 3 partitions, family and colleague both have a low rate utility. Family mean and median are both about 83% and colleague has 86% in mean and 83% in median utility. Staring from 4 partitions, family’s two utility factors are staying above 90%, which we see it as sufficient.

Colleague’s mean utility values for 4 and 5 partitions are much lower than its mean utility, which means there are a small number of users with very low colleague utility. With a median utility of 92% and 91%, we could say for most of users the utility requirements could be fulfilled. And starting from 6 partitions, both mean of median colleague utilities are above 90%.

Overall, lookup table method has a poor performance in utility when we use 3 partitions. And for 4 and 5 partitions, in most cases utility requirements could be fulfilled, but in some cases it has a very low utility rate. And for 6 partitions and above, there is no issue of utility.

5.9.3.2 Privacy: oversharing

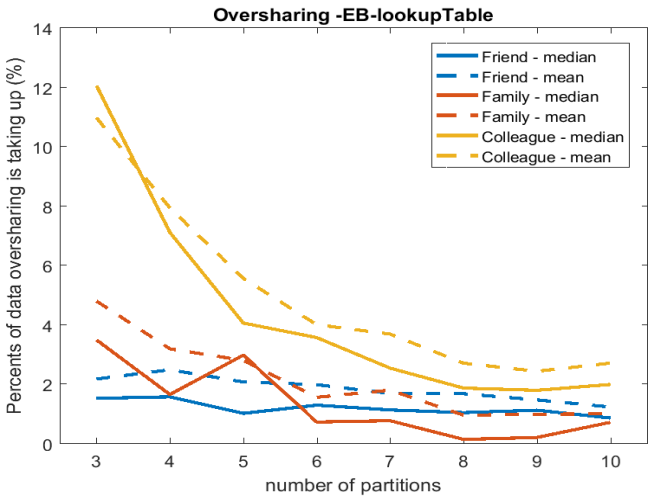


Figure 42. Oversharing evaluation - EB-lookupTable

This figure shows the oversharing evaluation for eigenbehavior lookup table method. In general, oversharing is decreasing as the number of partitions is increasing. For family and friend, oversharing rates are always in a low range below 5%. For colleague, it has 11-12% oversharing for 3 partitions and about 8% for 4 partitions, which is not so few. Staring from 5 partitions, oversharing rates are reduced below 6%. So in term of oversharing, eigenbehavior lookup table method has issues when there are 4 or fewer partitions. And for 5 and above partitions, all oversharing requirements could be fulfilled.

Overall, eigenbehavior lookup table method has excellent performance in user profile privacy protecting. But when partitions number is 3, it has very poor performance in oversharing and utility. When partitions number is 4 or 5, the utility and oversharing performances are fine, but also depending on how strict the standard is. For 6 partitions or above, this method has excellent performance without any utility and oversharing requirements issues.

5.10 Direct venue

5.10.1 User profile: earth mover distance

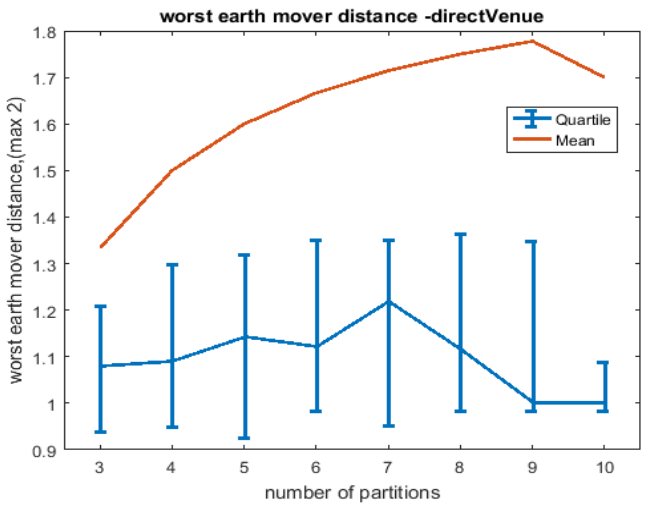


Figure 43. Worst earth mover distance – directVenue

This figure shows the earth mover distance evaluation results for direct venue method. It is quite clear this method has perfect performance in user profile privacy protecting. Even the lowest worst earth mover distance is about 1, which already means the two distributions look totally different. We only have 10 venue types, and all the partitions already have a large earth mover distance from origin venue distribution. Also due to the fact we calculate earth mover distance with normalized values, it is reasonable worst earth mover distance has drop down from 7 partitions and mean distance decreases from 9 to 10 partitions.

5.10.2 Social circle

5.10.2.1 Utility

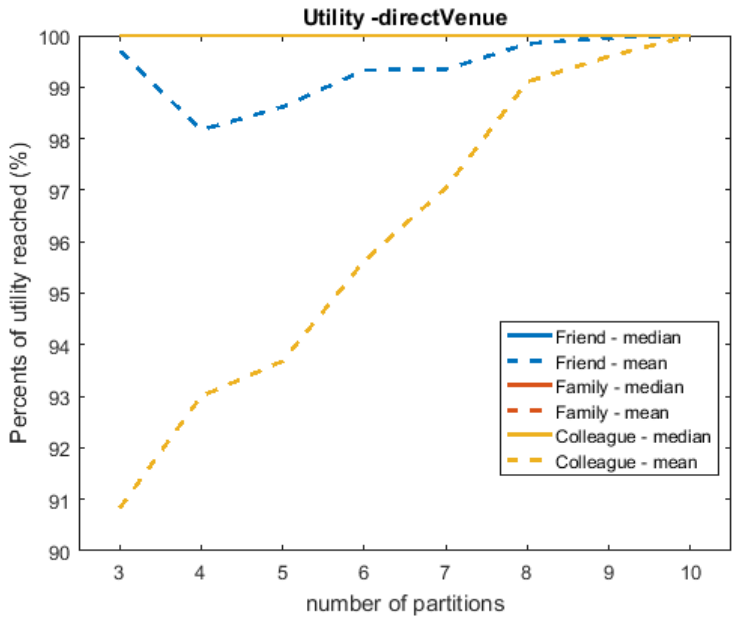


Figure 44. Utility evaluation – directVenue

This figure shows the utility evaluation results for direct venue method. All median utility rates are 100% and even the lowest mean colleague's utility is about 91%. It is an excellent result for direct venue.

5.10.2.2 Privacy: oversharing

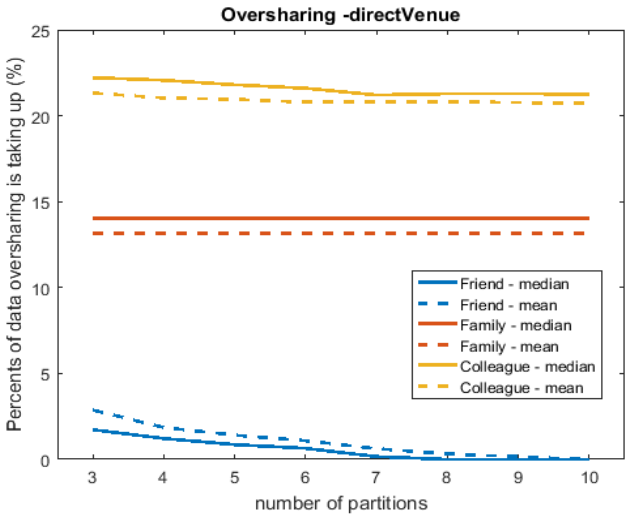


Figure 45. Oversharing evaluation – directVenue

This figure shows oversharing evaluation results for direct venue method. Colleague and family have a stable oversharing rate about 22% and 14%. Friend's oversharing remains below 3%. In term of oversharing, this method has an extremely poor performance.

Because in this method, check-ins with same venue type will always be sent to a same partition, it is very inflexible when we try to find the combination of partitions for social circles. For every venue type, the social circle will either include or exclude all check-ins. And it brings oversharing.

Overall, for direct venue method, even though it has nearly perfect performance in user profile privacy protection and utility, due to the draw back in oversharing, it is not a feasible partitioning method.

5.11 Overall Evaluation

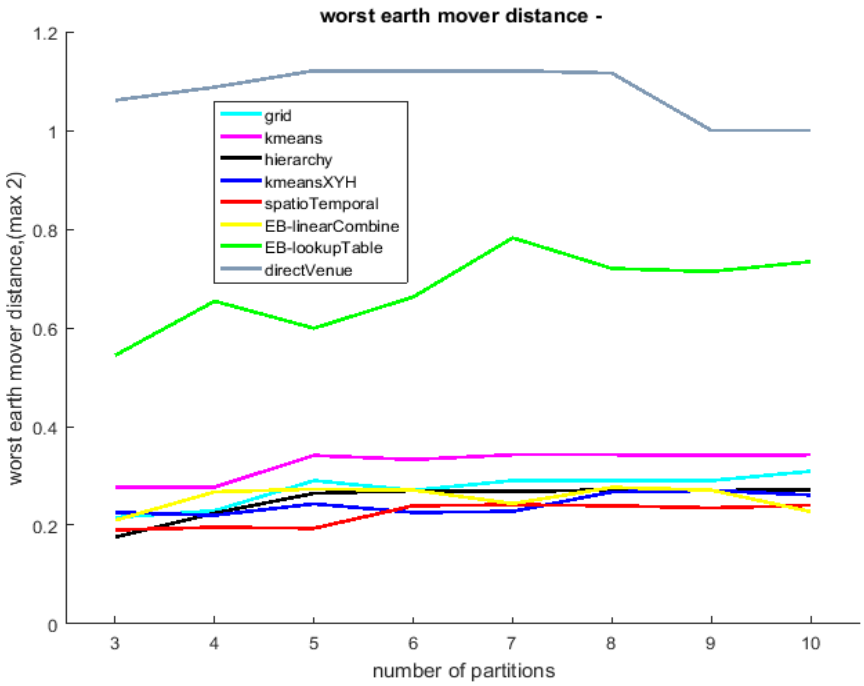


Figure 46. Worst earth mover distance - all methods

This figure shows worst earth mover distance for all methods. Among all partition method we proposed, direct venue no doubt has the best worst earth mover distance. Apart from direct venue, eigenbehavior-lookup table is significantly better than the rest. The rest methods are in a similar level in term of user profile privacy protecting, among which K-means is slightly better. But K-means' best distance is still about 0.35, which is seen as pretty similar to origin venue distribution.

In direct venue method we partition check-ins only by venue type. It is expected that partition's venue distributions from this method have the least similarity with origin venue distribution. In eigenbehavior-lookup table it is essentially partitioning check-ins based on venue type as well, but different venues could be mixed in each partition. So it also has a pretty high value is earth mover distance, but not as high as direct-venue's.

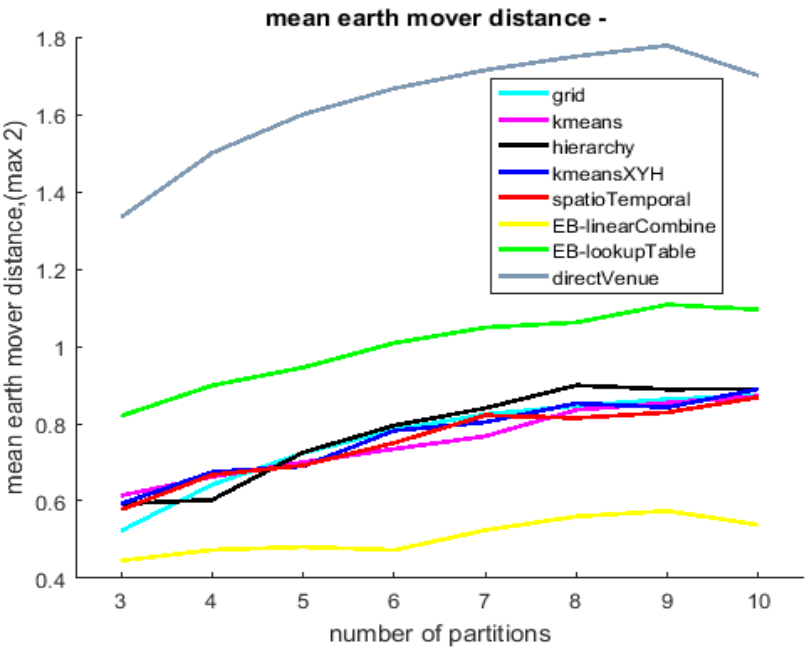


Figure 47. Average earth mover distance - all methods

This figure shows average earth mover distance for all methods. We could classify them according to performance into 4 levels. Direct venue and eigenbehavior-lookup table take 1st and 2nd level respectively. They are obviously better in this aspect. The rest of methods except eigenbehavior-linear combination are in the 3rd level and eigenbehavior-linear combination alone has the worst performance in average earth mover distance.

Similar to worst earth mover distance, it is expected that from direct venue, eigenbehavior-lookup table to those in 3rd group earth mover distance values are reducing. However, different from last figure, eigenbehavior linear combination has significantly low value comparing with other methods.

As what we discussed in its evaluation, due to many reasons this method in many cases works like totally random partitioning. While for other methods, for example grid, there would be some partitions with low amount of check-ins assigned, the venue distribution's earth mover distance with origin data could vary, so the average earth mover distance is higher.

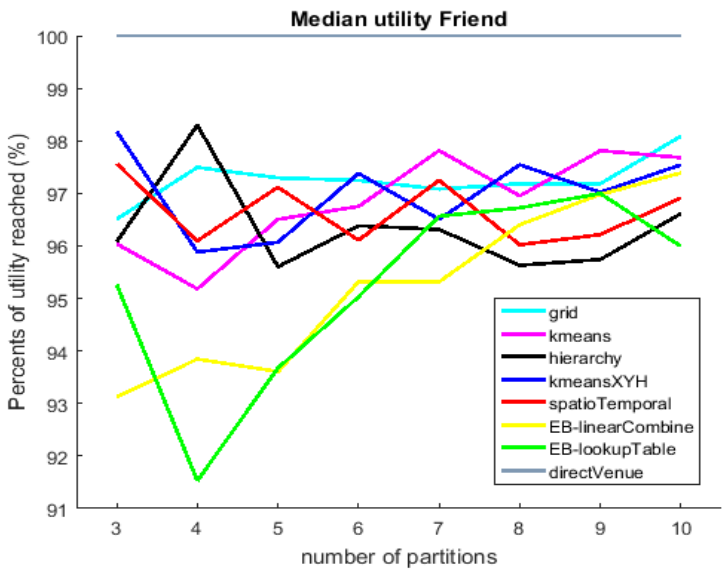


Figure 48. Utility evaluation friend - all methods

This figure shows median utility of friend social circle for all partitioning methods. All values in this figure are at least about 92%, there is no issue with friend circle's utility.

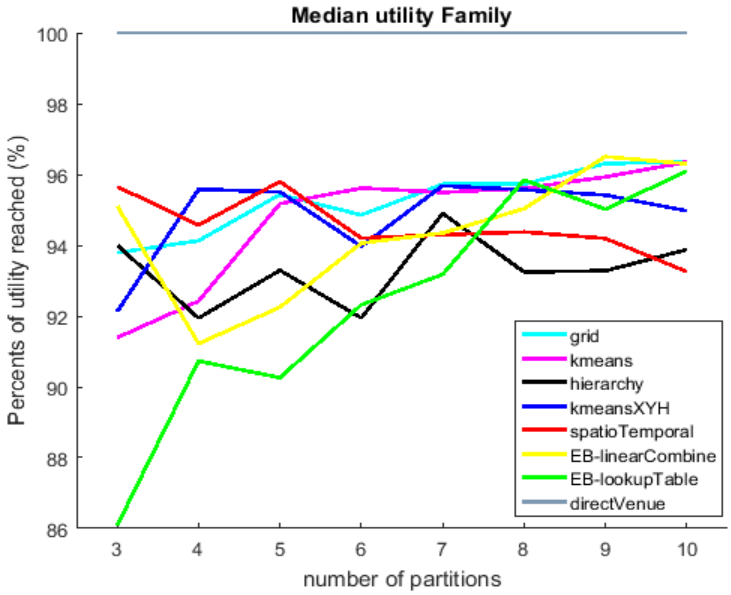


Figure 49. Utility evaluation family - all methods

This figure shows median utility of family social circle for all partitioning methods. If we consider 90% is the minimal requirement score, all the partitioning methods expect eigenbehavior-lookup table when partitions number is 3 satisfy the requirement.

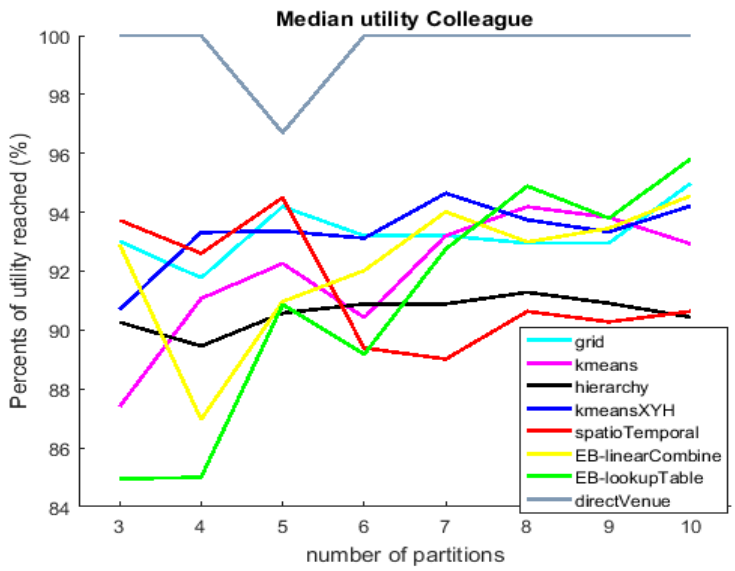


Figure 50. Utility evaluation colleague - all methods

This figure shows median utility of colleague social circle for all partitioning methods. Except a few points, all values here are above 90%. The only exceptions are eigenbehavior-lookup table for 3 and 4 partitions, eigenbehavior-linear combination for 4 partitions and K-means for 3 partitions.

Overall, for utility's evaluation, only when number of partitions is lower or equal to 4, some partitioning methods have a low utility rate. And even in those cases when utility rate is low, the value is still at least 85%, depending on the requirements, it might be acceptable. In addition, maybe a better solution of finding combination method could be proposed; the performance could be further enhanced. In general, there is no big issue with utility.

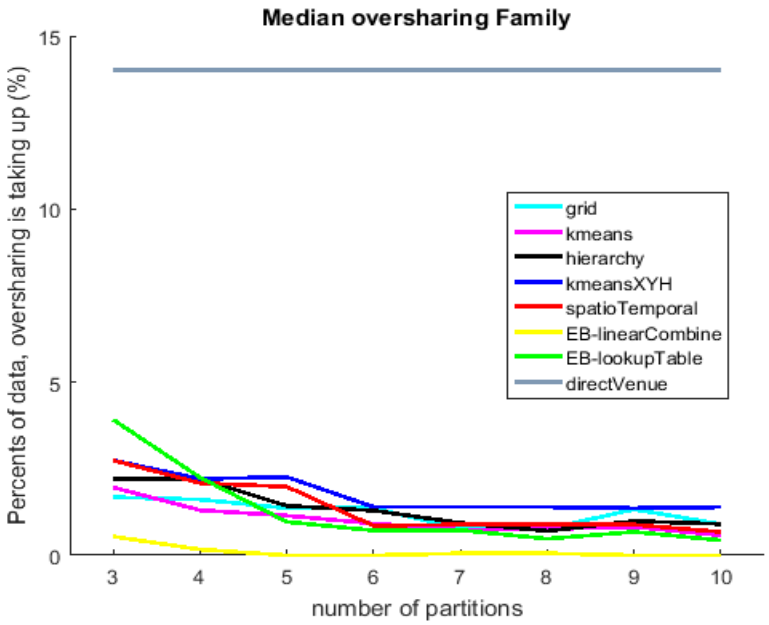


Figure 51. Oversharing evaluation friend - all methods

This figure shows median utility of family social circle for all partitioning methods. Direct venue has a significantly high oversharing rate for all numbers of partitions, which is about 14%. Expect direct venue, all other methods over share very little. Most of them are below 3%.

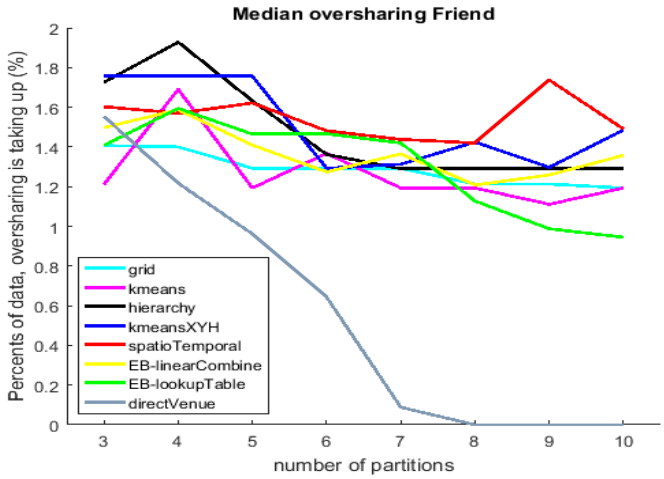


Figure 52. Oversharing evaluation family - all methods

This figure shows median utility of friend social circle for all partitioning methods. All of the oversharing rates are below 2%, which is an excellent performance. But since our friend's oversharing requirements is very loose, it is not beyond our expectation.

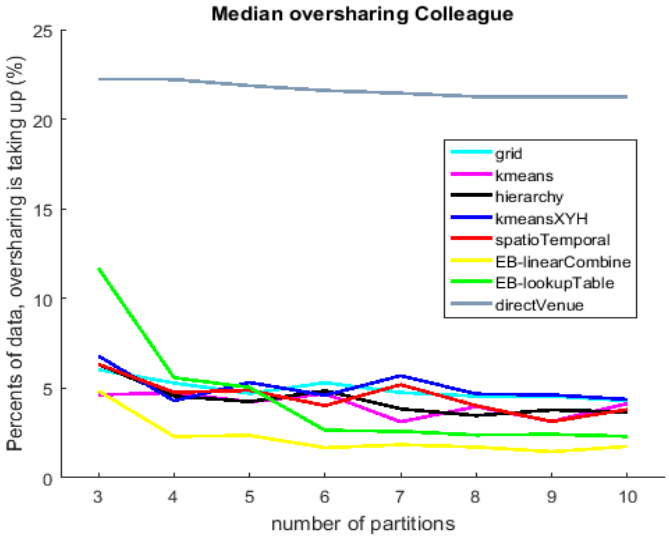


Figure 53. Oversharing evaluation colleague - all methods

This figure shows median utility of colleague social circle for all partitioning methods. Direct venue still has significantly high oversharing rates for all numbers of partitions, which are fluctuating between 23% and 21%. Expect direct venue, when partitions number is 3, eigenbehavior-lookup table also has a high oversharing rate of about 12%. However, oversharing rate is very little for all other situations.

Overall, for oversharing requirements, direct venue has a very poor performance. We have discussed the reason in direct venue's evaluation part. For the rest of methods, only for 3 partitions eigenbehavior-lookup table shows too much oversharing. Oversharing rates in all other situations fulfill the requirements.

In summary, after evaluating all the methods we have implemented. Direct venue method has good performance in user profile privacy protection, but meanwhile oversharing is too high. Eigenbehavior-lookup table can protect user profile privacy and also fulfills utility and oversharing requirements if more than 3 partitions are used. If there are 3 partitions, this method is proven to have a high oversharing rate. For the rest of methods, due to the poor performance in user profile privacy protection, none of them is capable for our system.

Generally, Eigenbehavior-lookup table has the best performance.

6 Summary and future work

The development of mobile device and service brings a challenge of user privacy protection. In order to protect user's location privacy, a system model in which location data are stored in some additional servers instead of non-trustable third party applications is proposed. In this thesis based on the system model, we have researched the possibility of privacy protection through data partitioning. For the generated data from mobile devices, we will apply a partitioning method to distribute data into different servers. Third party applications will get accesses of only selected servers. Since the servers are not trustable as well, we have to ensure user profile privacy protection in each server. Besides, we have to make sure social circle's utility requirements could be reached and meanwhile not too much data more than necessary would be shared.

We have proposed and also implemented 8 partitioning methods. After evaluating them in terms of user profile privacy protection, social circles' utility and oversharing, the method eigenbehavior-lookup table turns out is the only capable method in our system.

Moreover, there are still rooms for improvements.

In our evaluation, the best method eigenbehavior-lookup table still has an oversharing issue when partitions number is 3. More optimization could be done in the future.

We have introduced three dimensions of information, spatial, temporal and semantic respectively, in the beginning. Partitioning methods with other kinds of dimensions' combination, such as combining all three dimensions, could be explored.

We have only considered long-term user profile privacy in this paper. However, short-term privacy, such as preventing real-time location prediction, should be evaluated too. We leave this for future work.

As we discussed in spatial-temporal evaluation part (chapter 5.7), the evaluation standards for user profile privacy needs improvements too. The current evaluation only considers venue distributions, a more comprehensive evaluation which could consider spatial distribution as well would be better.

7 Bibliography

[1] Golmack, S. (2017, 02). Retrieved from https://www.smashingmagazine.com/2017/02/current-trends-future-prospects-mobile-app-market

[2] Bosomworth, D. (2015). Mobile marketing statistics 2015. Leeds: Smart Insights (Marketing Intelligence) Ltd.

[3] www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks

[4] Kapadia, A., & Tsang, P. P. (2008). Social circles: Tackling privacy in social networks.

[5] Riaz, Z., Dürr, F., & Rothermel, K. (2015). Optimized Location Update Protocols for Secure and Efficient Position Sharing. International Conference and Workshops on Networked Systems (NetSys).

[6] Riaz, Z., Dürr, F., & Rothermel, K. (2016). On the Privacy of Frequently Visited User Locations. Proceedings of the 17th International Conference on Mobile Data Management:MDM'16. Porto, Portugal.

[7] Cho, E., Myers, S. A., & Leskovec, J. (2011). Friendship and Mobility. Proceeding of the 17th ACMSIGKDD International COnderence on Knowledge Discovery and Data Mining. San Diego,California,USA.

[8] Eagle, N., & Pentland, A. S. (2009). Eigenbehaviors: identifying structure in routine. Behavioral Ecology and Sociobiology, pp. 63:1057-1066.

[9] Wernke, M., Skvortsov, P., Dürr, F., & Rothermel, K. (2014). A classification of location privacy attacks and approaches. Personal and Ubiquitous Computing, 18(1), 163-175.

[10] Dürr, F., Skvortsov, P., & Rothermel, K. (2011). Position sharing for location privacy in non-trusted systems. Proc. of the IEEE International Conference on Pervasive Computing and Communications (PerCom), (pp. 189-196).

[11] Milgram, S., & van Gasteren, L. (1974). Das Milgram-Experiment. Reinbek: Rowohlt.

[12] Aranganayagi, S., & Thangavel, K. (2007, December). Clustering categorical data using silhouette coefficient as a relocating measure. In Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on (Vol. 2, pp. 13-17). IEEE.

[13] Corpet, F. (1988). Multiple sequence alignment with hierarchical clustering. Nucleic acids research, 16(22), 10881-10890.

[14] Kurz, G., Gilitschenski, I., & Hanebeck, U. D. (2013, June). Recursive nonlinear filtering for angular data based on circular distributions. In American Control Conference (ACC), 2013 (pp. 5439-5445). IEEE.

[15] Wagner, D., Lopez, M., Doria, A., Pavlyshak, I., Kostakos, V., Oakley, I., & Spiliotopoulos, T. (2010, September). Hide and seek: location sharing practices with social media. In Proceedings of the 12th international conference on Human computer interaction with mobile devices and services (pp. 55-58). ACM.

[16] Consolvo, S., Smith, I. E., Matthews, T., LaMarca, A., Tabert, J., & Powledge, P. (2005, April). Location disclosure to social relations: why, when, & what people want to share. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 81-90). ACM.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature