The International Workshop on Urban Computing (UrbComp 2012)

August 12, 2012 - Beijing, China, Held in conjunction with KDD 2012

Website: http://www.cs.uic.edu/~urbcomp2012/

Program Chairs

Yu Zheng, Microsoft Research Asia, China

General Chairs

Ouri E. Wolfson, University of Illinois at Chicago

ACM ISBN: 978-1-4503-1542-5
Conference ID - 2012-30939

Program Committee

Overview

With the rapid progress of urbanization and civilization on earth, urban computing is emerging as a concept where every sensor, device, person, vehicle, building, and street in the urban areas can be used as a component to probe city dynamics to further enable city-wide computing for serving people and their cities. Urban computing aims to enhance both human life and urban environment smartly through a recurrent process of sensing, mining, understanding, and improving. Urban computing also aims to deeply understand the nature and sciences behind the phenomenon occurring in urban spaces, using a variety of heterogeneous data sources, such as traffic flows, human mobility, geographic and map data, environment, energy consumption, populations, and economics, etc.

Recently, real-world data reflecting city dynamics becomes widely available, including, e.g., users' mobile phone signal, GPS traces of vehicles and people, ticketing data in public transportation systems, user-generated content (like tweets, micro-blog, check-ins, photos), data from transportation sensor networks (camera and loop sensors) and environment sensor networks (temperature and air quality), as well as data from the Internet of Things. As a result, we are ready to carry out real urban computing activities that lead to better and smarter cities. For example, we can identify different functional regions [1], find smart driving directions [3][7][10], glean the problematic city configurations [4], detect anomalies in road traffic flows [6], and enable smart recommendations [2][5][8][9]. By better sensing and understanding the city dynamics we are more likely to design effective strategies and intelligent systems for improving urban lives. Examples of urban computing projects can be found on http://research.microsoft.com/en-us/projects/urbancomputing/default.aspx.

Representative Publications

[1] J. Yuan, Y. Zheng, X. Xie. Discovering regions of different functions in a city using human mobility and POIs. KDD 2012

[2] Ling-Yin Wei, Yu Zheng, Wen-Chih Peng, Constructing Popular Routes from Uncertain Trajectories. KDD 2012.

[3] Jing Yuan, Yu Zheng, Xing Xie, Guangzhong Sun, T-Drive: Enhancing Driving Directions with Taxi Drivers' Intelligence. Transactions on Knowledge and Data Engineering (TKDE).

[4] Yu Zheng, Yanchi Liu, Jing Yuan, Xing Xie, Urban Computing with Taxicabs, UbiComp 2011.

[5] Jing Yuan, Yu Zheng, Liuhang Zhang, Xing Xie, Guangzhong Sun, Where to Find My Next Passenger? , UbiComp 2011.

[6] Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan and Xing Xie. Discovering Spatio-Temporal Causal Interactions in Traffic Data Streams. KDD 2011.

[7] Jing Yuan, Yu Zheng, Xing Xie, Guangzhong Sun. Driving with Knowledge from the Physical World. KDD 2011.

[8] Yu Zheng, Xing Xie. Learning travel recommendations from user-generated GPS traces. In ACM Transaction on Intelligent Systems and Technology (ACM TIST), 2(1), 2-19.

[9] Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, Wei-Ying Ma. Recommending friends and locations based on individual location history. In ACM Transaction on the Web, 5(1), 2011.

[10] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, Yan Huang. T-Drive: Driving Directions Based on Taxi Trajectories. In ACM SIGSPATIAL GIS 2010.

Table of Contents

Inferring land use from mobile phone activity*

Jameson L. Toole†
Massachusetts Institute of Technology
77 Mass. Ave
Cambridge, MA, USA
jltoole@mit.edu

Michael Ulm
Austrian Institute of Technology
Vienna, Austria
michael.ulm@ait.ac.at

Marta C. González
Massachusetts Institute of Technology
77 Mass. Ave
Cambridge, MA, USA
martag@mit.edu

Dietmar Bauer
Austrian Institute of Technology
Vienna, Austria
dietmar.bauer@ait.ac.at

ABSTRACT

Understanding the spatiotemporal distribution of people within a city is crucial to many planning applications. Obtaining data to create required knowledge, currently involves costly survey methods. At the same time ubiquitous mobile sensors from personal GPS devices to mobile phones are collecting massive amounts of data on urban systems. The locations, communications, and activities of millions of people are recorded and stored by new information technologies. This work utilizes novel dynamic data, generated by mobile phone users, to measure spatiotemporal changes in population. In the process, we identify the relationship between land use and dynamic population over the course of a typical week. A machine learning classification algorithm is used to identify clusters of locations with similar zoned uses and mobile phone activity patterns. It is shown that the mobile phone data is capable of delivering useful information on actual land use that supplements zoning regulations.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining, spatial databases GIS*

*Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. UrbComp'12, August 12, 2012. Beijing, China. Copyright 2012 ACM 978-1-4503-1542-5/08/2012 É$15.00.

†Please direct all correspondence to Jameson Toole at jltoole@mit.edu

General Terms

Design, Measurement, Human Factors

Keywords

Urban Computing, Human Mobility, Land Use, Dynamic Population, Mobile Phone Data, Computational Social Science

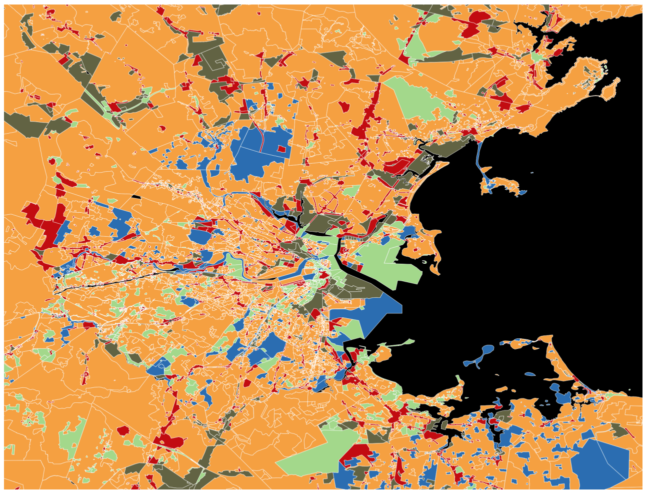


Figure 1: Zoning regulation for the Boston area. Color code: orange - Residential, red - Commercial, gray - Industrial, blue - Parks, green - Other.

1. INTRODUCTION

In describing the "organized complexity" of cities, Jane Jacobs notes that a "park's use depends, in turn, on who is around to use the park and when, and this in turn depends on uses of the city outside the park itself." [9] Where people live, work, and play is intimately related to the time and distance required to move to and from these locations [7]. Understanding how individuals are distributed in space and time is crucial to making effective and efficient planning decisions within cities. For example, the location choices of residents and firms is influenced by and determines the demand for mobility. Restaurants want to maximize patronage by choosing a popular location and individuals want to maximize their access to amenities.

How a particular area of a city is used is determined, in part, by the zoning regulations implemented and enforced by local governments. These regulations impact the structure of a city by dictating where housing or office space can be located. Zones of a kind share common usage. The central business district (CBD) for instance is populated during office opening hours whereas when offices are closed, relatively few people are found in these zones. Different zones relate to different land use which is related to different population size to be found at any given time in the zone. In practice, however, many zones feature different usage which might also differ somewhat from intended use. As an example zoning information for the Boston area is shown in Figure 1. Note, that zoning areas are not only restricted to land but also cover parts of rivers, lakes and the sea.

There is a large body of work dedicated to understanding the spatiotemporal dynamics of population and its relation to land use [10, 1, 5]. Measurements of human mobility within cities has traditionally been made via travel surveys. These surveys require subjects to record data on where they are moving to and from in the observation period (typically one day or a whole week), how they are doing so, and why. However, because surveys typically feature in-person interviews and demand a high workload for each subject, this method of data collection is expensive and limited.

Given these limitations, travel surveys suffer from relatively small samples (usually below tens of thousands of individuals), capture only short periods for each individual, and are updated infrequently. Fortunately, over the past decade a new type of measurement instrument has made its way into the pockets of people in nearly every culture and country. Each of the roughly 6 billion mobile phones currently in use [1] is capable of recording the location of calls, SMS, and data transmissions to within a few hundred meters. Moreover, these data are also collected centrally by mobile phone providers for billing purposes. With these data come enormous opportunities to improve our understanding of human mobility patterns.

In particular, call detail records (CDR) data, which provide information on the location of mobile phones any time a call is made or a text message is sent, contain much information on the distribution of persons in a region. This information can be obtained at low costs. Moreover, aggregated data only contains the number of active phones in a given area during a given time interval. This method of data collection provides much higher levels of anonymity reduces the risk any breach of individual information. Given the (imperfect) relation between the distribution of persons and active phones in a region the question arises as to whether the distribution of the numbers of active mobile phones can be used in order to infer land usage in a given zone.

To have such a measurement method would be very advantageous. Corresponding results can be used to monitor the use of all zones of a given zone class. Zoning regulation that all zones of one class share a common usage whereas the usage might differ for a number of reasons. Knowledge on different usage can be used to understand demand for mobility infrastructure across space and time. Monitoring the usage over time allows to detect changes in habits of the population as well as shifts in usage which may indicate ongoing regional developments.

Consequently this work investigates the potential of applying aggregated CDR data in order to infer dynamic land use, i.e. to understand how the population of different areas of a city changes with time and according to specific zoned land uses. The work centers on supervised classification of regions according to given zoning regulations. We demonstrate that CDR data can be used in order to classify zones of different types with reasonable accuracy. To this end, normalization techniques are discussed to highlight differences between zones. Then, the application and result of random forests for the classification is described in detail.

2. MOBILE PHONES AND HUMAN MOBILITY

Mobile phones have proven good instruments to measure human behavior. In one of the first studies utilizing these devices, Eagle and Pentland [6] were able to decompose mobile phone activity patterns of university students and employees into regular daily routines. Moreover, these patterns were found to be predictive of an individual's characteristics such as their major or employment level (i.e. graduate student). Subsequent research has built upon this work, scaling up in both geographic extent and sample size. González et al [8] studied data from nearly one-hundred thousand anonymous mobile phone users to reveal persistent regularities in the statistical properties of human mobility. Highlighting the remarkable predictability of human behavior, Song et al [12] estimated that it is theoretically possible to predict individual movements of users with as high as 93% accuracy using only data from mobile phones.

Mobile phone data has also been used to study how space is used over time. Reades et al [11] used mobile phone network data from Rome, Italy, to link mobile phone activity to commercial land uses. Measuring mobile phone activity in 1km by 1km grid cells, they employ a form of principal component analysis to identify the dominant activity patterns. The authors qualitatively interpret areas of the city exhibiting this signal as Commercial, though actual zoning information is not introduced. They then decompose activity across the city to identify regions with similar patterns of usage. Similarly, Soto et al [13] use CDR mobile phone data at the cell tower level to identify clusters of locations with similar activity. Qualitative agreement between these clusters and land uses were observed.

Calebrese et al [4] have applied similar decomposition and clustering techniques to classify locations on a university campus as classrooms, dormitories, etc. By analyzing wifi activity across 3000 wifi-access points, the authors used unsupervised, non-parametric techniques to identify clusters of similarly used locations. These locations naturally fit into location profiles such as "lecture hall" or "dormitory." Finally, CDR data have proven useful to detect movement at

[1]http://www.itu.int/net/ITU-D/index.aspx

the census tract scale [3]. Location data from calls helped to measure origins and destinations for trips across the Boston Metropolitan area. However, no attempt was made to associate such trips with land uses.

Other data sources such as points of interest (POIs) as well as GPS data collected from taxi fleets have been combined with unsupervised learning algorithms to identify the rich structure of different functional sections of a Beijing [14]. To date, however no studies exist that employ supervised learning techniques to combine traditional data sources on land use such as zoning regulations and CDR data. This study aims to investigate the link between zoned land use and mobile phone activity on a common spatial partitioning of the greater Boston area into regions of homogeneous land use. For each region the temporal profile of active phones is used in supervised classification techniques in order to identify patterns characteristic for a specific zoning classification. The corresponding patterns will be interpreted in detail.

3. DATA SOURCES

Two data sources are used in this work: mobile phone activity records and zoning regulations. For the Boston metro region, anonymized CDR provide the location of a mobile phone by triangulating signal strengths from surrounding cell towers, unlike traditional CDR data, in which record the location of a call as the location of the mobile phone tower. This provides slightly higher accuracy and allows us to measure calls continuously across space rather than at points where towers are located. Triangulation by this method is accurate to within a few hundred meters depending on the tower density. These data make it possible to measure the amount of phone activity (counts of the number of calls and texts) that occurs within a given area and time window. In this study we use three weeks of CDR data for roughly 600,000 users in the Boston region home to roughly 3 million people. Though mobile phone data come from specific set of carriers, the market share of these carriers is between 30% and 50%.

In addition to mobile phone activity, we obtain zoning classifications for the Boston metropolitan area. The Massachusetts Office of Geographic Information (MassGIS) aggregates uses into five categories: Residential, Commercial, Industrial, Parks, and Other. We are careful to note our assumption that actual land use and zoning classification are closely related while acknowledging that zoning regulations are only a proxy of actual land use imposing restrictions.

4. COMMON SPATIAL REPRESENTATION

The first obstacle to studying the relationship between phone activity and land use is the reconciliation of the spatial dimensions of the data: While the location of the phone activities are recorded as coordinate pairs, zoning data is provided in polygons at roughly the parcel scale. The spatial partitioning of phone and population data is rarely the same as zoning parcels. To reconcile all data sources as well as to reduce the influence of noise (due to inter alia sources localization estimation noise) in the data, we transform both to the same uniform grid. A lattice is laid over the analysis region such that every cell in the lattice measures 200 by 200 meters. Different grid sizes have been tested, 200 meters proved to be a good aggregation level; being coarse enough to reduce the noise level and detailed enough in order not to mix many parcels of different zoning areas.

In order to reduce the high noise level average hourly time series of phone activity are computed. Here, the average is computed for each hour within a day of the week. Only cells with mobile phone activity above a certain threshold are used in the analysis.

With respect to zoning data, each cell is given a single zoning classification based on the most prevalent (in terms of fraction of area covered) use within the area.

Potential pitfalls of this method arise due to large heterogeneity in population density. Downtown areas are much more densely populated than the suburbs, a characteristic that is reflected in other spatial divisions like census tracts. This leads to sparse mobile phone activity in rural regions. However, the small grid size used in this analysis retains detailed information about block to block zoning regulations in dense urban areas. Figure 2 displays actual zoned parcels versus the gridded approximations.

Table 1 shows the frequency of each zoning class in the grid. The vast majority of land, nearly 75% of cells, are zoned as Residential. Other uses appear in roughly equal fractions.

Table 1: Tabulation of Boston zoning. The land use profile of the city is dominated by residential use accounting for nearly 75%. Other uses share roughly the same percentage of remaining land.

Zone Use	Category Index	Count	Percentage
Residential	1	23322	74.28
Commercial	2	1854	5.90
Industrial	3	2236	7.12
Parks	4	1941	6.18
Other	5	2045	6.51

5. DESCRIPTIVE STATISTICS

We first examine the relationship between mobile phone activity and land use at the macro, city-wide scale. Figure 3 displays time series of mobile phone activity averaged over all cells of a given zoning classification. Examining absolute counts (first row) reveals that the average activity level in different zoning classifications differs greatly. While residential areas only show a maximum activity of roughly 50 events per hour, commercial cells reach approximately 100 events on average.

The number of activities within different cells shows huge differences. The downtown area of Boston shows orders of magnitude higher activity levels than typical residential zones. In order to allow for classification based on relative mobile phone activity, time series are normalized using a z-score. By definition, the normalized time series have zero mean and unit standard deviation. Mathematically, the normalized activity of cell (i,j) is given by:

$$a_{ij}^{norm}(t) = \frac{a_{ij}^{abs}(t) - \mu_{a_{ij}^{abs}}}{\sigma_{a_{ij}^{abs}}} \quad (1)$$

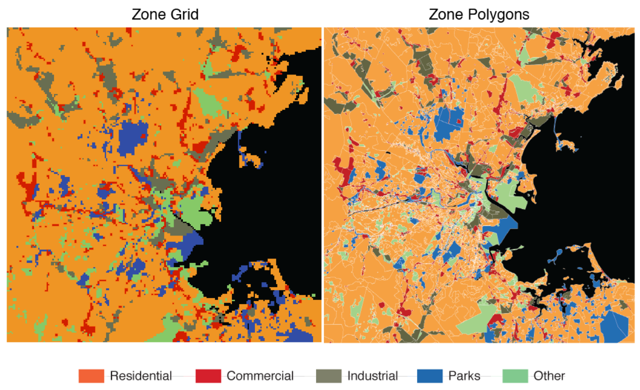


Figure 2: To improve computational efficiency and reconcile all mobile phone and traditional data sources, we create a uniform grid over the city. Zoning polygons (right), are rasterized to cells 200m by 200m in size (left). For cells where more than one zoning class exists, the most prevalent class is used. Given the small size of these cells, this data transformation provides an accurate map of the city while improving computational efficiency.

The second row of Figure 3 (a) shows the average (over cells of one zoning class) normalized activity. These profiles are remarkably similar for all zoning classes showing the strong circadian rhythm of the city. Residents wake up, go to sleep, and wake again the next day. The rise and fall of activity in each zone, however, is not solely the result of users moving into and out of a region, but is instead also partly due to an uneven distribution of phone use across the day. To account for this, during each hour, we subtract the average normalized activity of the entire region from the normalized activity at each given cell. The corresponding spatially de-meaned series will be referred to as *residual activity.* Residual activity can be interpreted as the amount of mobile phone activity in a region, at a given time, relative to the expected mobile phone activity in the whole city at that hour. Mathematically, it is calculated as follows:

$$a_{ij}^{res}(t) = a_{ij}^{norm}(t) - \bar{a}^{norm}(t) \qquad (2)$$

where $\bar{a}^{norm}(t)$ is the normalized activity averaged over all cells at each particular time. Averaging the residual activity for each zoning classification reveals patterns related to travel behavior. The last row of Figure 3 (a) and (b) provide the residual activity averages across zoning classes for weekdays and weekends. The most notable signal is the inverse relationship between residual activity in residential and commercial areas: While residential areas on average show higher than expected activity during the night and lower than expected during weekdays. As expected, the opposite is true for commercial zones. Somewhat surprisingly, the normalized activity does not show these features strongly. Only the residual activity demonstrates the expected behavior. There, also higher than average activity in parks on the weekend afternoons is visible.

Residential areas have higher residual activity in the early morning hours and late at night, while commercially zoned cells have a peak period during the day and show much lower activity levels late at night. These patterns most likely reflect the 9-to-5 business hours of offices and stores. More subtle patterns are also visible. In Boston, much of the CBD is zoned as Other or Mixed use. We see that residual phone activity in this zoning type has peaks in the early morning hours on Saturday and Sunday, suggesting these areas support night life on the weekends. These city-wide time series show that mobile phone activity and land use are linked at the highest level of aggregation. By treating phone activity as a proxy for the spatial distribution of people at a given time period the expected patterns of concentration of people in the CBD and inner city region during the working day, and the shifts induced by the commuting behavior are visible in the residual activity levels.

We note that because residual activity is relative to absolute call volume as well time of day, it is not affected by differences in mobile phone usage across zoned uses provided those differences are persistent in time. For example, it does not affect measurements if individuals are twice as likely to make a phone call in a commercial zone than a residential zone as long as this propensity is constant across all hours of a week.

Figure 4 displays the spatial distribution of normalized activity (top row) and residual activity (bottom row) at three time instants. Not shown in the plots are the absolute activity levels which are distributed much like population density. The CBD of Boston has orders of magnitude more activity than the rest of the city. Mapping the logarithm of absolute activity over time once again only reveals the circadian rhythm of the city which strongly dominates the differences in land usage which consequently are not seen in these plots.

In the spatial distribution of the normalized activity the dominance of the CBD is less pronounced. Nevertheless, the circadian rhythm still dominates the differences between different zones. From this perspective, Boston appears as a monocentric region, with small pockets of density located on an urban ring roughly 20km from the CBD.

By way of contrast, the spatial distribution of residual activity reveals a much richer structure. In the early morning hours, residual activity is located on the periphery of the region. During the day, this activity becomes heavily concentrated in the CBD or in small subcenters on the urban ring. Later in the evening, activity again returns to residential areas on the periphery, away from centers. This suggests some correlation between commuting patterns and the spatial distribution of residual activity.

6. CLASSIFYING LAND USE BY MOBILE PHONE ACTIVITY

In the last section we observed correlation between residual mobile phone activity and land use on the macro scale. Fluctuations in mobile phone activity mimics intuition of population changes related to commuting and recreational trips. In this section we investigate the question whether usage of cells of one zone class are homogeneous. This will be done by performing supervised classification based on features extracted from the residual activity time series and the classes provided by the zoning regulations as labels. Though previous work in this area has employed unsupervised learn-

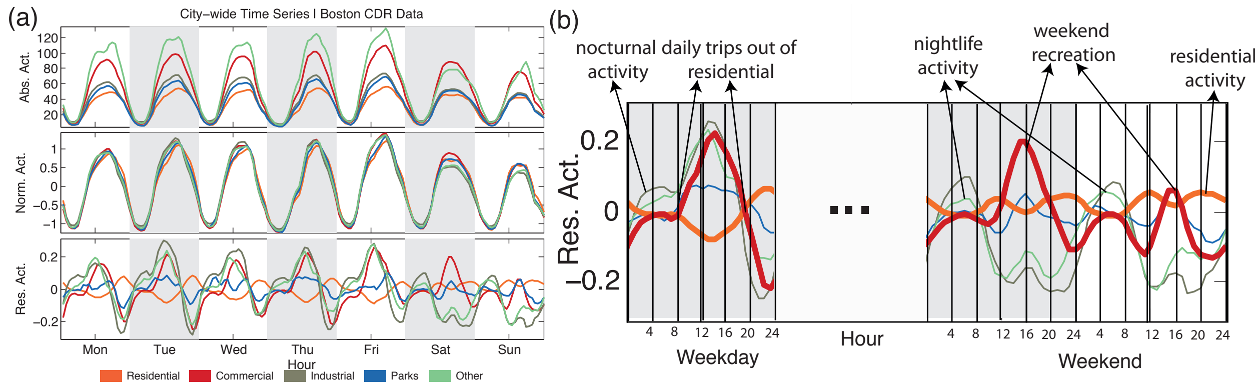


Figure 3: (a) Plots are shown for three different time series of average mobile phone activity within each of five land use. The first plot shows absolute activity (number of calls and SMS messages). The second plot displays z-scored time series. The bottom plot shows residual activity. (b) More detailed view of average (over cells of the same zoning class) residual activity.

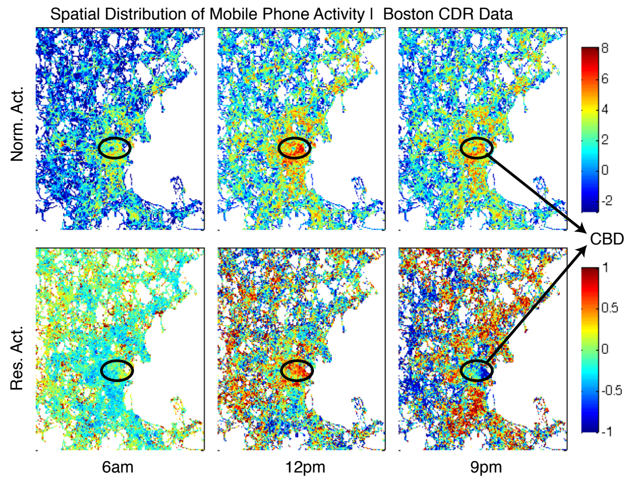

Spatial Distribution of Mobile Phone Activity I Boston CDR Data
Norm. Act.
Res. Act.
CBD
6am
12pm
9pm

Figure 4: Spatial distribution of absolute and residual phone activity over the course of a day. While absolute mobile phone activity is dominated by population density and sleep and wake patterns, residual activity reveals flows into and out of the city center over the course of a day.

ing techniques, access to extensive zoning data in a mature, regulated city such as Boston makes supervised learning an attractive option. Cross validation is used to test performance.

We implement the *random forest* approach described by Breiman [2]. Other approaches including neural network based classifiers have been tested and led to similar results. Random forests are useful for their ability to efficiently classify data with large numbers of input variables (such as long time series). Rather than make comparisons for every feature of the data every time, a number of random subsets are chosen to more efficiently search the space. This does not come at the cost of accuracy as random forests have been shown to have high performance on a variety of datasets [2]. Moreover, random forest classifiers allow weights to be introduced so that more frequently occurring classes do not overwhelm smaller ones. This feature will be exploited later to control for the large share of residentially zoned locations.

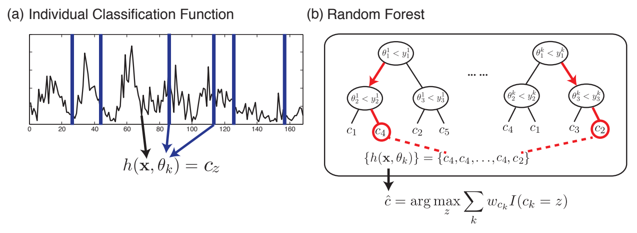

(a) Individual Classification Function
(b) Random Forest
$h(\mathbf{x}, \theta_k) = c_z$
$\{h(\mathbf{x}, \theta_k)\} = \{c_4, c_4, \ldots, c_4, c_2\}$
$\hat{c} = \arg\max_z \sum_k w_{c_k} I(c_k = z)$

Figure 5: (a) Shows the inputs to each decision tree $h(\mathbf{x}, \theta_k)$. A time series of residual phone activity, $\mathbf{x}$, is input and activity at a random subset of times , θ_k (denoted by the blue bars), is chosen to make comparisons. (b) A depiction of the random forest shows a number of different trees making predictions based on a different set of random times. Each tree casts a weighted vote for a certain classification. A final classification, $\hat{c}$, is made by counting these votes.

A random forest, $\{h(\mathbf{x}; \theta_k), k = 1, ...\}$, is constructed from a set of decision trees as visualized in Fig. 5. The training data is used to determine the parameter vectors θ^k. Least squares or maximum likelihood estimation can be used to find these configurations. To obtain a single prediction for each input time series, a voting scheme is implemented. Each tree votes for a class based on its prediction. These votes can be weighted (weights denoted by w_{c_k}) so that votes for one class count more or less than votes for a different class. The weighted votes are summed and a single zoning class prediction, $\hat{c}$ is chosen for the original input time series.

For the calculations we use a MATLAB implementation of the random forest algorithm released by Jaiantilal [2]. Our implementation uses 49 input features which are computed for each location as the input feature vector $\mathbf{x}$. These features include a 24-hour time series of residual mobile phone

[2]`http://code.google.com/p/randomforest-matlab/`

activity during an average weekday as well as a 24-hour time series of residual activity for an average weekend-day. The final feature is the mean of the location's absolute activity on any given day. Additional features such as the variance of mobile phone activity were tested, but none aided prediction. The output of the algorithm is a zoning classification for each location. Cross validation is used to test accuracy. We create 500 trees for each forest and define total accuracy as the fraction of correctly classified cells on the validation part of the sample.

Our first set of results include all five zoning classifications: Residential, Commercial, Industrial, Parks, Other. When all land use classes are included, however, we face a major challenge with classification. As noted above, nearly 75% of all cells are primarily residential. The next most common zoned use is Industrial at 7%. Because of our definition of total accuracy, the most naive classifier, simply assigning Residential to everything, will achieve 75% total accuracy, but will fail to capture any diversity in use. To guard against this, we weight the voting system so raise or lower the required votes in order to choose a given classification. The maximum of the weighted votes then provides the predicted class. Systematic variations of the weights on a (coarse) grid led to a choice of weights where the criterion applied was maximum classification accuracy for all classes but residential.

Finally, we note that the random forest classifier uses local information only to make a prediction. Given the size of our grid cells, it is reasonable to assume that land use does not differ greatly from each 200m by 200m tract of land to the next. To incorporate neighborhood information into our predictions, we implement a second pass algorithm. After the classifier has made a prediction for a cell, we examine the predictions for each of that cell's neighbors. If the majority of neighboring cells were predicted to be a land use that differs from the cell in question, that cell is switched to the majority use of its neighbors. In practice, this results in some spatial smoothing of noisy classification data. We find that performing the second pass provides gains of 2-10% overall accuracy for each classifier.

Even with vote weighting and the second pass algorithm, we achieve only modest results. Table 2 shows 54% accuracy over the whole city. This implies that demanding equal classification accuracy for all classes reduces overall accuracy by about 20%. Figure 6 displays the spatial distribution of correctly and incorrectly classified locations. We note, however, that the algorithm does capture some spatial patterns in the data and that our intra-use accuracy is relatively high for Commercial and Industrial uses. Parks and Other mixed uses remain difficult to classify.

To account for the tendency of the algorithm to over-predict residential use, we remove cells zoned as Residential from consideration. This leaves a nearly equal share of the remaining four uses: Commercial, Industrial, Parks, and Other. Table 3 and Figure 7 display results for this sub-classifier. Now, the zone with the largest share is commercial use, which only accounts for 33% of non-residential zones. Intra-use accuracy has improved significantly for Parks and Other mixed uses. Whereas the random forest including residential uses could only correctly classify 2% of zones classified for Parks, the sub-classifier, excluding Residential, correctly predicts 30% of park cells. A similar improvement from 10% to 34% is also observed for the Other or mixed use category. The share of classes incorrectly classified as Residential roughly is distributed onto Parks and Others in the classifier without the Residential category, while commercial and industrial zones are not affected heavily. One hypothesis for this effect is that many cells while classified as Residential in rural areas are not fully developed and thus used as parks and in the city center show mixed usage. Including the large class of residential zones masks this effect.

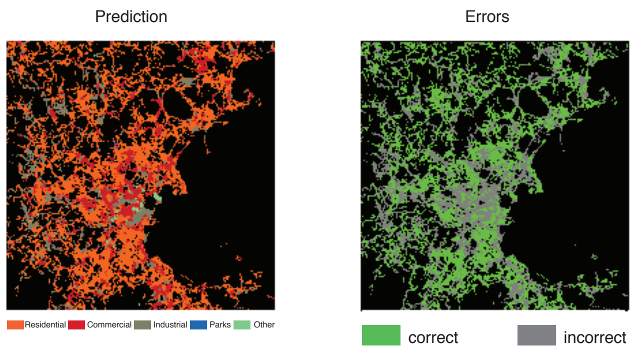


Figure 6: Left plot: zoning map as predicted from mobile phone data using the random forest classification algorithm. Right plot: spatial distribution of where the algorithm predicts land use correctly and where it fails. In general, these errors seem randomly distributed in space, suggesting that errors are not the result of some spatial correlations such as population density. For comparison to actual zoning, see the left panel of Figure 2.

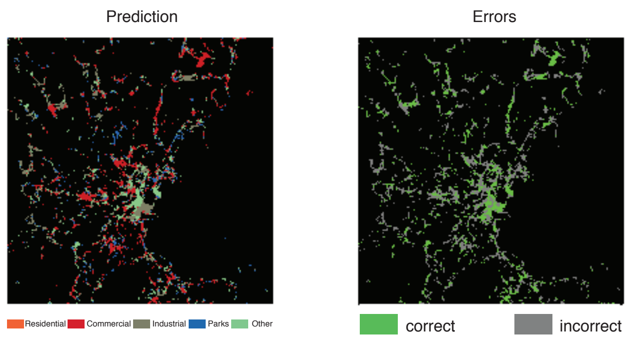


Figure 7: The left plot shows the city zoning map with residential areas removed as predicted from mobile phone data using the random forest classification algorithm. The right map displays the spatial distribution of where the algorithm predicts land use correctly and where it fails. Without residential areas to predict, the algorithm performs significantly better at predicting other uses. For comparison to actual zoning, see the left panel of Figure 2.

The goal of the supervised learning algorithm is to make correct predictions of actual zoned use. Incorrectly classified cells are labeled as errors, but how an area is zoned is

Table 2: Random forest classification results. The threshold refers the total number of phone events required in each cell over period of data collected to be considered for classification. Total accuracy is defined as the fraction of correctly classified cells. The share refers to the percentage of cells actually zoned for each class of use. Element (i,j) of the confusion can be interpreted as the fraction of actual zoned uses of class i that were classified as use j by the random forest. Thus the high percentages in the Res column can be interpreted as the algorithm heavily favoring classification as residential due to its overwhelming share of overall uses.

Total Accuracy:	0.54				
	Res	Com	Ind	Prk	Oth
Land Share:	0.74	0.09	0.08	0.04	0.05
Vote Thresh:	0.60	0.10	0.10	0.10	0.10
	Confusion Matrix				
	Res	Com	Ind	Prk	Oth
Res	0.62	0.21	0.15	0.01	0.01
Com	0.30	0.48	0.19	0.00	0.02
Ind	0.33	0.27	0.38	0.00	0.02
Prk	0.52	0.26	0.18	0.02	0.02
Oth	0.37	0.28	0.25	0.00	0.10

Table 3: Random forest classification results. In this case, residential land has been removed from consideration. The algorithm is now able to correctly predict much larger fractions of rarer land uses.

Total Accuracy:	0.40				
	Res	Com	Ind	Prk	Oth
Land Share:	0.00	0.33	0.31	0.16	0.20
Vote Thresh:	N/A	0.30	0.30	0.20	0.20
	Confusion Matrix				
	Res	Com	Ind	Prk	Oth
Res	N/A	N/A	N/A	N/A	N/A
Com	N/A	0.50	0.19	0.11	0.19
Ind	N/A	0.27	0.37	0.12	0.24
Prk	N/A	0.31	0.18	0.29	0.21
Oth	N/A	0.26	0.24	0.15	0.34

not necessarily the same as how it is used. As an example the area termed "Back Bay" containing some of Boston's most busiest shopping streets, Boylston and Newbury, is classified as residential, as is the campus of MIT. Clearly these areas have a different usage than residential areas in the suburbs. A political and idiosyncratic process for setting and updated zoning regulations may lead to broad or unenforced development standards. In light of this, errors made by our classification algorithm may be due to incomplete zoning data rather than actual mistakes. To examine this possibility further, we analyze prediction errors more closely. Figure 8 displays a detailed partitioning of classifier results. We compare average residual activity across three groups of cells: (I) All cells correctly predicted to be a given use. (II) All cells of another use incorrectly predicted to be the given use. (III) All cells of a given use incorrectly predicted to be some other use.

Reviewing residential use, we see that Group I is defined as all residential cells correctly predicted to be residential. The average activity pattern is the most dominant pattern of residual activity for residential land use. We find that the residual activity in non-residential cells predicted to be residential (Group II) closely follows the pattern found in Group I. This strongly supports our hypothesis that though some zones are not classified as residential in the data, their phone activity patterns suggest they are used in similar ways. In contrast, the residual activity in residential cells incorrectly classified as some other use (Group III) displays the inverse pattern. This suggests our algorithm is identifying cells that are zoned as residential use but that do not share activity characteristic of that zoning class in reality.

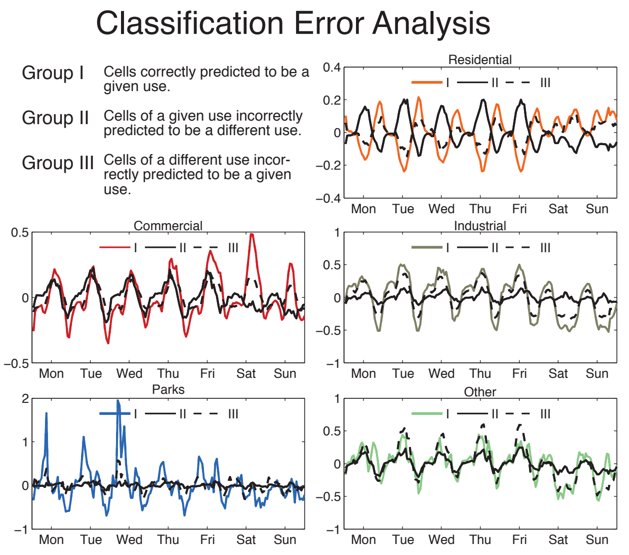


Figure 8: An analysis of classification errors. We consider three groups: (I) Cells correctly predicted to be a given use (II) Cells of a given use incorrectly predicted to be some other use (III) Cells of some other use incorrectly predicted to be a given use. For example, Group I includes all residential areas correctly predicted to be residential. Group II, residential cells predicted to be some other use (i.e. Commercial), have average activity that is the inverse of Group I, suggesting these locations were misclassified because they display fundamentally different activity patterns. Group III represent cells of other uses such as Commercial that behave like Residential. This error analysis suggests that our algorithm is clustering locations based on both their zoned use as well as the dominant patterns in mobile phone activity.

7. CONCLUSION

In this article, we examined the potential of CDR data to predict land usage. We demonstrated that aggregate data shows the potential to differentiate land usage based on temporal distribution of activities. While the absolute activity is dominated by the circadian rhythm of life, eliminating this rhythm reveals subtle differences between the five main land use categories Residential, Commercial, Industrial, Parks and Other. The addition of a temporal dimension to zoning classification may aid strategic planning decisions related to land use.

As the data are available at a high spatial resolution, we investigated the capabilities to infer land use on a fine grid of 200 by 200 meters. We found that supervised classification based on labeled zoning data provides estimated land use classifications which show better accuracy than random assignment. At the same time accuracy is worse than classifying every zone as Residential, the dominant category.

Reasons for this lack of accuracy might be found in the nature of the data used: actual usage might differ from the zoning regulations and Residential is often confused with Parks and Other zones. Omitting residential zones, the classification accuracy for Parks and Other zones greatly increases while industrial and commercial zones classification accuracies are not heavily affected. For rural areas where residential land might not be fully developed this is plausible. For urban zones the distinction between Residential and Other zones might also be subject to temporal changes as mixed use is prevalent. Finally, analysis of prediction errors reveals that the algorithm fails to correctly classify areas because they have fundamentally different mobile phone activity patterns. This suggests that there may be heterogeneity in how land is actually used, despited its official zoned classification.

Thus the main conclusion is that the CDR data shows some potential to infer actual land use both on an aggregate level and on a higher spatial resolution. However, zoning data might not be the optimal data source to infer actual land use and hence act as ground truth to guide the supervised learning algorithm. In this respect, our analysis suggests that mobile phone activity may be used to measure the heterogeneity in how space is used that cannot be captured by simple and broad zoning classifications. Moreover, the incorrect predictions made by our algorithm may suggest updates to traditional zoning maps so as to better reflect actual activity or highlight areas where more planning oversight is needed.

Both topics will be investigated further. Larger sample sizes in the form of longer time series might lead to a reduction in noise levels and hence increase the classification accuracy. It may also be advantageous to expand the set of features used in prediction. Although our aim was to keep this space relatively low dimensional to aid interpretation, the complexity of intracity mobility may demand more. However, given our results suggest that a modest fraction of the city can be classified at very high resolution from relatively simple features. The algorithm itself may be improved by additional consideration of balancing more prevalent uses with those more scarce. Finally, other data sources such as points of interest (POIs) will be used as ground truth in the supervised learning instead of zoned use. This may clarify whether the deviations in classification between the zoning regulations and the mobile phone usage dynamics are due to wrong zonings or deficiencies in the measurement technology using CDR data.

We hope this information will be useful to make effective and efficient choices of locations for both public and private resources. In addition to potential applications, we hope that tools and techniques developed and applied above will prove useful to merging traditional and novel data.

8. ACKNOWLEDGEMENTS

This work was made possible in part by the MIT - Xerox Fellowship as well as an National Science Foundation Graduate Research Fellowship. In addition, Marta C. González acknowledges awards from NEC Corporation Fund and the Solomon Buchsbaum Research Fund. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

9. REFERENCES

[1] D. Banister. Reducing the need to travel. *Environment and Planning B: Planning and Design*, 24(3):437–449, 1997.

[2] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Oct. 2001.

[3] F. Calabrese, G. Di Lorenzo, L. Liu, and C. Ratti. Estimating Origin-Destination Flows Using Mobile Phone Location Data. *IEEE Pervasive Computing*, 10(4):36–44, Apr. 2011.

[4] F. Calabrese, J. Reades, and C. Ratti. Eigenplaces: Segmenting Space through Digital Signatures. *IEEE Pervasive Computing*, 9(1):78–84, Jan. 2010.

[5] R. Cervero and K. Kockelman. Travel demand and the 3Ds: Density, diversity, and design. *Transportation Research Part D: Transport and Environment*, 2(3):199–219, Sept. 1997.

[6] N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, May 2006.

[7] K. T. Geurs and B. van Wee. Accessibility evaluation of land-use and transport strategies: review and research directions. *Journal of Transport Geography*, 12(2):127–140, June 2004.

[8] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, June 2008.

[9] J. Jacobs. *The death and life of great American cities.* Vintage Books, 1961.

[10] K. Maat, B. van Wee, and D. Stead. Land use and travel behaviour: expected effects from the perspective of utility theory and activity-based theories. *Environment and Planning B: Planning and Design*, 32(1):33–46, 2005.

[11] J. Reades, F. Calabrese, and C. Ratti. Eigenplaces: analysing cities using the spaceÿ - ÿtime structure of the mobile phone network. *Environment and Planning B: Planning and Design*, 36(5):824–836, 2009.

[12] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási. Limits of Predictability in Human Mobility. *Science*, 327(5968):1018–1021, Feb. 2010.

[13] V. Soto and E. F. Mart'ınez. Automated land use identification using cell-phone records. In *Proceedings of the 3rd ACM international workshop on MobiArch*, HotPlanet '11, pages 17–22, New York, NY, USA, 2011. ACM.

[14] J. Yuan, Y. Zheng, and X. Xing. Discovering regions of different functions in a city using human mobility and pois. *KDD*, 2012.

Estimation of Urban Commuting Patterns Using Cellphone Network Data

Vanessa Frias-Martinez
Telefonica Research, Madrid, Spain
vanessa@tid.es

Cristina Soguero
Telefonica Research, Madrid, Spain
soguero@tid.es

Enrique Frias-Martinez
Telefonica Research, Madrid, Spain
efm@tid.es

ABSTRACT

Commuting matrices are key for a variety of fields, including transportation engineering and urban planning. Up to now, these matrices have been typically generated from data obtained from surveys. Nevertheless, such approaches typically involve high costs which limits the frequency of the studies. Cell phones can be considered one of the main sensors of human behavior due to its ubiquity, and as a such, a pervasive source of mobility information at a large scale. In this paper we propose a new technique for the estimation of commuting matrices using the data collected from the pervasive infrastructure of a cell phone network. Our goal is to show that we can construct cell-phone generated matrices that capture the same patterns as traditional commuting matrices. In order to do so we use optimization techniques in combination with a variation of Temporal Association Rules. Our validation results show that it is possible to construct commuting matrices from call detail records with a high degree of accuracy, and as a result our technique is a cost-effective solution to complement traditional approaches.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms,Experimentation,Measurement.

Keywords

Commuting Patterns, O-D Matrix, Call Detail Records.

1. INTRODUCTION

Commuting patterns are typically represented using commuting matrices, which are a particular case of O-D matrices. O-D matrices characterize the transitions of a population between different geographical regions representing the origin (O) and destination (D) of a route. When building commuting matrices the geographical areas representing origin(O) and destination (D) capture where people live and work. Typically O and D are the same set and represent the towns or neighborhoods of the geographical area under study. Each element of the commuting matrix (i, j) defines the percentage of individuals that live in O_i and work in D_j.

Typically, National Statistical Institutes carry out periodical surveys asking different segments of the population about their commuting patterns [16]. The information obtained is used as input for O-D generation techniques. However, such approach typically involves high costs and the data collected has spatio-temporal limitations, which implies that the matrices generated typically only represents a snapshot of the commuting patterns over time.

In recent years, cell phones have become a pervasive technology with users carrying them at almost all times. The ubiquity of these platforms has transformed cell phones into one of the main sensors of human behavior. In fact, every time a subscriber makes or receives a phone call, or an SMS, or an MMS, information regarding the interaction as well as the geolocation of the user (in the form of the tower used for the communication) is logged for billing purposes. As a result we can find in the literature a variety of studies focussing on using cell phone data for estimating traffic and commuting patterns [8][18] . Following this trend, in this paper we explore the use of the location information contained in Call Detail Records as a means to compute the commuting patterns of a population expressed as an O-D matrix. Such approach overcomes the limitations posed by the use of other proxies (like smart cards, surveys or social security records) and it can be carried out as often as necessary with very limited costs.

Compared to the literature, our approach has the following contributions: (1) We base our study in Call Detail Records, which are already available for billing purposes in a telco operator, and not in specific measurements and/or traces obtained from the cell phone network. As a result our approach is based on a big part of a population and not on a limited number of traced cell phones; (2) We present a new technique for defining and constructing O-D matrices based on a new temporal variation of association rules (TAR, Temporal Association Rules) and (3) Our technique is designed to capture the different cultural commuting schedules of different urban areas.

2. CELLULAR INFRASTRUCTURE

In order to compute the commuting patterns of a population from geolocated cell phone logs, we first give a brief overview about how these pervasive networks work. Cell phone networks are built using a set of base transceiver stations (BTS) that are in charge of communicating cell phone devices with the network. Each BTS tower has a geographical location typically expressed by its latitude and longitude. The area covered by a BTS tower is called a cell. Each cell is typically divided in three sectors, each one covering 120 degrees. At any given moment, one or more BTSs can give coverage to a cell phone. Whenever an individual makes a phone call, the call is routed through a BTS in the area of coverage. The BTS is assigned depending on the network traffic and on the geographic position of the individual.

CDR (Call Detail Record) databases are generated when a mobile phone connected to the network makes or receives a phone call or uses a service (e.g., SMS, MMS, etc.). In the process, and for invoice purposes, the information regarding the time and the BTS tower where the user was located when the call was initiated is logged, which gives an indication of the geographical position of a user at a given moment in time. Note that no information about the exact position of a user in a cell is known. Also, no information about the location of cell phone is known or stored if no interaction is taking place.

From all the data contained in a CDR, our study uses the encrypted originating number, the encrypted destination number, the time and date of the call, the duration of the call, and the latitude and longitude of the BTS tower used by the originating cell phone number and the destination phone number when the interaction happened. In order to preserve privacy, all the information presented is aggregated and original records are encrypted. No contract or demographic data was considered or available for this study.

3. PROBLEM DEFINITION

A commuting matrix $CM[O, D]$ represents the percentage of population that commutes on an average daily basis from an origin geographical area O to a destination geographical area D. Typically O and D represent the same set of towns, and as a result a commuting matrix is usually a square matrix. Two commuting matrices can be defined: the home-work commuting matrix $CM[H, W]$ and the work-home commuting matrix $CM[W, H]$. In the first case, each row of the commuting home-work matrix $CM[H, W]$, H_i represents the percentage of population that lives in geographical area H_i and commutes to each geographical area W_j. The diagonal of the matrix expresses the percentage of the population that lives and works in the same town. Symmetrically, the work-home commuting matrix $CM[W, H]$ accounts for the population that works in the geographical area W_i and commutes back home to each one of the geographical locations H_j (columns). From this explanation, being N the number of geographical areas considered, it follows that $\sum_{j=1}^{j=N} CM[H_i, W_j] = 1 \forall i \in [1, ..., N]$ and $\sum_{j=1}^{j=N} CM[W_i, H_j] = 1 \forall i \in [1, ..., N]$.

Traditionally, such commuting matrices are computed by National Statistical Institutes (NSIs) that run surveys and questionnaires across the population under study and determine the commutes that citizens carry out on a daily basis. These mobility matrices are typically available at census bureaus. However, as stated earlier, such surveys are expensive and thus carried out every certain number of years.

The goal of this paper is to present a mechanism to estimate the commuting matrix of a geographical area from the information contained in CDR records that can approximate the values provided by traditional questionnaire-based approaches. For that purpose, two mechanisms need to be defined: (1) the construction of commuting matrices from CDR data and (2) an optimization process that identifies which behavioral patterns better define commuting when using CDR data.

4. ESTIMATING COMMUTING MATRICES FROM CDR

In this section we will present the mechanisms needed to characterize the commuting patterns of a population from call detail records (CDR).

4.1 From CDRs to Commuting Matrix

To compute a commuting matrix from CDRs we first need to identify the geographical areas in the region under study that we are going to use as either *home* or *work*. Given that the goal of this paper is to present an alternative method to generate commuting matrices, for each particular case we will select as regions the same ones considered by corresponding NSI. We assign to each region the set of BTSs geographically included in them (i.e. the towers that give coverage to that area). As a result each geographical area considered $g_i, i = 1, ..., N$, with N the total number of geographical areas considered, can be characterized by a set of BTSs $g_i = \{bts_1, bts_2, ..., bts_k\}$.

Once these areas have been characterized, we need to compute – from the CDRs– the individuals that called from an origin area at some point in time and later show calling activity at a destination area. These associations will populate the home-work and work-home commuting matrices.

We can formalize this problem using Association Rules [1]. Association Rules (ARs) were introduced by Agrawal *et al.* as a technique to discover specific item relationships in itemsets [1]. Specifically, given an itemset $X = X_1, X_2, ..., X_n$, an Association Rule of the type $X \rightarrow Y$ implies that whenever X is satisfied, Y is also satisfied, with a given support and confidence. Formally, being P the probability of an itemset:

$$support(X \rightarrow Y) = P(X \bigcup Y) \qquad (1)$$

$$confidence(X \rightarrow Y) = P(Y|X) = \frac{P(X \bigcup Y)}{P(X)} \qquad (2)$$

Often times, Association Rules(AR) are used to find the tuples that satisfy minimum support and confidence values in a dataset. ARs are calculated using the *Apriori* algorithm presented in [1]. In our context, we seek association rules $H_i \rightarrow W_j$ and $W_i \rightarrow H_j$ that identify tuples characterizing the home to work and work to home commutes. Furthermore, we require these events to happen in a temporal order *i.e.*, the home-work matrix $CM[H, W]$ is populated with pairs of events $H_i \rightarrow W_j$ such that the interaction at a home location H_i always happens earlier in time than an interaction event at work location W_j; analogously, the work-home matrix $CM[W, H]$ is populated with pairs $W_i \rightarrow H_j$ where an interaction event at work location W_i always happens before an interaction at a home location H_j. Because traditional Association Rules do not consider any temporal order, we present a technique designed to capture these elements: *Temporal Association Rules (TARs)*.

4.1.1 Temporal Association Rules

Temporal Association Rules extend association rules by introducing temporal constraints in the relationship between antecedent and consequent [12][6]. For our context, we propose a new Temporal Association Rule (TARs) where items X and Y are required to happen within a specific time interval. Specifically, each association

Algorithm 1: $CM_{CDR} = CMTAR(CDR, (t_{O,start}, t_{O,end}), (t_{D,start}, t_{D,end}))$

```
CM[O,D]
for each Subscriber $S$ do
  for $i = 1, ..., |CDR|$ do
    if $time(CDR_i) \in [t_{O,start}, t_{O,end}]$ then
      $O = location(CDR_i)$
      for $j = i, ..., |CDR\ within\ 24h|$ do
        if $time(CDR_j) \in [t_{D,start}, t_{D,end}]$ then
          $D = location(CDR_j)$
          $CM(O, D) + +$
        end if
      end for
    end if
  end for
end for
for each pair $(O_i, D_j)$ do
  $CM[O_i, D_j] = CM(O_i, D_j) / \sum_{j=1}^{j=N} CM(O_i, D_j)$
end for
```

Figure 1: ***CMTAR algorithm for the construction of an O-D matrix using Temporal Association Rules (TAR).***

rule $X \rightarrow Y$ is characterized not only by its support and confidence, but also by time intervals at which items X and Y need to happen *i.e.,* $X[T_O] \rightarrow Y[T_D]$, where T_O is the time interval when the antecedent (or origin O) has to happen and T_D the time interval when consequent (or destination D) has to happen. Also while in traditional Association Rules, antecedents and consequents can have more than one element, in our approach X and Y contain just one element, i.e. one geographical area, indicating the Origin(O) and the Destination(D).

In order to reveal commuting patterns from CDRs, we seek to identify the temporal association rules whose confidence represents the percentage of individuals that are at an origin location O_i during a time interval $T_O = [t_{O,start}, t_{O,end}]$ and move to a destination location D_j where they are present during a time interval $T_D = [t_{D,start}, t_{D,end}]$, formally:

$$O_i[t_{O,start}, t_{O,end}] \rightarrow D_j[t_{D,start}, t_{D,end}] \quad (3)$$

Note that $t_{O,end}$ happens before $t_{D,start}$. In our framework, O_i and D_j represent geographical regions and the temporal association rules will either reveal commuting patterns from home to work locations (with O=home location and D=work location) or work to home commutes (with O=work and D=home).

In order to construct a commuting matrix CM, we propose CMTAR, a TAR-based algorithm (see CMTAR Algorithm in Figure 1) that receives as input a set of CDRs and a pair of time intervals T_O and T_D. The algorithm produces as output a Commuting Matrix obtained from CDR records (CM_{CDR}) for the corresponding time intervals. CMTAR identifies for each subscriber S within the CDR dataset, all the pairs $O_i \rightarrow D_j$ such that O_i happens within the interval $[t_{O,start}, t_{O,end}]$ and D_j happens no later than 24 hours within the interval $[t_{D,start}, t_{D,end}]$. Each element of the commuting matrix $CM_{CDR}[O, D]$ is populated with the confidence values associated to each Temporal Association Rule (TAR) $O_i \rightarrow D_j$, with $i, j = 1, ..., N$ (see Equation (2)).

From an implementation perspective, we have implemented CMTAR using a modified *Apriori* algorithm designed to capture the temporal characteristics of TAR. The algorithm assumes that the set of CDRs are grouped for each subscriber S by date and time, being $|CDR|$ the number of CDR entries.

4.2 Optimizing Time Intervals

CMTAR constructs a Commuting Matrix CM_{CDR} using CDR and a set of time intervals that define the Temporal Association Rules. The problem is how to identify which temporal ranges best capture the behavioral fingerprint for the commuting matrix. The objective is to identify the time intervals for the origin and destination of the Temporal Association Rules (T_O and T_D) that produce a Commuting Matrix from CDR (CM_{CDR}) as similar as possible to the original Commuting Matrix provided by the corresponding National Statistics Institute (CM_{NSI}).

A first approach could use brute force to test all possible time intervals, and compute the similarity between CM_{CDR} and CM_{NSI}, being the best solution the one with the highest similarity value. However, due to the large amount of CDR data such approach is not computationally feasible. We propose to use optimization techniques to identify the optimal time intervals that best characterize the commuting patterns. In the following sections, we will present the use of Genetic Algorithms (GA) and Simulated Annealing(SA) to implement the optimization process. Both techniques have been shown to be useful in similar problems [9], and although they are both stochastic, they explore the candidate populations using significantly different approaches.

In our context, for each pair of time intervals T_O and T_D that the optimization technique evaluates, we first need to compute CM_{CDR} using the CMTAR algorithm. In order to evaluate its accuracy, we measure the similarity between CM_{NSI} and CM_{CDR}. As explained, each row in CM_{CDR} represents the set of confidence values for the corresponding TARs for all commutes departing from each geographical area O_i to any destination location ($O_i \rightarrow D_*$). Similarly, each row in CM_{NSI} represents the confidence of the associated TAR from each geographical area O_i to geographical areas D_*. Thus, in order to evaluate the accuracy of CM_{CDR} we need to evaluate the similarity of each row with the corresponding row of CM_{NSI}. For that purpose, we use Pearson's correlation[14] to analyze the similarity between each origin location O_i in CM_{CDR} with CM_{NSI} and the final similarity value is given by the average Pearson correlation across all origins. Formally the similarity between CM_{NSI} and CM_{CDR} is obtained as:

$$c(O_i) = Pearson(CM_{CDR}[O_i, D*], CM_{NSI}[O_i, D*] \quad (4)$$

$$similarity = \sum_{i=1}^{N} |c(O_i)|/N \quad (5)$$

4.2.1 Optimizing Time Intervals with GA

Genetic Algorithms (GA) are search algorithms based on the mechanics of natural selection tailored for vast and complex search spaces [2]. A GA starts with a population of abstract representations (called chromosomes) of candidate solutions (individuals) that evolves towards an improved sets of solutions. A *chromosome* is composed of several genes that code the value of a specific variable of the solution. Each gene is typically represented as a string of 0s and 1s. During the evolution, individuals from one generation are used to form a new generation, which is (hopefully) closer to the optimal solution. GAs use a fitness function in order to evaluate the quality of the solution represented by a specific individual. In each generation, GA creates a new set of individuals obtained from recombining the fittest solutions of the previous generation (crossover), occasionally adding random new data (mutation) to prevent the population from stagnating. This generational evolution is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

In the context of identifying the best time intervals for constructing CM_{CDR}, GA takes as input the set of phone calls (CDRs) from a geographical region and CM_{NSI}, that defines the optimization objective. Each candidate solution produced by GA is designed to capture the time intervals at which commuters call from origin and destination locations. In order to do that, we define a chromosome composed of four different genes. The first two genes represent the starting time and the finishing time at which subscribers make phone calls from the origin locations O. The last two genes represent the starting time and the finishing time at which subscribers make phone calls from destination locations D. Each gene is composed of five bits, which accounts for the 24 hours of the day. Given that we require that $[t_{O,start}, t_{O,end}]$ happens before $[t_{D,start}, t_{D,end}]$, whenever the newly computed chromosomes does not satisfy this restriction, we assume that T_O happens the natural day before T_D.

The fitness of each candidate solution is evaluated using Equation (5), i.e. we define the fitness function as the accuracy of the mobility matrix CM_{CDR} with respect to the NSI mobility matrix, CM_{NSI}. As a first step to evaluate the fitness of a candidate solution, CM_{CDR} has to be generated using CMTAR algorithm with the time slots defined by the genes of the candidate solution.

For example, if a candidate solution proposed by the GA has the values [(06,09),(17,22)], CMTAR computes the temporal association rules $O_i \rightarrow D_j$ that represent calls made or received at location O_i during a morning interval (6am to 9am) and at location D_j during a night period (5pm to 10pm). The confidence values are then used to generate CM_{CDR}, whose fitness is evaluated using CM_{NSI} with Equation (5).

4.2.2 Optimizing Time Intervals with SA

Simulated Annealing (SA) is a probabilistic method designed to find the global minimum of a cost function that may posses several local minima[11]. It works by emulating the physical process whereby a solid is slowly cooled so that its structure is frozen at a minimum energy configuration [3].

The SA metaheuristic starts from a random initial configuration and seeks to find solutions that minimize an energy function $E(x)$ as the temperature T decreases. At each step, the solution explored is accepted as long as the Acceptance Probability Function (APF) that depends both on the energy and on a varying temperature has a higher value than a randomly selected number:

$$P(E(s), E(new), T) > random(0, 1) \quad (6)$$

The APF is selected such that the smaller the value of T the less "uphill" solutions are allowed to be explored, and as T decreases, the more the "downhill" solutions are favored. Such an approach guarantees that the process does not get stuck in local minima reaching a good approximation to a global minimum. This process is repeated multiple times at each temperature value to allow the system to stabilize before decreasing T again.

In our context, SA takes as input CDRs and CM_{NSI}, and outputs CM_{CDR} and the intervals $T_O = [t_{O,start}, t_{O,end}]$ and $T_D = [t_{D,start}, t_{D,end}]$ that best characterize commuting patterns. For that purpose, SA explores randomly selected time intervals seeking the ones that decrease the candidate's energy $E(x)$ until a global minimum is found. Each candidate solution explored by SA is defined as a set of two intervals, one representing the time interval at origin $[t_{O,start}, t_{O,end}]$ and another representing time interval at destination $[t_{D,start}, t_{D,end}]$. Each time in the intervals is represented as a number in $[0, 24]$, checking that T_O happens before T_D. If this condition is not satisfied, the process assumes that T_O happens the natural day before T_D.

Whenever SA explores a new candidate solution, it randomly selects for each time t of each slot a new value from its neighborhood. Given that SA seeks to minimize the energy function, we define it as one minus the correlation coefficient between CM_{CDR} and CM_{NSI} obtained by Equation (5). Finally, the temperature T is decreased following a geometric decrement such that $T_{new} = \alpha * T_{old}$.

5. EXPERIMENTAL EVALUATION

In this section we present an evaluation of the mechanism we have proposed to generate the commuting matrix for the region of Madrid using CDR data. The state has a population of 6.5M and a size of 8,000 Km^2, with the city of Madrid concentrating 3.3M in population, and the rest corresponding to the 48 municipalities in which the region is divided. Figure 2 presents the map of the region and the division in municipalities.

5.1 Datasets

We have used two sources of information from the year 2009: (1) the NSI mobility matrices for the state of Madrid and (2) a CDR dataset of cell phone calls made and received in the state.

NSI matrices represent the home-to-work ($CM_{NSI}[H, W]$) and work-to-home ($CM_{NSI}[W, H]$) commuting patterns during 2009 for the 48 municipalities shown in Figure 2. These municipalities are considered as the Origin O and Destination D sets. Such

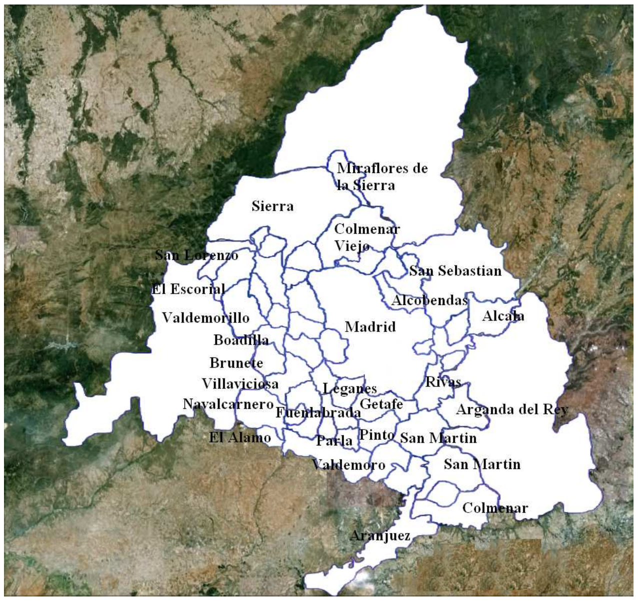


Figure 2: ***Geographical division of the municipalities in which the region of Madrid is divided (including the names of some of them). The association between BTS towers and geographical areas is defined by this borders.***

matrices were built by the local NSI after gathering information regarding the municipality where a person lived and the municipality where a person worked.

The second source of information is a CDR dataset that contains all phone calls, SMS and MMS, that were collected from BTS towers located in the state of Madrid during October and November of 2009, which account roughly for 3.5M unique phones and around 300M interactions. This dataset also includes the geolocation of the BTS towers. In order to filter out mobility patters not related to commuting, we only consider CDR data from Monday through Thursday. Similarly, all bank holidays were filtered. From the two months of traffic available for this study, we will use the data from October for the optimization process, and the data from November will be used to validate the results.

In order to guarantee privacy we implemented a set of elements: (1) All records were anonymized; (2) Data collection and anonymization was done by a third party that was not involved in the analysis; (3) No individual demographic data was available or requested for this study and (4) The information presented is always aggregated in order to further guarantee privacy.

5.2 GA and SA: Configuration

Genetic Algorithms and Simulated Annealing are used to search for the temporal intervals that best represent the times at which people commute using CDRs for the Madrid region. We carry out a total of four experiments: (1) the construction of $CM_{CDR}[H, W]$ using Genetic Algorithms and (2) using Simulated Annealing; and (3) the construction of $CM_{CDR}[W, H]$ using Genetic Algorithms and (4) using Simulated Annealing. The optimization process is the same in all cases, but while the first two use $CM_{NSI}[H, W]$ as the goal of the optimization, the second two use $CM_{NSI}[W, H]$.

For the experimental evaluation, we have used the JGAP implementation of Genetic Algorithms [13] and our own implementation of Simulated Annealing following the description presented in [3]. Both approaches use the CMTAR Algorithm to construct CM_{CDR} for each set of time slots considered, which we have implemented in Java.

Size	Temporal Range	Correlation
10	[20, 21][9, 16]	0.8050
20	[20, 21][9, 10]	0.8219
50	[20, 21][9, 10]	0.8219

Table 1: Optimization results when using Genetic Algorithms for the home-to-work [H,W] commuting matrix.

Size	Temporal Range	Correlation
10	[14, 16][20, 24]	0.9029
20	[15, 16][20, 24]	0.9029
50	[15, 16][20, 23]	0.9059

Table 2: Optimization results when using Genetic Algorithms for the work-to-home [W,H] commuting matrix.

In our experiments, GA uses a distributed architecture where a set of 16 genetic algorithms are run in parallel to explore the quality of different time intervals. Specifically, each process is initialized with a randomly generated population of a set of individuals. At every generation, the reproduction is carried out for a 90% of the total population; the crossover is executed with a 35% of pairs of the selected population by randomly selecting a gene in each individual and exchanging its content with its partner; and the mutation is executed for each gene with a probability of 1/12 and by randomly creating a new gene. The fittest individual is always moved to the next generation, and all the other individuals have a probability of being brought to the next generation proportional to their fitness value. Each process is executed on one core and runs in parallel with the other processes in our architecture of dual-core Intel processors. For our experiments we considered three different population sizes $10, 20, 50$.

On the other hand, the SA implementation starts with an initial temperature of $T_0 = 5$ and decreases its value with the function $T_{new} = 0.65 * T_{old}$ until a threshold value of $T_n = 0.1$ is reached. This cooling criteria allows us to explore a sufficiently large amount of temporal intervals without making the process too long. At each temperature, the SA evaluates three different time intervals and keeps the one that yields the best commuting matrix when compared to CM_{NSI}. Finally, we define as neighborhood solutions the set of temporal intervals that are within a range of four hours before and after the last time explored *i.e.*, $t_{new} \in [t_{old} - 4, t_{old} + 4]$. All the parameters here described were selected because they represented the best performing values across a large evaluation set.

5.3 Optimization Results

In this section, we discuss the results after running GA and SA for constructing $[H, W]$ and $[W, H]$ mobility matrices.

Table 1 and Table 2 show the results after applying GAs for the home to work and work to home commuting matrices, respectively. The tables shows the optimum Temporal Range obtained for each population size considered and the value of the fitness function

Temporal Range	Correlation
[21, 22][11, 16]	0.7863
[21, 22][12, 16]	0.7844
[21, 23][10, 16]	0.7840
[21, 23][14, 18]	0.7808

Table 3: Optimization results when using Simulated Annealing for the home-to-work [H,W] commuting matrix.

Temporal Range	Correlation
[10, 16][20, 23]	0.8949
[14, 17][20, 21]	0.8787
[15, 16][20, 23]	0.8781
[10, 17][21, 22]	0.8724

Table 4: Optimization results when using Simulated Annealing for the work-to-work [W,H] commuting matrix.

(given by Pearson correlation). The Temporal Range is expressed by two intervals, the first one indicates the temporal condition for the origin location and the second one for the destination location.

In Table 1 we observe that using CDRs to compute home-to-work commuting matrices for the region of Madrid we achieve correlation rates of up to 0.82 when compared to the NSI matrices (*ground truth*). This result was obtained with an initial population of 20 candidate solutions and for time slots that define origin as the interactions that took place between 8pm to 9pm of the previous day, and destination as the interactions that took place between 9am to 10am. Smaller populations yielded worse correlation results whereas larger populations did not improve the results. On the other hand, Table 2 shows that the work-to-home mobility matrices computed by GA achieve correlation rates when compared to NSI matrices of up to 0.9059 with an initial population of 20 individuals. In this scenario, the algorithm uses the calls made from 3pm to 4pm to detect the origin location and calls made from 8pm to 11pm (of the same day) to identify the destination location.

Tables 3 and 4 present the results obtained when using SA. For the $[H, W]$ commuting matrices, the best coefficient obtained was of 0.7863 with origin location detected between 9pm and 10pm of the previous day and destination location determined from calls made from 11am to 4pm. In the case of the work-to-home commuting matrices, the highest correlation coefficient is of 0.8949 with a temporal range of 10am to 4pm to detect the origin location and 8pm to 11pm to detect the destination location.

In general the correlation values provided by GA are better than the ones provided by SA. Also, we observe that in both cases, the work-to-home commuting matrices are better modelled from CDRs than the home-to-work (0.90 to 0.82 when using GA, and 0.87 to 0.78 when using SA). This result might be related to the fact that people make more cell phone calls during the day than early in the morning or at night, which provides a larger number of *geographical points* to model commutes from work-to-home than vice versa. Also, it might be an indication that the home-to-work commuting follows a less direct route (*e.g.*, taking kids to school), thus adding noise to the available data.

Finally, the average execution time for GA when considering the best solutions obtained for a population of 20 is $2,890$ minutes, while the average processing time for SA for the best solution is $2,699$ minutes.

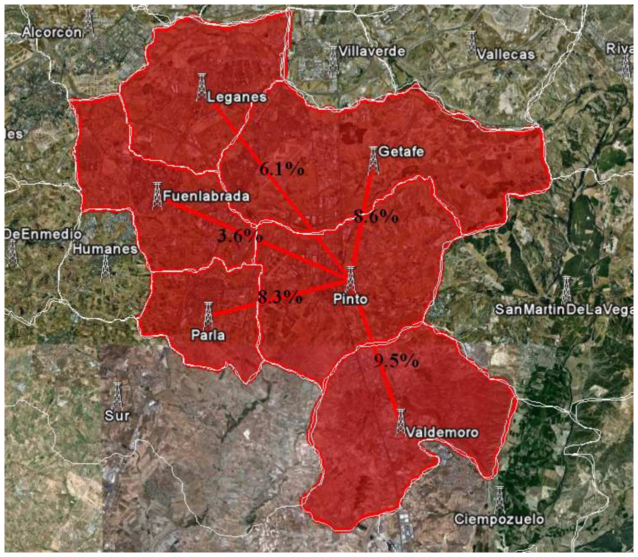


Figure 3: ***Visualization of the Commuting Matrix obtained for the municipality of Pinto in southern Madrid, showing the top five municipalities with the highest confidence value for the work-to-home commuting.***

6. VALIDATION

The experimental results described in the previous section have shown that CDRs can be used to construct commuting matrices that are as good as the one provided by NSI.

In our context, the goal of the validation is to assess whether the time intervals identified for the $[H, W]$ and $[W, H]$ commuting matrices are valid to estimate the commuting matrices of other years, in order to show that CDRs can be used to generate commuting matrices without the need of NSI data. Ideally, the validation process would consider the commuting matrices obtained by the NSI for 2010 and CDR data from 2010, and validate the time intervals using the similarity between CM_{CDR} and CM_{NSI}. Nevertheless, so far, no commuting matrices for 2010 or 2011 have been published by the local NSI.

Considering that limitation, we implement a validation process that uses the 2009 $CM_{NSI}[H, W]$ and $CM_{NSI}[W, H]$ matrices and the November 2009 CDR dataset. The intervals we are going to use are the ones obtained by the GA-based optimization: $[20 - 21][09 - 10]$ for the home-to-work commute and $[15 - 16][20 - 23]$ for the work-to-home commute. Finally, the validation is done by calculating the similarity between the CDR matrix obtained and the NSI matrix using Equation (5).

Table 5 shows for both home-to-work and work-to-home commutes the Temporal Range used, the correlation values obtained during the Optimization process using the October 2009 CDR dataset, and the Validation correlation between CM_{CDR} and CM_{NSI} using the November 2009 CDR dataset (with its corresponding standard deviation). We observe that the Validation correlation coefficients are within a 10% of the correlation values obtained in the Optimiza-

	Temporal Range	**Optimization(Oct09)**	**Validation(Nov09)**
Home-To-Work	$[20, 21][9, 10]$	0.8219	0.765 ($\sigma = 0.46$)
Work-To-Home	$[15, 16][20, 23]$	0.9059	0.9322 ($\sigma = 0.16$)

Table 5: Validation results for the $[H, W]$ and $[W, H]$ commuting matrices obtained with November 2009 CDR data.

Municipality	**% of Population**	**H-W Correlation**	**W-H Correlation**
Madrid	50%	0.9995	0.5818
Alcobendas	2%	0.9885	0.8210
San Fernando	1%	0.9120	0.7411
Moraleja de Enmedio	0.0007%	0.0935	0.9895
Villa Conejos	0.0004%	0.1256	0.9972

Table 6: Individual Correlation values for H-W- and W-H for a set of representative municipalities.

tion process. It is noticeable that in the case of the work-to-home commuting there is a slight increment in the correlation, which, in line with the results discussed in the previous section, being probably caused by an increase of the CDR data available during the time slots considered.

These results show that, although with some differences, the optimization process provides a good approximation of the time intervals needed to compute commuting matrices, and as a result future commuting matrices can be directly estimated from CDR data. This allows for constructing O-D matrices with much more frequency at a fraction of the cost. The reason for the different values between the NSI- and the CDR-generated matrices is mainly caused by the fact that the NSI generates the commuting matrix strictly using individuals that have a declared work location. As a result, CM_{NSI} does not capture any non-work related mobility (which in itself is very difficult to capture using questionnaire-based approaches). Our CDR approach captures all types of mobility (work, leisure, shopping, students, etc.), so the fact that using CDR data we can not completely correlate the results with the NSI is because our matrix contemplates more situations and as such is more realistic.

6.1 Commuting Patterns by Municipality

The correlation coefficient between CM_{CDR} and CM_{NSI} represents an average value between each individual row-to-row correlation. In an attempt to understand the commuting patterns for individual municipalities, we compare the rows of each CDR-based mobility matrix with the rows of its NSI counterpart. Our objective is to do a preliminary study to understand whether there are stronger correlations between both matrices for specific municipalities. Figure 3 presents a visualization of the work-to-home commuting matrix CM_{CDR} for the municipality of Pinto using November 2009 CDR data. It shows the top five TARs with the highest support, *i.e.,* the top municipalities where people that work in Pinto live.

Table 5 shows that the standard deviations for home-to-work and work-to-home correlations are 0.46 and 0.16, respectively. These results reveal that there exist large differences in the correlation values across municipalities, especially for the home-to-work commuting patterns. Table 6 presents the individual correlation coefficients for a set of representative municipalities for the home-to-work and work-to-home commute, including the percentage of the population than they represent. We can observe that the home-to-work correlation coefficients are higher when the municipality has a large number of citizens, *i.e.,* larger cities tend to have more predictable home-to-work commuting patterns than smaller ones. On the other hand, larger municipalities tend to be less predictable in their work-to-home commutes (have smaller correlation values) than smaller towns. This is probably due to the fact that in larger cities citizens tend to do other activities once they get out of work as opposed to smaller towns where people tend to go directly to home. Thus, although on average home-to-work patterns appear to be less predictable than the work-to-home ones (as shown in Tables 1 and 2), that is only the case for small municipalities. In large ones, the opposite holds, whereby the larger the city, the more predictable the home to work mobility matrices are (when compared to the work to home mobility matrix).

These preliminary results seem to indicate that incorporating the size of the municipalities in the optimization process could improve the final correlation values. Also, we consider that having more data to generate the O-D matrix will, to some extent, mitigate the current limitations regarding the predictability of small municipalities (consider that because we only use Monday through Thursday, in the end we have 17 days of traffic for the optimization process).

7. RELATED WORK

The construction of O-D matrices has been typically studied by transportation and urban planning research. Traditional solutions are based on questionnaires and/or in the combination of questionnaires with traffic information. Such solutions typically focus on generalization techniques that construct matrices from partial data. The main approach used to obtain traffic data information is electronic toll collection[10]. This approach is limited because the information provided only reflects a partial view of the route. A possible solution for these limitations is the use of GPS data. In this case, the information contains complete routes but the amount of data available is even more limited [17]. The studies done up to now focus mainly on GPS data available from taxi or bus fleets[17] which highly limits the conclusions.

The use of CDRs to model commuting patterns solves to a large extent the previous limitations. A variety of studies can be found in the literature: Caceres et al.[4] uses GSM simulated traces to construct origin-destination data to measure the flow of vehicles, Zhang et al. [18] presents a model to transform cellular counts into vehicular counts in order to construct commuting matrices, and Sohn et al. [15] introduced cell phone probes in the network to identify trajectories and estimates O-D matrices using handoffs. Our approach has a set of differential factors with these previous studies: (1) we use CDR data that does not contain any handoff information. Handoff information consists on storing the sequence of towers used during a conversation and although they provide more information, cell phone operators do not keep such data due to privacy concerns (also consider that information would actually be useful only if using cell-phones was allowed while driving); and (2) our approach focusses on showing that the traditional questionnaire-based approaches for estimating O-D matrices can be approximated by the technique we present. While the state of the art mainly presents techniques to construct O-D matrices and assumes that the quality of the data will imply good results, in our case the technique we propose uses the information contained in questionnaire-based O-D matrices to tune the parameters. From this perspective, our work has elements in common with Calabrese et al. [5] in the sense that the validation of the technique is done with external O-D matrices. The difference in our case is that we also use that same information to identify the best parameters to construct O-D matrices with CDR in order to approximate the values of traditional approaches. This allows us to present a technique that can be adapted to capture the different cultural schedules of different urban areas.

Some authors identify the construction of O-D matrices using CDR as the identification of home and work for each user, using that information to aggregate origin-destination patterns. The work by Frias-Martinez *et al.* [7], Isaacman *et al.*[8] and Calabrese et al. [5] present algorithms to detect home and work by identifying highly used cell-phone towers. Nevertheless the use of such algorithms has strong limitations that affect the construction of O-D matrices, mainly: (1) the error introduced by the algorithms in the estimation of the locations (which in general is not measurable due to the lack of ground truth data); and (2) the fact that the coverage is limited by the availability of information for each user, *i.e.*, home and work can only be detected for individuals that have a minimum amount of interactions with their cellphone. Depending on the context, this requirement can filter more than 80% of individuals[7], with the corresponding bias in the final matrix.

8. CONCLUSIONS

Traditional methods for the estimation of mobility matrices suffer from a variety of limitations, mainly the bias of the information collected and the cost of gathering such information. To overcome these issues, we have presented a method based on the data collected by cell phone infrastructures to generate commuting matrices. In the literature we can find similar approaches, but in our case we have focussed our study on showing that we can replicate the information contained in questionnaire-based O-D matrices.

Our approach is implemented with CMTAR, a TAR-based algorithm designed to construct commuting matrices from CDR data. The combination of CMTAR with optimization techniques provides an approach that identifies which parameters need to be used to construct commuting matrices that are as similar as possible to the NSI matrices. Our experimental evaluation and validation has showed that we can compute commuting matrices with a high level of accuracy using CDR, and as a result our CDR generated matrices can be used for the same purposes as traditional matrices.

9. REFERENCES

[1] Agrawal, R., Imielinski, T., Swami, A.N. . Mining association rules between sets of items in large databases . In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

[2] Goldberg, D. In *Genetics algorithms in search optimization and machine learning, Addison Wesley*, 1989.

[3] Bertsimas, D., Tsitsiklis, J. Simulated Annealing . *Statistical Science*, 8(1):10–15 , 1993.

[4] B. F. Caceres N., Wideberg J.P. Deriving origin destination data from a mobile phone network. *Intelligent Transport Systems, IET*, 1(1):15–26, 2007.

[5] Calabrese F., Di Lorenzo G., Liu L., Ratti C. Estimating origin-destination flows using mobile phone location data. *IEEE Pervasive Computing 10(4)*, pages 36–44, 2011.

[6] Frias-Martinez, E. and Karamchety, V. A Customizable Behavior Model for Temporal Prediction of Web User Access Sequences . In *LNAI 2703*, page , 2003.

[7] Frias-Martinez, V. and Virseda, J. and Rubio, A. and Frias-Martinez, E. Towards large scale technology impact analyses: Automatic residential localization from mobile phone-call data . In *Int. Conf. on Inf. & Comm. Technologies and Development (ICTD)*, 2010.

[8] Isaacman, S. and Becker, R. and Cáceres, R. and Kobourov, S. and Martonosi, M. and Rowland, J. and Varshavsky, A. Identifying important places in peoples lives from cellular network data. *Pervasive Computing*, pages 133–151, 2011.

[9] A. R. Kianmehr K. A fuzzy prediction model for calling communities. *Int. J. Netw. Virtual Organ.*, 8(1/2):75–97, 2011.

[10] Kwon, J., Varaiya, P. Real-Time Estimation of Origin-Destination (O-D) Matrices with Partial Trajectories from Electronic Toll Collection Tag Data. *Transportation Research Record no. 1923*, pages 119–126, 2005.

[11] Laarhoven, P.J.M., Aarts, E.H.L. Kluwer Academic Publisher. In *Simulated Annealing: Theory and Applications*, 1988.

[12] Mannila H., Toivonen H., Inkeri Verkamo A. . Discovery of Frequent Episodes in Event Sequences . *Data Mining and Knowledge Discovery*, 3(1):259–289 , 1997.

[13] Meffert, Klaus et al. . JGAP - Java Genetic Algorithms and Genetic Programming Package . In *http://jgap.sf.net*, 2008.

[14] J. Rodgers and W. Nicewander. Thirteen ways to look at the correlation coefficient. *American Statistician*, pages 59–66, 1988.

[15] K. Sohn and D. Kim. Dynamic origin–destination flow estimation using cellular communication system. *Vehicular Technology, IEEE Transactions on*, 57(5):2703–2713, 2008.

[16] U.S. Census Bureau. *www.census.gov*, 2011.

[17] Veloso M., Phithakkitnukoon S., Bento C., Fonseca N.,Olivier P. Exploratory study of urban flow using taxi traces. In *The First Workshop on Pervasive Urban Applications (PURBA)*, 2011.

[18] Zhang Y., Qin X., Dong S., Ran B. Daily O-D Matrix estimation using cellular probe data. In *89th Annual Meeting Transportation Research Board*, 2010.

Identifying users profiles from mobile calls habits

Barbara Furletti
KDDLAB - ISTI CNR
Pisa, Italy
barbara.furletti@isti.cnr.it

Lorenzo Gabrielli
KDDLAB- ISTI CNR
Pisa, Italy
lorenzo.gabrielli@isti.cnr.it

Salvatore Rinzivillo
KDDLAB - ISTI CNR
Pisa, Italy
salvatore.rinzivillo@isti.cnr.it

Chiara Renso
KDDLAB - ISTI CNR
Pisa, Italy
chiara.renso@isti.cnr.it

ABSTRACT

The huge quantity of positioning data registered by our mobile phones stimulates several research questions, mainly originating from the combination of this huge quantity of data with the extreme heterogeneity of the tracked user and the low granularity of the data. We propose a methodology to partition the users tracked by GSM phone calls into profiles like resident, commuters, in transit and tourists. The methodology analyses the phone calls with a combination of top-down and bottom up techniques where the top-down phase is based on a sequence of queries that identify some behaviors. The bottom-up is a machine learning phase to find groups of similar call behavior, thus refining the previous step. The integration of the two steps results in the partitioning of mobile traces into these four user categories that can be deeper analyzed, for example to understand the tourist movements in city or the traffic effects of commuters. An experiment on the identification of user profiles on a real dataset collecting call records from one month in the city of Pisa illustrates the methodology.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

General Terms

Algorithms

Keywords

GSM Data, User profiles, SOM

1. INTRODUCTION

4.4 billions or users worldwide, 838 GSM networks spread in 234 countries, 1.44M new GSM subscribers every day are only a few of the impressive numbers that witnesses the enormous diffusion of the GSM phenomena since its first network launched at the beginning of '90 [7]. This massive quantity of mobile phones are moving everyday with their human companions, leaving tracks of theirs movements. These tracks represents the mobility of millions of people in the Earth surface and the opportunities to use these data for analysing and understanding human mobility are tremendous. Research literature has seen a growing interest in techniques for analysing mobility of users based on GSM position data. This research has also been driven by an increasing number of applications that has found in mobile phone data a good partner for discovering interesting results on people behavior. The advantages of relying on these kind of data, compared to standard survey based data collection, is that they offer a wide coverage of the people presence in an area, they are heterogeneous from the point of view of the tracked person and they tend to be up to date and easily upgradeable with new automatic data collection. However, these huge quantity of humans location data comes with a price. Due to privacy reasons, the telecommunication provider must anonymize the data. Thus the analysis that can be done on such data does not distinguish the different user profiles. Therefore the heterogeneity of these data, besides being a strong point, is also a weak point. The mobility analysis that can be performed may suffer from biases due to the wide difference in mobility behavior of tracked users. How to determine, among all the positions collected in a city, which ones correspond to specific categories of users such as residents or visitors? Is it possible to distinguish them looking at they mobile usage?

In this paper we face this problem proposing a methodology to partition a population of users tracked by GSM mobile phones into four predefined user profiles: residents, commuters, in transit and tourists/visitors. Several applications may benefit from the analysis of a partitioned set of users based on this mobility characteristics. For example, being able to distinguish between residents and commuters may help in traffic management to better understand how traffic is affected by the residents mobility compared to the commuters. Having identified the tourists/visitors, it is essential to study how the city is receiving people from outside and how their movements are affecting the city. Again, being able to combine the mobility of resident population with the temporary population (like commuters, visitors or people in transit) may give a measure of the sustainability of the incoming population with respect to resident one. The population on a territory consumes resources like water, air and produces negative effects on the surroundings, like garbage, pollution, noise. In the cases where these resources are limited, the incoming tourist population may break the sustainable equilibrium of such resources. Thus, the ratio between residents and incoming people should be monitored in order to prevent critical situations.

The methodology introduced here aims at inferring the population profile among the GSM call positioning data, namely GSM Call Records (CDR) identifying, with a certain degree of approximation, which calls may correspond to predefined users categories among residents, commuters, in transit and tourists. The basic idea of the methodology is to perform two steps: the top-down step and the bottom-up step, thus combining the deductive power of queries on the call records to the inductive power of a machine learning step based on the SOM [5] technology. More in particular, the top down step tends to identify classes of users based on a predefined call behavior that may approximate a given typology of users. Just to give an example, users that generally call any time during the day or night for a long period may be considered resident, while users that only calls in a restricted period may be considered in transit or tourists. Since the border between these definitions may be not so crisp when data are sparse (e.g. how clearly to distinguish between tourists and in transit, or with a person that makes few calls?) a bottom-up step is performed to compute sets of users with similar calling behavior integrating the results of the top-down step.

The profiling methodology we are proposing is accompanied by an experiment studying the behavior of user profiles analysing mobile phone positioning data, namely GSM Call Records (CDR). Each dataset collects the (anonymized) records storing the location and duration of calls of mobile phone users in the GSM network. The dataset of CDR has been collected by an Italian telephone company in the area of Pisa, in Tuscany, Italy. The city counts about 90,000 inhabitants and it is the location of an ancient and prestigious University that attracts more than 10,000 students from everywhere in Italy. Pisa is well known all over the word as being a tourist attraction for its leading tower. There are estimations that every year one million tourists visit Pisa. Therefore this area is suitable to perform an experiment in trying to infer the different users profiles moving in the city.

The structure of the paper follows. After a selection of related works presented in Section 2, Section 3 introduces some basic concepts used though the paper like the Call Records definition and the user profiles. The methodology is introduced in the following Section 4 where the two steps top-down and bottom-up are presented. The experiments are illustrated in Section 5, while Section 6 draws the conclusions.

2. RELATED WORKS

The use of GSM traces for studying the mobility of users is a growing research area. An increasing number of approaches propose to use GSM data for extracting presence and/or movement patterns.

Famous experiments on analysing GSM data for studying people movement have been run on Rome [3] and Graz [11]. They use GSM data to realize a real-time urban monitoring systems. They get detailed real time data by installing additional hardware on top of the existing antennas to get an improved location of the users in the networks. The final objective is to realize a wide range of services for the city such as traffic monitoring and tourists movement analysis.

A different approach comes from Schlaich et al. [12] where the authors exploit the GSM handover data - the aggregated number of users flowing between cells - to perform the reconstruction of vehicles trajectories. The objective is to study the route-choice-behavior or car drivers in order to determine the impact of traffic state.

Another use of GSM data is the identification of interesting users places as in [1], where the authors propose a method for the identification of meaningful places relative to mobile telephone users, such as home and work points. They use GSM data (both calls and handovers) collected by the phone operator. The localization precision is the cell which is the same accuracy level of the identified interesting points. They distinguish between personal anchor points like home, work and other person-related places as the locations each user visits regularly, as for example a gym.

In Pereira et al. [9], the authors exploit cellular phone signaling data[1], focusing on the prediction of travel demand for special events. Similar to the previous approach, their analysis identifies the home location: here is defined as starting point of people's trips. However, they observed that mobility data are dependent on mobile phone usage, and this may bias the results. Therefore their proposed to integrate the GSM dataset with external data (e.g. ticketing statistics or taxi trips) with the aim of increasing the quantity and the quality of the data, in particular in term of spatial resolution.

Quercia et al. [10] uses GSM data for recommending social events to city dwellers. They combine the locations estimated by mobile phone data of users in the Greater Boston area and the list of social events in the same area. After extracting the trajectories and stops from GSM calls, they crawled the events from the web. Then, they divide the area of Boston in cells and locate each events and each stop in the corresponding cell. Therefore, by crossing the events and the stops, they identify a set of potential users participating to events.

Mobile phone records are analysed also in [2] where the authors propose a visual analytics framework to explore spatio-temporal data by means of SOM (Self-Organizing Map) analysis. They propose a method to cluster the dataset by either of the two dimension and evaluate the resulting aggregation on the other one. Altough they show the potentialities of using SOM for analysing mobile phone records, they do not focus on identifying user profiles.

All these approaches, as well others that can be found on the literature offer different perspectives on how GSM data can be exploited to study the human mobility and the huge potentialities of these kinds of data. Differently from these approaches, the aspect we want to study in this paper is to characterize the user profile based on the call habits of the tracked users.

There is a broad research area that focuses on the inference of significant places or activities from mobility data represented as GPS trajectories [8]. Examples of these works are in [15, 6]. In paper [6] the authors infer activities carried out by moving people (e.g. AtHome, Shopping) and the transportation mode. The inference of activities is based

[1]These data consist of location estimations which are generated each time when a mobile device is connected to the cellular network for calls, messages and Internet connections.

on temporal patterns (since different activities have different temporal duration), the location where people stopped, transition relations, since one activity may or may not be followed by another, and a number of common sense constraints. Authors of the paper [15] infer the similarity between users based on their GPS trajectories. They associate to a user trajectory the semantic location history - the sequence of Points of Interests visited - that is used to measure the similarity between different users. However, the GPS data offers a better spatio-temporal granularity level compared to GSM data so many of the techniques available for GPS cannot be used for GSM call record and new methods have to be invented.

3. BASIC CONCEPTS

The objective of this work is to propose a methodology for user profiling in GSM data. We propose a method to infer a possible segmentation of a population of GSM users into different behavior categories. This is an essential step for better understand and study people mobility from unsupervised mobility data.

Indeed, being able to differentiate population ranges enables a number of new applications where the mobility behavior is relative to a specific user segmentation. However, when the mobility data is not directly annotated with the user profile, the association of an anonymous trajectory to a given segment is far to be trivial.

The strategy we propose here is based on the identification of *residents*, *commuters*, *people in transit* and *visitors/tourists* from GSM call records.

GSM network and Call Data Records.

GSM (Global System of Mobile communications) network allows the mobile phone communications based on a system of antennas that transmit the signal to a spatial area that is called Local Area Network. All mobile phones inside that area may receive the signal and therefore are connected to the network. When they are connected they are enabled to make calls or send SMS (Short Text Messages) to another GSM phone connected to the network. When a call is engaged, the telephone company registers data about the location and duration of the call for billing purposes. These data are called Call Data Record (CDR) [13] and we can simplify the standard format as follows:

$< Caller\_ID, ID\_Cell\_Start, Start\_Time, ID\_Cell\_End, Duration >$

where: *Caller_ID* is the anonymous identifier of the caller, *ID_Cell_Start* and *ID_Cell_End* are the identifiers of the cell where the call starts and ends respectively, *Start_Time* is the date and time when the call starts, and *Duration* is the call duration.

Mobile users profiles.

We are interested in inferring the profile of users moving in a city. For the sake of the current study, stated A the spatial area under analysis, the categories we are interested in, are the following:

Resident. A person is resident in an area A when his/her home is inside the A. Therefore the mobility tends to be from and towards his/her home.

Commuter. A person is a commuter between an area B and an area A if his/her home is in B while the workplace is in A. Therefore the daily mobility of this person is mainly between B and A.

In Transit. An individual is "in transit" over an area A if his/her home and work places are outside area A, and his/her presence inside area A is limited by a temporal threshold T_{tr} representing the time necessary to transit through A. In other words, the user does not perform any main activity inside A. Depending on the application this temporal threshold T_{tr} may vary from few minutes to few hours.

Tourist or Visitor. The definition given by The World Tourism Organization defines tourists as people "traveling to and staying in places outside their usual environment for not more than one consecutive year for leisure, business and other purposes" [14]. We can rephrase and formalize this definition as: a person is a tourist in an area A if his/her home and work places are outside A, and the presence inside the area is limited to a certain period of time T_{to} that can allow him/her to spend some activities in A. In particular here the presence has to be concentrated in a finite temporal interval inside the time window. Should also be "occasional" therefore, he/she does not appear anymore during the observation period. It is also important to point out the distinction that this definition includes not only the classical "tourism" as visiting cultural and natural attractions, but also the activities related to work, visiting relatives, health reasons, etc.

4. METHODOLOGY

The proposed analytical process is based on a step-wise approach: first, domain knowledge is used to label each user according to a set of rules that define each profile; second, the profiles that do not fit in any of the hypothesis templates are analyzed by means of a machine learning approach to determine relevant groups of users according to their calling behavior. Therefore when an individual makes (at least) a phone call inside a network cell we say this individual is *present* into the cell area. The presence pattern is then defined by temporal constraints on the detected presence. However, these definitions combined with the characteristics of the GSM call data may give misleading classifications. For example, a resident user who rarely calls may be misclassified as a tourist or a person in transit, while a resident that only use phone at work may be classified as a commuter. Again, defining a good threshold to identify tourists may be difficult and certainly depends on the application. Although the "in transit" profile is well defined once a temporal threshold is fixed, the other profiles, especially the "tourist" population, is characterized by a "fuzzy" and non clear characterization.

To face this problem the user profile methodology we introduce here proposes the combination of a deductive and an inductive technique that we name *top-down* and *bottom-up*. In the top-down approach a set of spatio-temporal constraints are used to describe the individual categories following the definitions given by the domain experts and that we introduced in Section 3. The constraints are then implemented in the mobility data management and mining system M-Atlas [4], through the use of the provided query language. In the bottom-up step the assignment of users to categories is refined using a clustering algorithm, namely Self Orga-

nizing Map (SOM) [5]. Clearly, since the top-down step is based on a set of rules provided by the domain experts, they may fail at classifying behaviors on the borders of the definitions. Therefore, all those individual that have few phone calls or whose phone calls behavior does not clearly fall into the well-defined categories remain unassigned.
The bottom-up approach aims at integrating the results of the first step by using a data-driven approach to identify relevant group of users that present similar behaviors that can be classified as one of the available profiles.

The advantages of the described technique lies on the fact that GSM data - due to the widespread use of mobile phone and the heterogeneous classes of their users - allows to analyze the mobility behavior of a huge amount of people and a broad range of users categories. Furthermore, the use of an inductive step allows a refinement of the preliminary results obtained with the top-down approach.

Top-down Approach.

During this phase the residents, commuters and in transit categories are retrieved from the CDR dataset with a proposed set of spatio-temporal constraints that depend on the time window of the data collection and reflect the indications given by the domain experts.

Resident users are those whose CDR data show a continuous presence in the monitored period during the late afternoon and night (since we assume that during this period individuals stay at home) and the weekly minimal presence to be reasonable to establish as home.

Users that tend to have a sparse presence of calls during the period but concentrated only during the weekdays in the conventional working/studying times, are classified as commuters. The assumption is that the commuters spend nights at the home place (an area outside our interest) and weekdays at the work/study place in the area of interest, and never appear during the weekends.

People in transit are directly identified by a simple constraint that limits the presence to a fixed time range depending on the dimension of the area under analysis. The constraint tries to encode the average time needed to cross the area without stops for activities. The idea is to capture people passing on motorways and freeways near a city or crossing the city by using urban roads. This gap can vary from less then 1 hour, in case of small towns, up to several hours (two, three or more), in the case of big cities.

For example, a simple query in SQL Like to identify people in transit is the following:
```
SELECT a.Caller_ID
FROM Cdr_Calls_Table a, Coverage_Table b
WHERE (a.Last_Call - a.First_Call) < Time_Limit AND
b.Location = 'Pisa'
```

where `Cdr_Calls_Table` is the table containing the CDR data, `Coverage_Table` is the table containing the spatial coverage of the network cells, `Last_Call` and `First_Call` are the timestamps of the last and first call respectively, and `Time_Limit` is the estimated time needed to cross the area of interest.

Of course a number of users whose call behavior does not precisely falls in these three definitions remains unclassified after this first step, the bottom-up step is thus necessary to analyse the unclassified set of users trying to assign a category basing on a temporal profile.

Bottom-Up Approach.

The bottom-up approach has the twofold purpose of both identifying tourists and refining the results obtained by the top-down phase for residents and commuters.

The behavior of each user is modeled by means of the concepts of space (where a call is started and where it is terminated) and time. We exploit these two dimensions to define a temporal profile for each user.
Given a user u, a *Temporal Profile* TP_u is a vector of call statistics according to a given temporal discretization. For example, using a time discretization by day and a measure of frequency of calls, each entry of TP_u would contain the number of calls performed by the user in the corresponding day. Since we are interested in a specific area, we define a *Space constrained Temporal Profile* TP_u^A as a Temporal Profile where only the calls performed in the cells contained within the area A are considered.
This spatial projection is crucial when studying commuters in order to distinguish the call behavior at work and at home. To explore different time patterns, we define also two time transformations of a (Space constrained) Temporal Profile: (i) *time projection* by a cycle period, where the time intervals of the vector are referred to relative position in a time cycle like week, month, and so on; (ii) *time shifting*, where the time intervals of the vector are shifted in order to have the first entry corresponding to the first activity of the user. Clearly, the available statistics can be chosen according to the specific analytical scenario. For example, it can be considered the number of calls, the total duration of the calls, or a boolean operator that yields true if at least a call has been performed in a specific time period. Figure 1 shows two examples of extraction of temporal profiles from the call behaviors of two users, using call frequency as measure for each cell.

The temporal profiles defined above can be analyzed according to their relative similarities by means of a Self Organizing Map [5]. A SOM is a type of neural network based on unsupervised learning. It produces a one/two-dimensional representation of the input space using a neighborhood function to preserve the topological properties of the input space. As most neural nets, a SOM constructs a map in a training phases using input examples and uses the map for classifying a new input vector. The procedure for placing a vector from data space onto the map is to first find the node with the closest weight vector to the vector taken from data space. Once the closest node is located, it is assigned the values from the vector taken from the data space. SOM forms a sort of semantic map where similar samples are mapped close together and dissimilar apart. In our case the weighted vectors used for the analysis are the Temporal Profiles extracted for each user. For example, using a discretization of one hour, a daily temporal profile for a user consists of a vector with 24 entries. The dimensions of the vector may change accordingly with different aspects of the analysis like, for example, the temporal profile in a week, in a single day, or in the whole period by hour (see Figure 1). The SOM algorithm produces a set of nodes, where each node represents a group of users with similar temporal profile. By analyzing the profile that describes each group, it is possible to

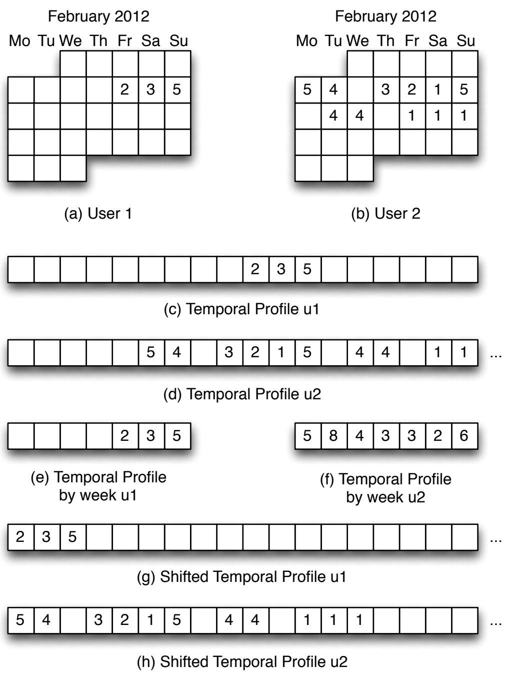

February 2012
Mo Tu We Th Fr Sa Su
(a) User 1
(b) User 2
(c) Temporal Profile u1
(d) Temporal Profile u2
(e) Temporal Profile by week u1
(f) Temporal Profile by week u2
(g) Shifted Temporal Profile u1
(h) Shifted Temporal Profile u2

Figure 1: Example of the extraction of temporal profiles from the call activities of two users. Each square represent a day and the number within a cell is the number of calls of the corresponding user in that day.

assign a class to the group itself, basing on the definitions of resident, commuter and tourist given above. In particular, we label as "tourists" the nodes that correspond to a temporal profile localized in a short and consecutive time period (as for example few days or a week). As stated before the temporal profile of a tourist can not be defined a priori because it has a wide variability depending on the season, the location and other unpredictable events. This method can thus help to discover the touristic degree of an area without using particular apriori knowledge. This phase is useful also to re-calibrate the top-down results. In fact, the analysis of the profiles emerging from the groups may give information about local habits and may suggests how to set the temporal constraints to adapt the model to the local habits of the area of study.

We perform this analysis from two perspectives dependent on the chosen temporal profile. In the first case we extract the temporal profiles for each user and we transform each profile by a left shifting operator in order to make the data more dense (see Figure 1 (g) and (h)), since the users who has visited the area in different time periods can be compared also by the length of their staying. Such operation, on one hand, loses an absolute temporal reference, thus is not directly possible to associate each entry to a specific time period. On the other hand, the SOM algorithm can easily identify group of users with compatible periods of visit to the area. In this way, for example, it is possible to identify the typical duration of presence of the users and, hence, assign the tourist/visitor class to the nodes with compatible temporal profiles. In the second case we extract the original temporal profile according to the absolute time alignment. In this case the resulting SOM tends to highlight similar and compatible presence profile of longer stay people, allowing to separate commuters and residents, by exploiting the calling habits of these users in particular during the weekends.

With respect to other clustering techniques, the SOM allows an easier and clearer visualization of the results. This technique seemed to us very useful for precessing the profiles input extracted from the GSM data, and visualizing the complex results.

5. EXPERIMENTS

We tested the proposed approach on a case study in the city of Pisa. We used a large dataset of GSM data collected in the province of Pisa by one of the Italian mobile operators. The data consist of around 7.8 million CDR records collected from January 9^{th} to February 8^{th} 2012. The data contains calls corresponding to about 232.200 users with a national mobile phone contract (no roaming users are included in the dataset). Our approach is based on a set of temporal constraints over the users' temporal profiles. As a preliminary validation analysis of the method, we analyze the temporal presence of users in the province of Pisa. Figure 2 shows the cumulative distribution of the duration of stay of users in the province: a point (x, y) on the chart represents the number of distinct users y that were observed in the area at most for x days. From the chart we can roughly partition the population on the basis of the domain knowledge: people staying less than four days are candidate visitors or in transit, while the others can be considered as resident/commuters. This is a very naive segmentation, however allow us to estimate how effective this approach is. In particular, using the four-days threshold the candidates resident commuters are around 107k. This number is compatible with the customer statistics provided by the telecom operator in the area, thus informally validating the proposed approach.

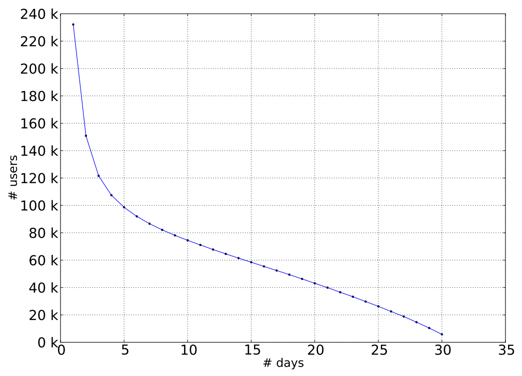


Figure 2: Cumulative distribution of the number of users per length of staying.

Since our aim is to study the mobility of residents and visitors in the area of Pisa, from the whole network we first selected the cells overlapping the urban area of the city. The urban center of the city is crossed by the river Arno and its corresponding cells are highlighted in pink in Figure 3. The larger gray area corresponds to the administrative territory of the city that includes also the seaside and the large park called *San Rossore*. Once we have determined the area of interest, we filtered the calls by considering only those calls performed in the selected cells.

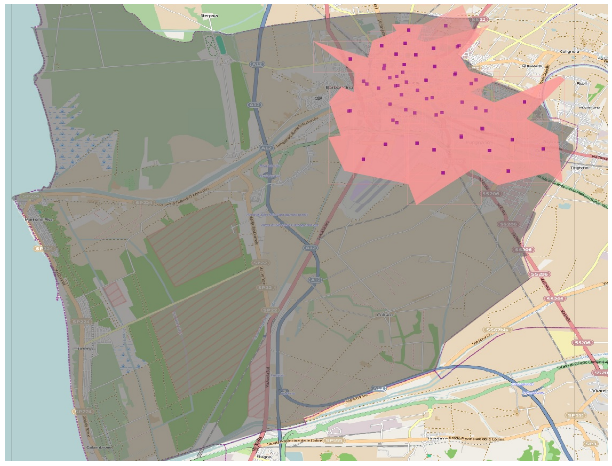


Figure 3: Coverage of the urban area of Pisa.

As mentioned in Section 4, the top-down approach consists in a set of spatio-temporal queries expressed in the query language of M-Atlas [4]. The spatial constraints, which are the same in all the cases, define the area of interest i.e., the urban area of Pisa and the corresponding GSM coverage. This coverage is expressed as a geometric intersection of the base station positions with the urban census surface. The temporal constraints are different for all the categories of users we want to identify as detailed in the following:

Resident
C1 - Temporal range: at least 1 call in [19:00 - 6:59] during the weekdays.
C2.1 - Daily presence: at least 2 distinct weekdays per week, that satisfy C1.
C2.2 - Daily presence: at least 1 day in the weekend without temporal range.
C3 - Weekly presence: at least 3 weeks, in which C1, C2.1 and C2.2 are satisfied.

Commuter
C1.1 - Temporal range: at least 1 call in [9:00 - 18:59] during the weekdays.
C1.2 - Temporal range: no calls in [19:00 - 8:59] during the weekdays.
C2.1 - Daily presence: at least 2 distinct weekdays per week, that satisfy C1.1 and C1.2.
C2.2 - Daily presence: never during the weekends.
C3 - Weekly presence: at least 3 weeks, in which C1.1, C1.2, C2.1, C2.2 and C3 are satisfied.

People in Transit
C1 - Temporal range: calls during at most 1 hour.
C2 - Daily presence: at most 1 day in which C1 is satisfied.
C3 - Weekly presence: at most 1 week, in which C1 and C2 are satisfied.

The result of the top-down approach is shown in Fig. 4. This method is able to capture only a low number of com-

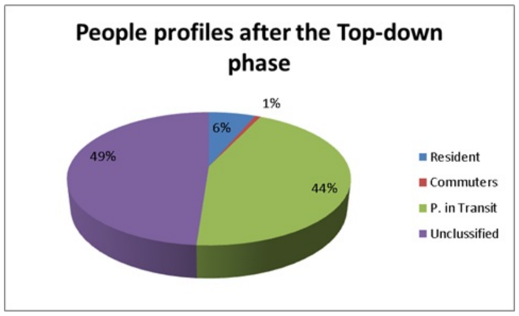


Figure 4: People profiles after the top-down phase.

muters and residents because the temporal constraints are very strict and selective. On the contrary, people in transit are well identified. The high percentage of this kind of users is justified by the presence of an highway and a freeway close to the town.

Thus, starting from the unclassified users, we can apply the SOM method to identify temporal profiles with similar characteristics, i.e. we can group together people who have the same calling patterns. In particular, we are interested in two aspects of the temporal profile: the duration of the stay in the city and the typical temporal location of a user call. To address the first problem, we perfomr a transformation of the temporal profiles by applying a temporal shift. The objective it to align all the user activities at the beginning of the time window. The results provided by SOM is shown in Figure 5. The resulting map shows a set of nodes, where each node contains a set of user profiles. For an immediate readability of the results, each node shows the cardinality of its the population, the circle is proportional to the population and the time chart shows the temporal distribution of the user activities in the specific time interval. In the map of Figure 5*(Left)*, the shifted temporal profiles consist of vectors of 31 entries, one for each day of the time window. Since we are dealing with rotated profiles, the extension of the temporal distribution in each node provides an immediate estimation of the duration of the stay of the corresponding users. From the map it is evident how the temporal profiles are grouped: on the bottom left corner of the figure there are the temporal profiles corresponding to short visits of the city; the upper right side of the figure shows the profiles that span for the whole period and it is possible to identify even nodes that present a clear commuter-like pattern with high frequency during the workdays and a smaller activity during weekends. It is important to point out the presence of three larger nodes corresponding to short visits ranging from one day (node with 5750 profiles) to three days (nodes in the upper left corner). The shifting transformation can be inverted to observe the actual temporal distribution of the activities during the period of study. Figure 5*(Right)* shows, for each node in the *(Left)* map, the corresponding absolute temporal distribution of the activities. The cardinality statistics are left as a reference between the two figures. It can be noted how the short-ranged temporal profiles are uniformly distributed across the whole period. For instance, the larger node containing temporal profiles of a single day presents a quite uniform presence of users across the month considered in the study. On the other hand, as it could be expected, the profiles with a larger extent do not vary too much, since

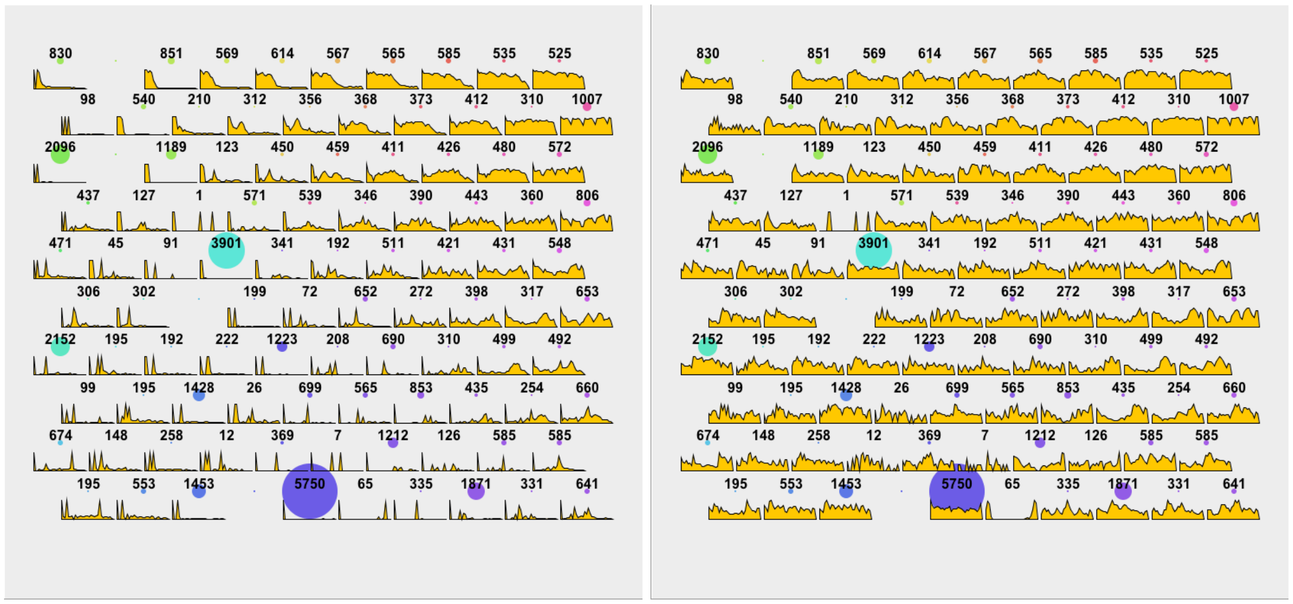


Figure 5: SOM clusters with rotated Temporal Profiles. *(Left)* Each node shows the distribution for shifted temporal profiles. *(Right)* Each node shows the actual temporal distribution for the corresponding set of users.

their width limits the shifting transformation.

To better understand the temporal distribution of user activities in the period, we apply the SOM method to the unshifted temporal profiles. The resulting SOM map is showed in Figure 6. In this map, we can notice how the commuter-

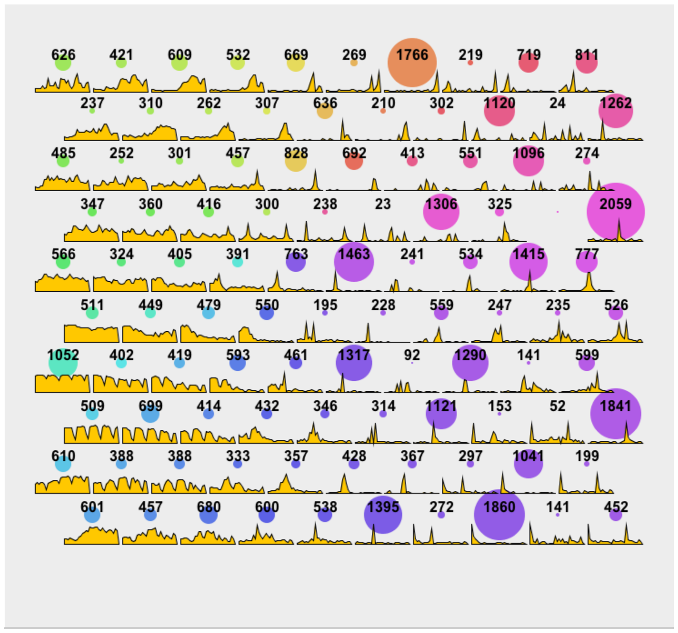


Figure 6: SOM clusters with (un-shifted) Temporal Profiles.

like patterns are even more evident in the bottom left corner. The population corresponding to these nodes is even larger than the nodes present in the shifted version. Actually, these nodes are contributed by users with different habits: the frequent callers have a regular temporal profile that remains unchanged even after the temporal shift; the infrequent callers, on the contrary, do not present such distribution by themselves but their aggregated distribution reconstructs this temporal pattern. The influence of personal calling activity may be crucial in some analysis, since a too specific distribution may be biased by an incomplete vision of the real phenomena, in particular when people do not use the mobile phone during their movements or activities. The case above of commuters distribution is just an example of this kind. This aspect is even more evident when we consider some of the larger node of the map in Figure 6. According to the node with 2059 entries (on the center right side of the figure), a lot of people were present in Pisa for a single day, specifically on January 26. Actually, that day, around 16 in the afternoon, an earthquake happened in northern Italy and it was perceived by the population in Pisa. That event triggered the need for a lot of users to communicate and call their relatives producing the peak we observed in the node. This particular example is emblematic in showing how a peak in the phone traffic not necessarily implies an increase in population density. In this case, the peak is due to infrequent callers that were forced to call by an external event.

6. CONCLUSIONS AND FUTURE WORKS

The wide coverage of GSM networks enables numerous applications for the understanding of people mobility behavior. However, the data anonymization combined with the heterogeneity of people that carry a GSM phone limits the analysis that can be performed due to the lack of a user profile. To face this problem in this paper we propose an approach for inferring user profiles from GSM data. In particular, we concentrate our efforts in identifying mobility profiles based on the presence of user in an area, that is in turn based on the call habits of that user. The methodology aims at identifying four categories of users: residents, commuters, in transit and tourists/visitors. The process in based on two phases: a top-down step where GSM Call Records data are queried, combined with a bottom-up step based on a machine learning algorithm to find homogeneous groups of users. We show - through an experiment run on a real dataset - how the process identifies users profiles. Finding the user profiles may enable wide spectrum of applications from traffic management - for example relating the commuter mobility with the resident one - or tourism - where the touristic flows in a city may be analyzed. The identification of user profiles is based on the analysis of a large dataset of call logs that carry almost no semantic information about the users' metadata. In fact, such data is usually subject to several restrictions (e.g. privacy) that does not allow the diffusion of demographic and personal information of a single individual. In this scenario, we try to compensate the deficiency of semantic information by relying on a unsupervised method to separate relevant groups with similar temporal profiles. Current work include the investigation of the accuracy problem. Indeed, the outcome of the method should be evaluated at least on two levels of accuracy: a quality measure of the resulting segmentation (based for example on classical measures of cluster quality like separateness and cohesion) and a validation assessment of the population of each group with a reference ground truth. The first task is straightforward using classical clustering methods, but does not guarantee the real adherence of the result to the reality. On the contrary, the latter is more challenging since it may assess the results against some form of ground truth and it is particularly useful when the input data do not come with a rich semantic information, like in the present case. We plan to investigate this latter issue from several points of view. For example, a possibility consists in the comparison of the profiles of resident users with the resident population measured by national statistical bureau. However, this ground truth describes only partially the phenomenon since we miss the dynamic aspect that are difficult to measure with precision with classical statistical measures like, for example, the continuously changing ratio of residents with commuters and tourists. Our proposal consists in the comparison of each of these profiles with a series of observations coming from different datasets. In particular, we are currently working on a project with the Municipality of Pisa to collect, integrate and analyze a set of statistical indicators about touristic presence and mobility. In this context, we plan to compare the temporal distribution of the extracted profiles with the data coming from several information sources related to the touristic presence in the city. Examples are the aggregated number of hotels reservations, issued museum tickets, social photo services (i.e. Flickr and Panoramio), microblogging services (i.e Twitter), trajectories of private vehicles equipped with GPS devices, touristic buses presence, etc. The method we propose is not a direct comparison of two variables, i.e. GSM profiles and the estimation of tourists in the city, but we aim at proposing a more complex methodology where different datasets are used to provide a specific vision on the same phenomenon, i.e. the touristic presence in the city, by means of the combination of the different points of view giving a complementary vision of the whole scenario. This is a very ambitious task that involves the management of different data sources. It is worth pointing out that this work would be relevant also in other in different application scenarios where a classical statistical validation is not possible.

7. REFERENCES

[1] R. Ahas, S. Silm, S. Järv, and E. Saluveer. Using mobile positioning data to model locations meaningful to users of mobile phones. *Journal of Urban Technology*, 17(1), 2010.

[2] G. Andrienko, N. Andrienko, P. Bak, S. Bremm, D. Keim, T. von Landesberger, C Poelitz, and T. Schreck. A framework for using self-organising maps to analyse spatio-temporal patterns, exemplified by analysis of mobile phone usage. *Journal of Location Based Services*, 4(3–4), 2010.

[3] F. Calabrese, M. Colonna, P. Lovisolo, D. Parata, and C. Ratti. Real-time urban monitoring using cell phones: A case study in rome. *IEEE Transactions on Intelligent Transportation Systems*, 12:141–151, 2011.

[4] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, C. Renso, S. Rinzivillo, and R. Trasarti. Unveiling the complexity of human mobility by querying and mining massive trajectory data. *VLDB J.*, 20(5):695–719, 2011.

[5] T. Kohonen. *Self-Organizing Maps.* Springer Series in Information Sciences. Springer, 2001.

[6] L. Liao, D. Fox, and H. Kautz. Location-based activity recognition using relational markov networks. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence, IJCAI'05*, 2005.

[7] Nokia. Nokia siemens networks. http://www.slideshare.net/NokiaSiemensNetworks/20-years-of-gsm-past-present-future-8512655.

[8] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. A. Macedo, N. Pelekis, Y. Theodoridis, and Z. Yan. Semantic trajectories modeling and analysis. *Accepted at ACM Computing Surveys*, 2012.

[9] F. C. Pereira, L. Liu, and F. Calabrese. Profiling transport demand for planned special events: Prediction of public home distributions, 2010. Available online at www.scienceDirect.com.

[10] D. Quercia, N. Lathia, F. Calabrese, G. Di Lorenzo, and J. Crowcroft. Recommending social events from mobile phone location data. In *International Conference on Data Mining, ICDM*, pages 971–976, 2010.

[11] C. Ratti, A. Sevtsuk, S. Huang, and R. Pailer. Mobile landscapes: Graz in real time, 2005. MIT Senseable City Lab.

[12] J. Schlaich, T. Otterstätter, and M. Friedrich. Generating trajectories from mobile phone data. In *Proceedings of the 89th Annual Meeting Compendium of Papers, Transportation Research Board of the National Academies*, 2010.

[13] Wikipedia. Call data record. http://en.wikipedia.org/wiki/Call_detail_record.

[14] Wikipedia. Tourism. http://en.wikipedia.org/wiki/Tourism.

[15] Xiangye Xiao, Yu Zheng, Qiong Luo, and Xing Xie. Finding similar users using category-based location history. In *GIS*, pages 442–445, 2010.

Characterizing Large-scale Population's Indoor Spatio-temporal Interactive Behaviors

Y-Q Zhang
Fudan University,
Shanghai, China
10210720048@fudan.edu.cn

Xiang Li[*]
Fudan University,
Shanghai, China
lix@fudan.edu.cn

ABSTRACT

Human activity behaviors in urban areas mostly occur in interior places, such as department stores, office buildings, and museums. Understanding and characterizing human spatio-temporal interactive behaviors in these indoor areas can help us evaluate the efficiency of social contacts, monitor the frequently asymptomatic diseases transmissions, and design better internal structures of buildings. In this paper, we propose a new temporal quantity: *'Participation Activity Potential' (P_{PA})* to feature the critical roles of individuals in the populations instead of their degrees in the corresponding complex networks. Especially for the people with high degrees (hubs in the network), *Participation Activity Potential* which is directly from the statistics of their daily interactions, can easily feature the rank of their degree centrality and achieve as high as 100% accuracy rating without building the corresponding networks by high-complexity algorithms. The effectiveness and efficiency of our new defined quantity is validated in all three empirical data sets collected from a Chinese university campus by the WiFi technology, a small conference and an exhibitions by the RFID technology.

Categories and Subject Descriptors

J.4 [**Social And Behavioral Sciences**]: Sociology; G.3 [**Probability And Statistics**]: Statistical computing; C.2.m [**Computer Systems Organization**]: Computer-communication networks—*Miscellaneous*

General Terms

Measurement, Experiment, Performance

Keywords

Participation activity potential, complex networks, close proximity interactions, degree centrality, human dynamics, temporal networks

[*]Correspondence and requests for materials should be addressed to X L

1. INTRODUCTION

With the rapidly urbanization, the populations are explosive increasing in the urban areas. More and more people's daily work, study and leisure are dependent on the public places of the urban area: subway stations, restaurants, café, museums, department stores, office buildings and etc. In these interior places, human daily interactions, e.g., face-to-face talks, hand touches, are named as human close proximity interactions (CPIs)[28], which impact our understanding of the respiratory disease transmissions, the efficiency of social contacts and the convenience of internal structures.

Since 1930's, social scientists have collected and studied human interactions and relationship data. The traditional technologies of data collections are inviting volunteers to complete questionnaires or hiring an observer to record data, and because of its time-consuming nature, the collected data is constrained to a small number of people. Nowadays, with advanced development in digital technologies, GPS, WiFi, RFID and mobile phones are widely used as sensors to locate people and build a huge volumes of spatio-temporal data, in the form of traveling trajectories[2, 15, 10, 24, 30, 17] and interactive traces[20, 4, 13, 14, 26].

Barabási, et al.[10, 24, 20] apply mobile phones to approximate human traces to mine the pattern of human mobility. GPS [2, 15, 30, 17] as a typical located technology, is widely used in tracing passengers' traveling trajectories. Barrat and their colleagues use the RFID technology to trace human CPIs in distinct rendezvouses [4, 13, 14, 26] and provide a 'reinforcement dynamics' to model human face-to-face interactions[25, 31]. By using wireless sensors, Salathé, et al. [23] record an American high school students' daily interactions and evaluate their respiratory diseases infectious risks.

With the concept of 'intelligent city' becoming reality, WiFi hotspots are distributed in every corner of the urban areas. The popularity of wireless devices, such as smartphones, laptop computers and tablets, further spurs the WiFi technology (or known as 802.11b) to assist in tracing human mobilities and interactions. Therefore, we use the WiFi technology to trace a Chinese university students' interactions over 3 months. These digital 'sensors' automatically record the human CPIs without additional observers or interviewers and avoid build-in errors caused by human memory[6, 7].

The data of human CPIs were traditionally very easy to map into a static network(contact network, or named aggregated contact network with individuals as nodes and interactions between them as links) to study the embedded social phenomena. In the past more than 10 years, researchers have witnessed the power of complex network theory to influence the spreading processes which help understand the biological and computer virus transmission. For instances, the small-world properties[29] and scale invariant power-law features of degree consequence distributions[1] can decrease the threshold of spreading processes[21, 22]. Moreover, in the scale-free physical and/or social networks, several 'super-connectors' (hubs in the networks) play the role of self-sustained sources to spread the infection to the rest of the system[3], which leads that there are many methods[5, 16] to find and vaccinate these 'super-connectors' in order to control the whole disease transmission.

However, with the development of complex network theory, the static networks as the body to study the spreading processes have faced with many new challenges[27, 18, 12, 8]. Empirically, human close proximity interactions are not stationary. A 'super-connector' can not simultaneously contact such huge number of neighbors. Very recently, the temporal network has become to attract researchers' attention [11]. In this study, we use three empirical data sets, two from the RFID technology (SocioPatterns) and one from the WiFi technology to uncover the distinct features between the temporal networks and static networks. Especially, we define a new temporal quantity, *'Participation Activity Potential'* to measure the critical roles of individuals in the corresponding networks, by which, 'super-connectors' in the temporal and static network can both be characterized effectively and efficiently.

The rest of the paper is organized as follows. In Section 2 we firstly introduce a series of transformed temporal networks and the corresponding aggregated static networks from three empirical data sets, and moreover, we define some preliminary concepts and notations used in this paper. In Section 3 we analyze the difference between the transmission graphs and the contact networks to investigate the typical features of 'super-connectors'. In Section 4 we define a new temporal quantity to characterize the rank of degree centrality of individuals in two benchmark networks, and conclude the whole paper in Section 5 with future steps of work.

2. FROM HUMAN CLOSE PROXIMITY INTERACTIONS TO NETWORKS

We firstly transform human close proximity interactions (CPIs) to the corresponding networks as shown in Figure 1, a schematic representation of human CPIs of an exampled case with the detailed descriptions see Example 1.

Example 1. There are three individuals A, B, C which have the corresponding CPIs AB, AC, BC (Figure 1(A)). Generally, these CPIs are the origin collected data, which records when and how long a pair of individuals have an interaction. Traditionally, the data is directly transformed to a contact network which is illustrated in Figure 1(D), where individuals are considered as nodes and individuals' interactions as links. The algorithms to build a contact network are classic and we show it in Appendix (*Algorithm 1*). In Figure 1(A), it is possible to find there exists an interval, $[t_2, t_3]$, three individuals contact with each other building a clique. However, the clique is not very clearly observable in the data of human CPIs directly. Therefore, in Figure 1(B), we define *'event interaction' (EI)* to focus on the clique. Each EI represents the interactions among all individuals in a given interval. The CPIs can be directly transformed to EIs (see

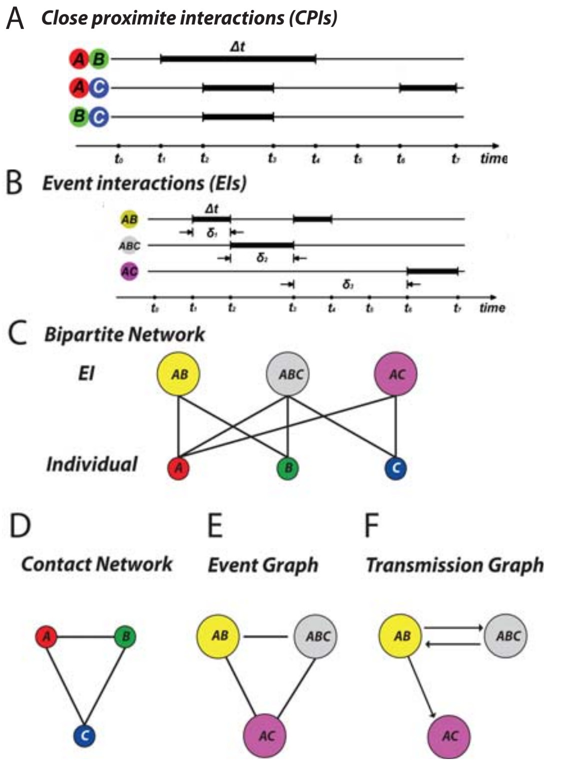


Figure 1: Schematic Representation of Human close proximity interactions (CPIs) and the corresponding (static/temporal) transformed networks.

Appendix, *Algorithm 2*). Therefore, it is easy to build a bipartite network (Figure 1(C)) exhibiting the affiliating relations between individuals and EIs. In a bipartite network, the links between individuals are based on the same EI two individuals affiliate with. As shown in Figure 1(E), an event graph is defined with the EIs: the links between the EIs are based on the same individuals two EIs both contain. Both contact networks and event graphs are static networks without temporal information when the link connects and how long the link persists to. In order to fuse temporal information, we further build a transmission graph based on the EIs. The links between the EIs consider not only their active interval, but also the order sequence of distinct EIs which is caused by the mobility of individuals contained in. The algorithm to build a transmission graph refers to *Algorithm 3* in Appendix.

To characterize the detailed human interactive behaviors, we define the following definitions.

Definition 1. The quantities of individuals:

1. *Self-activity duration*: If an individual j is active during the interval $[t^{begin}, t^{end}]$, the self-activity duration of individual j is $\Delta t^{SA}(j) = t^{end} - t^{begin}$.[1]

[1]As shown in Figure 1(A), during the interval of $[t_1, t_4]$, A has two records of CPIs with the overlapped duration, and $A's$ self-activity duration is the maximal duration of the CPIs.

2. *Self-activity frequency*: Given an observational interval T (without specially statement, T is the maximum observation interval), an individual j is active n times. Denote the self-activity frequency of individual j is $n^{SA}(j) = n$.[2] Therefore, the total self-activity duration of individual j is defined as

$$\Delta t^{SA}_{sum}(j) = \sum_{m=1}^{n^{SA}(j)} \Delta t^{SA}_{m}(j)$$

3. *Self-activity potential*: Given an observational interval T, the self-activity potential of individual j is defined as

$$P_{SA}(j) = [\frac{\Delta t^{SA}_{sum}(j)}{n^{SA}(j)}]^{p} n^{SA}(j) \qquad (1)$$

where p is an independent variable belongs in $[0, 1]$.

Definition 2. The quantities of event interactions(EIs):
Event interaction duration: If an EI e_i is active during the interval $[t^{begin}, t^{end}]$, the event interaction duration of e_i is defined as $\Delta t^{EI}(e_i) = t^{end} - t^{begin}$

Definition 3. The quantities of a bipartite network:

1. *The rate of participation*: If an individual j participates in r event interactions, the rate of participation of individual j is defined as $r(j) = r$.

2. *Event interaction size*: If an event interaction e_i contains s individuals, the size of event interaction e_i is $s(e_i) = s$.

Definition 4. The quantities of a contact network:
Degree of individuals in the contact network: If an individual j has k distinct neighbors in the contact network, the degree of individual j is defined as $k^{CN}(j) = k$.

Definition 5. The quantities of a transmission graph and the corresponding aggregated transmission graph:

1. *Event interaction frequency*: Given an observation interval T, an EI e_i is active n times in the transmission graph. Denote the frequency of EI e_i $n^{TG}(e_i) = n$. Therefore, the total self-activity duration of EI e_i is defined as

$$\Delta t^{EI}_{sum}(e_i) = \sum_{m=1}^{n^{TG}(j)} \Delta t^{EI}_{m}(j)$$

.

2. *Self-activity potential:*Given an observational interval T. Define the self-activity potential of EI e_i as

$$P^{EI}_{SA}(e_i) = [\frac{\Delta t^{EI}_{sum}(e_i)}{n^{TG}(e_i)}]^{a} n^{TG}(e_i) \qquad (2)$$

where a is an independent variable belongs in $[0, 1]$.

3. *Degree of EIs in the aggregated transmission graph*: If an event interaction (EI) e_i has been directed by k neighbors in the aggregated transmission graph, the in-degree of EI e_i is $k^{ATG}_{in}(e_i) = k$. Similarly, if the EI e_i directs to k neighbors, the out-degree of EI e_i is $k^{ATG}_{out}(e_i) = k$. Without considering the direction in the transmission path (or known as 'weakly connected'[19]), the degree of EI e_i is $k^{ATG}(e_i)$, which is equal to the number of all neighbors.

In this paper, there are three empirical data sets of human CPIs investigated. By the WiFi technology, we have collected the data of human CPIs in a Chinese campus during the 2009-2010 fall semester (3 complete months). Each student, teacher, and visiting scholar has a unique account to access the Campus WiFi system, which will automatically record their devices' MAC addresses to build a log with the MAC address of the WiFi access points(APs) they access to, and the connecting/disconnecting time as well. The logs indicate that when and where the person use the WiFi network. In the public classrooms, the spatial distance between any two individuals accessing to the same AP is as close as less than 8 meters[3]. From the logs of each individual, we can build human CPIs which is named as 'FudanWiFi09'.

The other two empirical data sets are both collected by the RFID technology freely achieved in the website of 'SocioPatterns' (http://www.sociopatterns.org), and a related research report refers to [14]. One of the data set was collected during the ACM Hypertext 2009 conference, where the 'SocioPatterns' project deployed the Live Social Semantics application. The conference attendees volunteered to wear radio badges which monitored their face-to-face interactions named as 'HT09'. The other data set contains the daily dynamic contact networks collected during the artscience exhibition 'INFECTIOUS: STAY AWAY' which took place at the Science Gallery in Dublin, Ireland, and we name it as 'SGInfectious'. In Table 1 we summarize the basic properties of these three empirical data sets.

Notice that these three data sets represent three different types of human proximity close interactions. The individuals in 'FudanWiFi09' have stable social ties: they can be classmates, teacher-student, lovers and so on. During 3 months, the interactions between the individuals may repeat in an acquaintance community. Individuals in 'HT09' are the attendees of the conference. Most of the people did not know each other. During the conference period of 3 days, they began to know each other, and the data represents the repeatable interactions in a stranger community. The individuals in 'SGInfectious' also have stable social ties, they know each other and then go with each other to visit the exhibition 'Infectious: stay away'. However, most of the involved visitors will not visit the exhibition again, which indicates that the data exhibits the unrepeatable interactions in an acquaintance community.

3. CONTACT NETWORKS AND TRANSMISSION GRAPHS

3.1 The differences between contact networks and transmission graphs

[2]The corresponding self-activity frequency is not the number of the given individual's CPIs, but the number of all unoverlapped self-activity duration.

[3]In the paper 'Two Categories of Interaction Dynamics of a Large-scale Human Population in a WiFi covered university campus' (unpublished), we give the details about the spatial distance between any two individuals accessing to the same AP.

Table 1: Three empirical data sets of human close proximity interactions(CPIs)

	FudanWiFi09	HT09	SGInfectious
Area	Campus	Conference	Mesume
Technology	WiFi	RFID	RFID
Collection duration	84 days	3 days	62 days
Number of individuals	17897	113	10970
Number of CPIs	884800	9865	198198
Spatial resolution(meters)	< 8	< 2	< 2
Types of CPIs	Acquaintances with repeat	Strangers with repeat	Acquaintances without repeat

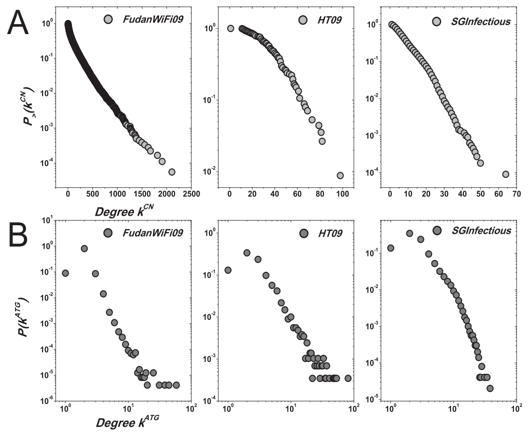


Figure 2: The degree distributions of the contact networks and the corresponding aggregated transmission graphs.

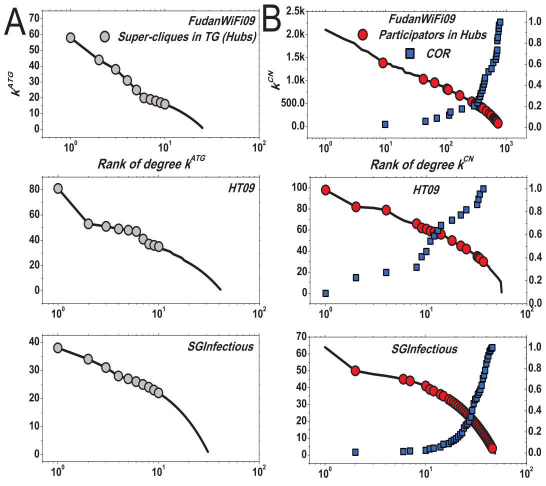


Figure 3: The analyses of hubs in the aggregated transmission graphs and contact networks.

We visualize the transformed networks from the three empirical data sets in Figure 2, illustrating the difference among the contact networks (static networks) and transmission graphs (temporal networks). The cumulative probability distributions of k^{CN} in Figure 2(A) from three empirical data sets are all exponential distributions although the corresponding number of sampled individuals are different (Table 1), i.e., the contact networks are homogeneous, implying that the static patterns of human CPIs are random and well-mixed. While in the transmission graph, the probability distributions of k^{ATG} exhibit the form of power-law (see Figure 2(B)), indicating that the aggregated transmission graphs are heterogeneous, and the temporal patterns of human CPIs contain many preference linking[1].

In the study of spreading processes, scale-free networks with the form of power-law degree probability distributions have been the focus for years, where the thresholds of spreading processes are approximate to zero[21, 22]. While in a homogeneous network such as an ER random graph, there exists a fixed threshold. Recent study further states that the hubs of such networks play a self-survived role in the dynamical spreading processes[3]. Therefore, we further compare the hubs in contact networks and transmission graphs which are named as 'super-connectors' and 'super-connecting groups', respectively. Figure 3(A) illustrates the relations between the degrees of the aggregated transmission graph(ATG) and the corresponding ranks, where the top 10 nodes ranked in the ATG are the 'super-connecting groups'. Each super-connecting group consists of at least two individuals. Naturally, we may regard these involved individuals as those 'super-connectors' in the contact networks. However, Figure 3(B) shows the opposite situation. The black line represents the relation between the degrees of individuals and the corresponding rank in the contact network, while the red circles exhibit the individuals of super-connecting groups in Figure 3(A). Strikingly, most of the members in the super-connecting groups have low degrees in the contact network. The blue rectangles represent the cumulative occurrence rates (COR) of each member, and around 90% of the individuals in the super-connecting groups are 'leaf' nodes in the contact network. Therefore, the degree distribution and the rank-degree relations of hubs present the obvious differences between static networks and temporal networks.

3.2 The analyses of contact network

In Definition 3, we define the participation rate of a given individual (r). Moreover in Definition 4 we define the degree of individuals in a contact network (k^{CN}). In Figure 4(A) we observe that k^{CN} increases with the growth of r. In other words, the potential of an individual becoming a 'super-connector' is due to his (her) participation capability.

With the defined self-activity potential P_{SA} of an individual, Figure 4(B) shows the individuals' degrees in the contact network is positive correlated with their self-activity

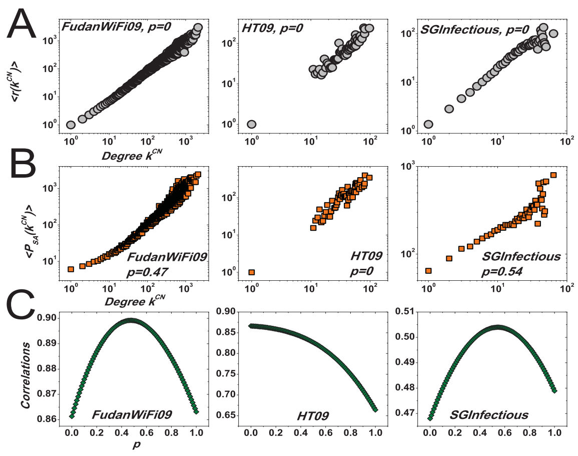


Figure 4: The individuals' self-activity potentials with their degree centrality in contact networks.

potentials, where a higher self-activity potential tells that the individual is more active. Therefore, the possibility of an individual becoming a 'super-connector' is also dependent on his (her) self-activeness.

Moreover, when $p = 0$, the self-activity potential degenerates to $P_{SA}(j) = n^{SA}(j)$, which is equal to the self-activity frequency. When $p = 1$, the self-activity potential degenerates to the total self-activity duration $\Delta t^{SA}_{sum}(j)$. Figure 4(C) exhibits the Pearson's Coefficient (PCC)[9] between the individuals' self-activity potentials $P_{SA}(j)$ and degrees k^{CN}, where p belongs in $[0, 1]$. The data of 'FudanWiFi09' and 'HT09' show their PCC can achieve as high as 0.9 and 0.85, respectively, indicating that the self-activity potentials of individuals have a very strong positive correlation with their degrees in the contact network. However, the PCC of the 'SGInfectious' data is less than 0.5, whose correlation is weaker. As summarized in Table 1, the types of CPIs of three empirical data sets are different: 'FudanWiFi09' and 'HT09' are the repeatable interactions while 'SGInfectious' represents unrepeatable interactions. In other words, the individuals in 'FudanWiFi09' and 'HT09' both build a time-invariant community, which does not hold in the case of 'SGInfectious'. In a time-invariant community, a larger self-activity potential may help the individual increase the possibility of contacting with other individuals. While in a time-variant community, new individuals join the community along with the old individuals leave, and every individual has an active lifetime, out of which the individuals will lose the capability to contact with other individuals in the community. Therefore, we conclude that in such (static) contact networks, super-connectors own two essential characteristics: high self-activity and sociability (participation capability). The 'leaf' individuals in the contact network are those with low self-activity and sociability. However, why can these 'leaf' individuals gather into the super-connecting groups in the (temporal) transmission graph? we need further explore the temporal transmission graphs in more detail.

4. PARTICIPATION ACTIVITY POTENTIAL

In a transmission graph, an EI has a self-activity potential P^{EI}_{SA} and a degree k^{ATG}. The crucial role of an individual in the transmission graph is featured by the degree of the corresponding EI. That is to say, when an individual participates in an EI having a higher degree, the individual is more important in the temporal network. However, since most of the individuals' participation rates are larger than one, they may participate in r EIs, whose degrees are r at most. Since the role of individuals in a temporal network is featured by the maximal degree, we define a new definition of the individuals' maximal participation degree as follows:

Definition 6. Maximal participation degree κ_{max}: If an individual has the participation rate r, he (she) participates in r EIs: $e_1, e_2, ..., e_r$, with the corresponding degrees in the ATG as $k^{ATG}_1, k^{ATG}_2, ..., k^{ATG}_m$ $(m \leq r)$, respectively, the maximal participation degree $\kappa_{max} = max(k^{ATG}_1, k^{ATG}_2, ..., k^{ATG}_m)$

4.1 Maximal Participation Activity Potential

Given an individual j participates in an EI e_i $(j \in e_i)$. Define the participation activity potential of individual j with EI e_i as follows:

$$P_{PA}(j, e_i) = P^{EI}_{SA}(e_i) = [\frac{\Delta T^{EI}_{sum}(j, e_i)}{n^{TG}(j, e_i)}]^a n^{TG}(j, e_i) \qquad (3)$$

The maximal participation activity potential of individual j comes:

Definition 7. Maximal participation activity potential: An individual j participates in a set of EIs

$$\Gamma(j) = [e_1, e_2, ..., e_i, ...e_{r(j)}]$$

The corresponding participation activity potential is:

$$\zeta^{P_{PA}}(j) = [P_{PA}(j, e_1), ..., P_{PA}(j, e_{r(j)})]$$

and, the maximal participation activity potential of individual j is:

$$P^{max}_{PA}(j) = max(\zeta^{PA}(j))$$

From Figure 5(A), we observe that the maximal participation activity potential P^{max}_{PA} increases with the growth of the maximal participation degree κ_{max}, indicating that the maximal participation activity potential features the maximal degree of the individuals in the ATG. In Appendix (*Algorithm 3*), we provide the algorithms of building transmission graph, and the computational complexity of the algorithms is $O(M^2)$ (M is the size of *EITimeTable*). However, the participation activity potential of individuals is identical with the self-activity potential of EIs, while the computational complexity of algorithms to calculate the self-activity potential is $O(N)$ (N is the size of EIs and $N \ll M$). If we can use the maximal self-activity potential to feature the critical role of individual replacing their degree centrality, we can dramatically reduce the computational complexity.

We further illustrate in Figure 5(B) the accuracy rating to feature the rank of κ_{max} by P^{max}_{PA}. All three empirical data sets show that the accuracy rating increases with the growth of κ_{max}, indicating that the maximal participation activity potential can feature the members of super-connecting cliques with the achieved accuracy rate as high as 100%.

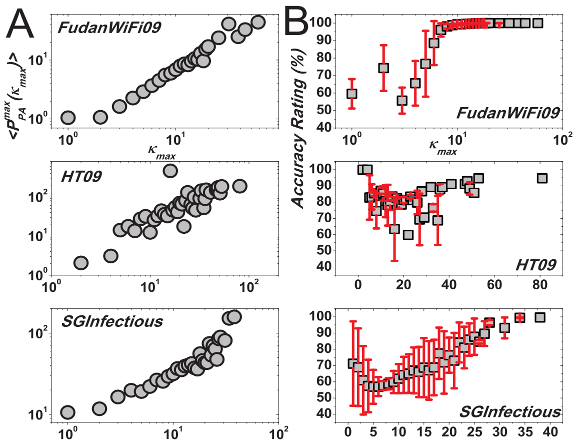


Figure 5: The maximal participation activity potential of three collected data sets.

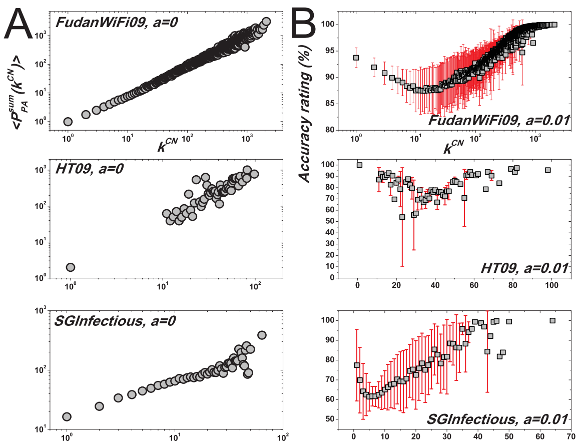


Figure 6: The sum of participation activity potential of three collected data sets.

4.2 Sum of Participation Activity Potential

We further define the sum of participation activity potential of a given individual j as follows:

Definition 8. Sum of participation activity potential P_{PA}^{sum}: Given individual j participates in a set of EIs:

$$\Gamma(j) = [e_1, e_2, ..., e_i, ...e_{r(j)}]$$

The corresponding participation activity potential is

$$P_{PA}(j, e_1), ..., P_{PA}(j, e_{r(j)})$$

Therefore the total participation activity potential of individual j is defined as:

$$P_{PA}^{sum}(j) = \sum_{m=1}^{r(j)} P_{PA}(j, e_m) \qquad (4)$$

Equation 4 shows that when $a = 1$, the sum of participation activity potential degenerates to the total participation duration

$$P_{PA}^{sum}(j) = \sum_{m=1}^{r(j)} \Delta T_{sum}^{EI}(j, e_m)$$

From Figure 1, we find that the total self-activity duration is identical with the sum of participation duration,

$$P_{SA}(j, p = 1) = P_{PA}^{sum}(j, a = 1)$$

While the definition of P_{SA}^{sum} also includes the participation rate and the activity of participation. Therefore, the sum of participation activity potential can replace the self-activity potential to feature the rank of degrees of individuals in the contact network.

As shown in Figure 6(A), in all three data sets, the sum of participation activity potential increases with the growth of corresponding degrees in contact networks, which indicates that super-connectors in the contact networks have higher sum of participation activity potentials. Moreover, we examine the accuracy rate of measuring rank of degrees in the contact networks by the sum of participation activity potential. In Figure 6(B) all of three empirical data sets support the findings that the sum of participation activity potential can reach the accurate rate as high as 100% of measure the super-connectors. Therefore, the critical role of individuals in the contact networks can be characterized by the sum of participation activity potential.

As discussed above, the maximal participation activity potential of a given individual features the corresponding rank of degree centrality in the transmission graph, and the sum of participation activity potential features the corresponding rank of degree centrality in the contact network. Sometimes, the individual with a high maximal participation activity potential also has the high sum of participation activity potential, e.g., some super-connectors in 'HT09' and 'SGInfectious'. However, most of the individuals with high maximal participation activity potentials do not have the corresponding high sum of participation activity potentials, leading that most of the members of 'super-connecting groups' are not the 'super-connectors', as shown in Figure 3(B).

5. CONCLUSIONS

In this paper, we have defined a new 'actor-related' quantity: *participation activity potential* which can characterize their structural relations. The maximal participation activity potential of the individuals feature their rank of degree centrality in the corresponding temporal network, achieving the accurate rate as high as 100% to measure the members of super-connecting groups. Besides, the sum of participation activity potential is effective to characterize the super-connectors in the contact network. In the view of more detailed spreading processes such as infectious disease prevalence and computer virus propagation, the members of super-connecting groups play a critical role to the spread over the whole networking system, therefore, this new proposed quantity deserves further extensive efforts to understand its significance in dynamical spreading processes of urban networking systems in near future.

Acknowledgments

We were grateful to the assistance from Bochun Wu, and the Informatization Office of Fudan University for the WiFi Data collection. We acknowledge the SocioPatterns to provide the data of human close proximity interactions collected by the RFID technologies. This work was partly supported by the National Key Basic Research and Development Program (No.2010CB731403) and the NCET program (No.NCET-09-0317) of China.

6. REFERENCES

[1] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[2] Y. Bu, L. Chen, A.-C. Fu, and D. Liu. Efficient anomaly monitoring over moving object trajectory streams. *In Proceedings of the 15th SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'09)*, pages 159–168, 2009.

[3] C. Castellano and R. Pastor-Satorras. Thresholds for epidemic spreading in networks. *Phys Rev Lett*, 105:218701, 2010.

[4] C. Cattuto, W. Van den Broeck, A. Barrat, V. Colizza, J.-F. Pinton, and A. Vespignani. Dynamics of person-to-person interactions from distributed rfid sensor networks. *PLoS ONE*, 5(7):e11596, 2010.

[5] R. Cohen, S. Havlin, and D. Ben-Avraham. Efficient immunization strategies for computer networks and populations. *Phys Rev Lett*, 91:247901, 2003.

[6] N. Eagle. *Machine perception and learning of complex social systems*. PhD thesis, Massachusetts Institute of Technology, 2005.

[7] N. Eagle, A. Pentland, and D. Lazer. Inferring friendship network structure by using mobile phone data. *P Natl Acad Sci U S A*, 106(36).

[8] J. Fernández-Gracia, V. M. Eguíluz, and M. S. Miguel. Update rules and interevent time distributions: Slow ordering versus no ordering in the voter model. *Phys Rev E*, 84:015103, 2011.

[9] K.-I. Goh, M. E. Cusick, D. Valle, B. Childs, M. Vidal, and A.-L. Barabási. The human disease network. *P Natl Acad Sci U S A*, 104(21).

[10] M. C. González, C. A. Hidalgo, and A.-L. Barabási. Understanding individual human mobility patterns. *Nature*, 453:779–782, 2008.

[11] P. Holme and J. Saramäki. Temporal networks. Phys Rep in press, 2012.

[12] J. L. Iribarren and E. Moro. Impact of human activity patterns on the dynamics of information diffusion. *Phys Rev Lett*, 103:038702, 2009.

[13] L. Isella, M. Romano, A. Barrat, C. Cattuto, V. Colizza, W. Van den Broeck, F. Gesualdo, E. Pandolfi, L. Ravà, C. Rizzo, and A. E. Tozzi. Close encounters in a pediatric ward: Measuring face-to-face proximity and mixing patterns with wearable sensors. *PLoS ONE*, 6(2):e17144, 2011.

[14] L. Isella, J. Stehlé, A. Barrat, C. Cattuto, J.-F. Pinton, and W. Van den Broeck. What's in a crowd? analysis of face-to-face behavioral networks. *J Theor Biol*, 271:166–180, 2010.

[15] J.-G. Lee, J. Han, and X. Li. Trajectory outlier detection: A partition-and-detect framework. *In Proceedings of the 24th International Conference on Data Engineering (ICDE'08)*, pages 140–149, 2008.

[16] S. Lee, L. E. C. Rocha, F. Liljeros, and P. Holme. Exploiting temporal network structures of human interaction to effectively immunize populations. *PLoS ONE*, 7(5):e36439, 2012.

[17] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma. Mining user similarity based on location history. *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, page 34, 2008.

[18] B. Min, K.-I. Goh, and A. Vazquez. Spreading dynamics following bursty human activity patterns. *Phys Rev E*, 83:036102, 2011.

[19] V. Nicosia, J. Tang, M. Musolesi, G. Russo, C. Mascolo, and V. Latora. Components in time-varying graphs. http://arxiv.org/abs/1106.2134, 2012.

[20] J.-P. Onnela, J. Saramäki, J. Hyvönen, G. Szabó, D. Lazer, K. Kaski, J. Kertész, and A.-L. Barabási. Structure and tie strengths in mobile communication networks. *P Natl Acad Sci U S A*, 104:18, 2007.

[21] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Phys Rev Lett*, 86(14).

[22] R. Pastor-Satorras and A. Vespignani. Epidemic dynamics and endemic states in complex networks. *Phys Rev E*, 63:066117, 2001.

[23] M. Salathé, M. Kazandjieva, J. W. Lee, P. Levis, M. W. Feldman, and J. H. Jones. A high-resolution human contact network for infectious disease transmission. *P Natl Acad Sci U S A*, 107(51).

[24] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási. Limits of predictability in human mobility. *Science*, 327:1018–1021, 2010.

[25] J. Stehlé, A. Barrat, and G. Bianconi. Dynamical and bursty interactions in social networks. *Phys Rev E*, 81:035101(R), 2010.

[26] J. Stehlé, N. Voirin, A. Barrat, C. Cattuto, L. Isella, J.-F. Pinton, M. Quaggiotto, W. Van den Broeck, C. Régis, B. Lina, and P. Vanhems. High-resolution measurements of face-to-face contact patterns in a primary school. *PLoS ONE*, 6(8):e23176, 2011.

[27] A. Vazquez, B. Rácz, A. Lukács, and A.-L. Barabási. Impact of non-poissonian activity patterns on spreading processes. *Phys Rev Lett*, 98:158702, 2007.

[28] S. Wasserman and K. Faust. *Social Network Anlysis: Methods and Applications*. Cambridge University Press, Cambridge, 1994.

[29] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[30] X. Xiao, Y. Zheng, Q. Luo, and X. Xi. Finding similar users using category-based location history. *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 442–445, 2010.

[31] K. Zhao, J. Stehlé, G. Bianconi, and A. Barrat. Social

network dynamics of face-to-face interactions. *Phys Rev E*, 83:056109, 2011.

APPENDIX

A. ALGORITHMS

A.1 Algorithms to build a contact network

Algorithms 1. CNBuilder: building a contact network from the records of human close proximity interactions.

Input: *CPIs:* a set of records about human close proximity interactions (Size: S) with the formalization as $(N_1, N_2, t_{start}, t_{end})$.

Output: *CN:* a list of pairwise nodes.

```
1: $CN \leftarrow$ an empty set {};
2: for Each record of CPIs i ($i \in (1, ..., S)$) do
3: If the pairwise nodes in record $i$ are not included in $CN$
4: $CN \leftarrow CN \bigcup (N_1(j), N_2(j))$
5: end
6: end for
7: return $CN$;
```

We assume the size of CN as n, then the computational complexity of the algorithm in the worst case is $O(\frac{(0+(n-1))n}{2} + (S-n)n) \approx O(Sn)$

A.2 Algorithms to build event interactions and a transmission graph

Algorithms 2. EIBuilder: building event interactions from human close proximity interactions.

Input: *CPIs:* a set of records about human close proximity interactions (Size: S) with the formalization as $(N_1, N_2, t_{start}, t_{end})$.

Output: *EITimeTable:* a list of starting and ending time of EIs. *EIs:* a list of event interactions.

```
1: $timeList \leftarrow$ an empty set {};
2: **for** Each record of CPIs i ($i \in (1, ..., S)$) **do**
3: **if** the starting and ending time of CPI $i$ are not including in $timeList$ **then**
4: $timeList \leftarrow timeList \bigcup (t_{start}(j), t_{end}(j))$
5: **end if**
6: **end for**
7: Return $timeList$;
8: Order $timeList$ by time ascend.
9: $EITimeTable \leftarrow$ a empty set {};
10: **for** Each record of $timeList$ j ($j \in (1, ..., M)$)[4] **do**
11: $nodeset \leftarrow$ an empty set {};
12: **for** Each record of $CPIs$ i ($i \in (1, ...S)$) **do**
13: **if** the record $i$ of CPIs with $t_{start} \leq t_j$ and $t_{end} \gg t_{j+1}$ and nodes of $i$ are not included in $nodeset$ **then**
14: $nodeset \leftarrow nodeset \bigcup N_1(i) \bigcup N_2(i)$;
15: **end if**
16: **end for**
17: $EITimeTable \leftarrow EITimeTable \bigcup nodeset \bigcup (t_j, t_{j+1})$;
18: **end for**
19: Return $EITimeTable$;
20: $EIs \leftarrow$ an empty set {};
21: **for** Each record of $EITimeTable$ k ($k \in (1, 2, ..., M)$) **do**
22: **if** the nodeset of record $k$ is not included in $EIs$ **then**
23: $EIs \leftarrow EIs \bigcup$ nodeset of $k$;
24: **end if**
25: **end for**
26: Return $EIs$;[5]
```

[4]M+1 represents the size of timeList.

The computational complexity of the algorithm to build *timeList* in the worst case is $O(MS)$. The computational complexity of the algorithm to sort *timeList* in the worst case is $O((M+1)^2)$. The computational complexity of the algorithm to build *EITimeTable* in the worst case is $O(SM)$. The computational complexity of the algorithm to build *EIs* in the worst case is $O(MN)$. Therefore, the computational complexity of the algorithm to build *EIs* from *CPIs* in the worst case is $O(MS)$

Algorithms 3. TGBuilder: building a transmission graph and the corresponding aggregated transmission graph from the *EITimeTable*

Input: *ETTimeTable*.

Output: *TG*: the transmission path from one EI to another. *ATG*: the aggregated transmission graph.

```
1: $TG \leftarrow$ an empty set {};
2: sort $EITimeTable$ by the ascending order of the staring time.
3: **for** Each record of $EITimeTable$ e ($e \in (2, ..., M)$) **do**
4: $bridgenodes \leftarrow$ an empty set {};
5: **for** Each record of $EITimeTable$ ei ($ei \in (e, ..., 1)$) **do**
6: **if** there are common nodes both in record e and record ei, while the common nodes are not included in $bridegenodes$ **then**
7: $TG \leftarrow TG \bigcup$ [nodeset of ei, nodeset of e, $t_{start}(ei)$, $t_{end}(ei)$, $t_{start}(e)$, $t_{end}(e)$];
8: $bridgenodes \leftarrow bridgenodes \bigcup$ nodeset of ei;
9: **if** the size of $bridgenodes$ is equal to the size of nodeset of ei **then**
10: **break;**
11: **end if**
12: **end if**
13: **end for**
14: **end for**
15: Return $TG$;
16: $ATG \leftarrow$ an empty set {};
17: **for** Each record of $TG$ t $t \in (1, 2, ...P)$ **do**
18: **if** the pairwise nodesets of record t are not included in $ATG$ **then**
19: $ATG \leftarrow ATG \bigcup$ the pairwise nodesets of record t;
20: **end if**
21: **end for**
22: Return $ATG$[6];
```

The computational complexity of the algorithm to build a TG in the worst case is $O(M^2)$ and the computational complexity of the algorithm to build an ATG in the worst case is $O(PH)$.

Regarding event interactions as vertices, we further introduce the following rules to link the successive vertices as the edges of the transmission graph: a) in the time series, a source EI is the closest EI prior to the sink EI; b) at least one user coexists in the source and sink EI; c) when there are several sources before one sink, any set of the shared users between the given source and sink EIs never intersect with each other(set).

[5]N represents the size of EIs.

[6]H is the size of ATG

Mining Traffic Incidents to Forecast Impact

Mahalia Miller
Dept. of Civil Engineering, Stanford University
& Hewlett Packard Labs
mahalia@stanford.edu

Chetan Gupta
Hewlett Packard Labs
chetan.gupta@hp.com

ABSTRACT

Using sensor data from fixed highway traffic detectors, as well as data from highway patrol logs and local weather stations, we aim to answer the domain problem: "A traffic incident just occurred. How severe will its impact be?" In this paper we show a practical system for predicting the cost and impact of highway incidents using classification models trained on sensor data and police reports. Our models are built on an understanding of the spatial and temporal patterns of the *expected* state of traffic at different times of day and locations and past incidents. With high accuracy, our model can predict false reports of incidents that are made to the highway patrol and classify the duration of the incident-induced delays and the magnitude of the incident impact, measured as a function of vehicles delayed, the spatial and temporal extent of the incident. Equipped with our predictions of traffic incident costs and relative impacts, highway operators and first responders will be able to more effectively respond to reports of highway incidents, ultimately improving drivers' welfare and reducing urban congestion.

Keywords

Geomining, event analysis, traffic prediction, cyber-physical

1. INTRODUCTION

Traffic congestion causes business losses due to increased travel time, requires increases in public infrastructure investment, and due to extra emissions, threatens urban air quality. While work zones, weather, fluctuations in normal traffic, special events, traffic control, and physical bottlenecks can all lead to congestion, a key contributor to congestion is traffic incidents–"events that disrupt the normal flow of traffic, usually by physical impedance in the travel lanes" [2].

The wealth of data from transportation sensor networks offers the opportunity to understand traffic incidents in order to make urban travel more efficient. Incident detection was among the first incident managment problems to be addressed. Researchers have proposed machine learning and rule-based approaches for detecting incidents or outliers at individual locations, e.g. [8, 11]. A recent extension is to detect and identify causal relations between anomolous events in traffic data streams from GPS traces [12]. This work offers insights into understanding the spatio-temporal impact of traffic events. Another approach uses data from fixed sensors to build a model of recurrent traffic congestion and retrospectively identify the spatio-temporal impact of an incident [10]. This work leaves open some specific problems: (1) the definition of the impact region is based on a fixed threshold instead of values tuned to expected conditions at different locations and times of day and (2) the definition of the impact region is limited to only the highway on which the incident originated.

A second area in incident management is forecasting the severity of an incident and its impact. Analytical formulae based on queuing or shockwave models have been used, often demonstrated on simulator data without validation from sensor recordings, e.g. [3]. Khattak *et al.* [9] use sensor data to build regression models for incident duration, but forecast delay using analytical formulae. In contrast, Garib *et al.* [5] have applied regression on historical data for predicting incident delay. However, they use incident duration as a predictor variable for incident delay, which means that delay can be predicted only after the incident is cleared. This work suggests the challenge of forecasting the incident impact promptly after an incident is detected to enable the best possible emergency response. Possible responses include variable message signs, reducing flow of traffic on upstream onramps, and prioritizing emergency responders.

In this paper, we propose a practical system for predicting the cost and impact of highway incidents using classification models trained on sensor data and police reports. We do this by applying machine learning models to the incoming stream of recordings from traffic sensors embedded in many highway systems. Our system allows for the rapid prediction of how severe a newly-detected incident will be by mining historical traffic sensor recordings as well as semi-structured police reports, weather data, and day of week to build a classification model of incident impact. The following are the paper's key contributions:

- Using historical traffic data from thousands of sensors, we build a baseline model that captures the expected conditions at each location and time of day over the study area.
- We predict the impact of an incident that has just started by using a classification model trained on his-

torical events. We show that combining traffic sensor recordings, initial police reports, weather data, and time of day we are able to reliably predict absolute and relative impact of a highway incident. We measure the impact by the duration of incident-induced delays and by economic losses from cumulative travel time delay [1].

- We demonstrate the high predictive power of our model on two real datasets of reported highway incidents from different regions, each over a two-month period. We also show the high level of transfer learning possible by training our model on one region and testing the model on incidents from a different year and region.

Although in this work our framework is specifically applied to traffic, our approach points a way toward studying the impact of events in other operational domains in which we have sensor data and external incident (event) logs [2].

2. INCIDENT IMPACT COMPUTATION

We first introduce relevant definitions and then outline an algorithm to identify the impact region for a particular incident considering the true road topology.

2.1 Preliminaries

In this work, we analyze contiguous sections of freeway. We use data from the California **Pe**rformance **M**easurement **S**ystem [3]) where tens of thousands of loop detectors fixed along major highways statewide measure quantities such as velocity, density, and flow every 30 seconds. For incidents, we use data from the California Highway Patrol (CHP)[4]. When an incident is reported, CHP notes details such as when and where it occurred. We provide further details in Section 4.1.

We model the regional highway network as a spatial network graph, $G = (U, E)$. Each node represents a detector, u_i (located at postmile x_i) and an edge $e_{i,j}$ exists for every pair of detectors such that there is a single road segment between two sensors where the road segment is part of a highway with PeMS detectors. The edge $e_{i,j}$ is directed such that j is upstream of i, where upstream is defined as in the opposite direction of normal traffic flow. In practice, at detector u_i, the vehicle count data is aggregated over all lanes to compute the total recorded flow in a given direction during a five-minute time window. This quantity known as *flow* is denoted by $q(i,t)$ with units in vehicles per hour (vph). The *average occupancy* of the highway over all lanes in the given direction is denoted by $\rho(i,t)$, with $0 \leq \rho(i,t) \leq 1$, where 0 indicates an empty highway and 1 indicates the theoretical limit of full coverage of the road by vehicles [4]. The *flow-weighted mean velocity* $v(i,t)$ at location i and time t is the chosen measure for the vehicle speed at a given location i and time t:

$$v(i,t) = \frac{\sum_{k=1}^{N_l} q_k(i,t) v_k(i,t)}{\sum_{k=1}^{N_l} q_k(i,t)} \tag{1}$$

where the speed $v_k(i,t)$ for each lane k and at detector i and at time t is computed according to the algorithm described by Chen [4].

Critical for determining abnormal traffic behavior is the identification of recurrent traffic conditions. Kwon [10] use a nearest-neighbor approach to find k days that had traffic conditions similar to the traffic conditions just before the start of an incident of interest. The average of the conditions at k-days is then taken to be the recurrent (or normal) behavior. For small k, the method leads to the possibility of choosing a date in the past with an incident—the opposite of trying to find recurrent non-incident conditions. Instead of using nearest-neighbors, we calculate the median of the speeds at each sensor location i and each point in time t and call it the recurrent velocity, $v^*(i,t)$. The median is robust to outliers, i.e. to incidents.

2.2 Understanding Cost of Delay

In this work, the economic loss due to lost productivity associated with an incident is directly proportional to the cumulative incident delay value according to the cost-benefit framework used in the State of California [1]. Thus, we will now detail the computation of the cumulative delay for all drivers in an incident impact region.

The delay on each road segment per unit of time is defined as the additional vehicle-hours traveled driving below free-flow speed, v_{ref}, per unit of time. In this study v_{ref} is 60 *mph* as is standard with other researchers using similar data [10]. This analysis further assumes that the flow, occupancy, and speed remain constant during every five-minute interval per road segment. We begin with the standard domain definition of delay $d(i,t)$ in the road segment of length l_i starting at detector i at time t. The delay essentially measures the vehicle-hours lost due to traveling below reference velocity [13]:

$$d(i,t) = l_i \times q(i,t) \times \max\left(\frac{1}{v(i,t)} - \frac{1}{v_{ref}}, 0\right) \tag{2}$$

where $v(i,t)$ refers to velocity at location i and time t as discussed in 2.1, $q(i,t)$ is the flow at location i and time t and v_{ref} is considered to be constant for all time t and location i with a value equal to 60 *mph*.

We are interested in the cumulative delay for all affected drivers, not simply the per segment per unit time delay, $d(i,t)$. Thus integrating $d(i,t)$ over a region A and time interval T the total delay over the spatial and temporal region defined by A and T would be given as

$$D_{tot} = \int_A \int_T d(i,t) du dt. \tag{3}$$

The total delay over region A and time T can have many causes. An approach as described by Kwon and Varaiya [10] is to separate total delay as follows:

$$D_{tot} = D_{inc} + D_{rec} + D_{rem} \tag{4}$$

[1]The economic losses in this context are the sum over the estimated number of affected vehicles of the extra time over expected travel times spent in the impact region of the incident multiplied by a cost per hour of lost productivity from the US Department of Transportation [1]

[2]This work is part of a larger effort called Live Operational Intelligence (LOI)[6], in which we are building systems for clients that analyze large amounts of streaming sensor and event data for supporting operational decisions. Interested industries include oil and gas production and drilling, energy delivery, logistics and transportation.

[3]Freeway Performance Measurement System (PeMS), http://pems.dot.ca.gov

[4]Also avaiable at http://pems.dot.ca.gov

where D_{tot} is total delay calculated from flow and speed, D_{inc} is the delay caused by non-recurring incidents such as collisions (D_{inc} is referred to as D_{col} in [10]), D_{rec} is the recurrent delay and D_{rem} is the remaining delay. Of our interest is D_{inc}, which is computed over a contiguous region called the *impact region* of an incident. The work of previous researchers to divide delay into components offers a general framework, but there is no work to date that mathematically derives expressions that capture the different terms. We now provide a derivation of the various components of delay.

We begin by defining an *impact region*, which we understand to be a contiguous region (contiguous in both space and time) beginning at the time and location of the incident, and where the traffic conditions deviate from the recurrent traffic patterns. This contiguous region goes forward in time and upstream in space, since after an incident, traffic backs up (Figure 1 illustrates the spread of the impact region with time for an example incident). Precisely, let the closest upstream sensor to incident I_a denoted by u_a and the time of incident I_a be t_a. Then:

Definition 1. *The impact region is collection of connected subgraphs A_t induced from G. A_t is constructed for each time t starting at $t = t_a$, such that for all nodes $i \in A_t$, either $v(i, t-1) < v^*(i, t-1)$ or $v(P(i), t-1) < v^*(P(i), t-1)$ where $P(i)$ is the parent of the detector i. At t_a, A_{t_a} is u_a.*

Now we can derive some related quantities. Substituting Equation 2 into Equation 3 and replacing the integral with summation (since measurements are discrete), Equation 3 can be rewritten as

$$\begin{aligned} D_{tot} = \quad & \textstyle\sum_T \sum_A l_i \times q(i,t) \\ & \times \max\left(\frac{1}{v(i,t)} - \frac{1}{v^*(i,t)} + \frac{1}{v^*(i,t)} - \frac{1}{v_{ref}}, 0\right). \end{aligned} \tag{5}$$

Equation 5 shows that the delay is trivially 0 when the current speed is greater than the free-flow speed, i.e., $v(i,t) \geq v_{ref}$. Equation 5 allows a rewriting of the total delay into two non-zero delay cases, as defined below:

1. Current speed is below the threshold speed and the free-flow speed, $v(i,t) < v^*(i,t) \leq v_{ref}$.

$$\begin{aligned} D_{tot} = & \textstyle\sum_T \sum_A l_i \times q(i,t) \times \left(\frac{1}{v(i,t)} - \frac{1}{v^*(i,t)}\right) \\ & + \textstyle\sum_T \sum_A l_i \times q(i,t) \times \left(\frac{1}{v^*(i,t)} - \frac{1}{v_{ref}}\right) \end{aligned} \tag{6}$$

2. Current speed is greater than the threshold speed but less than the free-flow speed, $v^*(i,t) \leq v(i,t) \leq v_{ref}$.

$$D_{tot} = \textstyle\sum_T \sum_A l_i \times q(i,t) \times \max\left(\frac{1}{v(i,t)} - \frac{1}{v_{ref}}, 0\right) \tag{7}$$

For the sake of simplicity, it is assumed that each spatio-temporal region of delay is caused by a unique delay-causing incident. So, when computing the impact region of an incident, we do not include regions corresponding to the start of a subsequent incident. Now, let A_t be the spatial extent of a particular incident's impact at time $t \in T'$ where T' is its temporal extent. Then,

$$\begin{aligned} & \textstyle\sum_T \sum_{A_t} l_i \times q(i,t) \times \left(\frac{1}{v(i,t)} - \frac{1}{v^*(i,t)}\right) = \\ & \quad \textstyle\sum_T' \sum_{A_t} l_i \times q(i,t) \times \left(\frac{1}{v(i,t)} - \frac{1}{v^*(i,t)}\right) \\ & + \quad \textstyle\sum_{T-T'} \sum_{A-A_t} l_i \times q(i,t) \times \left(\frac{1}{v(i,t)} - \frac{1}{v^*(i,t)}\right). \end{aligned} \tag{8}$$

Combining Equation 4, which states that $D_{tot} = D_{inc} + D_{rec} + D_{rem}$, with Equations 6, 7 and 8 yields the following definitions for D_{inc}, D_{rem} and D_{tot}:

Definition 2.

$$\begin{aligned} & \textit{If } v(i,t) < v^*(i,t), \\ & \quad D_{inc} = \textstyle\sum_{T'} \sum_{A_t} l_i \times q(i,t) \times \left(\frac{1}{v(i,t)} - \frac{1}{v^*(i,t)}\right)) \\ & \quad D_{rem} = \textstyle\sum_{T-T'} \sum_{A-A_t} l_i \times q_i(t) \times \left(\frac{1}{v_i(t)} - \frac{1}{v^*(i,t)}\right) \\ & \quad D_{rec} = \textstyle\sum_T \sum_A l_i \times q(i,t) \times \left(\frac{1}{v^*(i,t)} - \frac{1}{v_{ref}(t)}\right) \\ & \textit{If } v(i,t) \geq v^*(i,t), \\ & \quad D_{inc} = D_{rem} = 0 \\ & \quad D_{rec} = \textstyle\sum_T \sum_A l_i \times q(i,t) \\ & \qquad \times \max\left(\frac{1}{v(i,t)} - \frac{1}{v_{ref}(t)}, 0\right). \end{aligned} \tag{9}$$

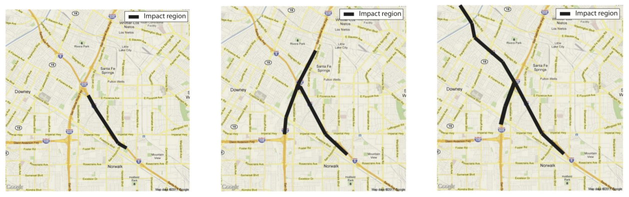
These equations precisely define quantities that can be understood as calculating the cumulative extra travel time experienced by all drivers during a certain time period and spatial region when traffic is not freely flowing ($v < v_{ref}$). The recurrent delay, D_{rec}, is the component of the currently experienced delay that drivers might always expect traveling through a region during a certain time of day. The incident delay, D_{inc}, is the cumulative extra time that all drivers spend in traffic associated with a given incident. The equations essentially codify our intuition that: if the current velocity $v(i,t)$ is below the expected velocity $v^*(i,t)$, then one component is the recurrent delay (from v_{ref} to $v^*(i,t)$) and the remaining component (from $v^*(i,t)$ to $v(i,t)$) is either incident delay if we are in an incident region or remaining delay otherwise. If the current velocity is above the expected velocity, all of the delay is the recurrent delay. The cost of an incident is then simply a constant dollar amount times D_{inc} (We take it to be \$11.20 for California [1].)

2.3 Computing Duration and Cost of Delay

Now that we have defined the impact region and cost of delay, we will briefly outline our algorithm for computing the delay associated with an incident.

We define the set of detectors of congested segments at time $t = 0$ as a group of detectors in which the speed at each detector is below the variable threshold speed, $v^*(i, 0)$, as defined in 2.1. Note that in published algorithms to date, e.g. [10], a fixed threshold speed such as $50mph$ has been used to determine membership in the preliminary impact region.

Intuitively, for each incident, $a \in A$, we begin at t_a with the closest upstream sensor u_a. Our algorithm continues in the next time steps $t > 0$ and a node is appended to the set, S_t, of nodes at time t, if the node is *congested*, i.e., if the flow-averaged speed is less than $(1-c)v^*(i,t)$, (where c is a constant), and the sensor was congested in the previous time step or is directly upstream of a congested sensor at time t. If the speed is greater than $(1-c)v^*(i,t)$ and less than $(1+c)v^*(i,t)$ and if the speed at the immediate downstream sensor was not between $(1-c)v^*(i,t)$ and $(1+c)v^*(i,t)$, the sensor is appended to the set of congested sensors at this time step. When both stop conditions, not congested and no upstream sensors at the previous time step, are met, time is incremented and the search upstream from u_a is repeated. The total process of identifying the impact region for a given incident a ends when $S_t = \emptyset$ at the end of



(a) $t = 4$ min (b) $t = 14$ min (c) $t = 29$ min

Figure 1: Spread of an impact region over time (measured since reported start of the incident) where the congested segments ($v(i,t) < v^*(i,t)$) are indicated by a bold black line.

a timestep iteration. This allows us to compute the impact region, A_t at any time t (In our experiments, we found $c = 0.05$ to be the most suitable). We skip the details for lack of space.

In contrast to the work to date on impact regions [10] in which the algorithm identifies congestion on one highway in isolation, our algorithm recursively searches upstream, thus identifying possible incident impacts to connecting highways. Our algorithm considers the true freeway topology. For example, Figure 1 shows an example from our Los Angeles dataset that illustrates the automatic incident impact region detection applied to real data. The incident starts in the southbound direction of the highway at the lower right of the black line shown in the figures. The incident then spreads upstream and the impact region spills over onto a connecting highway temporarily. This figure also illustrates how our algorithm considers the true road topology.

We are also interested in predicting how long the incident impact lasts. We define the duration, t_{tot}, as the duration of the contiguous incident impact region characterized by A' and T'. In contrast, the PeMS traffic database system used in California (URL mentioned before) computes the "duration" of an incident by subtracting the last time stamp of a police report from the first report. Because this method does not account for effects of an incident after police log reports, we instead computed the "duration" directly from the sensor recordings. We compared the durations we computed directly from the impact region with those of the police incident logs and found that impact regions tend to last longer than the PeMS "duration". In the data, we see examples such as one incident with one report of a crash and a second and final report of an officer dispatched. In this example, our algorithm will compute a longer duration because we measure until traffic conditions return to recurrent conditions.

3. PREDICTING INCIDENT IMPACT

3.1 Problem Definition

The impact of an incident can be characterized in multiple ways. One possibility might be how much the slowdown will be at a particular location or another might be predicting a metric over the entire impact region, as previously defined. As mentioned early, we predict the latter—the macro level impact caused by an individual incident. The problem is: "*Given an incident just occurred, what will its impact be?*" We define impact in two ways: (i) The monetary productivity costs of the cumulative non-recurrent *delay* in Equation 9 that will be caused by a particular incident (ii) The *duration* which can understood to be the temporal extent of the impact region. Both metrics are moderately correlated and are different measures of impact. The cost of delay is a constant factor accounting for the value of lost time times the integration of the delay magnitude over space and time, which could cause incidents of different duration, but differennt magnitudes, to have the same delay value.

Ideally, we would aim to predict the precise numerical values for the two quantities of interest. We tried several regression techniques and models such as regression trees. We found that predicting the precise values (for either cost or duration) is a difficult problem. For example, the least root relative square error obtained with regression trees for duration was close to 100% in many cases, meaning that our error would be the same as if were to make the trivial prediction of the overall mean as the predicted value. To overcome this, we mapped the problem to predicting the impact as a class variable. This makes the problem more tractable and from the domain expert level no less useful. Traffic operators are very often satisfied with knowing the general range of the impact, namely "Will the impact be negligible, moderate, or severe?" Or, "Given two incidents, which one will have a greater impact?" Answering these questions is the first step to solving the domain-specific need of identifying how to prioritize limited resources to mitigate the effects of incidents reported to emergency dispatchers.

3.2 Building the Feature Vector

A big challenge in predicting impact is building a feature vector that combines data from disparate structured and unstructured data sources. The first step is to collect sensor recordings near the incident location. We map a reported incident to the closest upstream sensor on the directed graph $G(U,E)$. Note that the impact of an incident typically spreads upstream, i.e. there is back-up behind an incident (as depicted in Figure 1). At this closest upstream sensor as well as one directly upstream and one farther upstream, we collect the current conditions including speed $v(i,t)$, expected speed $v^*(i,t)$, and road occupancy $\rho(i,t)$. Figure 2 depicts the spatial relationship.

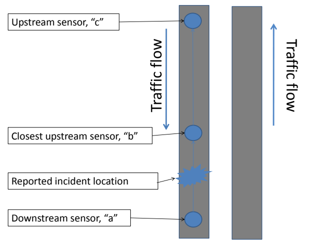


Figure 2: Diagram of sensor locations upstream ("b", "c') and downstream ("a") of the incident. ©Mahalia Miller

In addition to the directly-measured quantities, we compute a few meaningful physical quantities such as $v_{diff}(i,t) = v^*(i,t) - v(i,t)$ at the three sensors in Figure 2. We also construct a few functions as part of the feature vectors. We aim to capture the physical relationships that define the traffic through these functions. For example, one function that we compute inspired from Equation 2 is: $f(i,t) = q(i,t) \times (\frac{1}{v(i,t)} - \frac{1}{v_{ref}})$.

We then include the second type of data–California Highway Patrol (CHP) reports since initial testing indicated an improvement in accuracy by including these extra features. In addition to basic information including reported start time of day, the structured part of the CHP records offers a type class (a total of 36 types), which we mapped to 9 types: *Traffic hazard, Collision+no/minor injuries, Collision+major injuries/ambulance, Natural weather hazard, Lane closure, Fire, Collision-no details, Hit and run, Other.* We also include features parsed from unstructured police logs in the first 2 minutes after the reported start of an incident.

Another feature we used is whether within one hour of an incident it was raining and if so, by how much.

In addition we considered features pertaining to the topology of the highways. As of now, we consider the number of lanes as a feature. In future work, we aim to add more features such as the proximity to an exit.

In addition to building feature vectors based on data for each incident, we construct feature vectors for possible pairwise combinations of incidents. This dataset is used for predicting which incident of a pair will have relatively higher impact.

3.3 Bin Selection

For every incident of interest, we compute the incident impact both in terms of cost of delay and duration. This is done using the algorithm and definitions introduced in 2. Once we have obtained these numerical quantities, we map the two prediction variables to classes or bins.

There are many ways to construct bins [7], such as *equi-width*, *equi-height* and *clustering*. We experimented with many of these approaches and found that either the bins lead to a very skewed distribution of elements across classes (such that it makes sense to just learn a trivial learner) or have no semantic meaning for domain experts. The latter is important since this an emerging application to be implemented in practice by internal company teams and external agencies. Finally, as reported in our results, we chose to do manual binning. This allows us to divide the range of values into sub-ranges, so that they have a semantic meaning. For example, for a range of duration d from $0-140$ minutes, the range of $d \leq 5$ indicates negligible impact, $5 < d \leq 30$ indicates moderate impact, $30 < d \leq 60$ indicates high impact and $d > 60$ indicates severe impact.

3.4 Building Prediction Models

Once we have constructed the feature vector and mapped the continuous prediction variables to discrete classes, we are ready to build prediction models. We can use classification models for prediction, where each bin becomes a class.

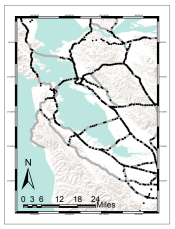
Before building classification models, we do feature selection. We remove those features that provide no or very little information gain. For prediction, we have constructed a suit of classification algorithms, including: (1) a classification tree model based on C4.5 trees with various settings, (2) an ensemble of short trees constructed using Adaboost, (3) a k-NN classifier (with various ks and distance weights) (4) a multiclass classification algorithm built with 5 models by varying C4.5 tree model parameters (5) a multi-layer perceptron and (6) radial-basis function network. For the decision trees we varied the confidence threshold (up to 0.2), minimum number of nodes at leaf nodes (1-5), and whether or not to force binary splits for nominal variables. For the nearest neighbor we varied the number of neighbors (from 1-7) and how they were distance weighted. The intuition was to select techniques from various classes of classifiers (tree-based, nearest neighbor, and function construction). We also put a minimum threshold for recall, precision for each class (typically set to 0.25). Then given a dataset, we choose the classifier that gives us the lowest error given that the recall and precision for each class is greater than the threshold.

A further refinement would be to introduce a cost matrix once the relative trade-offs between different types of prediction error are quantified for a particular transportation district.

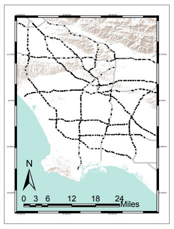
4. EXPERIMENTAL RESULTS

4.1 Dataset Description

As mentioned earlier, we extracted our traffic sensor datasets from PeMS. We chose two different regions and seasons to help test the versatility of our framework. The first dataset is two months of sensor recordings in the Los Angeles area (Caltrans District 7), namely January and February 2009. The dataset includes 28, 818,432 recordings from 1696 mainline (open freeway) detectors. Note that the full dataset includes detectors on ramps and freeway to freeway connectors, but we consider only a subset (1696 detectors here) that are embedded the main highway lanes. The second dataset is July and August 2011 for the San Francisco Bay Area (Caltrans District 4) and is 32,587,200 recordings from 1825 mainline detectors. Figure 3 shows the spatial distribution of sensors and freeways in the two regions.



(a) San Francisco Bay Area



(b) Los Angeles Area

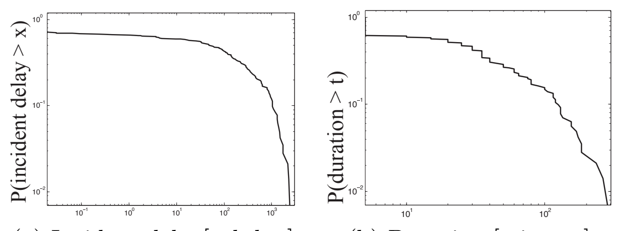
Figure 3: Map of the freeway sensors (black dots) and major highways (in grey) used for predicting the impact of incidents. ©Mahalia Miller

We focused on incidents starting on I-5 in eastern Los Angeles and US Route 101 in Silicon Valley in the San Francisco Bay Area, because they provide samples with high traffic volumes and high potential impact on productivity. For the first dataset, we specifically investigated incidents on southbound interstate I-5 between postmiles 116.9 and 130 and their effect on northbound and southbound I-5 traffic as well as on connecting highways. There were 173 incidents in this subset of the corridor during January and February 2009. This second dataset focuses on incidents on southbound US Route 101 between postmiles 400 and 410, which numbered 244 during the study period, and their effect on US Route 101 and connecting highways. We investigated weekday incidents only and used recurrent speeds calculated for weekdays only to be consistent.

For incident details, we used two types datasets from the California Highway Patrol (CHP) as collected by PeMS and also freely available in near real-time from CHP [5]. First, there were structured logs with incident start times and locations. Second, the incident details were semi-structured with free text police logs. After parsing the data with aid from the CHP dictionary of shorthand (can be found at the URL mentioned above) and including some common misspellings, we extracted features such as the number of officers on the scene, if a truck was involved, if a tow truck was mentioned, and the number of vehicles reported. For example, one line in the police log for a particular incident is: "11779132,01/23/2009 00:00:00,RP IN A GRY MERZ BENZ VS WHI BIG RIG L/UP94757,ADD". This police log was made available one minute and 2 seconds after the reported start of the incident and says that the reporting party (RP) was in a gray Mercedes Benz and in a crash with a white semi-truck (BIG RIG). From this line of raw text, our natural language processor identifies that two vehicles are involved. Additionally, it notes that a truck is present. These clues are then added to the feature vector.

In addition, we extracted hourly rainfall data from the California Department of Water Resources databases by mapping each sensor to the closest weather station [6].

The distribution of both prediction variables is quite skewed and in Figure 4(a) we plot the distribution of incident delay (cost divided by a constant factor) in log-log scale from the LA dataset and observe the high concentration of low delay events. We observe that the distributions follow a exponential distribution. Second, in Figure 4(b) we see a similar trend for the incident duration identified by our method.



(a) Incident delay[veh-hrs] (b) Duration [minutes]

Figure 4: Probability of an incident having an impact greater than x, where x is incident delay in (a) and incident duration in (b) (Complementary CDF, in log-log scale). Note the high number of low magnitude events.

4.2 Results

We evaluate our technique for its accuracy in predicting the impact class, measured by either incident cost or duration. The results are obtained with 10-fold cross validation. We do 10 runs of 10-fold cross validation and present results averaged over the 10 runs. For each run we choose the maximum cross validation accuracy (recall that in our model every run had a suite of classifiers) obtained in the run and the overall accuracy result is the average over these runs. For each maximum accuracy, we note the precision, recall and f-measure for each class and present the average of these values. We first present results for cost and then for duration. We present results rounded of to the second decimal place.

[5]California Highway Patrol (CHP) Traffic Incident Information Page, http://cad.chp.ca.gov

[6]California Department of Water Resources California Data Exchange Center,http://cdec.water.ca.gov/intro.html

4.2.1 Predicting Cost

In Table 1 we provide the accuracy results for predicting the cost of delay on the San Francisco Bay Area (SF) dataset. For each experiment, we describe the number of classes, the bounds for the classes, the counts in each class and the overall accuracy. For example, experiment SF-3 has three classes, with buckets being, $cost \leq \$10$, $\$10 < cost \leq \$250$ and $cost > \$250$. We have chosen semantically meaningful buckets and not buckets that might give us higher accuracy. A cost of less that \$10 can be thought to be insignificant, a cost of \$10 to \$250 can be thought to be moderate, \$250 to \$1000 as high and a cost in excess of \$1000 as exorbitant. We also provide the precision, recall and f-measure for each class in Table 1. This is useful since for a skewed distribution we need to make sure that we have reasonable recall and precision for classes with fewer examples. We obtain good results for two classes both in terms of accuracy and precision/recall for each of the individual classes. In particular, we can decide with high degree of confidence if the cost of an incident is going to be low (less than \$10). As expected, as the number of classes increases, our accuracy goes down, but even for four classes (experiment SF-4) we have reasonable results. Table 1 also details the results from the Los Angeles region (LA) dataset. It is interesting to note the cost of incidents are in general higher and the bin boundaries are shifted accordingly (The range of \$250 - \$1000 is removed because we have very few training examples). Since traffic in LA is notoriously bad, these results are expected.

4.2.2 Predicting Duration

In Table 2 we show the results for the SF and LA datasets for predicting the temporal duration. The bin boundaries are in minutes, so the buckets for the three experiments are (1) $duration \leq 5min$ and $duration > 5min$, (2) $duration \leq 30min$ and $duration > 30min$, and (3) $duration \leq 5min$, $5min < duration \leq 30min$ and $duration > 30min$. For emergency responders, a major decision support need is identifying a "false alarm" or near 0 duration event. In this 2 class paradigm, our model is over 90% accurate for LA, and 87% for SF.

4.2.3 Comparing Incidents

Finally, from a relative magnitude perspective, we are interested in knowing which of two incidents will have a higher impact, whether measured by incident cost of delay or temporal duration. Table 3 shows that our model can predict the relative magnitude of a pair of incidents with high accuracy.

4.2.4 Extensions: Transfer Learning and Multi-tiered Classification Model

To investigate the versatility of our proposed framework we asked the following question: to what extent will a model trained on one region transfer to another region? Table 4 show the accuracy for a model trained on SF incidents and directly applied to predict the impact class of incidents in LA. The results suggest that we do transfer learning from region to another and demonstrates the versatility of our framework. In comparison to the LA results when trained on LA data, the LA results when trained on SF data generally show only marginal losses in accuracy. This result is encouraging and suggests that for practical deployment,

a) Summary of experimental results for cost

Experiment	Classes	Bounds ($)	Counts	Accuracy
SF-1	2	10	114, 124	95.59
SF-2	2	100	157, 81	87.86
SF-3	3	10, 250	114,54,70	81.09
SF-4	4	10, 250, 1000	114,54,34, 36	73.73
LA-1	2	10	75, 97	91.40
LA-2	2	250	94,78	88.84
LA-3	2	1000	120, 52	82.33
LA-4	3	10, 1000	75, 45, 52	75.87

b) Precision, recall and f-measure for cost

Experiment	Class	Precision	Recall	f-Measure
SF-1	1	0.95	0.96	0.95
	2	0.97	0.95	0.96
SF-2	1	0.91	0.91	0.91
	2	0.82	0.83	0.82
	1	0.92	0.96	0.94
SF-3	2	0.62	0.55	0.58
	3	0.75	0.76	0.76
	1	0.94	0.96	0.95
SF-4	2	0.61	0.62	0.62
	3	0.51	0.46	0.48
	4	0.47	0.46	0.47
LA-1	1	0.90	0.91	0.90
	2	0.93	0.93	0.92
LA-2	1	0.91	0.88	0.90
	2	0.86	0.89	0.88
LA-3	1	0.87	0.88	0.87
	2	0.72	0.68	0.70
	1	0.87	0.92	0.89
LA-4	2	0.56	0.53	0.55
	3	0.73	0.72	0.72

Table 1: Results for predicting cost. We showed high overall accuracy and high precision and recall for each class.

a) Summary of experimental results for duration

Experiment	Classes	Bounds (min)	Counts	Accuracy
SF-1	2	5	146, 92	87.22
SF-2	2	30	188, 50	81.05
SF-3	3	5, 30	146, 42, 50	72.99
LA-1	2	5	85, 87	91.51
LA-2	2	30	118, 54	79.65
LA-3	3	5, 30	85, 33, 54	71.98

b) Precision, recall and f-measure for duration

Experiment	Class	Precision	Recall	f-Measure
SF-1	1	0.89	0.91	0.90
	2	0.85	0.82	0.0.83
SF-2	1	0.83	0.96	0.89
	2	0.63	0.24	0.35
	1	0.92	0.96	0.94
SF-3	2	0.0	0.0	0.0
	3	0.47	0.84	0.61
LA-1	1	0.92	0.91	0.91
	2	0.91	0.92	0.92
LA-2	1	0.95	0.74	0.83
	2	0.62	0.92	0.74
	1	0.91	0.86	0.88
LA-3	2	0.43	0.38	0.40
	3	0.62	0.71	0.66

Table 2: Results for predicting duration of an incident. We show high overall accuracies and good precision and recall for most experiments.

Measure	Region	Accuracy
Cost	San Francisco	96.01
Cost	Los Angeles	94.89
Duration	San Francisco	96.3
Duration	Los Angeles	92.21

Table 3: Prediction accuracy (%) for pairwise relative greater magnitude of incident *cost* and *duration*. Our model is able to reliably predict the relative magnitude of the impact.

we will not need to train separate models for each region. The model-building part of this exercise was a bit tricky since CHP in LA and CHP in SF had different "types" for incidents and the distribution of variable values could be different. While we might expect that collecting more data regarding a particular incident as time passes might improve our prediction, we find no significant improvement in accuracy in our specific application. One reason might be that most incidents are of short duration and for longer duration events, the first few minutes are not full descriptors of the full progression of the incident impact. More likely, however, is that we have a limited number of events greater than 15 minutes, for example, and thus need more events to get a significant improvement over the multi-class baseline model.

4.3 Discussion

The overall classification results are encouraging. In addition, we discovered models that brought new insights to the domain experts. For example, the pruned C4.5 tree depicted in Figure 5, which was discovered for predicting false alarms in the Los Angeles dataset (70/30 train/test) has over 90% accuracy. The root node of the classification tree makes a decision based on how different the speed at a sensor (two sensors upstream from the incident) is from the recurrent speed. Then, if the speed is significantly below the recurrent speed for that location and time of day, the model predicts that an accident is occurring. If not, the model checks how densely packed the road is. If the road is relatively empty, then the model predicts the report as a false alarm. If not, the model checks to see if any police report mentioned vehicles involved within the first two minutes. If so, the model predicts that an accident is occuring. If not, the model classifies the report as a "false alarm" (meaning that impact would be negligible). This automatically created and pruned tree is consistent with intuition and offers insights, such as that being only k *mph* slower than recurrent conditions at a certain time of day and location given an incident report is a good indicator that it is not a false alarm. More generally, these models show which variables are important for prediction.

In addition, our approach is practical and feasible in real time since both feature extraction and model execution are quick.

As with most data mining problems, when modeling real-world physical systems, having good features is critical. In this regard, we found that the features we constructed that were trying to imitate the underlying physical models were

Measure	Bounds	Accuracy
Cost	\$10	90.07
Cost	\$10, \$100	86.05
Duration	5 Minutes	91.28
Duration	5, 30 Minutes	73.84

Table 4: Prediction accuracy (%) by each bin selection choice for k classes of *incident delay* trained on the SF dataset and tested directly on LA: the model results suggest good transfer learning.

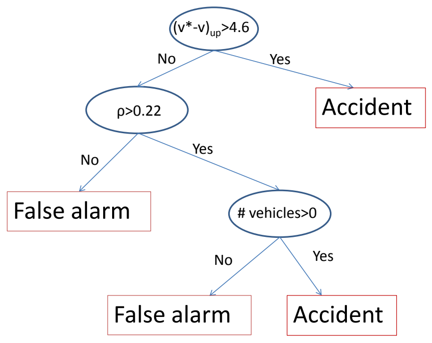


Figure 5: Classification tree for predicting "false alarms" for LA, which predicts with >90% accuracy and is consistent with intuition of domain experts.

often good predictors. This might point to a way for constructing models for physical systems–namely that we look at some equations from domain experts and capture some of those relationships in our feature vectors. Furthermore, in this work we collected features of different types describing: the incident, changes in the state variable (such as deviation from recurrent velocity), the environment (such as rain), and the topology (refinement is ongoing). We have shown that this exercise also improves the accuracy of our models.

Furthermore, for the predicted variable it is important to first choose a physically useful quantity to predict (such as the ranges for cost or duration) even at the expense of accuracy in models. Making such a choice often causes a problem with class distribution and makes an already challenging problem even more so. Nevertheless this is needed to get useful models that domain experts want to use.

Our results show that given a cyber-physical system, machine learning techniques can help predict the impact of an incident. We believe that this way of obtaining details of incidents from some event database, computing impact using sensor data and building a model that correlates the two can be extended to other domains.

5. CONCLUSIONS

In this paper we have proposed a system for rapid prediction of the cost and impact class of highway incidents that demonstrates the real-world applicability of a predictive model based on classification models trained on available urban data. The feature vector is built from structured data from sensor networks on highways as well as semi-structured text collected at different points in time. With experiments on real-world data, we have demonstrated that our models are good predictors of incident impact. Thus, our work supports a decision–response to a highway incident–that until now has relied on human expertise. In addition to advancing the use of machine learning for highway operations in practice, these results open the door to further studies on applying statistical methods to traffic management and related applications. We are discussing these results with HP Services and a government agency and expect to deploy the underlying techniques to real-time operations.

6. REFERENCES

[1] *Valuation of Travel Time in Economic Analysis*. Office of the Secretary of Transportation, U.S. DOT, 2003.

[2] *Traffic congestion and reliability : linking solutions to problems*. Federal Highway Administration, 2004.

[3] S. Boyles, S. Boyles, and S. T. Waller. A stochastic delay prediction model for real-time incident management, 2007.

[4] C. Chen, Z. Jia, J. Kwon, and E. van Zwet. Statistical methods for estimating speed using single-loop detectors. *Submitted to the 82nd Annual Meeting of the Transportation Research Board*, 2002.

[5] A. Garib, A. E. Radwan, and H. Al-Deek. Estimating magnitude and duration of incident delays. *Journal of Transportation Engineering*, 123(6):459–466, 1997.

[6] C. Gupta, K. Viswanathan, L. Choudur, M. Hao, U. Dayal, R. Vennelakanti, P. Helm, A. Dev, S. Manjunath, S. Dhulipala, and S. Bellad. Better drilling through sensor analytics: a case study in live operational intelligence. In *Proc. of the 5th International Workshop on Knowledge Discovery from Sensor Data*, SensorKDD '11, pages 8–15. ACM, 2011.

[7] C. Gupta, S. Wang, U. Dayal, and A. Mehta. Classification with unknown classes. In M. Winslett, editor, *Scientific and Statistical Database Management*, volume 5566 of *Lecture Notes in Computer Science*, pages 479–496. Springer Berlin / Heidelberg, 2009.

[8] A. Ihler, J. Hutchins, and P. Smyth. Adaptive event detection with time-varying poisson processes. In *Proc. 12th KDD*, pages 207–216, New York, NY, USA, 2006. ACM.

[9] A. Khattak, X. Wang, and H. Zhang. Incident management integration tool: dynamically predicting incident durations, secondary incident occurrence and incident delays. *Intelligent Transport Systems, IET*, 6(2):204 –214, 2012.

[10] J. Kwon, M. Mauch, and P. Varaiya. Components of congestion: Delay from incidents, special events, lane closures, weather, potential ramp metering gain, and excess demand. *Transportation Research Record*, 1959:84–91, 2006.

[11] X. Li, Z. Li, J. Han, and J.-G. Lee. Temporal outlier detection in vehicle traffic data. In *ICDE '09*, pages 1319 –1322, 2009.

[12] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xing. Discovering spatio-temporal causal interactions in traffic data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 1010–1018, New York, NY, USA, 2011. ACM.

[13] A. Skabardonis, P. Varaiya, and K. F. Petty. Measuring recurrent and nonrecurrent traffic congestion. *Transportation Research Record*, 2003:118–124, 2003.

Sensing Places' Life to make City Smarter

Prof. Stéphane Roche
Center for Research in Geomatics
Université Laval
Québec, QC, Canada
+1-418-656 7841
stephane.roche@scg.ulaval.ca

Prof. Abbas Rajabifard
Centre for SDIs and Land Administration
The University of Melbourne
Melbourne, Victoria, Australia
+61-3-8344 0234
abbas.r@unimelb.edu.au

ABSTRACT

This paper explores the smart city concept and proposes an innovative way of sensing urban places' life using aggregation of devices sensors (cameras…) and human sensors (VGI, geosocial networks) datasets. The paper also discusses the need of an enabling geospatial information platform to facilitate data discovery and access in order to support smart cities' operations. Indeed, in this context, Spatial Data Infrastructure plays an important role and acts as an enabling platform linking governments authoritative spatial information with crowd sourced, voluntary information initiatives.

Categories and Subject Descriptors

- Urban sensing and city dynamics sensing
- Smart recommendations in urban spaces
See also http://www.acm.org/class/1998/

General Terms

Measurement, Design, Security, Human Factors.

Keywords

Smart city, sensors, place, spatial enablement, Volunteer Geographic Information, crowdsourcing, enabling platform, SDI.

1. INTRODUCTION

Our living world becomes more and more complex every day and faces growing challenges. Urban areas are especially under pressure, as their population will exponentially grow in the coming decades. Various economical, environmental, social, demographical, etc. issues and stakes will lie ahead in the near future. These issues are even stronger while the age of real-time and location-aware information accelerates social and spatial changes. Building "smarter" (intelligent) urban environments and communities is one of the potential solutions that is currently proposed and explored to tackle these challenges.

Smart City is one the most fashionable answers that was firstly proposed, and has been used as a marketing artefact by companies like IBM (Smarter Planet initiative), CISCO (Smart+Connected Communities division) or last but not least and more recently Intel (Collaborative Research Institute for Sustainable Connected Cities), and also by some of the biggest cities in the world (Rio de Janeiro, Singapore, London…). A smart city is roughly described as a platform or a system of systems, essentially based on three components: Sensors, Networks and Engagement (actuators).

But is this 'Smart Cities' approach makes cities smart enough to meet the current and future challenges? The answer is certainly NO [1], but it s not obvious. Making a city smarter is neither only a technological infrastructure issue, nor a managerial one. It is also and essentially providing citizens a better and safer way of living in urban areas, in the places where they live, work, have fun, consume… Therefore, sensing life in those places is a major stake to understand new city dynamics and then to design better living urban environments. The Singapore Live project running by the MIT SENSeable City Lab. is a good example of such a stake.

With the many challenges facing society today at multiple scales, location has emerged as a key facilitator in decision-making. Location data is now commonly regarded as the fourth driver in the decision-making process. The location provides more intelligent data analysis due to improved analytical and visualisation capabilities. Location-based services and information are the basic components needed to dynamically describe and represent places' life [2]. We live in the Global Location Age. "'Where am I?' is being replaced by, 'Where am I in relation to everything else?'" [3]. Indeed with the exponential growing of location-based social networks (geosocial), Geoweb 2.0 and geoinformation crowdsourcing, citizens are increasingly involved in the production of geographic information. This kind of information, voluntarily produced and diffused by people, mainly refers to the places they live/use. Indeed, people whom live in a place are often the "experts" of this place [4]. The active engagement of citizens is then a major requirement for smarter operations of a city, and this needs for citizens to be spatially (digitally) enable [1]. Capturing the sense of places is then a major stake for urban communities that look for a smarter way of development. Smart city infrastructure has to be based on an Enabled platform for aggregating formal data sensed by device sensors and sensible knowledge sensed by citizen sensors.

This stake also poses fundamental research problems: How to efficiently and dynamically senses urban places' life? What kind of mechanisms should be developed to actively engage citizens as active sensors? What should be included in a Volunteer Geographic Service – VGS to make this service useful for engaging citizens in smarter city operations?

This paper precisely aims at providing recommendations for sensing urban places' life by using aggregation of devices sensors

UrbComp'12, August 12, 2012. Beijing, China. Copyright 2012 ACM 978-1-4503-1542-5/08/2012 …$15.00.

(cameras…) and human sensors (VGI, geosocial networks) location-based data. It also discusses the data infrastructure that is needed to facilitate data access, discovery and linkages. Section 2 provides a brief overview of smart city challenges, and especially argues that a city could not be smart without spatially enabled citizens. Section 3 discusses what the concept of Place does really mean in the smart city context, and to what extent place does matter to make city smarter. Section 4 finally develops the proposed methodology, based on a work in progress, for sensing places' life in smart city, and provides example applications (e.g. crisis management).

2. HOW TO MAKE CITY SMARTER?

2.1 Smart city weaknesses

A recent Wired.uk.co paper entitled "Surely there's a smarter approach to smart cities ?" [5] provides a really good overview of the weaknesses of current smart city deployment. Based on the analysis of the IBM and Cisco's projects, this paper concludes that this approaches are weak. The main weaknesses (this is not exhaustive) are the followings:

- These smart city approaches are to much technologically driven and mainly reflect the most basic functionalities of the Internet of Things,
- These are based on a "One-size fits all" and very top-down strategic approach "to sustainability, citizen well-being and economic development",
- The 'smartness' is limited to efficiency and there is most of the time nothing really develop to improve and increase the flexibility and adaptability of the city's operations,
- Citizens are not really considered in these projects, except as passive sensors (for instance by unconsciously providing their location through their smart-phone activities),
- These initiatives are not really scalable, only small scale is really taken into account, and city is considered as an indivisible whole, as a single coherent and predictable unit.

2.2 Smart city requirements

The Latin etymology of intelligent ("intelligentare") refers to the ability to learn about/from someone or something, understand, interact with one's environment and, act relevantly. This ability particularly consists of adaptability to changes in the environment and, capacity for knowledge from this environment.

Then "smart city" refers to the capability of a city (or a community) to understand events or phenomena that characterized its internal and external dynamics (crisis, transport issues, road traffic, socio-economical transformations, demographical changes, etc.).

Moreover, in the current hypermodern context, a smart city must also be able to identify the main components of an event (where, when, who, what, how), to analyze it, to provide location-aware (and contextual) sense to it and, to react (actuate) properly (and in real-time – at least compliant with the nature of this event).

In order to achieve these goals, urban infrastructures (transport, communication, water supply or energy...) have of course to be redesigned and upgraded by articulating their material components with existing capabilities of new active and communicant sensors and actuators. This refers to the Web 4.0, the Internet of Things, the Web of communicating objects.

But this also requires the cities' capacities of analysis, diagnosis and communication to be substantially and significantly improved. This requirement is very much in line with the issue of "big data" that increasingly grows while data coming from various sensors are multiplied and accumulate. This refers to the Web 3.0, the semantic web, this of meaning, of sense, of signs.

Last but not least, urban society that is to say citizens, institutional organizations, public-private stakeholders also have to develop or increase (new) adhoc skills to act in this smart urban environment. It must negotiate new rules to work all together, new modes of opened governance, adapted contribution mechanisms (as crowdsourcing), new way of participation. This refers to the Web 2.0, the participatory geoweb [6], this of geosocial, user generated contents, participation, social networks and open data.

Major issues also arise as regard as citizens' digital and spatial literacy. This active and engaged citizen (see section 3.2) is indeed the main driving force of a "smart city". The growing amount of location-based contents generated by connected - anytime and anywhere - user (citizens' produsers equipped with smart phones) ; the exponential growth of volunteered geographic information (Foursquare: 20 million users and 2 billion check-ins) trough social networks (most of which are Location-Based social networks) become the basic features of the Spatially enabled society. This is defined as an "evolving concept where location, place and any other spatial information are available to governments, citizens and businesses as a means of organizing their activities and information" [7].

From a practical point of view, and in the smart city context, "spatial enablement" refers to the individuals' (or collective) ability to use any geospatial information and location technology as a means to improve their spatiality, that is to say, the way they interact with space and other individuals on/in/through space. Spatiality is the dynamic component of place making.

Therefore, more fundamentally a spatially enabled citizen is characterized by his ability to express, formalize, equip (technologically and cognitively), and of course consciously -or unconsciously- activate and efficiently use his spatial skills.

Smart urban solutions have to be built on the vision of citizens as active sensors on one hand and on spatial enablement of citizens via social networks on the other hand. These solutions have to be inline with improvement of navigation related spatial skills using geographical information and techniques for annotating spaces with digital information. These kind of solutions have also to be built on the potentials offered both by embedded sensors to crowdsource the process of collecting geo-referenced information about places in the city and social networks to disseminate this information and democratize access to it.

3. MORE PLACES, LESS SPACE

3.1 Back to the concept of Place

In this section we argue that places, and sensing urban place especially, are central as a way to understand cities and then to smartening its urban management and design. But what the concept of Place does really mean in the smart city context, and to what extent place does matter to make city smarter?

Place has different meaning depending on the field of interest. As mentioned by Mike Goodchild [8] "*The concept of place has a long history in geography and related disciplines, but has been plagued by a fundamental vagueness of definition*". From a social perspective, Place is considered as "an expression of context", an expression of the "value of linking individual behavior to context" [8]. Place is often used in the sense of community or neighbourhood, implying an informal relationship to an area surrounding the individual's place of residence [8].

From a more geographical point of view, Michel Lussault [9] argues, "spaces are socially constructed"; and all space is a formal arrangement of artefacts, materials and ideas, and is characterized by specific attributes like scale, metric, substance and configuration. He considers space as a meta-concept that is contextually and contingency shaped as places, areas (territories) or networks. In this framework, place are space where Euclidian / metric distance doesn't matter.

The concept of place does not certainly mean the same in the location age context than it meant before. However the potential of location-based technologies is not limited to the ability to say "where" we are or "who" and "what" are close to us. It could also provide extended capabilities to users, such as accessing new forms of virtual spaces or increasing physical space in which they live (by adding digitals artefacts for instance). A sort of "informational thickening" of places and networks (already identified in the seminal book of Mitchel [10], appears from these new practices. This hypermodern urban contexts and spatially enabled society are indeed characterized by its wikinomics [11], where places of interest could be identified, sensed and characterized by citizen seen as sensors (geosocial 2.0).

For Manuel Castells [12], this new form of location-based social networks characterizes the Society of Flows. Castells said "*Time and Space do not disappear but they are transformed*", one individual could possibly be in two different spaces in the same time, people constantly evolve in overlapped multifunctional networks. Then, places become multifunctional and multi-meaningful. Castells finally argues, "*The space of flows dominates the space of places*".

3.2 Does Place still matter for smart city?

As it was highlighted before, people increasingly live in high-density urban, often high rise and multi functional buildings. These increasingly urbanised populations will predominantly live in multi-level, multi-purpose, highly engineered, high-rise developments. Cities require significant infrastructure above and below the ground. Rapidly expanding vertical cities and their populations will experience a range of new environmental, social and economic challenges. With this in mind, then the question would be, does place still matter in a spatially enabled society? Contrary to the Castells' arguments, we believe the answer is undoubtedly yes. But it's not obvious, and the reality more complex than a simple tension between place and flow.

While concepts of place and network are becoming central in the "spatially enabled society", the relevance of the concept of territory (space-area) is under erosion. Geographers conceive territories as areas, which are sensitive to Euclidian distances, bounded by identifiable and clear boundaries, and characterized by continuous components and phenomena. Most of those territories gradually disappear. In the same time, networked places, characterized by a spatial and temporal discontinuity of their embodied components become preeminent. In this context of hyper-modernity, places become hyper-places where physical Euclidean distances do not still relevant, but in which other forms of distance (time, connectivity, digital, social...) structure social relationships and human spatiality.

Our daily living spaces may not be reduced to a horizontal layer, physically accessible. Instead, "virtual information" that are mixed (mashed-up) are fully part of its. This ability to choose to live into (with) places, essentially hybrid, in their informational thickness, allows individuals to precisely "be fully in their world". Spatiality in this context is essentially based on co-spatiality. This capacity for engagement / disengagement (for navigation through the entire informational thickness of places), is actually part of all spatial actions undertaken by individuals, and in this sense refers probably to a new type of spatial skills that should be analyzed.

Further, in the context of cities, the lack of an efficient and effective three dimensional solution limits the ability of the public to visualize and communicate 3D developments, the ability of architects, engineers and developers to capitalize on the full potential of 3D city models; the ability of governments and developers to visualize multi-level developments resulting in increased costs and delays; and the ability of land registries to administer a title registration system that can accommodate these increasingly complex multi-level developments.

Having said that, in the general context of geocommunication (communication based on location and mobility), places and then (by linking) networks become the preeminent forms of space (much more than area). More precisely, the construction of spaces is mainly based on the aggregation of places; those aggregations are more or less organized, more or less sustainable. Those places may refer to temporal sequences of positions (location + relations) of one mobile and communicant individual, which together generate a space-network (formalizing a "route"). But they can also refer to locations (in a given time) of various individuals, all connected, all members of a "social network", and in this case space contextually formalizes the social network. It can finally be a hybridization of the first two, that is to say dynamic space-networks, based on the tangled routes and crossings of the members of a given social network. In very rare cases, places become territories with fuzzy and shifting boundaries. Then these space-areas formalize the spatial extent of the interactions occurred between the members of a social network.

Spontaneous and localized contributions of individuals (localized tweets, Facebook Places or foursquare chek-in) are most often materialized by Points of Interest (POI). This POI could be considered as new forms of spatial projections of social relationships and human spatiality. These contributions raised through mobile technologies, but accessible from different platforms (Internet, PDAs...), act as socio-spatial mediators, in charge of telling where people are, what they are doing, with whom, what they have in mind, who and what are there with us (or around, close to us), what they expect from others, what they think about the place, etc. [13]. These contributions are emerging as an essential way to access the sense of places. Every place tells a story, and user contributions are its more relevant medium.

Due to the relative pre-eminence of place to understand urban environment, and following Mike Goodchild [14] idea, it is time to move from a space-based to a place-based geospatial infrastructure. This is what the following section is about.

4. SENSING PLACES' LIFE

4.1 SDI as an Enabling Platform

Spatial Data Infrastructure (SDI) is an integrated, multi-levelled hierarchy of interconnected SDI based on partnerships at corporate, local, state/provincial, national, regional (multi-national) and global levels. This enables users to save resources, time and effort when trying to acquire new datasets by avoiding duplication of expenses associated with the generation and maintenance of data and their integration with other datasets. SDIs are increasingly focussing on large scale people relevant data (land parcel based data or build environmental data) with the result that today it is suggested most SDI activity worldwide is at this level. A central aspect in understanding these developments is the evolution of mapping, and the growth of land administration systems and national mapping initiatives in different countries.

Data Infrastructures are now in place that enables individuals to position themselves and navigate to a chosen destination by multiple routes, identifying nearby places and services of interest. In this context, SDI and spatial technologies are now used routinely in decision making to support city planning and forecasting and at the same time to address some of the world's most pressing societal problems. Many countries now recognize SDI as an essential modern infrastructure such as information communication technology (ICT), electricity or transportation.

Harlan Onsrud [15] defined SDI as a network-based solution to provide easy, consistent, and effective access to geographic information and services to improve decision making in the real world in which we live and interact.

Spatially enabled society is "dependent on the development of appropriate mechanisms to facilitate the delivery of data and services". To be spatially enabled an organization has to: (1) accommodate in its very operational logic, a more effective and transparent political and electoral process by making relevant geographical information accessible to citizens; (2) foster economic market improvement through the development and diffusion of public geographical information products and services; and (3) allow monitoring environmental sustainability by using spatial indicators provided by distributed sensor networks [16]. Therefore there is a general agreement on the need for a "service-oriented infrastructure on which citizens and organizations can rely" to have access to geographical information and location-based services [17]. Such infrastructure (basically close to the last generation of SDI) is seen as the key basis to any spatially enabled society, since it provides stakeholders with faster and direct information updating and downloading capabilities; and deploys mobile and monitoring applications offering augmented and virtual reality capabilities for instance [18].

4.2 Citizens as sensors

User generated geographic content and geo-crowdourcing are indeed two other major characteristics of spatially enabled society as well as smart city. Spatially enabled citizens increasingly use technology, particularly mobile technology, to voluntarily contribute and provide local information and share place-based knowledge on their networks. Users become both producers and consumers of this information. Citizens, as sensors, are able to provide their (social) network with real-time information about their spatial experiences: recording and sharing personal memories, reporting on inefficiencies and problem areas within the city, or rating the services provided in different locations. In this type of user-contribution-based service, community is shaped through LBS and, in return, these services rely on community, considered as a source of information. This concept of "citizens as sensors" [19] is also an important issue for Spatial Data Infrastructures (SDIs). Spatially enabled citizens could be considered as a dynamic source of information to feed the SDI data flows [20], as well as the monitoring system of smart cities.

If citizens can unconsciously provide useful information to fuel smart city (when their traces, their spatial behaviors, or even their tweets for instance are tracked and analyzed to better understand new dynamics in the city) they also can consciously participate to city life and actuation. Several urban computing projects using human (e.g. drivers or vehicles) as sensors, have already demonstrated to what extent this kind of bottom up approaches, could provide city decision/policy makers useful and relevant information about city life [21, 22]. Similarly, in the context of spatial enablement citizens could take advantage of existing Spatial Data Infrastructures -SDI- while creating and sharing spatial knowledge, as sensors in their own right. To this effect, volunteered Geographical Information -VGI- becomes the most prolific sources of information to characterize places. Based on both such SDI resources and their own local knowledge, citizens as sensors, could not only provides place-based information, but also Volunteered Geographical Services – VGS, and then make them more and more engaged as active smart city actuator [23].

As we saw previously, spatially enabled society and smart cities have a lot in common, and they both benefit from Spatial Data Infrastructures as enabling platforms improving access, sharing and integrating spatial data and services. Yet they are still conceptual and technical challenges to achieve a fully functional system [16]. Smart city as actuating source of spatial enablement might probably provide solutions to overcome these challenges.

4.3 Cameras sensors

Distributed cameras across cities which are used for many applications such as traffic management, or public safety can also be useful for other potential applications. These cameras can be considered as sensors network for further data collection and analysis for live city application of related movement and planning or other related areas. With this in mind, this research aims to integrate imagery from the city's camera sensor network to create an immersive and coherent 3D visualisation of streets and their real-time dynamics and events. The camera sensor network will include live cameras around the city, traffic control sensors and VGI. High-level image processing, including feature extraction from live video streaming, will allow for accurate and detailed information to be readily available in real-time. A dynamic live city has applications as diverse as urban planning, disaster management, asset management and intelligent transportation systems.

In addition, recent advancement in digital imaging and computing technologies allows for efficient image analysis for 3D object reconstruction. Meanwhile, video technologies are pursued for analysis of dynamic scenes, moving object detection and tracking. In this project, we will further explore vision technology to fuse the static virtual city with temporal information such as the movement of pedestrians and cars obtained from videos.

4.4 Aggregation

In order to aggregate place-based information in a dynamic way and to ensure traceability capability, we propose an enabling urban platform based on a WikiGIS framework. The WikiGIS applies wiki management and integration strategies to geo-data–geometry (shape and location)–graphic attributes–descriptive attributes, and not solely to location-based texts, as in geoblogs for instance. More precisely, a WikiGIS is a geographic information system (GIS), created online (on the Internet) through collective interventions (in this case, data coming from camera-sensors together with data coming from citizens), which involves interactions between participants, followed by the combination and traceability of their contributions into coherent geospatial representations that are open to improvements [24].

In this context, a potential application will be an experiment system which delivers live city allowing for visualizing the dynamics and movement in the city, thus, stimulating visual-based thinking, space perception, and imagination. Spatially distributed live events can be observed on the live city, just as on the spot. Such spatially enabled live events in the context of virtual city scenes facilitate enhanced comprehension of objects, events and patterns, in relation with incidents in the nearby streets. Further, to support this idea, a component also would be to develop a holistic decision-support system and a smart Spatial Platform for city management, in particular in the context of disaster response, and recovery. The urban disaster integrated decision-support system will provide a dashboard for the strategic, tactical, and operational decisions arising during disaster response. This system will enhance the cognitive abilities of decision makers, allowing them to explore multiple scenarios in parallel, to visualize the consequences of their decisions under various assumptions, to refine the decisions continuously over time.

Spatial data is often heterogeneous and distributed in networks as it might be created by city's inhabitants, government or private sectors using different techniques, existing IT systems and readings form the physical environment [25, 26]. In addition, spatial data require interpretation to be relevant in emergency situations. In an emergency situation, geospatial data should be brought together from different sources, integrated in homogenous formats and, analysed and converted to momentous information. Currently such a workflow is not automatic and requires a substantial input by an individual, which is often not effective in emergency response scenarios.

The aim of this experiment is to develop and implement a smart spatial platform that provides access to multiple distributed data sources (including data from cameras and live inputs) automatically integrates geospatial data, intelligently analyses data and generates user friendly information to support decision making. The proposed platform consists of four main components as illustrated in following Figure.

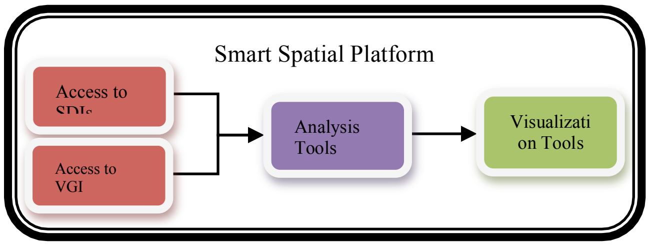


Figure 1. Smart Spatial Platform.

The first component is access to SDI which provides spatial data from different sources. SDI provides access mechanism for the smart platform to connect and download data from distributed sources. The SDI also provides a standard access method such as internationally compliant web service approaches.

The second component is access to VGI. This is the spatial data that are crowed sourced. The third component is the analysis tool that uses optimized models that can replace the human modelling skill and knowledge for decision making such as planning or disaster management. It includes in particular a VGS for validating VGI data, based on comparison algorithm and diff. operators (data triangulation) using SDI data and physical space description (sensed by cameras) as references to elaborate geographical rules [27]. The fourth component is the information visualization tool. It provides analyzed data in a user-friendly and intuitive manner, both in 3D (cameras sensors) and 2D, so the data can be effectively used in different situations.

5. CONCLUSION

This paper proposes a place-based urban platform that aims to facilitate data access, discovery and linkages in a smart city context. This wiki-platform is fed both by camera-sensors information and citizen-sensors trough their location-based network. By sensing urban places' life, this platform contributes to the smart operations of cities.

The work related in this paper is a work in progress. The platform is currently under development and an experimental protocol should be designed and applied to urban crisis management by the end of this summer. This paper provides illustrations of how to proceed to move urban infrastructures and city components towards smarter operational modes. Geospatial technologies, information and methods are powerful means to smartening the world by providing multi-sensing capabilities, building models, improving analyses and capacities to understand and react by feeding actuating technologies or engaged people.

But more important for a smart city is its capability to capture the sense of places. A city is not a machine, but rather made by people local actions and feeling. This could not be captured and represented without active citizens sensors (VGI, crowdsourcing…) connected to location based-social networks.

Capturing the sense of places, in a dynamic and 3D way, is then a major stake for urban communities that look for a smarter way of development. Smart city infrastructure has to be based on an Enabled platform for aggregating formal data sensed by device sensors and sensible knowledge sensed by citizen sensors.

6. ACKNOWLEDGMENTS

Our thanks to the Social Sciences and Humanities Research Council of Canada and Université Laval (research grant) for their funded support. We acknowledge the members of the Centre for SDIs and Land Administration and Centre for Research in Geomatics for their suggestions and inputs, however views expressed in this paper are those of the two authors.

7. REFERENCES

[1] Roche, S., Nabian, N., Kloeckl, K., and Ratti, C., 2012. Are 'Smart Cities' Smart Enough? In Rajabifard, A. and Coleman, D. (eds.) *Spatially Enabling Government, Industry and Citizens: Research development and perspectives*, GSDI Association Press, Chapter 12, 215-236.

[2] Roche, S., 2011. Je me localise, tu te localises… nous nous localisons : la géocommunication dans une société à compétences spatiales. In *recueil de textes du colloque « Spatialités et modernité : Lieux, milieux et territoires »,* (Pau, France, October 13 - 14, 2011), 7 p.

[3] Introduction of PennState Geospatial Revolution Project: http://geospatialrevolution.psu.edu/

[4] Goodchild, M.F., 2009. Neogeography and the nature of geographic expertise. *Journal of Location Based Services* 3,2, 82–96.

[5] http://www.wired.co.uk/news/archive/2012-04/17/potential-of-smarter-cities-beyond-ibm-and-cisco

[6] Scharl, A.; Tochtermann, K., 2007. *The Geospatial Web: How Geobrowsers, Social Software and the Web 2.0 Are Shaping the Network Society*. Springer, London, UK.

[7] Williamson, I., Rajabifard, A. and Holland, P., 2010. Spatially Enabled Society, In *Proceedings of the FIG Congress 2010*, "Facing the *Challenges – Building the Capacity"*, Sydney, Australia.

[8] Goodchild, M.F., 2011. Formalizing place in geographic information systems. In Burton, L.M., Kemp, S.P., Leung, M.-C., Matthews, S.A. and Takeuchi, D.T. (eds.), *Communities, Neighborhoods, and Health: Expanding the Boundaries of Place*, Springer, New York, 21–35.

[9] Lussault, M., 2007. *L'homme spatial: la construction sociale de l'espace humain*, Seuil, Paris.

[10] Mitchell, W. J.,1996. *City of bits*, MIT Press, Cambridge, MA.

[11] Tapscott, D. and William, A. D., 2009. *Wikinomics: How Mass Collaboration Changes Everything*, Penguin Group, New-York.

[12] Castells, M., 2011. City of Bits, Space of Flows: My Dialogue with Bill Mitchell´s Theory of Urbanism in the Digital Age, *SA+P William J. Mitchell Symposium*, (Media Lab, MIT, Cambrige, MA, October, 11, 2011).

[13] Lindqvist,J., Cranshaw, J., Wiese, J., Hong, J., and Zimmerman, J., I'm the Mayor of My House: Examining Why People Use foursquare - a Social-Driven Location Sharing Application, *CHI 2011* (Vancouver, BC, Canada, May 7–12, 2011).

[14] Goodchild, M.F., 2012. Reflections and Visions, final keynote presentation, Global Geospatial conference, (Québec City, Canada, May 14 – 17).

[15] Onsrud, H., 2011. Afterword, In Nedovic-Budic, Z., Crompvoets, J. and Georgiadou,Y. (eds) *Spatial Data Infrastructure in Context: North and South*, Taylor & Francis, CRC Press, London.

[16] Rajabifard, A., 2009. Realizing Spatially Enabled Societies – A Global Perspective in Response to Millennium Development Goals, *18th UNRCC-AP Conference*, (Bangkok, Thailand, October, 26-30, 2009).

[17] Rajabifard, A., Feeney, M.E.F., Williamson, I.P., and Masser, I., 2003. *Development of Spatial Data Infrastructures: from Concept to Reality*, National SDI Initiatives, Chapter 6, Taylor & Francis, U.K.

[18] Uitermark, H., Van Oosterom, P., Zevenbergen, J., and Lemmen, C., 2010. From LADM/STDM to a Spatially Enabled Society: a Vision for 2025, *The World Bank Annual Bank Conference On Land Policy And Administration*, (Washington D.C., USA, April, 26-27, 2010).

[19] Goodchild, M.F., 2007. Citizens as sensors: Web 2.0 and the volunteering of geographic information. *Geofocus,* 7,8–10.

[20] Craglia, M., 2007. Volunteered Geographic Information and Spatial Data Infrastructures: when do parallel lines converge? *Position paper for the VGI Specialist Meeting*, *NCGIA*, (Santa Barbara, December, 13-14, 2007).

[21] Yuan, J., Zheng, Y. and Xie, X., 2012. Discovering regions of different functions in a city using human mobility and POIs, *KDD 2012*.

[22] Yuan, J., Zheng, Y., Zhang, L., Xie, X. and Sun, G., 2011. Where to Find My Next Passenger?, *13th ACM International Conference on Ubiquitous Computing, UbiComp 2011.*

[23] Savelyev, A., Janowicz, K., Thatcher, J., Xu, S., Mülligann, C. and Luo, W., 2011. Volunteered Geographic Services: Developing a Linked Data Driven Location-based Service, *SSO '11*, (Chicago, Illinois, November 1, 2011).

[24] Roche, S., Mericskay, B., Batita, W., Bach M., and Rondeau, M., 2012.WikiGIS Basic Concepts: Web 2.0 for Geospatial Collaboration. *Future Internet*. 4, 1, 265-284. http://www.mdpi.com/1999-5903/4/1/265/pdf

[25] Daqing, Z., Bin, G., and Zhiwen, Y., 2011. The Emergence of Social and Community Intelligence. *IEEE Computer*. 44,7, 21-28.

[26] Robinson, R., 2012. How Cities Can Exploit the Information Revolution, posted June 9 on The Urban Technologist Blog http://theurbantechnologist.com/2012/06/09/how-cities-can-exploit-the-information-revolution/

[27] Goodchild, M.F., and L., Li, 2012. Assuring the quality of volunteered geographic information. *Spatial Statistics*. 1, 110–120.

[28] Jackson, M., Gardner, Z., and Wainwright, T., 2011. *Location-awareness and Ubiquitous Cities: A report to the U-City Research Institute Yonsei University, S. Korea*, (Centre fo Geospatial Science, The University of Nottingham, June, 2011, 70 p).

City-scale Traffic Simulation From Digital Footprints

Gavin McArdle
National Centre for Geocomputation
National University of Ireland Maynooth
Maynooth, Co. Kildare, Ireland
gavin.mcardle@nuim.ie

Eoghan Furey
National Centre for Geocomputation
National University of Ireland Maynooth
Maynooth, Co. Kildare, Ireland
eoghan.furey@nuim.ie

Aonghus Lawlor
National Centre for Geocomputation
National University of Ireland Maynooth
Maynooth, Co. Kildare, Ireland
aonghus.lawlor@nuim.ie

Alexei Pozdnoukhov
National Centre for Geocomputation
National University of Ireland
Maynooth, Co. Kildare, Ireland
alexei.pozdnoukhov@nuim.ie

ABSTRACT

This paper introduces a micro-simulation of urban traffic flows within a large scale scenario implemented for the Greater Dublin region in Ireland. Traditionally, the data available for traffic simulations come from a population census and dedicated road surveys which only partly cover shopping, leisure or recreational trips. To account for the latter, the presented traffic modelling framework exploits the digital footprints of city inhabitants on services such as Twitter and Foursquare. We enriched the model with findings from our previous studies on geographical layout of communities in a country-wide mobile phone network to account for socially related journeys. These datasets were used to calibrate a variant of a radiation model of spatial choice, which we introduced in order to drive individuals' decisions on trip destinations within an assigned daily activity plan. We observed that given the distribution of population, the workplace locations, a comprehensive set of urban facilities and a list of typical activity sequences of city dwellers collected within a national road survey, the developed micro-simulation reproduces not only the journey statistics but also the traffic volumes at main road segments with surprising accuracy.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining, spatial databases and GIS*; I.6.5 [**Simulation and Modelling**]: Model Development—*modelling methodologies*

Keywords

urban mobility, spatial choice, location based social networks, agent based traffic modelling

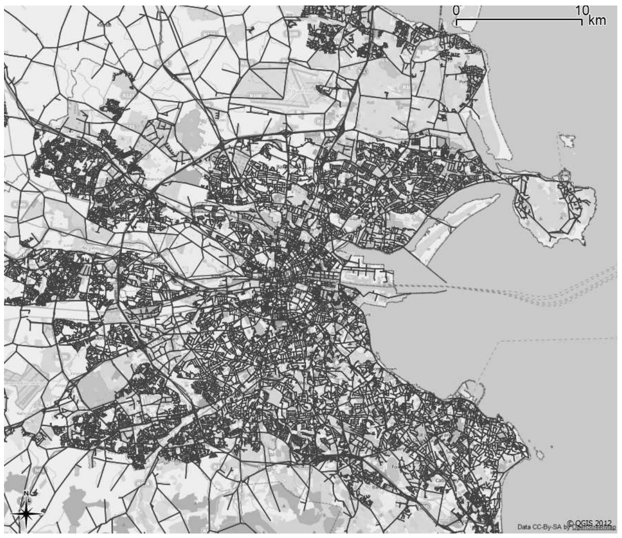


Figure 1: The road network in Greater Dublin region used for modelling.

1. INTRODUCTION

New opportunities in the study of human mobility arise from the availability of digital traces of movement such as the check-in data of location-based social network services or Call Detail records of cell phone usage. They allow for uncovering details about urban mobility previously unavailable from traditional travel surveys, such as an evidence of a long-tail in the daily trip distribution indicating that some individuals cover distances orders of magnitudes larger in their typical trips than a majority of other users [7]. Check-in services such as Foursquare [19] or geo-referenced Twitter [21, 20] provide new insights for transportation modelling by providing quantifiable evidence about the purpose of individuals' travel whether for shopping, leisure or recreation, or meeting friends and visiting family. These activities generate a considerable amount of road traffic which needs to be accounted for in transportation models. However, they are not covered in detail by traditional travel surveys [3] and rely on perceived rather then measured trip lengths and are likely to contain biases.

1.1 Contributions of this work

In this paper we investigate the usefulness of digital footprints of individual movement for calibrating human mobility models within an urban traffic micro-simulation framework. We implemented a large scale realistic working day scenario for the Greater Dublin region in Ireland. Particularly, the presented approach includes the following novel contributions.

- We introduced a spatial choice model of the radiation type for selecting destinations of individual trips (Section 2.1), with interpretable parameters and a simple calibration scheme (Section 2.3).
- The model is applied for facility choice based on a dataset of points of interest and transitions statistics observed via geo-referenced Twitter messages and Foursquare check-ins in Ireland (Section 2.2).
- Geographical layout of a social network observed in country-wide cell phone data is used as a proxy for modelling destination choice of the socially related trips (Section 2.4).
- The developed methodology is applied for destination choice in shopping, leisure and socially related journeys which account for major part of the traffic flows but are not available from traditional surveys.
- These activities are integrated into a realistic traffic scenario calibrated on the daily plans generated in accordance with a census of population, workplace locations, daily activities and departure times (Section 3), and validated on the measured traffic volume counts at major roads in Greater Dublin region (Section 4).

The paper is organised as follows. Section 2 gives an overview of spatial interaction approaches to urban mobility studies with particular focus on spatial choice modelling. We describe the developed adaptation of the radiation model in Section 2.1, which is then applied within a comprehensive framework of activity-based micro-simulation of traffic flows. This framework is built on the MatSim platform [4] and is described in Section 3. It uses a dataset of places of work locations to model commute flows. The necessary technical details on the datasets used in model development are given in Appendix 1. Our experimental results presented in Section 4 show that the proposed spatial choice model produces accurate estimates of the daytime traffic volumes at major roads. We highlight and interpret some characteristic traffic volumes patterns and compare this model to a naive baseline nearest neighbour method where all individuals choose a closest facility for their destination. We discuss the possible origins of the surprisingly accurate predictions in Section 5 which concludes the paper.

2. URBAN MOBILITY

Traditional transportation planning and forecasting frameworks stem from travel surveys on origin-destination flows and apply gravity laws [28], intervening opportunities [24], competing destinations [13] or an overarching constrained entropy maximisation framework [27] to investigate the trip distribution. A more flexible approach using activity-based models, focuses on modelling travel demand based on the activities that people need to perform in the course of a day. This framework is usually implemented in a micro-simulation system where each agent is assigned an activity chain and performs destination choice in the context of this agenda. Meanwhile, the mobility of individuals is still relatively unexplored within these frameworks. Empirical evidence confirms that regular commute is a dominating mobility pattern [23], which also governs occasional fluctuations as people tend to arrange their travel plans by considering accessibility and convenience with regard to their primary locations such as home and work. This rational paradigm and availability of digital footprints opens new ways to enrich activity-based models in transportation modelling and urban planning. Facility choice thus becomes a key element of the model performance.

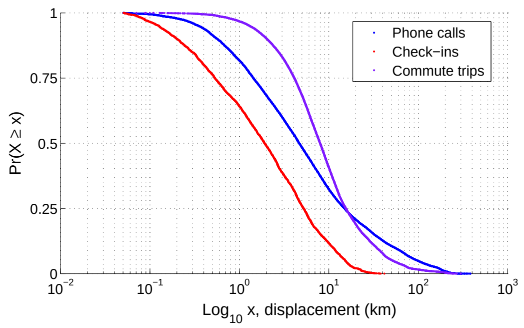


Figure 2: A probability of observing a commute trip, a phone call or a check-in displacement longer than x kilometers.

Attempting to predict the locations where people travel for work, recreation, shopping and to live is a significant challenge with a long research history. Both professionals and academics have carried out considerable work over the last century in dealing with this challenge and many positive findings have emerged. Factors that are taken into consideration regarding the choice of location include: travel distance and time, size of the store or facility, range of products or services and overall quality and price considerations. Models of estimating residential location choice include various logit models [2, 16, 10]. In modelling the choice of leisure facilities some recent developments are based on hollow space time prisms which are derived from leisure trip length statistics [14].

2.1 Radiation model of spatial choice

The radiation model [22] is inspired by the theory of intervening opportunities [24] and applies emission-absorption ideas to compute probabilities of interactions for a set of origins and destinations of known capacities. It is a destination-constrained parameter free model where distance decay is replaced with rank-based decay similarly to intervening opportunities. We applied this idea at an individual level to derive a probability of choosing a particular facility from a set of facilities of the given type with known capacities.

In our model we assign every individual an emission threshold z^i which determines a minimum level above which a particular driving trip will become worthwhile. For example, in case of shopping destination choice process, an individual with a large threshold z who is planning a shopping trip would have high or perhaps very specific demands which would have to be overcome and so is less likely to be absorbed by a nearby facility or shop. We assume there is some preselected distribution which describes this demand, $p(z)$. As there is no information on which kind of shopping trip an individual plans to undertake, we consider that a particular location choice of an individual at location i is based on the probability $P_{m_i}(z)$ that a maximum threshold drawn from $p(z)$ after m_i repetitions is equal to z. Suppose that each possible destination facility at location j has a certain probability to satisfy that demand $P_{n_j}(> z)$, which is given by a maximum threshold extracted from $p(z)$ after n_j repetitions, where n_j is the capacity of a facility at j. We must also account for the probability that none of the intervening facilities could absorb the traveller $P_{s_{ij}}(< z)$ where s_{ij} is the total facility capacity in a circle centred on i of the radius equal to the distance between locations i and j. Then, the probability that a person at location i with a demand threshold m_i makes a trip to a facility at j with capacity n_j and no other closer facility, is given by

$$P(1|m_i, n_j, s_{ij}) = \int_0^\infty dz P_{m_i}(z) P_{s_{ij}}(<z) P_{n_j}(>z), \qquad (1)$$

We perform the integral in a similar fashion to the radiation model [22] and find

$$P(1|m_i, n_j, s_{ij}) = \frac{m_i n_j}{(m_i + s_{ij})(m_i + n_j + s_{ij})}. \qquad (2)$$

In our adaptation we aimed at a model where an unknown distribution $p(z)$ of the demands of individuals deciding to commence a car trip can be integrated out.

2.2 Mobility data

The movement dataset we used to model shopping, leisure and recreation trips is a combination of Twitter data collected in [20] and an Irish subset of the Foursquare dataset described in [8]. It contains a total of 107218 check-in events posted by 5287 unique users. Characteristic trip lengths contained in this dataset as compared to commute distances and call lengths are presented in Figure 2. Other summary statistics plots including temporal descriptors of users activities and their mobility are presented in Figure 5. We use this data to assess the parameters of the characteristic trip length for non-working activities.

2.3 Facility choice and parameter fitting

In contrast to the original radiation model where the inputs are the known populations of the origin and destination, we have a quantity m_i which relates to the choice of facilities in a region. Good quality public datasets on facility capacities are not readily available. We can make some estimate of our parameter m_i for a given region from the user-inputted data on OpenStreetMaps, but the overall coverage of this dataset is somewhat sparse. Instead we have found that we can substitute the m_i for a given location with an average facility choice m_{opt} for the entire region. We have devised a simple method to determine the optimum value for this parameter (Figure 3). The facilities are ranked according to the distance to the trip origin and for each facility we use the radiation model (2) to find the probability that a trip to the facility will be made. The data clearly show a long tail, confirming that longer trips to lower capacity facilities become increasingly unlikely. In trying to find a good value of m we see that if we set it too large, $m > m_{opt}$, this implies the high or specific demand which can not be satisfied by nearby facilities and indeed it can be seen in Figure 3 the probability

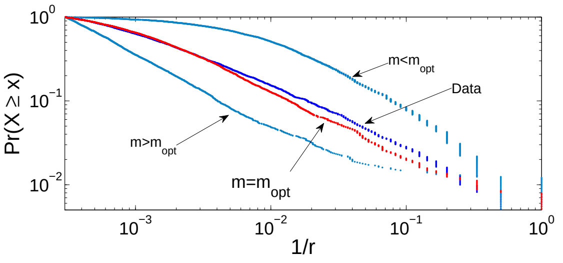


Figure 3: Fitting the m parameter in the inverse rank cumulative probability plot, log-log scale.

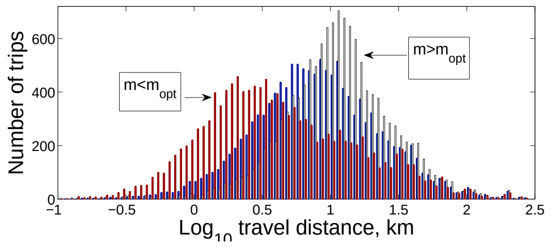


Figure 4: Impact of m parameter on the trip length histogram. X-axis is in log_{10} scale, km.

to undertake a trip to a distant facility is significantly increased. Conversely, if m is too small, an individual is much more likely to make a trip to a nearby facility. The optimal value of the parameter m_{opt} is found by minimising the difference between the rank cumulative probability as found by the radiation model and the observed data. The trip length distributions resulting from a facility choice dictated by a radiation model are shown in Figure 4. Again, it is seen that the $m < m_{opt}$ results in a shorter average path length, and conversely for $m > m_{opt}$. The optimal m_{opt} reproduces the trip length distribution which we find from a database of check-ins. An example of the theoretical analysis of the trip length distribution under generic multiplicative spatial choice models can be found in [25].

2.4 The geography of social networks

Empirical evidence [3, 21, 9] suggests the importance of social influence on the formation of atypical patterns of mobility. People visit family members or friends, and join them in recreation, leisure, tourism or shopping trips. It was observed that a probability of befriending a person is inversely proportional to the number of closer people, i.e. a spatial rank of the person [18]. Social networks also possess distinct community structure which often show geographical patterns both at inter-city [11] as well as intra-city scales [26]. One can use the characteristic distances and geographical layout of these interactions as a proxy for socially related travel such as journeys to visit friends and family.

The geographical layout of the major communities detected in a cell phone communication network in Greater Dublin area is presented in Figure 6 (taken from [26]). The community structure is clearly influenced by the underlying geography. Given that it is much more likely to observe social links between members of a community than across different ones, we have simulated a social network for the population of the agent which reproduces the characteristic link length distribution, node degree and community struc-

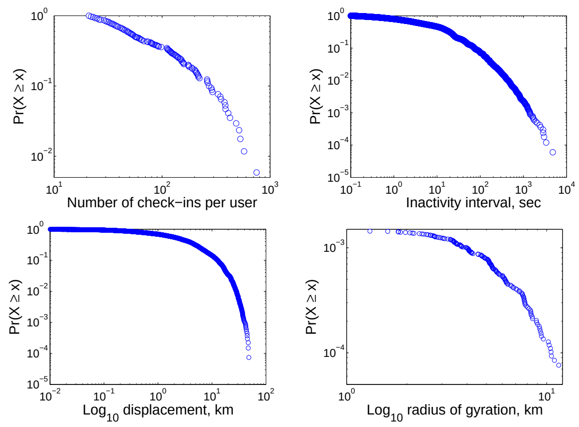


Figure 5: Empirical CDF for the observed check-ins dataset show lack of power law tails due to limited sampling time, relatively low number of messages registered per user, and bounded geographical area of observation.

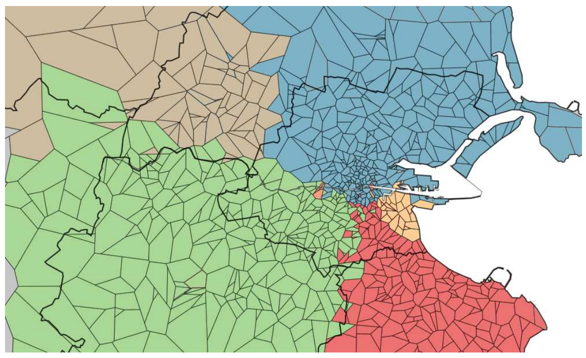


Figure 6: Geographical layout of the major communities detected in a cell phone comunication network in Greater Dublin area [26].

ture which we find in the communication network. We will present further details on this social network generation algorithm elsewhere, and instead show the resulting statistics of the network in Figure 7.

3. TRAFFIC SIMULATION

Agent-based micro simulation is an effective way to model and predict traffic. In this approach, each agent is considered as an individual with an ability to make their own decisions and manage their daily activities to get the greatest return. MATSim [4] and SUMO (Simulation for Urban MObility) [6] are two examples of software frameworks implementing agent-based traffic simulations.

In MATSim each agent is assigned a plan which represents the desires of that agent for the day, for example, one desire is the departure time for work. The plan is altered through different iterations of the simulation in order to maximise an individual agent's utility score. Travelling is seen as having a negative or neutral score whereas spending time at home has a positive score. Each iteration tries to minimise travel time to increase the overall utility score. The iterations should continue until the system has reached a relaxed state, known as a Nash Equilibrium where future iterations will produce little improvement in the utility scores

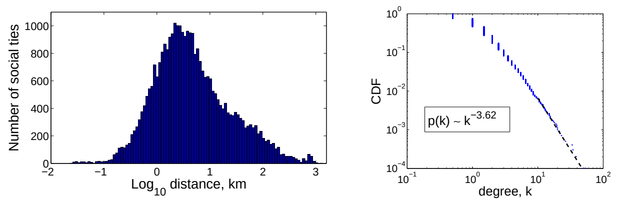


Figure 7: A histogram of a social tie length (left) and a node degree (right) of the simulated social network.

of agents. The variables which MATSim can alter during iterations are the departure time, the route choice and optionally, the location choice for secondary activities, such as leisure and shopping [14]. Route choice is determined using the A-Star algorithm [17], while time choice is achieved using local random mutation [5].

Activity chains, which represent the sequence of activities undertaken by individuals, are a key input for micro simulations. Generally they are derived from data recorded from travel surveys and reveal patterns of activities that people carry out on a normal day. For example, the percentages of people who travel to a shop immediately after work can be calculated and integrated into traffic simulations in order to predict which shopping locations will receive which portions of the road traffic. MATSim has been used by Horni et al. [14] in this way. By combining data from the Swiss census regarding work, education and home locations of citizens, with a activity chains collected by the Swiss National Travel Survey, traffic flows were produced for an average day. Originally MATSim employed an entirely time based utility function to calculate where individuals could travel within the time allocated for shopping trips, however this was discovered to be insufficient. Therefore, the model was extended to consider further variables such as shop size or the density of shops in a given area [14].

3.1 Implementation

We use MATSim to generate a traffic simulation for the Greater Dublin Area and compare two location choice models in the simulating traffic flows. One approach considers a nearest neighbour algorithm while the other uses a variant of the radiation model detailed in Section 2.1. MATSim has specific data requirements, including the road network and agent plans for the study area. Below, the details of the how this data was prepared for the simulation of traffic in the Greater Dublin Area are provided.

3.1.1 Network

MATSim requires a network consisting of nodes and links. The nodes represent road intersections while the links are the road segments joining these intersections. Using tools provided by MATSim, the OpenStreetMap(OSM) road network for Dublin was extracted and transformed into the appropriate structure. All roads in an area of approximately 200kms squared around Dublin City were extracted from OSM, Figure 1. Additionally, all major roads (national routes and motorways) in Ireland were obtained. Additional information provided by OSM including the speed limits, class of road and type of road were also obtained. This permits the simulation to determine the flow capacity of road segments which is used in route choice.

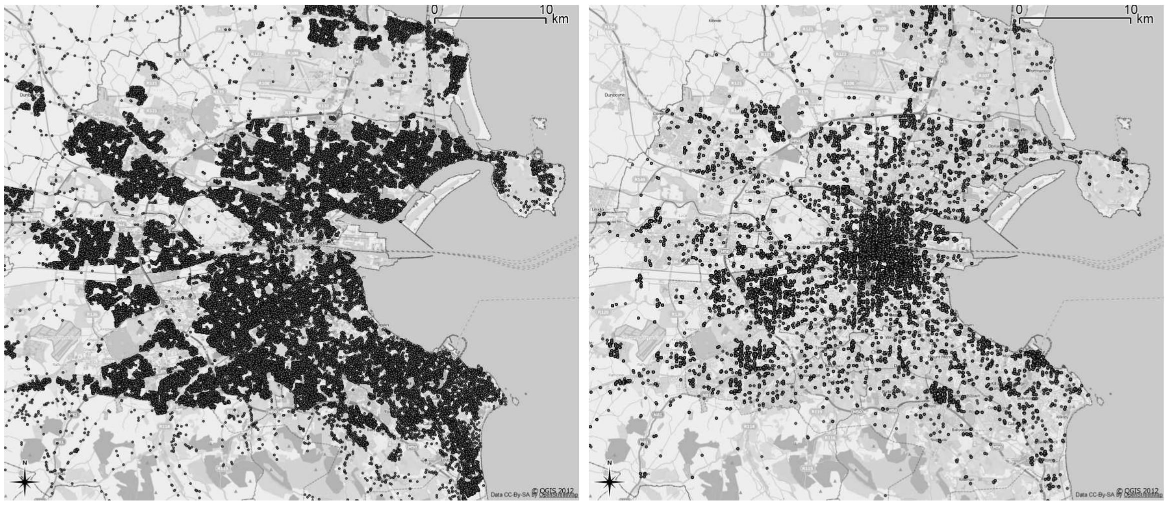


Figure 8: Home (left) and work (right) locations.

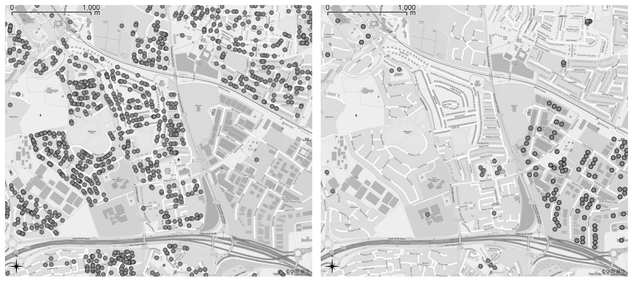


Figure 9: A close up view on home (left) and work (right) locations highlight a typical segregation of residential areas and industrial zones.

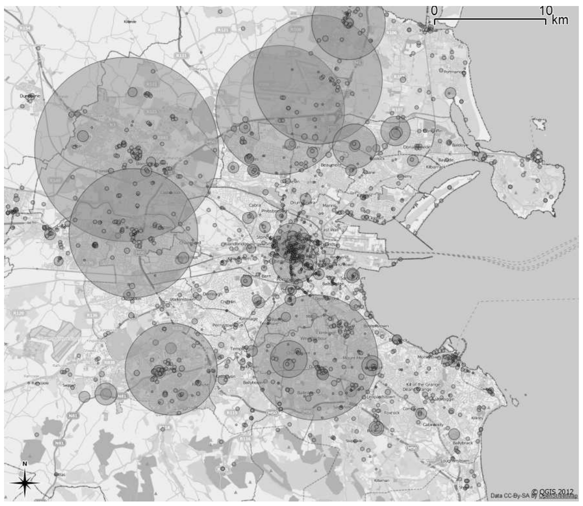


Figure 10: Locations of facilities in the area. The estimated capacities are denoted by circle size (largest correspond to major shopping malls).

3.1.2 Population and Demand

MATSim also requires a population, which it will model. The population represents the home and work locations of individuals within the study area. The locations are shown in Figure 8. Figure 9 highlights the contrast between a residential and industrial area. Additionally each individual needs to have a plan, which consist of the desired activities they will perform during the day. The plans or activity chains which include the sequence and duration of activities represent the demand on the network. Further details on the datasets used within the micro simulation framework are given in Appendix 1. One important aspect of creating the demand is the choice of location where activities will occur.

3.2 Facility choice implementation

The location of various activity types (schools, gyms, pubs, restaurants, shops, etc.) were extracted from public datasets including the OSM and a points of interest database of an in-car GPS navigator. Additionally, the capacity of each of these facilities was estimated. The resulting facilities are shown in Figure 10 where the size of the circle on the map represents the capacities. This formed the input for determining the location choice during the generation of the day plan for each agent. The plans were assigned according to the survey as described in Appendix 1 and contained the following activity types: school/education, shopping, personal business, visiting family/friends, social/entertainment, sport/leisure, and doctor/medical facility. The radiation model was applied for each individual choice over all alternatives amongst the facilities of a given type. That is, the same model was applied both at a strategic choice (a school's location) and a tactical choice level (a pub or a restaurant). For a social visit, a location to visit was assigned by sampling a home or work location of a friend from the simulated social network (Section 2.4). In addition to the radiation model for location choice described in Section 2.1, a nearest neighbour location choice was used as a naïve baseline approach. This model randomly selects a facility within a 4km threshold of the agents' current or future location. A sample of 50,000 individuals was randomly selected from the available data for home and work locations. In total 50 route replanning

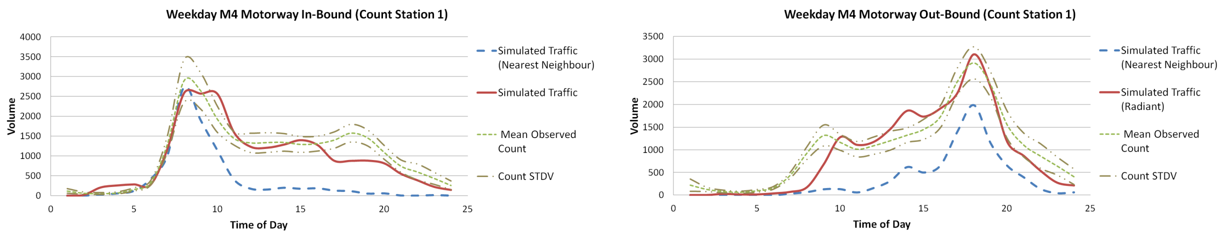


Figure 11: Traffic volume counts and simulation results for location 1.

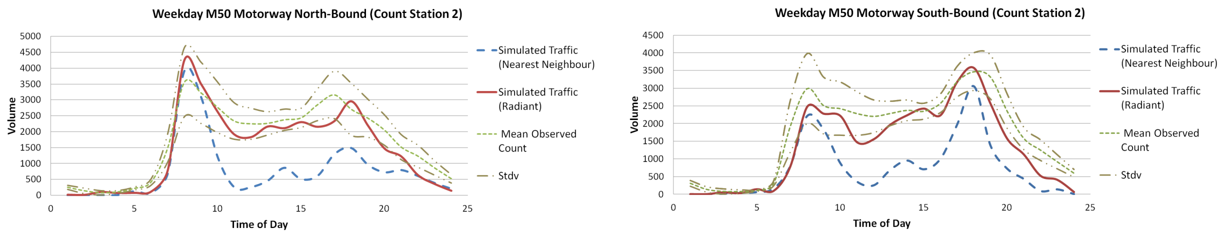


Figure 12: Traffic volume counts and simulation results for location 2.

iterations of the simulation were used.

MATSim outputs several pieces of data which are useful to assess the effectiveness of the simulation. Firstly an animation which shows the movement of the 50,000 agents over the road network can be rendered to assess traffic volumes at different periods of the day. The total distance travelled and trip duration of individuals is also produced. Finally, count data which shows the number of cars passing each road segment (link on the network) for each hour of the day is provided and served as a key validation measure for assessing the model performance.

4. RESULTS

The count data obtained with a simulation is compared to observed count information provided by the National Roads Authority (NRA) in Ireland which counts traffic using hardware embedded in road surface at specific locations. We have aggregated hourly average volumes for a typical working day over a summer period of 2006. Figure 14 shows a summary of the volume of cars using each road segment on the network during a 24 hour period returned by the simulation with radiation location choice. Not surprisingly, the motorway (M50 motorway) which surrounds the city sees the highest volume of traffic and so is assigned the darkest colour. Figure 14 also shows the physical location of all NRA count stations used to validate the simulation and highlights the three (labelled 1 - 3) that are presented in the paper in Figures 11-13. These were chosen due to their diverse geographic locations. The count information for these count stations was extracted from the data returned by MATSim. As 50,000 agents represents approximately 10% of vehicular traffic, the counts were scaled appropriately. The results were plotted alongside the mean NRA observed count data which were calculated by averaging the count data for weekdays (Tuesday to Thursday) from the published count statistics.

Figures 11 to 13 show the count data for each count station. All the graphs emphasise the importance of location choice for secondary activities. The nearest neighbour approach (dashed line) successfully detects the time of day that the morning and evening peak in traffic occurs. This is due to the fact that home and work locations are obtained from census data. For the remainder of the day when secondary activities are occurring, the nearest neighbour model significantly underestimates the volume of traffic as individuals fail to travel for better opportunities and instead select activity locations which are in close proximity to them.

This is in contrast to the radiation model (solid line) which produces accurate count data throughout the day. Significantly, the volumes at the morning and evening peaks occur within 2 standard deviations of the mean observed counts for traffic on the M50 motorway (Figures 12,13). Similarly for the remainder of the day, the volume closely follows the mean observed count data. Figure 11 shows the count data for station 1 which is on the M4 motorway that connects Dublin to cities in the West of Ireland. Here, the peak for the morning out-bound traffic appears later than the observed mean values and the evening peak for in-bound traffic appears absent. This anomaly can be explained by the experiment set-up in which only individuals working in Dublin are considered. The simulation does not capture those that live in the Greater Dublin Area and work outside the city. Therefore this road is underused in the morning for out-bound commuter traffic and likewise in the evening for in-bound traffic.

5. CONCLUSIONS

It is not uncommon to observe the accuracy of models and the forecasts of volumes within 40% interval of the measured flows [12], and the observed fit can be considered as surprisingly good for a generic approach undertaken in this study. A major impact on the quality of the results is due to the amount, high detail and spatial resolution of the home to work data available for the region. Nevertheless, the newly introduced universal radiation spatial choice model was shown to perform superior to the nearest facility choice and was able to reproduce midday traffic volumes at a variety of major roads. The exact geography of social ties makes an essential contribution to its performance. We will study the influence of model components on the traffic system in

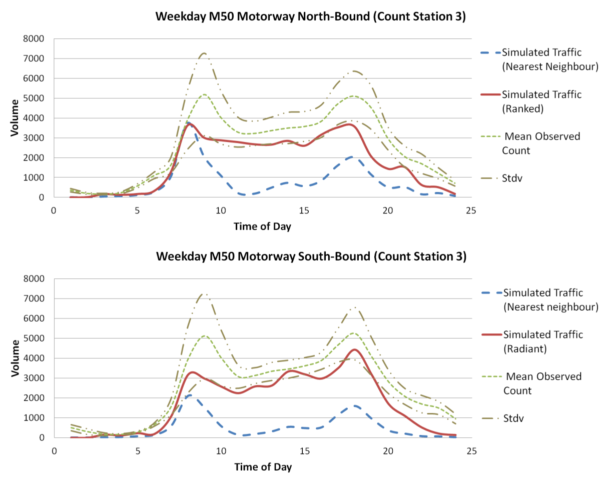


Figure 13: Traffic volume counts and simulation results for location 3.

more detail. Particularly, we are interested in the impact of stratification effect that emerges in the coupled consideration of mobility and social influence on facilities choice [15]. Also, the simulated volumes will be increased to the levels at which the impact of congestion on the route and destination facility choice can not be neglected.

Acknowledgments

Research presented in this paper was funded by a Strategic Research Cluster grant (07/SRC/I1168), Stokes Lectureship award by Science Foundation Ireland under the National Development Plan, and the Complexity-NET programme Complexity in Spatial Dynamics (COSMIC). The authors gratefully acknowledge this support. We would like to thank Fergal Walsh for producing Figure 6 and Caroline Treacy for assistance in spatial data handling.

Appendix 1.

The population data was obtained from the Irish National Census which is conducted every five years. The most recent census was conducted in 2011, however as the data is still being collated, the 2006 Census was used for generating the Dublin Simulation. POWCAR (Place of Work - Census of Anonymised Records) is a subset of the full census which provides the home location and work location of individuals, the mode of transport used to commute and the time at which individuals leave their residence. The home location is anonymised by giving it at an electoral division level while the work location is presented a 250 metre grid level. The time of departure is presented as discrete 30 minute intervals for the morning period and several modes are encoded in the means of transport.

To simulate movement within the Greater Dublin Area, individuals whose place of work is within the Dublin are extracted. Furthermore, only those who use a private car or van to get to work are considered. For finer grained location data, we assign each individual in the POWCAR dataset buildings to represent their home and work locations. GeoDirectory, a commercial database, which contains the location of every building in Ireland was used. The database contains the coordinates of buildings, the electoral division they are in and the class of building (commercial, residential or both). Using this database, each individual is assigned a random residential building in their electoral district and a random commercial building with a 250 metre buffer of the work location declared in the POWCAR dataset. This data was combined with the departure time information. For the discrete values, a random time instant in the 30 minute departure segment was selected.

The demand is represented by activity chains and day plans collected via The Irish National Travel Survey (INTS) [1]. This Survey was carried out in 2009 as part of the Quarterly National Household Survey. Over 7000 participants were randomly selected and issued with a travel diary to record all journeys for a period of twenty four hours on a day that was allocated to them. The information gathered included journey origin and destination type (home, work, school, etc.), time of departure and arrival, mode of transport, purpose of trip, distance travelled and the time of each journey.

Activity chains, with durations were extracted from the INTS and relative frequencies of all travel sequences were calculated. This enabled a probability to be applied to each one so that for each individual in the POWCAR dataset, a day plan was generated. Once the sequence of events is determined, the duration of the activities needs to be defined by randomly selecting from all of the durations associated with the specific activity chain that has been chosen.

6. REFERENCES

[1] *National Travel Survey Report.* Central Statistics Office, Government of Ireland, 2009.

[2] J. Abraham and J. Hunt. Specification and estimation of nested logit model of home, workplaces, and commuter mode choices by multiple-worker households. *Transportation Research Record: Journal of the Transportation Research Board*, 1606(-1):17–24, 1997.

[3] K. W. Axhausen. Social networks, mobility biographies, and travel: survey challenges. *Environment and Planning B: Planning and Design*, 35:981–996, 2008.

[4] M. Balmer, K. Meister, M. Rieser, K. Nagel, and K. Axhausen. *Agent-based simulation of travel demand: Structure and computational performance of MATSim-T.* ETH, Eidgenössische Technische Hochschule Zürich, IVT Institut für Verkehrsplanung und Transportsysteme, 2008.

[5] M. Balmer, B. Raney, and K. Nagel. Adjustment of activity timing and duration in an agent-based traffic flow simulation. *Progress in activity-based analysis*, pages 91–114, 2005.

[6] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. SUMO - simulation of urban mobility: An overview. In *SIMUL 2011, The Third International Conference on Advances in System Simulation*, pages 63–68, Barcelona, Spain, October 2011.

[7] D. Brockmann, L. Hufnagel, and T. Geisel. The scaling laws of human travel. *Nature*, 439(7075):462–465, 2006.

[8] Z. Cheng, J. Caverlee, K. Lee, and D. Z. Sui. Exploring millions of footprints in location sharing services. In *ICWSM*, 2011.

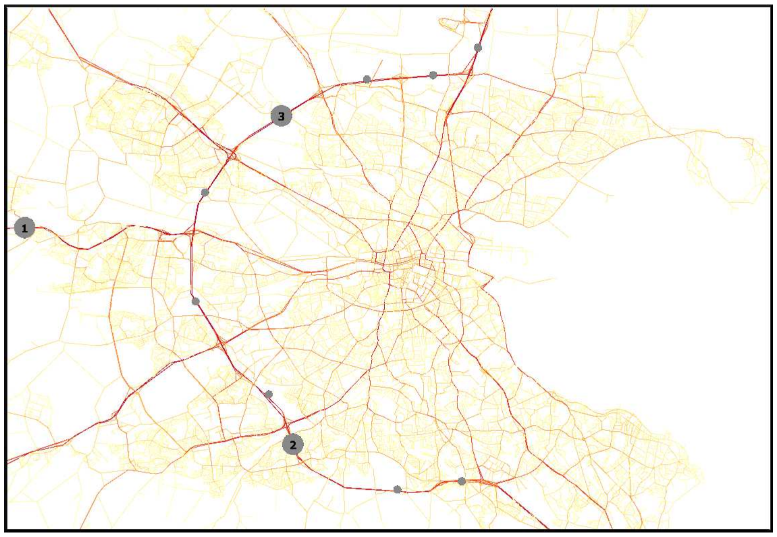


Figure 14: The modelled traffic volumes in Greater Dublin region and locations of counters used for model validation.

[9] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 1082–1090, New York, NY, USA, 2011. ACM.

[10] N. Eluru, C. Bhat, R. Pendyala, and K. Konduri. A joint flexible econometric model system of household residential location and vehicle fleet composition usage choices. *Transportation*, 37(4):603–626, 2010.

[11] P. Expert, T. S. Evans, V. D. Blondel, and R. Lambiotte. Uncovering space-independent communities in spatial networks. *Proceedings of the National Academy of Sciences*, 108(19):7663–7668, May 2011.

[12] B. Flyvbjerg, M. K. Skamris Holm, and S. L. Buhl. Inaccuracy in traffic forecasts. *Transport Reviews*, 26(1):1–24, 2006.

[13] A. S. Fotheringham. A new set of spatial-interaction models: the theory of competing destinations. 1(15):15–Ű36, 1983.

[14] A. Horni, D. Scott, M. Balmer, and K. Axhausen. Location choice modeling for shopping and leisure activities with matsim. *Transportation Research Record: Journal of the Transportation Research Board*, 2135(-1):87–95, 2009.

[15] A. Lawlor, C. Coffey, R. McGrath, and A. Pozdnoukhov. Stratification structure of urban habitats, June 2012. Pervasive Urban Applications workshop (PURBA'12) at PERVASIVE'2012.

[16] B. Lee and P. Waddell. Residential mobility and location choice: a nested logit model with sampling of alternatives. *Transportation*, 37(4):587–601, 2010.

[17] N. Lefebvre and M. Balmer. *Fast shortest path computation in time-dependent traffic networks*. ETH, Eidgenössische Technische Hochschule Zürich, IVT, Institut für Verkehrsplanung und Transportsysteme, 2007.

[18] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11623–11628, 2005.

[19] A. Noulas, S. Scellato, R. Lambiotte, M. Pontil, and C. Mascolo. A tale of many cities: universal patterns in human urban mobility. *arXiv:1108.5355v4 [physics.soc-ph]*, 2011.

[20] A. Pozdnoukhov and C. Kaiser. Space-time dynamics of topics in streaming text. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, LBSN '11, pages 8:1–8:8, New York, NY, USA, 2011. ACM.

[21] A. Sadilek, H. Kautz, and J. P. Bigham. Finding your friends and following them to where you are. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 723–732, New York, NY, USA, Feb. 2012. ACM.

[22] F. Simini, M. C. Gonzalez, A. Maritan, and A.-L. Barabasi. A universal model for mobility and migration patterns. *Nature*, (484):96–100, 2012.

[23] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, Feb. 2010.

[24] S. A. Stouffer. Intervening opportunities: A theory relating mobility and distance. *American Sociological Review*, 5(6):845–867, 1940.

[25] D. Veneziano and M. C. Gonzalez. Trip length distribution under multiplicative spatial models of supply and demand: Theory and sensitivity analysis. *CoRR*, abs/1101.3719, 2011.

[26] F. Walsh and A. Pozdnoukhov. Spatial structure and dynamics of urban communities, June 2011. The First Workshop on Pervasive Urban Applications (PURBA).

[27] A. G. Wilson. *Entropy in Urban and Regional Modelling*. Pion, London, United Kingdom, 1970.

[28] G. K. Zipf. The p1 P2/D hypothesis: On the intercity movement of persons. *American Sociological Review*, 11(6), 1946.

Exploiting Large-Scale Check-in Data to Recommend Time-Sensitive Routes

Hsun-Ping Hsieh[1], Cheng-Te Li[1], Shou-De Lin[1,2]
[1]Graduate Institute of Networking and Multimedia
[2]Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
{d98944006, d98944005, sdlin}@csie.ntu.edu.tw

ABSTRACT
Location-based services allow users to perform geo-spatial check-in actions, which facilitates the mining of the moving activities of human beings. This paper proposes to recommend time-sensitive trip routes, consisting of a sequence of locations with associated time stamps, based on knowledge extracted from large-scale check-in data. Given a query location with the starting time, our goal is to recommend a time-sensitive route. We argue a good route should consider (a) the popularity of places, (b) the visiting order of places, (c) the proper visiting time of each place, and (d) the proper transit time from one place to another. By devising a statistical model, we integrate these four factors into a goodness function which aims to measure the quality of a route. Equipped with the goodness measure, we propose a greedy method to construct the time-sensitive route for the query. Experiments on Gowalla datasets demonstrate the effectiveness of our model on detecting real routes and cloze test of routes, comparing with other baseline methods. We also develop a system *TripRouter* as a real-time demo platform.

Categories and Subject Descriptors
H.2.8 [**Database Management**]: Database Applications – *Data mining*.

General Terms
Algorithms, Management, Performance.

Keywords
Time-sensitive query, trip recommendation, check-in data.

1. INTRODUCTION
Location-based Services (LBS), such as Foursquare [1] and Gowalla[2], allow users to perform check-in actions that pin the geographical information of current location and time stamp onto their personal pages. The rapid accumulation of user check-in records can not only collectively represent the real-world human activities, but also serve as a great resource for location-based recommendation systems. Since the user-moving records implicitly reveal how people travel around an area with rich spatial and temporal information, including longitude, latitude, and check-in timestamp, one reasonable application leveraging such user-generated check-in data is to recommend the travel routes. Indeed, much existing work recommends routes using GPS trajectories [2][14] or geo-tagged photos [1][4][18].

In this paper, instead of purely relying on past moving trajectories to recommend traveling paths, we propose a novel time-sensitive trip route recommendation framework that takes advantage of the user-check-in data. We argue that a proper route recommendation system should consider the following factors when designing a route:

- **The popularity of a place.** Popular landmarks by definition should attract more visitors.
- **The proper time to visit a place.** In general, the pleasure of visiting a place can be significantly diminished if arriving at the wrong time. Some places have a wider range of visiting time while others are constrained to certain particular time slots. For example, most people do not want to visit a beach during boiling hot noon, but rather arrive in the late afternoon to enjoy the sunset scene. Or certain ball game events usually take place at particular time period (e.g. in the evening). As shown in Figure 1, as derived from the Gowalla check-in data described in Section 5, visitors visit some places with higher probability during certain time slots. For example, people usually visit the Empire State Building from about 12:00 to the mid night (note that this place is famous for its excellent night view), (b) people tend to visit the Madison Square Garden in the early evening for a basketball game, (c) the proper time to visit the Central Park is during daytime, and (d) Time Square is preferred from afternoon to midnight.
- **The amount of time required to transit from one place to another.** The transit time between places is highly correlated to visiting the next places at proper time. To find the next place with the proper visiting time, we should consider the amount of time spent on traveling from the current location to the next. For example, one has bought tickets to a football game at a stadium 2 hours away. He will logically choose to start traveling toward the stadium 2 hours ahead of the official kick off time instead of going to a nearby museum 30 minutes away.
- **The visiting order of places.** The visiting order of places is important as it depends on the nature of places and human preference. For example, going to the gym first then going to nearby restaurant for dinner is a better plan than the other way around since it is unhealthy to exercise right after a meal.

While some places are extremely sensitive to the visiting time, the others (e.g. movie theaters) might not possess such strict constraint. An intelligent route recommendation system should

[1] Fouresquare: https://foursquare.com/

[2] Gowalla: http://gowalla.com/

consider such diversity and be able to create a route that has higher chance of satisfying users' needs. This paper argues that by exploring the check-in data, it is possible to design a statistical model to achieve such goal.

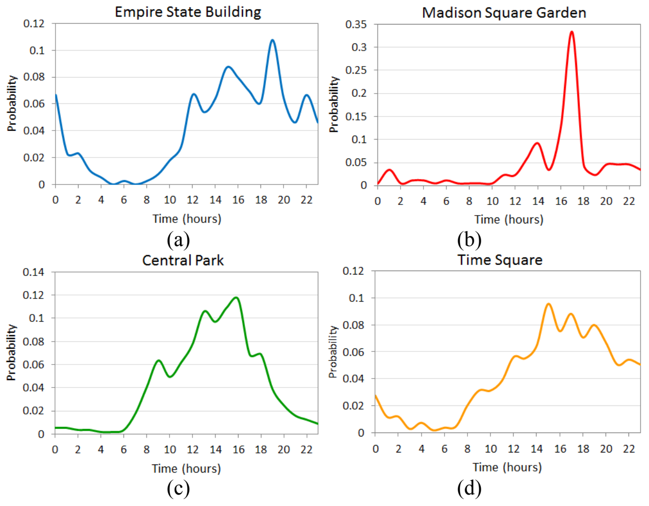


Figure 1: The distribution of the visiting probability at each time unit (hour) for (a) Empire State Building, (b) Madison Square Garden, (c) Central Park, and (d) Time Square. These distributions are derived from the Gowalla check-in data.

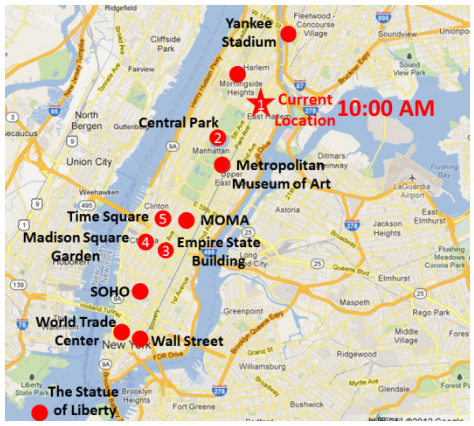


Figure 2: An illustration of recommending a trip route for a query located at the star position with the time stamp 10:00 AM, in Manhattan area, New York City. The goal is to find a trip route connecting some check-in locations with the consideration of the mentioned four requirements.

The online check-in data provide plenty of explicit or implicit information that allows us to fulfill the abovementioned requirements for the sake of planning a proper trip route. First, we can derive from the check-in data the number of people who have visited a certain place, and thus derive the popularity of places. Second, users in LBS tend to perform check-in actions to keep track of their trips in traveling days. As a result, we can obtain and consider the visiting order of places. Third, the check-in records contain the visiting time stamps of locations. Users in LBS are able to collectively reveal the proper visiting time of places. Fourth, followed by the check-in time stamps from existing routes, we are able to have the transit time between places. Equipped with such elements, we utilize the check-in data to recommend the trip routes. Let us use Figure 2 as an example to elaborate the major idea of our time-sensitive trip route recommendation. Assuming a certain user starts to travel from his New York City hotel, marked with a star in Figure 2, at 10:00 AM. There are several popular attractions he/she can visit in a day, including the four famous places mentioned in Figure 1. If the user wants to visit all four places, a possible trip route consists of going to Central Park first, followed by Empire State Building, Madison Square Garden, and finally Time Square.

Formally, the goal of this paper is to construct a time-sensitive route from the check-in data. Given a starting location with a time stamp as the time-sensitive location query, we propose to find a sequence of check-in locations as the trip route, in which each location can be visited at the proper time with the proper transit time from one place to another in the route. The benefit of such time-sensitive trip route is three-fold. First, the user can maximize the capability/price value on visiting each place. Second, with the suggested transit time, users are able to control their schedule more accurately and manage their time effectively. Third, the trip planner can recommend users one or more attractions along the way.

We propose a statistical approach with a greedy search algorithm to construct the time-sensitive routes with respect to the query. The method consists of two phases. In phase one, we measure the quality of a route by devising a goodness function, which integrates the abovementioned four requirements. In phase two, with the query location and time, we greedily find next visiting places by optimizing the goodness function.

We summarize the contributions of this paper in the following.

- We propose a novel time-sensitive trip route recommendation problem using the check-in data in location-based services. We fulfill the idea by developing a *TripRouter* system based on the real-world Gowalla check-in data.
- Conceptually, we argue that a good route should consider four elements: (a) the popularity of a place, (b) the visiting order of places, (c) the proper visiting time of a place, and (d) the proper transit time between places.
- Technically, we devise a goodness function to measure the quality of a route. By exploiting some statistical methods, we model the four requirements of a good route into the design of the goodness function. In addition, for the given time-sensitive location query, we develop a greedy algorithm to search for the route by optimizing the goodness function.

This paper is organized as follows. We describe the related work in Section 2. Section 3 devises the goodness of a route and elaborates the greedy route search algorithm. We evaluate the proposed method in Section 4 and demonstrate the *TripRouter* system in Section 5. Section 6 concludes this work.

2. RELATED WORK

Route Planning by GPS Trajectory Data. There is lots of related work about route planning using the GPS trajectories. J. Juan et al. [14] [15] find the fastest routes to a destination. Z. Chen et al. [2] and L.-Y. Wei et al. [11] search for popular and attractive trajectories for recommendation. Z. Chen et al. [3] find the top-k trajectories connecting some user-given locations. H. Yoon et al. [13] and Y. Zheng et al. [17] propose the itinerary recommendation by considering user preference based on mined trajectory attributes. Y. Zheng et al. [16] [17] aim to discover interesting and classical travel sequences. L.-A. Tang et al. [10] finds the top-k nearest neighboring trajectories with the minimum aggregated distance to some query locations. L.-Y. Wei et al. [12]

construct the top-k routes which sequentially pass through the query locations within the specified time span. Though there are many successful proposals to solve different kinds of route planning problems, the issues of proper visiting time of places and proper transit time between places are never tackled. To achieve such goal, this work proposes to generate the time-sensitive trip routes using check-in data.

We use Table 1 to summarize the differences between our work and other relevant studies. Here we list some important issues about route planning, including: whether it allows the Query of certain Locations (QL), and whether it considers the following ideas: Popularity (PO), Visiting Order (VO), Visiting Time (VT), Transit Time (TT), User Preference (UP), Distance (DI), Travel Duration (TD), and Top-*k* retrieval (TK).

Table 1: Summarization of differences between this paper and other related work.

	QL	PO	VO	VT	TT	UP	DI	TD	TK
[14][15]		■	■			■	■	■	
[2]	■	■	■						
[11]		■	■						■
[3]	■		■						■
[13]	■	■	■		■	■		■	■
[16][17]	■	■	■			■			■
[10]	■						■		■
[12]	■	■						■	■
This work	■	■	■	■	■			■	

Route Recommendation Using Social Media. The rapid rise of social media applications generates huge-volume geo-spatial data of human activities, such as geo-tagged photos in Flickr and check-in records in Foursquare. Both geo-tagged photos and check-in data can reveal how people sequentially visit places in an area. Using geo-tagged photos, Y. Arase et al. [1] mine frequent route patterns for recommendation. A.-J. Cheng et al. [4] propose personalized travel recommendation based on personal profiles and visual attributes of geo-tagged photos. X. Lu et al. [7] and T. Kurashima et al. [6] construct routes based on user preference of must-go destinations, visiting time, and travel duration. Z. Yin et al. [18] mine and rank trajectory patterns from geo-tagged photos and diversify the ranking results. L.-Y. Wei et al. [12] infer the top-*k* routes traveling a given location sequence within a specified travel time from uncertain check-in data. Different from these work, we aim to perform knowledge discovery to construct the time-sensitive routes.

3. METHODOLOGY

3.1 Basic Definitions

Definition 1: Location. A location l_i is a tuple, $l_i = (x_i, y_i)$, where x_i is the longitude, y_i is the latitude.

Definition 2: Route of Check-in Locations. A route is a sequence of locations with the corresponding time stamps, denoted by s, s=<(l_1,t_1), (l_2,t_2), ..., (l_n,t_n)>, where n is the number of locations. Throughout this paper, we focus on recommending single-day route, which implies t_n-t_1 is no more than 24 hours.

Definition 3: Time-sensitive Query. We define the *Time-sensitive Query* as $Q = (l_q, t_q)$, where l_q is the initial location of a user, and t_q is the starting time for this trip.

Definition 4: Time-sensitive Route. Given a time-sensitive query, we define the output *Time-sensitive Route* as a sequence of check-in locations s_r = <(l_1,t_1), (l_2,t_2), ..., (l_k,t_k)>, where $l_1 = l_q$, $t_1 = t_q$, and k is the number of locations in the route, which can be either specified by users or determined using existing time constraint of the trip.

In the following we will describe how to measure the quality of a time-sensitive trip route. Based on the proposed goodness definition, we are able to search and recommend better time-sensitive routes given an initial time-sensitive query.

3.2 Measuring the Quality of a Trip Route

In order to construct a high-quality route for recommendation, we need to first design a proper metric to measure the quality of any given route. We propose that a good trip route should consider the following four factors: (a) the popularity of a place, (b) the proper visiting time of a location, (c) the proper transit time traveling from one location to another, and (d) the visiting order of places in the route. We attempt to model these factors into the goodness function, and utilize such function to greedily selecting locations for the construction of the final trip route.

3.2.1 Route Popularity

A popular place, by definition, should be somewhere that attracts more visitors in general. If a route contains more popular places, it has higher potential to satisfy a user. The popularity of a place can be represented by the number of check-in actions performed at that place. In our goodness measure of a route, we first consider the popularity of places in the route. We define the relative popularity of a location l_i as:

$$pop(l_i) = \frac{N(l_i)}{N_{max}}$$

where $N(l_i)$ is the number of check-in of the location l_i, and N_{max} is the maximum number of check-in among all the locations in the check-in data. Given a route s=<(l_1,t_1), (l_2,t_2), ..., (l_n,t_n)>, we define the popularity-based goodness function $f_{pop}(s)$ as:

$$f_{pop}(s) = \left(\prod_{i=1}^{n} pop(l_i)\right)^{\frac{1}{n}}$$

3.2.2 Proper Visiting Time

The check-in data reveals that while some locations (e.g. park and movie theater) are popular regardless of the visiting time in a given day, other locations (e.g. stadium and beach) are more attractive during certain time period of the day. We propose to learn such time-dependent popularity of each location from the check-in data. We begin from defining the *Temporal Visiting Distribution* as the following.

Definition 5: Temporal Visiting Distribution (TVD) of a Location. We define a *Temporal Visiting Distribution* for a location l, $TVD_l\,(t_i)$, as the probability distribution of a randomly picked check-in record of l occurs at time t_i. For example, in a 24-hour span, *TVD* can be a legal probability distribution shown in Figure 3. *TVD* can easily be learned from check-in data, representing how popular a place is at a given time.

Using *TVD*, we can determine whether it is proper to visit a place at a given time. For example, assuming we want to know how well a decision is to visit a place at 8:00AM, given the location's *TVD* is represented as the green dotted line in Figure 3. To do that, we propose to first generate a thin Gaussian distribution $G(t; \mu, \sigma^2)$ whose mean value μ is 8 with a very small variance σ^2 (e.g. standard deviation is 1). And then we can transform the original task into measuring the difference between the Gaussian distribution with the learnt *TVD* of such location. Here we use the *symmetric Kullback-Leibler* (*KL*) *Divergence* between $G(t;\mu,\sigma^2)$ and $TVD_l(t)$ to represent the fitness of the assignment. The formal mathematical definition of a fitness score between a place *l* and a time *t* can be defined as

$$D_{KL}(G(t;\mu,\sigma^2)||TVD_l(t))$$

$$=\sum_x G(x;\mu,\sigma^2)\,log\frac{G(x;\mu,\sigma^2)}{TVD_l(x)}+\sum_x TVD_l(x)\,log\frac{TVD_l(x)}{G(x;\mu,\sigma^2)}$$

Conceivably, a smaller KL value indicates better match between the assignment and the distribution learned from data.

Consequently, we formally define the temporal visiting goodness function $f_{visit}(s)$ of a route s=<(l_1,t_1), (l_2,t_2), ..., (l_n,t_n)>, as a combination of the popularity of places together with the fitness of each location over time, in the following equation.

$$f_{visit}(s)=\left(\prod_{i=1}^{n}D_{KL}\big(G(t;t_i,\sigma^2)||TVD_{l_i}(t)\big)\times\frac{1}{pop(l_i)}\right)^{\frac{-1}{n}}$$

If the places in a route *s* are visited during the proper time period, the $f_{visit}(s)$ value would become higher.

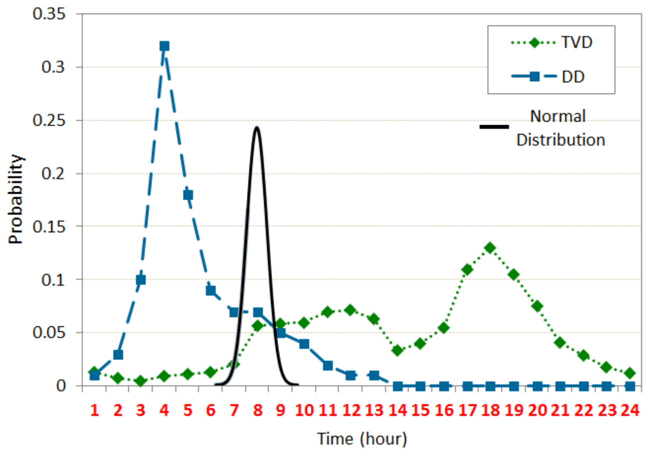


Figure 3: Examples of the temporal visiting distribution (TVD) (the green dotted curve) for a certain location *l_i*, and the duration distribution (DD) (the blue dashed curve) between location *l_i* and *l_j*. The black solid curve represents a normal distribution of a particular time assignment to measure the fitness values.

3.2.3 *Proper Transit Time Duration*

To schedule a good trip route, another key element to be considered is the visiting time of each place as well as the transit time from one place to another. Although the check-in data cannot explicitly tell us the above two kinds of information, we can simply treat the duration between two checked-in places as the summation of the visiting time of the first place plus the transportation time from the first to the second place. Such duration can further be utilized to evaluate the quality of a trip. Here we propose the *Duration Distribution*, as defined in the following, to model such 'visiting plus transit time' between places.

Definition 6: Duration Distribution (DD) between Two Locations. We define the *Duration Distribution (DD)* between locations l_i and l_j as the probability distribution over time duration *t*, $DD_{l_il_j}(t)$, which can be obtained from the following random experiment: randomly pick two consecutive check-in records (l_i, t_i), (l_j,t_j) of a person, and calculate the probability that t_j-t_i=*t*.

Again, we consider only one-day trip, and therefore treat the outcome space of *DD* between hours 0 through 24. For example, any legal probability distribution between hours 0 through 24 can be a *DD* (e.g. the blue dashed line in Figure 3).

Similar to what we do to *TVD*, given a pair of locations l_i and l_j together with an assignment of a given duration Δ among them, we can model Δ as a thin Gaussian distribution and compare it with $DD_{l_il_j}(\Delta)$ using symmetric KL divergence. Consequently, for a route *s*=<(l_1,t_1), (l_2,t_2), ..., (l_n,t_n)>, it is possible to know how good the route is based on the durations between places by defining a goodness function of duration:

$$f_{duration}(s)=\left(\prod_{i=1}^{n-1}D_{KL}\big(g(t;\Delta_{i,i+1};\sigma^2)||LTD(l_i,l_j)\big)\right)^{\frac{-1}{n-1}}$$

A route *s* with higher value of $f_{duration}(s)$ indicates such route can be visited with proper "transit+staying" time between places.

Here we use Figure 4 as an illustration to summarize our idea of utilizing *TVD* and *DD* to measure the goodness of a trip route. Given a route *s*=<(l_1,t_1), (l_2,t_2), ..., (l_n,t_n)>. We use symmetric KL divergence to measure the visiting fitness of each location l_i by calculating a $D_{KL}(l_i)$ value between TVD_{li} and a narrow Gaussian distribution. We also use KL divergence to measure the fitness of each transition $l_i \rightarrow l_j$ and derive a $D_{KL}(\Delta_{ij})$ between $DD_{l_il_j}$ and a thin Gaussian distribution. Eventually we compute the geometric mean of such D_{KL} values to be the time-related route goodness.

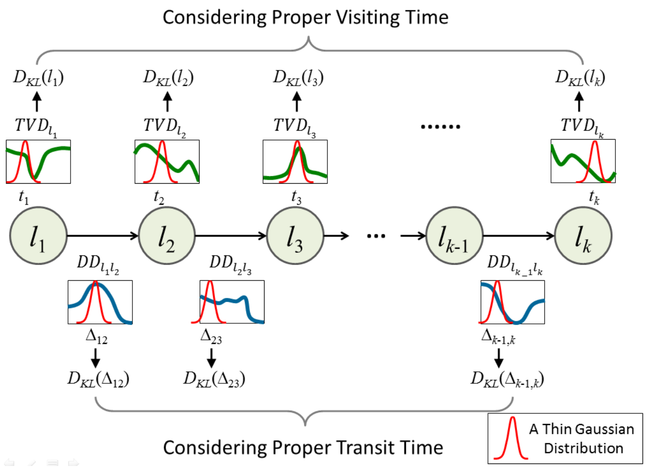


Figure 4: For a route s=<(l_1,t_1), (l_2,t_2), ..., (l_k,t_k)>, we compute 2*k*-1 values of KL-divergence and then take the geometric mean of such values as the time-dependent goodness of a route.

3.2.4 *Proper Visiting Order*

Due to the characteristic of each place, there might be certain latent patterns about the order of the places to be visited. With the check-in data, we are able to learn such orders and exploit them to evaluate the quality of a route. For example, going to restaurant for dinner and then going back to hotel is better than the other way around. In this section, we propose to exploit the idea of the *n*-gram language model to measure the quality of the order of visits in a trip route. Using the check-in corpus, we can first generate the n-gram probabilities of locations. Then, given a route *s*=<(l_1,t_1), (l_2,t_2), ..., (l_n,t_n)>, we can compute its *n*-gram probability. We consider such *n*-gram probability as the goodness of visiting order. Technically, we use the average value of the probabilities of uni-gram, bi-gram, and tri-gram to estimate the goodness of orders. Note that the uni-gram probability is corresponding to the popularity-based route goodness. We can formally write the probabilities as follows.

$$P_{uni}(s) = f_{pop}(s)$$

$$P_{bi}(s) = (P(l_1)P(l_2|l_1)P(l_3|l_2)\cdots P(l_n|l_{n-1}))^{\frac{1}{n}}$$

$$P_{tri}(s) = (P(l_1)P(l_2|l_1)P(l_3|l_1l_2)\cdots P(l_n|l_{n-2}l_{n-1}))^{\frac{1}{n}}$$

Therefore, the goodness of visiting order of a route can be defined:

$$f_{order}(s) = \frac{(P_{uni}(s) + P_{bi}(s) + P_{tri}(s))}{3}$$

Higher $f_{order}(s)$ value represents better quality of route. Note that we utilize the add-one technique for smoothing.

3.2.5 Final Goodness Function

Here we integrate the goodness measures of the proper visiting time, the proper transit time duration, and the proper visiting order into the final goodness function $f(s)$. The final goodness function contains two parts. The first part is the average value of the temporal visiting goodness $f_{visit}(s)$ and the location transition goodness $f_{duration}(s)$. The second part is the visiting order goodness $f_{order}(s)$. We use a parameter α ∈ [0,1] to devise a linear combination of such two parts. The final goodness function $f(s)$ is defined in the following.

$$f(s) = \alpha \times \left(\frac{f_{visit}(s) + f_{duration}(s)}{2}\right) + (1-\alpha) \times f_{order}(s)$$

A route s with higher value of $f(s)$ will be considered as a better route. Experiments in Section 5.3 suggest $\alpha \approx 0.9$ being more effective on measuring route quality. Such result exhibits the usefulness of the proposed time-sensitive route recommendation.

Algorithm 1. *TimeRoute* algorithm

Input: (a) $RouteDB$: routes extracted from the check-in data;
(b) $Q = (l_q, t_q)$: the time-sensitive location query;
(c) k: the number of locations in the final route.
Output: a time-sensitive route $s_r = \langle(l_1,t_1),(l_2,t_2),\ldots(l_k,t_k)\rangle$.

```
1:  $s_r = \langle(l_1 = l_q, t_1 = t_q)\rangle$.
2:  for $i = 2$ to $k$ do:
3:    $C_i = \{l_c | l_{i-1} \rightarrow l_c$ in $RouteDB\}$.
4:    $f_{max} = 0$.
3:    for each $l_c \in C_i$ do:
4:      $s_{tmp} = s_r + \langle(l_c, t_c)\rangle$.
5:      Compute the goodness $f(s_{tmp})$.
6:      if $f(s_{tmp}) > f_{max}$ do:
7:        $s_r = s_{tmp}$.
8:        $f_{max} = f(s_{tmp})$.
9:  Return: $s_r$.
```

3.3 Greedy Algorithm *TimeRoute*

In this section, we formally describe the problem of time-sensitive trip route recommendation based on the proposed goodness measure. And then we propose a greedy algorithm, *TimeRoute*, to construct the time-sensitive routes for a given query.

Problem Definition. Given (a) the routes extracted in the check-in data, (b) the time-sensitive location query $Q = (l_q, t_q)$, and (c) the number k of locations in the final route, the goal is to construct a route s_r=<(l_1=l_q,t_1=t_q), (l_2,t_2), ..., (l_k,t_k)> to optimize $f(s_r)$.

To solve this problem, we devise a greedy algorithm, *TimeRoute*, to achieve the local-optimal solution. The basic idea is to select next place based on the goodness function $f(s)$. Starting from the query location (line 1 in Algorithm 1), when selecting next location l_i (i >2), we first identify a set of candidate locations C_i by collecting locations which have been ever followed by l_i (line 3). Then for each location in the candidate set C_i, we select the candidate l_c with the maximum goodness value given the existing route, and append it to the final route s_r. (line 4-8). Such procedure will terminate when k spots are identified in the route.

4. EXPERIMENTS

4.1 Dataset and Data Analysis

We utilize the Gowalla dataset [5] which has been exploited for location-based analysis in several places, such as [8] and [9]. The Gowalla dataset contains 6,442,890 check-in records from Feb. 2009 to Oct. 2010. The total number of check-in locations is 1,280,969. Considering a route as a sequence of check-in locations of a user within a day, we construct the route database *RouteDB* containing 2,605,867 routes, among them 1,469,130 has only length one and are not used. The average route length is 4.09, without considering length-1 routes. Figure 5 shows the distribution of the route length, which is highly-skew and heavily-tailed. Figure 5 also shows that people usually do not prefer visiting too many locations in a day, but with some exceptions. Figure 6 shows the distribution of the time duration between two places. It indicates that people consider places closer to where they are when they are planning the next destination.

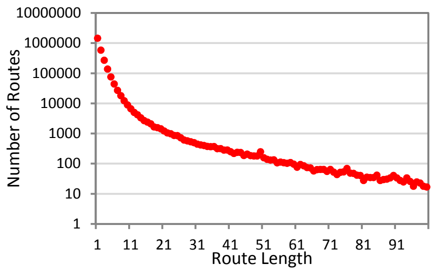


Figure 5: Distribution of route length in *RouteDB*.

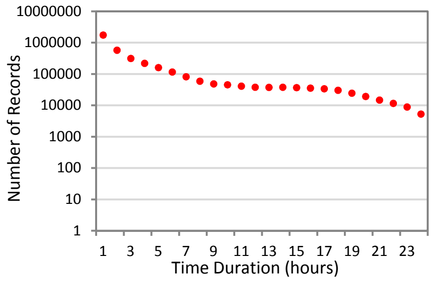


Figure 6: Distributions of time duration in *RouteDB*.

Table 2: The statistics of *RouteDB* and the three subsets.

	Total Number of Check-ins	Avg. Route Length	Variance of Route Length	Distinct Check-in Locations
RouteDB	6,442,890	4.09	48.04	1,280,969
New York	103,174	4.46	71.24	14,941
San Francisco	187,568	4.09	58.36	15,406
Paris	14,224	4.45	75.73	3,472

From *RouteDB*, we extract three subsets of the check-in data, which corresponds to cities of New York, San Francisco, and Paris. Some statistics are reported in Table 2. We can find the average route lengths and their variance in New York and Paris are significantly longer than average. Figure 7 shows the distribution of route length in the three subsets of check-in data while Figure 8 shows the distribution of the time duration.

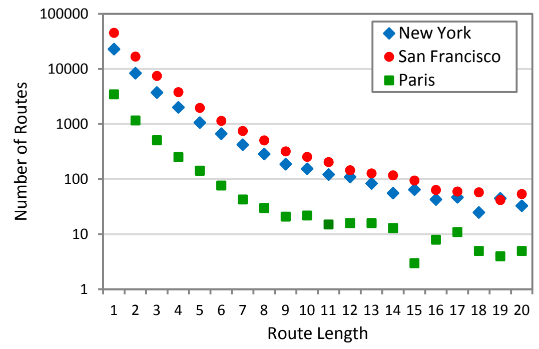


Figure 7: Distribution of route length for three cities (hour).

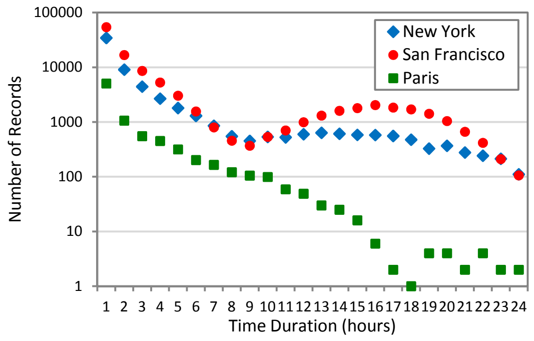


Figure 8: Distribution of time duration for three cities (hour).

4.2 Evaluation Plan

In this section, we introduce two evaluation plans for verifying the performance of our proposed method using the data in these three cites, and compare the results with several baseline methods.

Experiment 1: **Pair-wise Time-sensitive Route Detection.** In this experiment, we would like to verify whether our goodness model can rank the existing routes higher than the non-existing ones. We first randomly choose one thousand real routes from the check-in data. Note that the time stamp is associated with each location *l*. For each route, we replace a portion of the locations with other locations in the city to generate a pseudo route. To make the task non-trivial, we adopt a replacing strategy to replace a location with a 'plausible' one instead of a randomly selected one. That is, to replace a location at position *i* of a route, we only choose from candidate locations that once appear right after the location at position *i*-1 (e.g. the bigram probability of them is non-zero) instead of simply picking a random location. Furthermore, after the replacement, we want to make sure the generated pseudo routes do not exist in the database. That is, there is no such route in the database of the same location sequences together with the same associated time stamps. As can be seen in Figure 11 to 13, the amount of replaced locations varies from 10% to 50% of the total number in a route. We then use our fitness model to examine each pair of the existing route and its pseudo route, and record how frequently our method ranks the correct one higher. Finally, we report the accuracy of our method and compare it with the baseline results. The accuracy is calculated as the number of successfully detected routes divided by the number of pair instances.

Similarly, we can generate another kind of pseudo route by perturbing the time stamps of certain locations in an existing route. For example, given an existing route s=<(l_1,t_1), (l_2,t_2),… (l_{i-1}, t_{i-1}), (l_i,t_i), (l_{i+1},t_{i+1}),..., (l_n,t_n)>, we change t_i to a different time t_j, where $t_{i-1} < t_j < t_{i+1}$. We expect a proper fitness function to assign lower score to such pseudo routes.

Experiment 2: Time-sensitive Cloze Test of Locations in Routes. Given some real trip routes with time stamp in each location, by removing some middle locations, the goal is to test whether a method can successfully identify the removed location. Higher hit rate indicates better quality of recommendation.

Baseline Approaches. To evaluate the effectiveness of our method, we design the following four baseline methods for both experiment 1 and experiment 2.

- **Distance-based Approach.** This method chooses the closest location to the current spot as the next spot to move to. It rates a route using the goodness function $f_d(s) = (\prod_{i=1}^{n} \frac{1}{D(l_i,l_{i-1})})^{\frac{1}{n}}$, where $D(l_i, l_{i-1})$ is the geographical distance between two consecutive locations.
- **Popular-based Approach.** This method chooses the most popular spot of a given time in that city as the next spot to move to. It rates the path using the goodness function $f_{pop}(s)$ as have been defined previously in Section 3.2.1.
- **Forward Heuristic Approach.** The forward heuristic chooses a location l_i that possesses the largest bi-gram probability with the previous location $P(l_i|l_{i-1})$ as the next location to move to. Its goodness function is $f_{forw}(s) = P_{bi}(s)$, as defined previously in section 3.2.4.
- **Backward Heuristic Approach.** The backward heuristic chooses a location l_i that possesses the largest bi-gram probability with the next location $P(l_i|l_{i+1})$ as the next location to move to. The fitness function can be described as $f_{backw}(s) = (P(l_1|l_2)P(l_2|l_3)\cdots P(l_{n-1}|l_n))^{\frac{1}{n}}$.

4.3 Experimental Results

Section 4.3.1 shows the results of Experiment 1 and Section 4.3.2 illustrates the outcome of Experiment 2. For both experiments, we implement four baseline methods to compare with our proposed *TimeRoute* method.

4.3.1 Pairwise Time-Sensitive Route Detection

In experiment 1, we first vary the number of replaced locations from 10% to 50% and report the accuracy of different methods. Figure 9 contains the results for New York City. Our fitness model can achieve around 97% accuracy in distinguishing the real routes from replaced ones. The accuracy scores of the forward and backward heuristics vary from 89% to 93%. The popular-based and distance-based methods do not do a good job here. Similar trend happens in Paris (Figure 11), but for San Francisco (Figure 10), our method shows much higher accuracy comparing with others. The results are not surprising because our method does consider the location preference over time and location order.

Figure 12 shows the results of creating pseudo paths by shifting time stamp for some locations. Again we vary the ratio of change from 10% to 50%. The results show that our model can almost perfectly detect such change, better than the popularity-based method (around 15% accuracy). The other competitors do not have the capability to distinguish such pairs because they do not consider time information during route generation, and therefore the fitness scores are identical for such pair of routes.

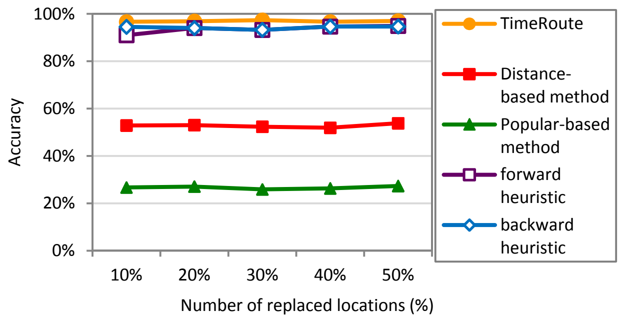


Figure 9: Accuracy by varying the number of replaced locations in New York.

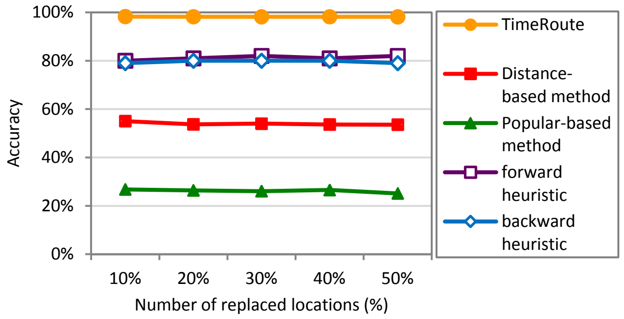


Figure 10: Accuracy by varying the number of replaced locations in San Francisco.

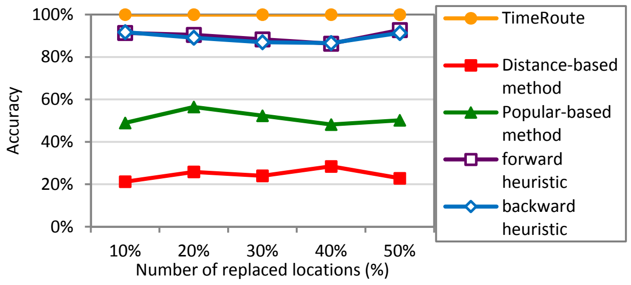


Figure 11: Accuracy by varying the number of replaced locations in Paris.

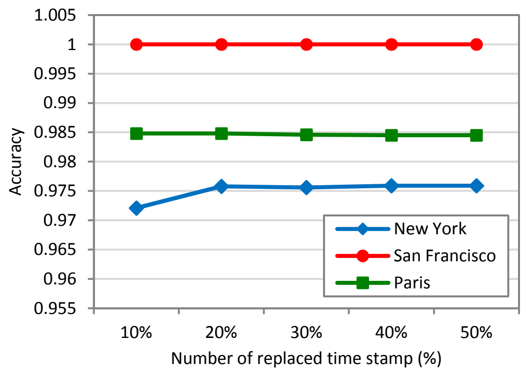


Figure 12: Accuracy by varying the number of replaced time stamp for our method in the three cities.

4.3.2 Time-Sensitive Cloze Test in Routes

In cloze experiment of locations in routes, we calculate hit rate by varying the position of missing location. Generally speaking, the preceding position of missing location obtains lower hit rate than latter one because the system can generally do better when more information is revealed.

As reported in Figure 13-15, the hit rates of the four baseline models are often lower than 10% in these three cities, while we can achieve 15%~40% hit rate.

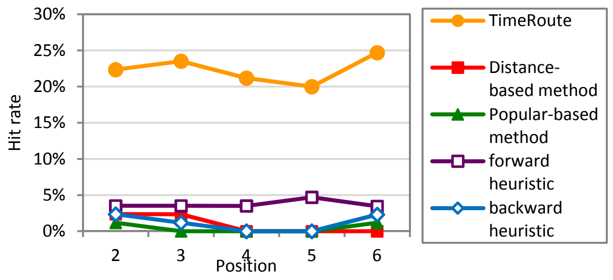


Figure 13: Accuracy by varying the position of missing location in New York.

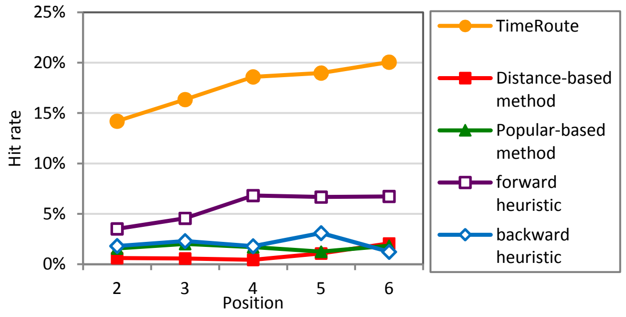


Figure 14: Accuracy by varying the position of missing location in San Francisco.

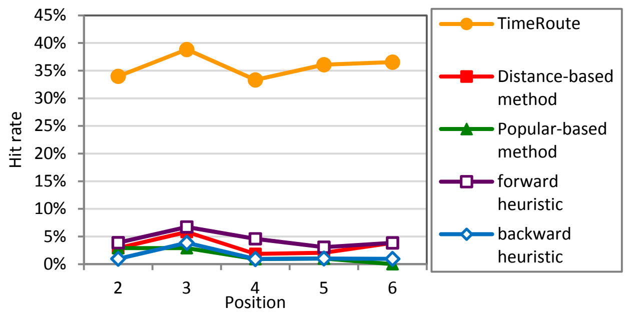


Figure 15: Accuracy by varying the position of missing location in Paris.

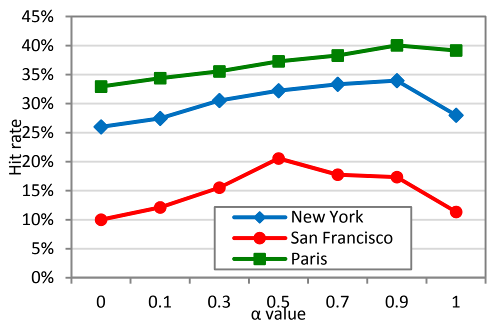


Figure 16: The impact of α on the time-sensitive cloze test for the three cities.

Impact of α: Next, we examine how sensitive our model is to the parameter α, ranging from 0 to 1. We use the hit rate of cloze test and the results are shown in Figure 16. In New York and Paris, the best α value is around 0.9. That is, much more weight is assigned to time-sensitive models than the visiting order on cloze test task. In San Francisco City, α performs well while varying from 0.5 to 0.9.

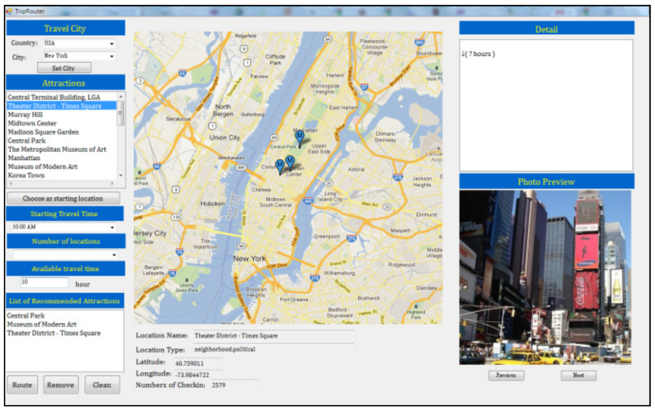
5. SYSTEM DEMONSTRATION

Using our model, we develop an online time-sensitive trip route recommendation system, called *TripRouter*. The system snapshot is shown in Figure 17. Users first determine the city they want to travel, and then select one location as their starting location, together with the starting time. *TripRouter* also allows users to specify their estimated travel time duration and the desired number of locations of such trip. We list the three major functions of *TripRouter* as below: (a) time-sensitive route recommendation, (b) displaying diverse information of locations and routes such as location attributes, route statistics, and some geo-tagged photos obtained from Flickr, and (c) recommending the transportation mode by querying Google Map API according to mined transit time duration.

Below we show three recommended routes querying from Central Park at different starting time, where the route length *k* is set as 4.

- **Central Park at 9PM:** Central Park (9AM) → New York City Center (11AM) → 5th Ave (5PM) → FAO Schwarz restaurant (7PM).
- **Central Park at 2PM:** Central Park (2PM) → The Museum of Modern Art (3PM) → Bergdorf Goodman (4PM) → Lee's art shop (7PM).
- **Central Park at 5PM:** Central Park (5PM) → 5th Ave (6PM) → Pulitzer Fountain (7PM) → Four season hotel (8PM).

The above examples tell us that our *TripRouter* system is able to recommend the best route based on the specified time and location.



Country: USA
City: New York

Central Terminal Building, LGA
Theater District - Times Square
Murray Hill
Midtown Center
Madison Square Garden
Central Park
The Metropolitan Museum of Art
Manhattan
Museum of Modern Art
Korea Town

Starting Travel Time: 10:00 AM

Available travel time: 10 hour

List of Recommended Attractions:
Central Park
Museum of Modern Art
Theater District - Times Square

Location Name: Theater District - Times Square
Location Type: neighborhood,political
Latitude: 40.759011
Longitude: -73.9844722
Numbers of Checkin: 2579

Detail: ↓(? hours)

Figure 17: The system interface of *TripRouter*.

6. CONCLUSION

This paper tries to address an important research question: how much the check-in data can provide in terms of designing a suitable trip route. The solution provided by us seems to be very encouraging as it shows that one can indeed squeeze a lot of knowledge from check-in data to design a time-sensitive trip route that has higher potential of satisfying the users. Note that our approach is mostly data-driven, which assures diverse results can be learned from different cities in which visiting patterns may vary with different culture and characteristics of the city. Ongoing work focuses on two directions: using maximum likelihood estimator to accurately model the visiting time duration of a place and transportation time between places, and further exploit the collaborative filtering approaches to take advantage of the user and location similarities.

7. REFERENCES

[1] Y. Arase, X, Xie, T. Hara, and S. Nishio. Mining People's Trips from Large Scale Geo-tagged Photos. In *ACM MM* 2010.

[2] Z. Chen, H. T. Shen, and X. Zhou. Discovering Popular Routes from Trajectories. In *IEEE ICDE* 2011.

[3] Z. Chen, H. T. Shen, X. Zhou, Y. Zheng, and X. Xie. Searching Trajectories by Locations: An Efficiency Study. In *ACM SIGMOD* 2010.

[4] A.-J. Cheng, Y.-Y. Chen, Y.-T. Huang, W. H. Hsu, and H.-Y. M. Liao. Personalized Travel Recommendation by Mining People Attributes from Community-Contributed Photos. In *ACM MM* 2011.

[5] E. Cho, S. A. Myers, and J. Leskovec. Friendship and Mobility: User Movement in Location-based Social Networks. In *ACM KDD* 2011.

[6] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura. Travel Route Recommendation Using Geotags in Photo Sharing Sites. In *ACM CIKM* 2010.

[7] X. Lu, C. Wang, J.-M. Yang, Y. Pang, and L. Zang. Photo2trip: Generating Travel Routes from Geo-tagged Photos for Trip Planning. In *ACM MM* 2010.

[8] S. Scellato, A. Noulas, R. Lambiotte, and C. Mascolo. Socio-spatial Properties of Online Location-based Social Networks. In *ICWSM* 2010.

[9] S. Scellato, A. Noulas, C. Mascolo. Exploiting Place Features in Link Prediction on Location-based Social Networks. In *ACM KDD* 2011.

[10] L.-A. Tang, Y. Zheng, X. Xie, J. Yuan, X. Yu, and Jiawei Han. Retrieving k-Nearest Neighboring Trajectories by a Set of Point Locations. In *SSTD* 2011.

[11] L.-Y. Wei, W.-C. Peng, B.-C. Chen, and W.-C. Peng. PATS: A Framework of Pattern-Aware Trajectory Search. In *MDM* 2010.

[12] L.-Y. Wei, Y. Zheng, and W.-C. Peng, Constructing Popular Routes from Uncertain Trajectories. In *ACM KDD* 2012.

[13] H. Yoon, Y. Zheng, X. Xie., and W. Woo. Social Itinerary Recommendation from User-generated Digital Trails, In *Personal and Ubiquitous Computing*, 2011

[14] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with Knowledge from the Physical World. In *ACM KDD* 2011.

[15] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-Drive: Driving Directions Based on Taxi Trajectories. In *ACM SIGSPATIAL GIS* 2010.

[16] Y. Zheng., L. Zhang, X. Xie, and W.-Y. Ma. Mining Interesting Locations and Travel Sequences from GPS Trajectories. In *WWW* 2009.

[17] Y. Zheng and X. Xie. Learning Travel Recommendations from User-generated GPS Traces. In *ACM TIST* 2011.

[18] Z. Yin, L. Gao, J. Han, J. Luo, and T. Huang. Diversified Trajectory Pattern Ranking in Geo-tagged Social Media. In *SDM* 2011.

Urban Point-of-Interest Recommendation by Mining User Check-in Behaviors

Josh Jia-Ching Ying, Eric Hsueh-Chan Lu, Wen-Ning Kuo and Vincent S. Tseng*
Institute of Computer Science and Information Engineering
National Cheng Kung University
No.1, University Road, Tainan City 701, Taiwan (R.O.C.)
{jashying, ericlu416, p76994165}@gmail.com, *Correspondence: tsengsm@mail.ncku.edu.tw

ABSTRACT

In recent years, researches on recommendation of urban Points-Of-Interest (POI), such as restaurants, based on social information have attracted a lot of attention. Although a number of social-based recommendation techniques have been proposed in the literature, most of their concepts are only based on the individual or friends' check-in behaviors. It leads to that the recommended POIs list is usually constrained within the users' or friends' living area. Furthermore, since context-aware and environmental information changes quickly, especially in urban areas, how to extract appropriate features from such kind of heterogeneous data to facilitate the recommendation is also a critical and challenging issue. In this paper, we propose a novel approach named *Urban POI-Mine (UPOI-Mine)* that integrates location-based social networks (LBSNs) for recommending users urban POIs based on the user preferences and location properties simultaneously. The core idea of UPOI-Mine is to build a regression-tree-based predictor in the normalized check-in space, so as to support the prediction of interestingness of POI related to each user's preference. Based on the LBSN data, we extract the features of places in terms of i) *Social Factor*, ii) *Individual Preference*, and iii) *POI Popularity* for model building. To our best knowledge, this is the first work on urban POI recommendation that considers social factor, individual preference and POI popularity in LBSN data, simultaneously. Through comprehensive experimental evaluations on a real dataset from Gowalla, the proposed UPOI-Mine is shown to deliver excellent performance.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; J.4 [**Computer Applications**]: Social and Behavior Sciences

General Terms

Measurement, Experimentation.

Keywords

Point-Of-Interest Recommendation, Urban Computing, Data Mining, Location-Based Social Network, User Preference Mining.

1. INTRODUCTION

The markets of Location-Based Services (LBSs) [7] in urban areas, including navigational services, traffic management and location-based recommendation, have grown rapidly in recent years. Due to the needs of effectively improving smart urban living, it is beneficial for these LBSs to be able to recommend users Points-Of-Interest (POIs) where they may be interested in. Thus, effective and efficient urban POI recommendation techniques for LBSs targeting urban mobile users are desirable. The intuitive idea for POI recommendation is based on the personal check-in behaviors of a user. Although such strategy may reflect the personal preference of users, the recommended results are always the POIs that the user has been to. Hence, nowadays, new breed of item recommendation methods, called *social-based prediction*, have emerged. Such recommendation methods usually use the *social properties* of users, mined from the collections of users' social network, to recommend the probably interesting items (POIs) for a user. Figure 1 shows an example of social network information, which typically consists of social links of end users and some user-generated data (i.e., textual information, check-ins, etc.). Among the social-based recommendation studies, Collaborative Filtering (CF) techniques [12] have been widely used for influence of interestingness of item for mobile user. However, they tend to recommend popular items which most of similar users (or friends) are interested in but might not match individual user's preference actually. It may lead to the sparse data problem [15]. Additionally, these social-based recommendation methods only provide the POIs that the user has visited. This may also lead to the lost of applicability for recommendations.

Although the issues of social-based recommendation systems based on the users' check-ins have been discussed in the literatures, existing studies mostly consider only on the social properties of among users [12]. Notice that social links and check-ins typically reflect users' living-spheres, e.g., the environment

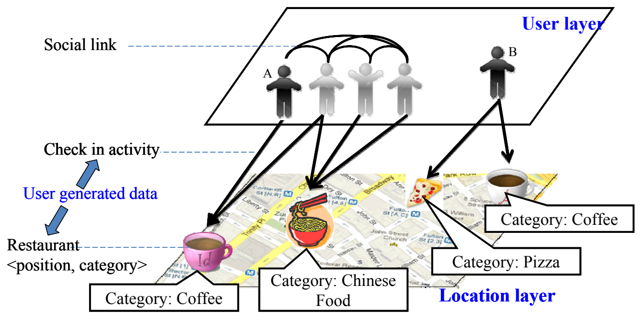


Figure 1. Location Based Social Network.

around users. As a result, CF-based recommendations are constrained by the users' living-spheres if only social properties are considered. Take Figure 1 as an example, both user A and user B favor to drink coffee. Suppose that user A and user B are not friends, and user A's friends always check in some Chinese food restaurants. Thus, some social-based Collaborative Filtering recommendations would recommend the Chinese food restaurant for user A. Additionally, such social-based Collaborative Filtering recommendation only consider the POIs that the user or her/his friends have been visited. It does not work well when previously unvisited POIs are considered. We argue that it is insufficient if merely using social links and check-in history to recommend POIs.

The notion of urban computing has been proposed by Zheng *et al.* [18] [19]. To achieve city-wide computing for serving people and their cities, Zheng *et al.* [18] [19] believe that every device, person, vehicle, building, and street in the urban area can be a sensor to understand city dynamics. Many POIs in Location-Based Social Network (LBSN) have been labeled with useful tags[16][17] such as coffee shop or pizza restaurant, which are crucial for assisting users in searching and exploring new places as well as for sensing POIs' properties. Although we also could map users' current position to some POI databases or road networks for understanding the tag of POI that the users stay in, this process is not work well in an urban area. The reason is that the POIs in an urban area are very crowded so that we cannot accurately tag the location just by catching users' positions. Figure 2 shows that there are two totally different types of restaurants in the same building. Fortunately, we could view each user of LBSN as a sensor to detect the semantic tags of POI for improving POI recommendation in an urban area. In Figure 1, restaurants are tagged with several semantic tags such as Coffee, Chinese Food, Pizza, etc. We observe that both user A and B have visited coffee shops, implying that their preferences in restaurant are very similar. Suppose user B is a loves to have pizza as shown in Figure 1. Thus, it is more reasonable recommending the pizza parlor than recommending the Chinese food restaurant for user A. According to above reasons, we exploit users' preferences from visited POIs to recommend users the POIs of LBSN in an urban area.

To address the above-mentioned problem, we propose a novel approach named *Urban POI-Mine (UPOI-Mine)* for recommending users the POIs in urban LBSN based on not only social factors but also users' preferences. As shown in (1), given a set of users U and a set of POIs P, the problem of POI recommendation can be formulated as predicting relevance score of a given POI for each user, relevance between user's preference and interestingness of POI,.

$$f(u,p) \rightarrow v, \text{where } u \in U, r \in P, \text{and } v \in [0,1] \qquad (1)$$

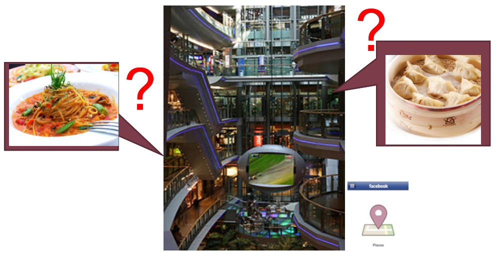


Figure 2. A scenario of POI in an urban area.

Hence, POI recommendation in LBSN can be addressed as a numerical value prediction problem. While numerical value prediction techniques have been developed for many applications, such as demographic prediction [4], bio life cycle analysis [10] and prediction of geographical natural [3], the problem has not been explored previously under the context of urban computing. Furthermore, the context-aware and environmental information changes quickly especially in urban areas. How to extract appropriate features to support the recommendation from such heterogeneous data is also a critical and challenge issue. To support POI recommendation based on users' preferences and social properties, we address this problem by learning a regression-tree based predictor to realize the prediction of relevance score. A fundamental issue is to identify and extract a number of descriptive features for each place in the system. Selecting the right features is important because those features have a direct impact on the effectiveness of the prediction task. As mentioned earlier, only considering the POIs that the user or her/his friends have visited do not work well. Therefore, we explore the users' preference in visited POIs and seek unique features of places captured in the users' preference.

By dealing with the observations prompted in the above examples, we extract features of user-POI pair in three different but complementary aspects: 1) Social Factor (SF), 2) Individual Preference (IP), and 3) POI Popularity (PP). Features extracted from Social Factor, corresponding to a given POI for a user, can be derived from all check-ins among the user's similar friends at the POI based on statistical analysis. In this paper, we extract check-in-based features (e.g., number of check-ins of friends who have a lot of common check-ins with the user) and spatio-based features (e.g., number of check-ins of friends whose living area is very close to the user) as relevance descriptions of specific POIs.

To involve the factor of user preference, we extract features from Individual Preference to capture the relatedness between users and POIs by exploiting the regularity of user check-in activities to POIs with the same tag. Since there are two kinds of tags are annotated on a POI, we could make good use of Individual Preference by deriving descriptive features of a given place from its "related" places. To facilitate extraction of features from Individual Preference, we build a preference table for each user that captures the relatedness between semantic tags and users by exploring regularities of user check-ins to the POI with the same semantic tag. We propose a family of preference formulations that capture the user-highlight and POI-category from the user check-in activities.

As mentioned earlier, such check-in data is very sparse. It leads to that the features extracted from both Social Factor and Individual Preference do not work well for a new user. We employ the popularity of POI to make a maximum likelihood estimation of the relative between user and POI. We argue that different types of POIs will be visited in different frequencies. For example, people may go to coffee shops every day but rarely visit French restaurants. Therefore, we normalize the check-ins of POI based on its semantic tag for representing its popularity. This popularity is thus treated as a feature of POI Popularity, along with features derived from Social Factor and Individual Preference, to feed the regression-tree model in the proposed UPOI-Mine approach.

This research work has made a number of significant contributions, as summarized below:

- We formulated the problem of POI recommendation in an urban area as the problem of relevance score prediction. This problem has not been explored previously in the research community.

- We proposed *Urban POI-Mine (UPOI-Mine)*, a new approach for urban POI recommendation by mining urban users' check-in behaviors. and propose UPOI-Mine to learn a regression-tree for estimating relevance score of each user-POI pair. In the proposed *UPOI-Mine*, we explore simultaneously the factors namely i) Social Factor, ii) Individual Preference, and iii) POI Popularity by exploiting the LBSN data to extract descriptive features.
- We used a real dataset, which was crawled from Gowalla (http://www.Gowalla.com/), to evaluate the performance of our proposed UPOI-Mine in a series of experiments. The results show UPOI-Mine delivers superior performance over other recommendation techniques in terms of the popular measures NDCG and MAE.

The rest of this paper is organized as follows. We briefly review the related work in Section 2 and provide our urban POI recommendation approach UPOI-Mine in Section 3. Finally, we present the evaluation result of our empirical performance study in Section 4 and discuss our conclusions and future work in Section 5.

2. RELATED WORK

In this chapter, we review and classify relevant previous studies into three categories: 1) Similarity Measurement, 2) Recommendation Systems, and 3) Mobility Prediction.

Similarity Measurement. For solving the problem of data sparse, many researches using Collaborative Filtering (CF)-based approach or item-based approach to estimate missing values. The fundamental problem of CF-based approach is how to evaluate user similarity and location similarity. In [12], Ellen Spertus *et al.* proposed six different measures for recommending online social networks. The six measures applied cosine distance, mutual information measure, TF-IDF and log-odds functions to measure the similarity of community.

Recommendation System. In recent years, rapid development of the Internet brings much information and various businesses. A variety of web sites provide huge data of music, images and commodity. How to recommend appropriate items to users is a critical problem. There are also a number of researches for recommendation systems. Traditional recommendation systems usually use CF-based method as the main concept. In [6], Tzvetan Horozov *et al.* proposed an enhanced CF solution for personalized POI recommendation. Some trust-based approaches such as [9], Paolo Massa and Paolo Avesani proposed a trust-aware recommender system. The system builds a trust metric and makes use of trust information to recommend items.

In [5], Mohsen Jamali and Martin Ester proposed TrustWalker. TrustWalker combines Random walk model and trust-based CF approach to predict the ratings of items for users. Trust-based CF approach uses trust values and user-to-item ratings to predict the item ratings. A trust value is the user-to-user rating. If you think someone's interest is similar with you, you can give this people a high trust value. TrustWalker can solve the "cold start problem" with the same precision. The content-based recommendation systems like [11], Chihiro Ono *et al.* used Bayesian network modeling user preference for recommending movies. Bayesian Networks is highly flexibility so it is appropriate for representing complex relations between users' preference and contexts. In [2], Souvik Debnath *et al.* proposed a way to hybrid CF and content-based recommendation system. The approach can determine the weight values of attributes by the linear regression, which obtained from a social network.

Mobility Prediction. A user's location is completely related to his social relations and personal information. LBSN has become a popular application. Many users join a LBSN and share their life, photo, music and location history with their friends. They also get interesting information from their friends. More and more researches on location recommendation with social networks have been proposed with the development of LBSN.

In [1], Betim Berjani and Thorsten Strufe proposed Regularized Matrix Factorization (RMF) recommender to recommend appropriate spots, which mean locations, for users. This research used LBSN data that is crawled from a LBSN website, Gowalla. RMF recommender is a personalized recommender for places. RMF recommender first maps users and spots to a joint latent factor space. RMF recommender exploits regularized Singular Value Decomposition (SVD) model to predict the ratings of users to spots.

In [13], Kenneth Wai-Ting Leung *et al.* proposed a Collaborative Location Recommendation (CLR) framework based on co-clustering. This approach considers location, user and activity to build Community Location Model (CLM) graph. Then CLR approach uses Community-based Agglomerative-Divisive Clustering (CADC) algorithms to iteratively merge and divide nodes in CLM graph. After clustering users, locations and activities by CADC algorithm, it gets refined clusters of similar locations which are visited by similar users and have similar activities. For clustering CLM graphs, it models the similarity as follows:

- Two users are similar if they have similar activity patterns and have visited similar locations.
- Two locations are similar if they are visited by similar users and take place in similar activities.
- Two activities are similar if they are done by similar users and have similar location sequence.

However, the refined clusters are still too large for recommendation. For this reason, this approach classified users into three types: Pattern Users, Normal Users and Travelers by their entropies of visited locations. Then, this approach can recommend locations according different type users. For example, for a traveler, the recommender first finds similar travelers to this user. Then, the recommender finds the locations, which are visited by these similar travelers and have similar activities. Finally, the recommender recommends these locations to the user.

In [14], Ye *at el.* proposed a CF-based POI recommendation framework. This framework fuses user preference influence, social influence and geographic influence to infer the check-in probability for a given user to visit a POI. This approach also exploits a power-law distribution to build the geographical influence among POIs and uses CF method to depict user preference influence and social influence.

3. URBAN POI-MINE (UPOI-Mine)

In the proposed UPOI-Mine approach, we design a two-phase algorithm, as shown in Figure 3, to address the problem of user check-in behaviors mining for urban POI recommendation. The first phase deals with the feature extraction (see lines 1 to 5 in Figure 3), while the second phase takes care of the restaurant

```
Input:  Social Links Set $L$
        Users' check-ins $C$
        POIs $P$
Output: relevance score of each pair of user and POI $S$
 1   Phase 1. Feature Extraction
 2   Feature Set $F \leftarrow \varnothing$
 3   $F \leftarrow F \cup SF(L, C, P)$
 4   $F \leftarrow F \cup IP(C, P)$
 5   $F \leftarrow F \cup PP(P)$
 6
 7   Phase 2. Feature Extraction
 8   Training Set $T \leftarrow F \cup relevanceScore(C, P)$
 9   Regression Tree $R \leftarrow$ M5Prime_Modeling($T$)
10   Relevance Score $S \leftarrow R(P)$
11   Return $S$
```

Figure 3. UPOI-Mine algorithm.

recommendation (see lines 7 to 11 in Figure 3). The task of feature extraction explores three aspects that are discussed in Introduction. For a pair of specific user and POI, we explore the Social Factor (SF) as population features which abstract the aggregated check-ins of the user's friends. On the other hand, we explore the Individual Preference (IP) between users and POIs in order to formulate descriptive features of a given user-POI pair. Moreover, to overcome the data sparse problem, POI Popularity (PP) is considered as a feature in our recommendation model. The features derived from Social Factor, Individual Preference and POI Popularity are used to learn a regression-tree model for predicting relevance score of each user-POI pair in the POI recommendation phase. Given a user-POI pair, the prediction by the regression-tree model estimates the relevance between the interestingness of POI and the user's preference. After checking all POIs, we obtain all qualified POIs for the user under examination.

3.1 Features from Social Factor

Our goal is to extract discriminative Social Factor features from check-ins among the user's similar friends at the POI. Intuitively, aggregating friends' relative check-ins of POI could be used for influencing the probability of that a user likes the POI. Formally, given a friend *f* and a set of POI *P*, the *f*'s relative check-ins of a POI *p* is formulated as (2).

$$relative\ check\text{-}ins(f, p) = \frac{checkins(f, p)}{\sum_{p' \in P} checkins(f, p')} \qquad (2)$$

Accordingly, given a user-POI pair (*u*, *p*), the features extracted form Social Factor could be generally formulated as (3).

$$SF(u, p) = \sum_{f \in F(u)} (relative\ check\text{-}ins(f, p) \times similarity(f, u)) \qquad (3)$$

where *F(u)* is the set of user *u*'s friends.

As mentioned above, we can significantly observe that measuring similarity of two users is the key of Social Factor features. Intuitively, living sphere of users and their friends could be utilized for measure the similarity between users and their friends

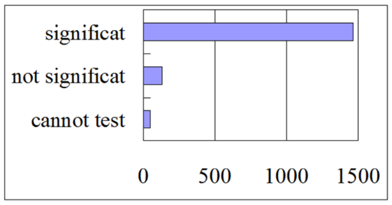


Figure 4. result of χ^2 test.

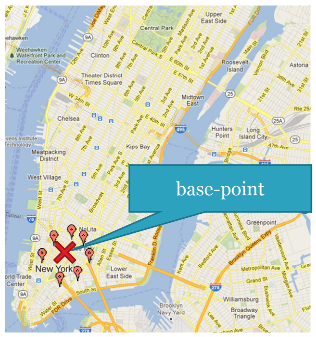


Figure 5. An example of base-point

due to the nature of living style and activities offered by their living areas. As a result, different similarities, naturally formed in aggregated behaviors of friends to various kinds of POI, are embedded in the friends' check-in activities. In a LBSN data, the most important information is user's common check-ins and distance among users for user similarity measurement. In the following, we propose to extract two population features to depict POI as below.

- Similarity by Common Check-ins (CheckSim) - We employ the χ^2 test for testing relation of check-in behaviors of Gowalla users and their friends. If the test shows that a relationship is significant, it means the user always checks in at a POI where her friends are also checked in. Based on the observations from the Gowalla dataset, shown in Figure 4, we find most of users will check in POI, which her friends also check in. Hence, the common check-ins is a good for measuring users' similarity. Based on the observations, we formulate the similarity of two users as (4).

$$\text{CheckSim}_{i,j} = \begin{cases} \cos(v_i, v_j) & \text{, if user i and j are friends} \\ 0 & \text{, otherwise} \end{cases} \qquad (4)$$

where cos() indicates cosine similarity and v_i indicates the check-in vector of user *i*. Take Table 1 as an example, the check-in vector of user *i* is <1, 0, 2, 5, 0>, and the check-in vector of user *j* is <0, 10, 0, 1, 0>. Thus, the CheckSim of user *i* and user *j* is

Table 1. An example of check-in log.

Restaurant ID	r1	r2	r3	r4	r5
user i	1	0	2	5	0
user j	0	10	0	1	0
user k	1	1	0	0	0
user l	1	1	1	1	5
total	3	12	3	7	5

$$\frac{(1\times 0)+(0\times 10)+(2\times 0)+(5\times 1)+(0\times 0)}{\sqrt{1^2+0^2+2^2+5^2+0^2}\times\sqrt{0^2+10^2+0^2+1^2+0^2}} \approx 0.09$$

- Similarity by Relative Distance (DisSim) - As discussed above, most people will check in the place following their friends who live nearby them. Based on the idea, this design idea of DisSim focuses on the distance between users and their friends. To do so, we must first identify users' living areas. We argue that most of a user's check-in activities will happen in her/his living area. Thus, for each user, we find out her/his top *k* frequently visiting POIs and treat the central of these POIs as her/his *base-point*, as shown in Figure 5. Accordingly, we formulate the DisSim of two users as follow.

$$DisSim(u,f)=\begin{cases}1-\dfrac{\text{Distance}(u,f)}{\max\limits_{f'\in F(u)}\{\text{Distance}(u,f')\}} & \text{, if user } u \text{ and } f \text{ are friends}\\ 0 & \text{, otherwise}\end{cases} \quad (5)$$

where Distance() indicates the Euclidean distance of two base-points and *F(u)* indicates the set of user *u's* friends.

3.2 Features from Individual Preference

As mentioned above, we could view each user of LBSN as a sensor to detect the semantic tags of a POI for improving POI recommendation in an urban area. In Gowalla website, there are two kinds of semantic tags, i.e., category and highlight, as shown in Figure 6. The category tag is annotated on a place when the place is created. Each place just have only one category tag, e.g., coffee, pizza, etc. On the other hand, any user (even the user never checks in the place) can arbitrarily annotate the highlight tag on a place.

Since the Gowalla will record the count of highlight tag, we could generally determine the possibility of that a tag *t* is annotated on a POI *p*, as shown in (6).

$$Possibility(t,p)=\frac{count(t,p)}{\sum\limits_{t'\in T(r)} count(t',p)} \quad (6)$$

where *count(t, p)* indicates the number of times the tag *t* is annotated on the POI *p* ,and *T(p)* indicates the set of tags of POI *p*. Take Figure 6 as an example, the possibility of that a tag 'coffee' is annotated on a POI is

$$\frac{2}{2+10+88}=0.02$$

Accordingly, given a user-POI pair (*u, p*), the features extracted form Individual Preference could be generally formulated as (7).

$$IP(u,p)=\sum_{t\in T(p)}(\text{Possibility}(t,p)\times\text{Personal Preference}(u,t)) \quad (7)$$

where *T(p)* indicates the set of tags of POI *p*.

We can significantly observe that measuring a user's individual preference of a semantic tag is the key of Individual Preference features. Intuitively, users' check-in history could reflect their preference of the type of POI. As a result, for each user, we aggregate the number of check-ins of the POI with the same tag to represent each user's personal preference of semantic tag. In the following, we propose to extract two features to depict users' preference.

- Preference in Category (CPref) - Based our observations from the Gowalla dataset, users' check-in activities are fluctuated. Some users frequently check in many places, and some users rarely check in at all. Hence, to measure individual user's personal preference of POIs, we need to normalize the aggregated number of check-ins by user's total check-ins. Based on above, we formulate a user *u*'s personal preference of a category tag *t* as (8).

$$\text{CPref}(u,t)=\frac{\sum\limits_{r\in C(t)} checkins(u,r)}{\#total\ check\text{-}ins\ of\ u} \quad (8)$$

where *C(t)* indicates the set of POIs with category tag *t*. Take Table 1 and Table 2 as an example, the check-in vector of user *i* is <1, 0, 2, 5, 0>. Thus, the user *i*'s personal preference of a category tag *A* is

$$\frac{1+0+0}{1+0+2+5+0}=0.125$$

Table 2. An example of category tag

Restaurant ID	p1	p2	p3	p4	p5
Category	A	A	B	C	A

Table 3. An example of highlight tag

Restaurant ID	p1	p2	p3	p4	p5
Highlight	a, b	b, c, d	a, d	a, c	g

Note that, as mentioned above, people will annotate highlight tag on a place and repeat the annotation many times. Based on the observation, to measure an individual user's personal preference of POIs, we could not directly normalize the aggregated number of check-ins by user's total check-ins. Take Table 1 and Table 3 as an example, suppose that we use user's total check-ins to normalize the aggregated number of check-ins, the user *i*'s personal preference of a highlight tag *a* is

$$\frac{1+2+5}{1+0+2+5+0}=1$$

It is clear that if we use user's total check-ins to normalize the aggregated number of check-ins, the total of personal preference of a user will be greater than 1.0 and the scale of total of personal preference of a user will be very fluctuated.

- Preference in Highlight (HPref) – Based on the above observation and condition of highlight tag, we use the summation of a user's total check-ins of each highlight tag for normalization of the user's personal preference of POIs. To do so, given a set of highlight tags *H*, we formulate a user *u*'s personal preference of a highlight tag *t* as (9).

$$\text{HPref}(u,t)=\frac{\sum\limits_{r\in hl(t)} Checkins(u,r)}{\sum\limits_{\forall t'\in H}\ \sum\limits_{r\in hl(t')} Checkins(u,r)} \quad (9)$$

where *hl(t)* indicates the set of POIs with highlight tag *t*. Take Table 1 and Table 3 as an example, the check-in vector of user *i* is <1, 0, 2, 5, 0>. Thus, the user *i*'s personal preference of the highlight tag *a* is

$$\frac{1+2+5}{(1+2+5)+(1+0)+(0+5)+(0+2)+(0)}=0.5$$

3.3 Features from POI Popularity

As discussed in Introduction, check-in data is very sparse. It leads to the features extracted from both Social Factor and Individual Preference does not work well for new users. We employ the popularity of POI to make a maximum likelihood estimation of the popularity of POI. However, the type of POI always affects

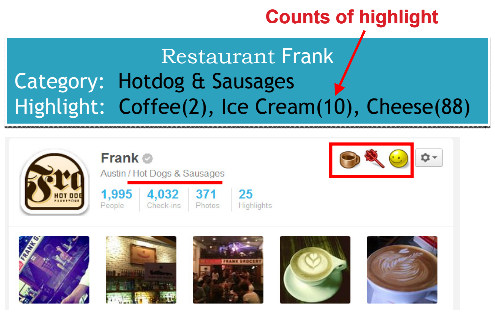

Counts of highlight
Restaurant Frank
Category: Hotdog & Sausages
Highlight: Coffee(2), Ice Cream(10), Cheese(88)
Frank
Austin / Hot Dogs & Sausages
1,995 People
4,032 Check-ins
371 Photos
25 Highlights

Figure 6. An example of semantic tag of a POI.

users' check-in will. For example, people may buy a cup of coffee in a coffee shop everyday but rarely go French restaurant. Base on this idea, we estimate the likelihood by conditional probability which is the probability of that users check in the POI given a category, so called relative popularity of POI. In the following, we propose to extract the feature to depict popularity of POI.

$$\mathrm{RP}(p) = P(p \mid C(p)) = \frac{\#total\ check\text{-}ins\ of\ p}{\sum_{p' \in C(t)} (\#total\ check\text{-}ins\ of\ p')} \quad (9)$$

where $C(t)$ indicates the set of POIs with category tag t. Take Table 1 and Table 2 as an example, the set of POIs with category tag A are $p1$, $p2$, and $p5$. The total check-in of POI $p1$, $p2$, and $p5$ are 3, 12, and 5, respectively. Thus, the popularity of POI $p1$ is

$$\frac{3}{3+12+5} = 0.15$$

3.4 POI Recommendation

After the phase of feature extraction, features derived from Social Factor, Individual Preference and POI Popularity are used as inputs for the POI recommendation phase to learn a regression-tree model. We choose M5Prime as the relevance score predictor because it has shown excellent performance in similar tasks [3]. The M5Prime is one kind of regression-trees, but it is not constrained by using M5Prime as the regression-tree model. When the model is trained, the training data first will be divided according to the decreasing of variance of target attribute. Then, each piece of training data will be used for building individual regression model. Meanwhile, the procedure of dividing training data will be recorded to build a decision tree as shown in Figure 7. This modeling process is NOT sensitive for bias data. In other words, M5Prime is not easily affected by data sparse problem.

In addition, M5Prime is an automatic and non-parametric model. Therefore, it is more convenient to use. Besides, as mention earlier, the LBSN data may growth rapidly, thus the efficiency is an unavoidable issue we should face. Fortunately, M5Prime needs lower run time. In our proposed UPOI-Mine, all user-POI pairs are used for building training model by M5Prime, i.e., a POI with high relevance score under examination is considered as the candidate with high probability that the user may like. A POI tends to be automatically recommended to a user if the POI is predicted as a high relevant score for the user.

4. EXPERIMENTS

In this section, we conduct a series of experiments to evaluate the performance for the proposed UPOI-Mine using Gowalla dataset . All the experiments are implemented in Java JDK 1.6 on an Intel Xeon CPU W3520 2.67GHz machine with 24GB of memory running Microsoft Windows win7. We first describe the data preparation on the Gowalla dataset and then introduce the evaluation methodology. Finally, we show our experimental results for following discussions.

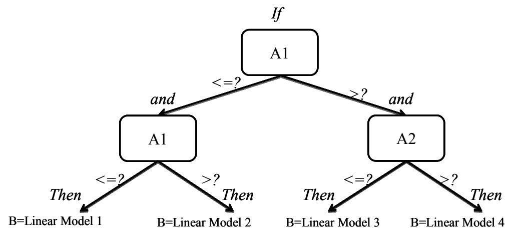


Figure 7. An example of M5Prime.

4.1 Gowalla Dataset

The Gowalla dataset is a check-in dataset collected from Gowalla website. The dataset contains 1,964,919 spots, 18,159 users, 5,341,191 check-ins and 392,246 friend links. We normalize the check-in for each user. Then, we use the normalized check-in as the ground truth, which is called "relevance score". We normalize the check-in values into the range [0, 5]. The process of normalization is described as (11).

$$\text{Relevance Score} = \begin{cases} 3, & \text{if } x = \text{avg} \\ 3 + \dfrac{x - avg}{\text{max - avg}} \times 2, & \text{if } x > \text{avg} \\ 3 - \dfrac{\text{x - avg}}{\text{min - avg}} \times 2, & \text{if } x < \text{avg} \end{cases} \quad (11)$$

where x indicates the real check-in number of a user. "min", "max" and "avg" indicate the smallest, largest and average check-in number of a user, respectively. Finally, we use the relevance score in our proposed UPOI-Mine to evaluate the score of POI based on social, location information and user preference factors.

4.2 Evaluation Methodology

UPOI-Mine is based on the ranking of POI score and thus can be viewed as an information retrieval system if we consider a user as a query term. Therefore, we employ the popular measurement Normalized Discounted Cumulative Gain (NDCG) [8] to measure the list of recommended POIs. NDCG is commonly used in information retrieval to measure the search engine's performance. A higher NDCG value for a list of search results indicates that the highly relevant items have appeared earlier (with higher ranks) in the result list. For each list of recommended POIs, we can obtain a score list, where the scores are provided by ground truth. Such list is called the relevance vector. For example, suppose that the prediction model estimates the relevance scores of POIs $F1$, $F2$, $F3$, and $F4$ are 4, 3, 2, and 1, respectively. Hence, the POIs will be ordered as $<F1, F2, F3, F4>$ by recommender. Suppose that the relevance score of ground truth for POI list $<F1, F2, F3, F4>$ is G =<2, 3, 0, 1>. That is the relevance scores of $F1$ and $F2$ are 2 and 3, respectively. The Discounted Cumulative Gain (DCG) of a relevance vector G is computed by Equation (12). The premise of DCG is that the highly relevant documents appearing lower in a search result list should be penalized as the graded relevance value is reduced logarithmically proportional to the position of the result. Here the parameter b is to control where we start to reduce the relevance value. For example, if the relevance vector is <2, 3, 0, 1> and b is set as 3, the $DCG[4]$ is $2+3+(0/\log_3 3)+(1/\log_3 4)$. (In our experiments, b = 2.)

$$DCG[i] = \begin{cases} G[i], & \text{if } i = 1 \\ DCG[i-1] + G[i], & \text{if } i < b \\ DCG[i-1] + \dfrac{G[i]}{\log_b i}, & \text{if } i \ge b \end{cases} \quad (12)$$

In particular, NDCG@p, measures the relevance of top p as shown in Equation (13).

$$NDCG@p = \frac{DCG[p]}{IDCG[p]} \quad (13)$$

where $IDCG[p]$ indicates the $DCG[p]$ value of ideal ranking list. For example, given a ranking list of 5 items with relevance as <4, 1, 3, 1, 1>, the ideal ranking list of this 5 items is <4, 3, 1, 1, 1>. NDCG ranges from 0 to 1. The higher NDCG is, the better a ranking result list is. In the above example, the NDCG @5 is

$$NDCG@5 = \frac{4 + \frac{1}{\log_2 2} + \frac{3}{\log_2 3} + \frac{1}{\log_2 4} + \frac{1}{\log_2 5}}{4 + \frac{3}{\log_2 2} + \frac{1}{\log_2 3} + \frac{1}{\log_2 4} + \frac{1}{\log_2 5}} = 0.913785$$

However, NDCG is not comprehensive, because it only focuses on ranking performance avoiding the absolute difference between estimated relevance and ground truth of relevance. For example, given a ranking list of 5 items with estimated relevance as <5, 4, 3, 1, 1>, the ground truth of relevance of this 5 items is <4, 3, 2, 2, 2>. We can observe that NDCG will be 1.0 that means the effectiveness of recommender is pretty good. Excepting the ranking performance, we also want to examine the how close our predictions are to the eventual outcomes. Therefore, we employ the Mean Absolute Error (MAE) to measure the list of recommended POIs as Equation (14).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| \qquad (14)$$

where f_i indicates the estimated relevance scores of POI i and y_i indicates the ground truth of relevance scores. The lower MAE is, the fewer error is. In the above example, the MAE is

$$\frac{1}{5} \times (|5-4| + |4-3| + |3-2| + |1-2| + |1-2|) = 1.0$$

4.3 Experimental Results and Discussions

We divide the experiment into two parts: internal experiment and external experiments. For the internal experiments, we first compare the performance of our social, location information and user preference factors. Then, we compare the effectiveness of every feature in all of factors. For the external experiments, we compare the performance of UPOI-Mine with TrustWalker [5] and CF-based model [14] in terms of NDCG and MAE.

4.3.1 Comparison of Various Features

This experiment evaluates how each factor and feature performs in the proposed UPOI-Mine in terms of NDCG@10 and MAE. Figure 8 shows the NDCG@10 value of UPOI-Mine, considering social, location information and user preference, respectively. We observe that the effect of user preference is better than that of other two factors. The result shows that the check-in behaviors of user-self still more important than that of friends. However, we still can use location information and social factor to provide recommendations when the user is new to this LBSN website. Figure 9 shows that the MAE is smallest when combining all factors. The reason is that the three factors can be complementary of one another when regression model is trained. In Figure 10 and Figure 11, we compare all features in terms of NDCG@10 and MAE respectively. We observed that the effect of category feature outperforms other features because the check-in of user-self is most important.

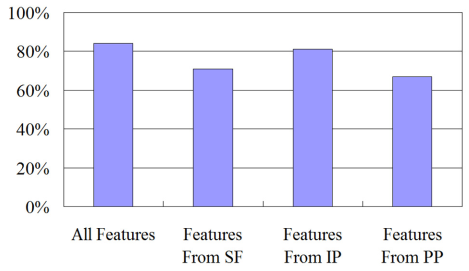


Figure 8. Comparison of deferent aspects in terms of NDCG@10.

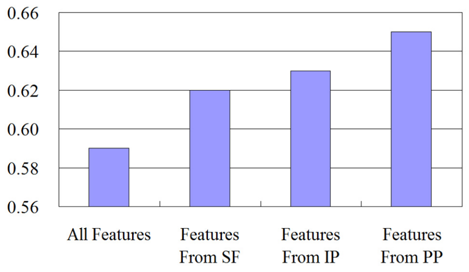


Figure 9. Comparison of deferent aspects in terms of MAE.

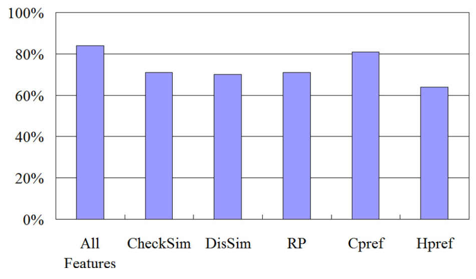


Figure 10. Comparison of deferent features in terms of NDCG@10.

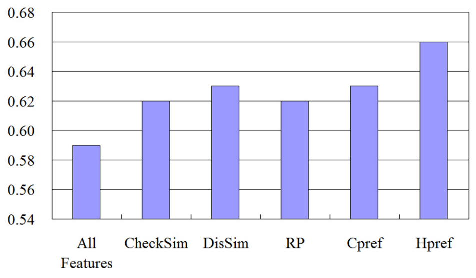


Figure 11. Comparison of deferent features in terms of MAE.

4.3.2 Comparison of existing recommender

This experiment evaluates the performance of our proposed UPOI-Mine comparing TrustWalker [5] and CF-based [14] in terms of NDCG@10 and MAE. TrustWalker is a trust-based CF method. It uses social factor to recommend items to users. CF-based POI [14] considers geographic influence, social influence and user preference influence with collaborative filtering to recommend locations to users. Figure 12 shows UPOI-Mine outperforms TrustWalker and CF-based POI in terms of NDCG@10 and MAE. The reason is that we consider check-in of user-self in the factor of user preference while other methods do not. We also observe that the MAE of CF-based POI is large, as shown in Figure 13. CF-based POI only considers the common check-in of two users when computing social influence and user preference influence. Because of the sparsity of LBSN data, considering common check-in will lead to many zero scores. So the reason of large MAE value is that many zero scores which is far away for the relevance scores of users while UPOI-Mine and TrustWalker use social relation to generate scores in social factor.

5. CONCLUSIONS

In this paper, we have proposed a novel approach named *Urban POI-Mine (UPOI-Mine)* for recommendation of interesting urban POIs by mining users' preferences. Meanwhile, we tackle the problem of mining user check-in behaviors in urban computing, which is a crucial prerequisite for effective recommendation of POIs in urban areas. The core task of POI recommendation in

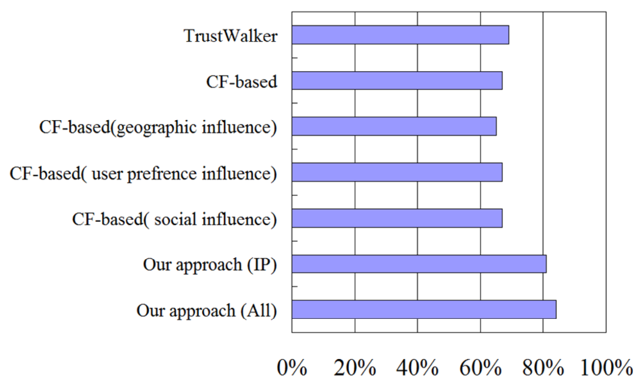


Figure 12. Comparison of various models in terms of NDCG@10.

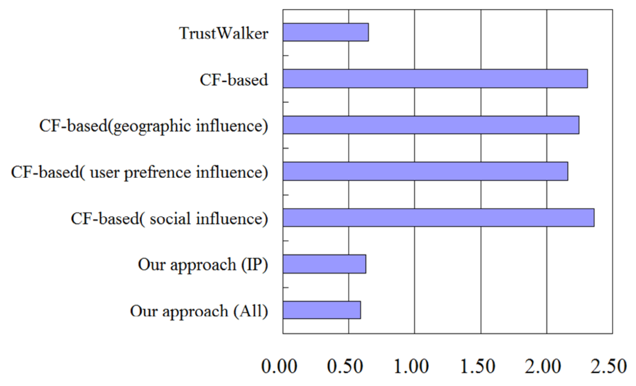


Figure 13. Comparison of various models in terms of MAE.

urban areas is nicely transformed to the problem of relevance score prediction. We evaluate the relevance score of each user-POI pair by learning a regression-tree. In the proposed *UPOI-Mine*, we have explored i) *Social Factor* (*SF*), ii) *Individual Preference* (*IP*), and iii) *POI Popularity* (*PP*) by exploiting the LBSN data to extract descriptive features. To our best knowledge, this is the first work on urban POI recommendation that considers social factor, preference relatedness and POI popularity in LBSN data, simultaneously. Through a series of experiments by the real dataset Gowalla, we have validated our proposed *UPOI-Mine* and shown that *UPOI-Mine* has excellent performance under various conditions.

6. ACKNOWLEDGMENTS

This research was partially supported by National Science Council, Taiwan, under grant no. NSC100-2631-H-006-001 and NSC100-2218-E-006-017.

REFERENCES

[1] B. Berjani and T. Strufe. A Recommendation System for spots in Location-Based Online Social Network. Proceedings of the 4th Workshop on Social Network Systems Article No. 4,2011.

[2] S. Debnath, N. Ganguly, P. Mitra. Feature Weighting in Content Based Recommendation System Using Social Network Analysis. Proceedings of WWW, pages 1041-1042, 2008.

[3] A. Etemad-Shahidi and J.Mahjoobi. Comparison between M5' model tree and neural networks for prediction of significant wave height in Lake Superior. Ocean Engineering36(2009)1175–1181.

[4] J. Hu, H.-J. Zeng, H. Li, C. Niu, Z. Chen. Demographic prediction based on user's browsing behavior. Proceedings of ACM WWW (WWW'2007) May 2007.

[5] M. Jamali, M. Ester . TrustWalker: A Random Walk Model for Combining Trust-based and Item-based Recommendation. Proceedings of KDD, pages 397-406, 2009.

[6] T. Horozov, N. Narasimhan, V. Vasudevan. Using location for personalized POI recommendations in mobile environments. Proceedings of SAINT, pages 124-129, 2006.

[7] E. H.-C. Lu, W.-C. Lee and V. S. Tseng, "A Framework for Personal Mobile Commerce Pattern Mining and Prediction," IEEE Transactions on Knowledge and Data Engineering (TKDE), Volume 24, Issue 5, May 2012, Pages 769-782.

[8] D. Manning, P. Raghavan and H. Schütze. Introduction to Information Retrieval. Cambridge University Press, 2008.

[9] P. Massa and P. Avesani ,Trust-aware Recommender Systems, In Proceedings of RecSys, pages 17-24, 2007.

[10] T. Menzies, J. S. D. Stefano, M. Chapman. Learning Early Lifecycle IV&V Quality Indicators. Proceedings of the Ninth International Software Metrics Symposium (METRICS'03)

[11] C. Ono, M. Kurokawa, Y. Motomura, and H. Asoh. A Context-Aware Movie Preference Model Using a Bayesian Network for Recommendation and Promotion, In Proceedings of UM, pages 247-257, 2007.

[12] E. Spertus , M. Sahami, O. Buyukkokten. Evaluating similarity measures: a large-scale study in the Orkut social network. In Proceedings of KDD, pages 678-684, ,2005.

[13] K. W.-T. Leung,D. L. Lee,W.-C. Lee. CLR: A Collaborative Location Recommendation Framework based on Co-Clustering. In Proceedings of SIGIR , pages 305-314 , 2011.

[14] M. Ye, P. Yin, W.-C. Lee, and Dik-Lun Lee. Exploiting Geographical Influence for Collaborative Point-of-Interest Recommendation. Proceedings of KDD, pages 1046-1054 , 2011.

[15] S.-J. Yen, Y.-S. Lee, C.-H. Lin and J.-C. Ying, Investigating the Effect of Sampling Methods for Imbalanced Data Distributions, Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC'2006), pp. 4163-1468, October 2006.

[16] J. J.-C. Ying, E. H.-C. Lu, W.-C. Lee, T.-C. Weng, V. S. Tseng. Mining User Similarity from Semantic Trajectories. In Proceedings of ACM SIGSPATIAL International Workshop on Location Based Social Networks (LBSN' 10), San Jose, California, USA, November 2, 2010.

[17] J. J.-C. Ying, W.-C. Lee, T.-C. Weng, V. S. Tseng. Semantic Trajectory Mining for Location Prediction. Proceedings of The 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS' 11), Chicago, IL, Nov. 2011.

[18] J. Yuan, Y. Zheng, and X. Xie, Urban Computing with Taxicabs, Proceedings of 13th ACM International Conference on Ubiquitous Computing (UbiComp 2011), Beijing, China, Sep. 2011.

[19] J. Yuan, Y. Zheng, X. Xie. Discovering regions of different functions in a city using human mobility and POIs. Proceedings of 18th SIGKDD conference on Knowledge Discovery and Data Mining (KDD 2012).

Intention Oriented Itinerary Recommendation by Bridging Physical Trajectories and Online Social Networks

Xiangxu Meng
College of Computer Science
National University of Defense Technology
Changsha, China
aiguokk@gmail.com

Xinye Lin
College of Computer Science
National University of Defense Technology
Changsha, China
lxytcmn@gmail.com

Xiaodong Wang
College of Computer Science
National University of Defense Technology
Changsha, China
xdwang@nudt.edu.cn

ABSTRACT

Compared with traditional itinerary planning, intention oriented itinerary recommendation can provide more flexible activity planning without the user pre-determined destinations and is specially helpful for those strangers in unfamiliar environment. Rank and classification of points of interest (POI) from location based social networks (LBSN) are used to indicate different user intentions. Mining on physical trajectories of vehicles can provide exact civil traffic information for path planning. In this paper, a POI category-based itinerary recommendation framework combining physical trajectories with LBSN is proposed. Specifically, a Voronoi graph based GPS trajectory analysis method is proposed to build traffic information networks, and an ant colony algorithm for multi-object optimization is also implemented to find the most appropriate itineraries. We conduct experiments on datasets from FourSquare and GeoLife project. A test on satisfaction of recommended items is also performed. Results show that the satisfaction reaches 80% in average.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications; I.2.8 [**Problem Solving, Control Methods, and Search**]: Scheduling

General Terms

Algorithms, Measurement, Experimentation.

Keywords

itinerary planning, trajectory mining, location-based system, multi-level categories

1. INTRODUCTION

According to the daily experience, the nature of travel arrangement is to look for a group of geographical locations which are connected with most convenient traffic paths available and best meet the personal demands. As described in reference [2], the users usually hope to find a set of objects to meet various needs. For example, a traveler may have the following needs – shopping, dining, accommodation, sightseeing. These needs can only be met by a set of different geographic locations. Of course, there are many kinds of criteria to evaluate the final choice. Some people want the best service, and others pursue the lowest cost. On the initial stage of itinerary planning, the users usually only have a general intention. The final decision will be made after collecting enough relevant information. During the collection process, they may ask a friend for recommendation, do research on some Travel Forums, or analyze traffic data, etc.

When someone plans to travel to a new place, his initial intention is usually not very clear. In order to analyze such initial intentions, we put out questionnaires to a group of students in National University of Defense Technology (lies in Changsha, Hunan, China, about 1500km away from Beijing). The questionnaire includes the following three questions:

1. How much do you know about Beijing? (A. Have been there many times, very familiar. B. Only been there 1-5 times, not very familiar. C. Never been there, not familiar at all.)

2. Please list your arrangement there for a one-day trip?

3. If someone is willing to help you to arrange your trip, what demands will you have?

Example 1: The answer of a volunteer who belongs to C class (Never been there, not familiar at all):

In the morning, he wants to have breakfast in a fast-food restaurant and visits some historical interests afterwards. At noon, he would like to have lunch in a restaurant with local taste. As for afternoon, he is willing to walk besides a lake in a park. Last but not least, he needs buy some souvenirs for friends and then go to a party in a bar which should be convenient to get to. Before going to all these places, he hopes to get as much information as possible, and an accurate prospect on both time and money cost on the traffic.

Result analysis (20 valid questionnaires in total are collected):

Question 1: Numbers in parentheses represent the number of volunteers belongs to each class: A. (4) B. (8) C. (8)

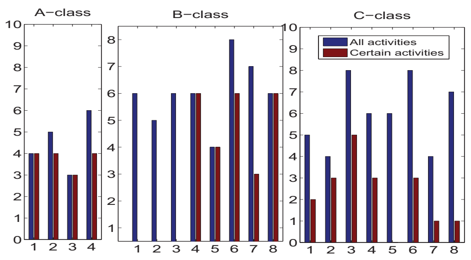


Figure 1: Results of questionnaire.

Question 2: For the convenience of analysis, the arrangements proposed by the users are divided into two categories: 1. Certain activities: activities that have locations and transportation requirements, for example, visiting the Imperial Palace in the morning. 2. General activities: General activities: activities that only have general intentions (without particular location requirement), for example, having Sichuan cuisine for lunch. As shown in Fig. 1, according to different choices made in question 1, users are divided into 3 classes, A, B and C. Users who are familiar with Beijing (A-class) are more likely to arrange certain activities. In contrast,most of the users of B and C class only have a general intention. Especially for the C-class users, the portion of general activities is higher than fifty percents. For more, 4 out of the 20 users do not have any precise preference of traveling locations.

Question 3: Five demands that have the highest occurrence rates are (the number in the parentheses indicates the number of users):1. unique/famous places(7); 2. least traffic time cost(5); 3.precise location description and best route(5); 4. avoidance of rush hour of travelers (4); 5. lower cost (4).

After the analysis above, it is reasonable to summarize the following features of a common user's itinerary planning for a new place.

1. The description of the demanded destination is usually a category but not a exact geographical location, such as "a famous attraction" , "a well-known snack bar" , etc.

2. Users demand precise information of each place they want to go such as exact geographical location, the traffic route, the shortest and average taxi time between any two locations.

3. Users are interested in knowing other people's review about the places.

4. Some of the users' demands have multiple optimizing objects, which are clearly expressed and possible to be mathematically described, such as least time cost on the traffic; some are very vague and hard to be described mathematically, such as the most famous location available.

It is very difficult for the computer to implement the aforementioned search or recommendation, because the object description is not clear, and the information in need is too extensive. Even a human guide will find it hard to give such recommendations, because it is impossible for him/her to know clearly about all the restaurants, hotels, attractions of a city, and the best traffic route and time among them. Besides, because the GIS-based shortest route search algorithm doesn't consider the real-time traffic change and the rush hours (weekends, people going to and off work), it is also very hard for the route search algorithm to figure out the best travel route and least time cost. To resolve such a problem and help the users to make the best decision, we jointly take in use of different users historical trajectories and information from social networks to provide a category based itinerary planning service. Luckily, both the current social networks and trajectory mining technologies can present effective support to such a service.

Currently, most mobile devices are able to position, based on the networks or GPS. This feature makes it very easy to collect the trajectory data of mobile users. In a much larger scale, the traffic control and city planning departments of a city are also collecting the trajectory data of quantities of vehicles and mobile devices. Some large enterprises are also doing so to carry on relative research. The T-drive project [11, 12]of Microsoft is an example. Based on the fact that the taxi drivers have the richest knowledge of the road systems of a city, researchers provide a real-time navigating service by mining the driving logs of a large number of taxis.

The location based social networks, such as Jiepang [1] and Foursquare, support the users with tagging, rating and reviewing places they have been to. These user-generated data have embraced the daily experience of millions of users, and are very valuable for the recommendation of itinerary planning.

All these technologies provide us the chance of realizing accurate itinerary recommendation. To our best knowledge, none commercial entities have yet provided itinerary planning recommendation service based on general demands. This paper will focus on the realization of general itinerary recommendation.

2. MODEL AND FRAMEWORK

2.1 Model of itinerary planning

It is easy to understand that an activity must be related to a location. Therefore, an activity can be denoted as a tuple with three elements:

$$A :=< P, T, condition >$$

P (Place) denotes the geographical position, which indicates an exact location, such as the Imperial Palace, the Olympic Park, etc.

T (Time) denotes the time when the activity takes place.

C (Condition) can be a constraint on any aspect of an activity. For example, it could be a constraint of having less cost than $\$100/person$, or traveling by taxi, etc.

It is important to notice that, the places are normally hierarchically organized. According to the analysis result from the questionnaires, the users usually won't give an exact depiction of the demanded destination. Instead, users are more likely to give a category such as bar, shopping mall, etc. In order to solve this problem, this paper takes use of the categorization of different locations of Foursquare. A category (C) is a set of different location points:

$$C := \{P_1, P_1, P_1, \ldots, P_n\}$$

[1] JiePang is one of the biggest location based social networks in China, www.jiepang.com.

The categories are hierarchical organized. Places belong to categories.

$$C_i \subset C_j, P_i \in C_j$$

A *path* indicates the connecting route between to places, i is the description, such as the time cost.

$$Path := < P_s, P_e, i >$$

A trip is a time-ordered description of several activities and the path belong them.

$$Trip := < A_1, Path_1, A_2, Path_2, \ldots, A_n >$$

In this paper, the interrogation mark (?) is used to indicate any ONE place, the asterisk mark (*) is used to indicate all the places in a category. By using the two marks, the user's vague demands of itinerary planning could be described as follows:

$$Need := \{ \begin{array}{l} <?C_1, T_1, condition_1 >, \\ <?C_2, T_2, condition_2 >, \ldots, \\ <?C_n, T_n, condition_n >>: Optimizer\} \end{array}$$

optimizer denotes the optimization goal, such as the shortest travel time or hot spots of all POIs. According to the model, example 1 can be described as follow:

$$\{ \begin{array}{l} <?FastFoodRestaurant, Workday \odot 8:00, Taxi >, \\ <?HistoricSite, Workday \odot 9:00, Taxi >, \\ <?ChineseRestaurant, Workday \odot 12:00, Taxi >, \\ <?Plaza, Workday \odot 13:00, Taxi >, \\ <?Mall, Workday \odot 18:00, Taxi >, \\ <?WineBar, Workday \odot 20:00, Taxi >, \\ : [MinTravelTime, AllPopular]\} \end{array}$$

2.2 Traffic information networks

To find the best trip, we need to build a cost function; the cost could be the time cost, the service evaluation, or the total distance along the trip. If we want to find the least time cost, the information of driving time between any two places would be necessary. Thus, we build up the semantic traffic map G of a city based on the taxis' trajectories:

$$G := \{P, Path\}$$

P is the set of all the geographical places; Path is the set of all the existing edges between these locations. To build such a map G, we need to get information of all the places and path between them. The detailed method of building the semantic traffic map is presented in Section 3.

2.3 Framework of Joint itinerary planning

This paper access information of geographical locations from social networks, and build up semantic traffic map by mining the GPS trajectories, as shown in fig.2. According to different cost functions, the system will present different trip recommendations for itinerary planning with detailed explanations respectively. The explanation includes valuable information for the user, such as the popularity of the places and the driving time between adjacent locations in the trip. The whole system consists of three repositories:

1. Repositories of location points: We crawling all location points, which are provided by different users and evaluation by millions of users, of the target city from location based social networks. This collaborative evaluated information can more objectively indicate the popular degree of location points;

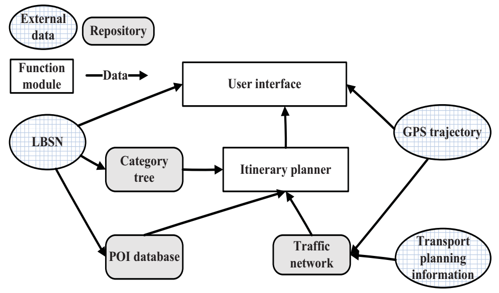


Figure 2: Framework of joint itinerary planning.

2. Categories of location points: It saves classification information of places. In this paper, we use hierarchy categories provided by foursquare;

3. Traffic information networks: It preserves the shortest travel time and the best path between any two arbitrary semantic location points of a city. We get semantic locations, which paid close attention by users, from public transit agencies, and mining taxi history data to obtain accurate shortest path information (section 3.2).

What's more, itinerary planner is responsible for recommending reasonable itineraries based on the information above, according to the user's requirements. The UI module calls the online map system (such as Google Map [2]) to visual display itineraries, and supports users to modify requirements interactively.

3. VORONOI GRAPH BASED TRAFFIC INFORMATION NETWORKS

Traffic departments of many countries and cities are collecting trajectories of public transport vehicles. These GPS data have high frequency and large time span, hence the data are very difficult to process. Besides, these trajectory data only contain the physical space and time stamps, so they cannot provide any semantic information [9], which the users can understand better and usually concern much more. To convert the physical trajectory into feasible semantic knowledge, there are two main obstacles: 1. How to determine the physical location point that the user understands best and knows most. 2. How to get the shortest path and driving statistics between any two location points. In the following part of this section, we propose the semantic position determining algorithm based on public transport stops, and the semantic traffic map building algorithm base on Voronoi diagram. These two algorithms make it possible to get the shortest time and optimized path between any two places.

3.1 Voronoi based semantic point building

Every city contains thousands or even more geographical locations, it is very hard for the residents to memorize all of them, not to say a stranger. In tradition, the most used method to describe some geographical location is to combine the "traffic network" and the landmarks. For example, No.163 of Haidian West Road, Beijing; or 400m east

[2]http://maps.google.com/

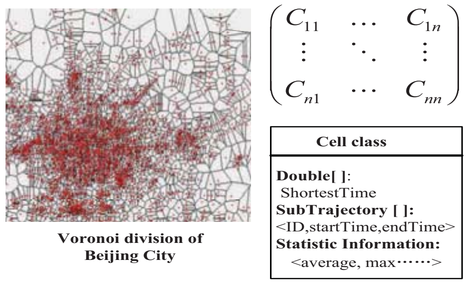

Voronoi division of Beijing City
$\begin{pmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{pmatrix}$
Cell class
Double[ ]:
ShortestTime
SubTrajectory [ ]:
<ID,startTime,endTime>
Statistic Information:
<average, max……>

Figure 3: Voronoi based semantic point.

to the south gate of the Olympic Park. Fortunately, when planning a stop or station for public transport, the related departments always take the population, physical distance, traffic situation and other factors into consideration. And these stops are commonly named with semantic marks that the residents know well. Concerning the planning principle of these stops, it is reasonable to assume that for any important location A of a city, there is always a stop B, so that the time cost from B to A is smaller than a given threshold, for example, 10 minutes. Based on the assumption above, this paper tries to build up a semantic map of a city, based on the stops of public transportation. As shown in Fig.3, the whole city area is decomposed using Voronoi diagram, taking the stops as seeds. Each cell has a unique id, and is the named after the seed stop within it. Thus each cell generates a semantic point with constant id and name(***SemiPoint***). Voronoi diagram insures that, for any geographical point(GPS-Point,***Point***), there is a corresponding nearest semantic point (in our case, a stop). So, a GPS trajectory can be described by a semantic trajectory(Semi-trajactory,***SemiList***), which is a serial of ***SemiPoints***.The information of all the stops can be accessed on the traffic department's website. Voronoi diagram is built by the classic plane sweep algorithm [3].

For each stop point, we treat it separately as start and destination to build up the semantic traffic map, which is stored in form of adjacent matrix(***Time-Matrix[Stop Num][Stop Num]***). An element of this matrix describes the traffic statistics between two semantic points. The specific method to generate these traffic statistics is determined by the administrator according to different demands, which is implemented by different statistic methods(***StatisticFunction()***). For example, 7 days a week, calculate the shortest time and best path of every 2 hours. Considering that there're only 7 days data in the demo system, we only calculate the shortest time and best path between any two semantic points for rush hours on weekdays and workdays.

3.2 Run through model based traffic information networks building

After decomposing the city map, we need to calculate the statistics between any two cells. In traditional GIS system, the best path is simply the shortest path which could easily be found, and the shortest time could be estimated according to the Euclidean distance. However, in the real life, traffic restrictions, one-way road, dynamic traffic flows, and many other factors together generate a very complicate situation, in which the shortest path usually is not the one with smallest time cost. Furthermore, there isn't any model or algorithm yet that can find the path with shortest travel time both precisely and independently.

Algorithm 1 Voronoi-based Time Matrix Building

```
    Input: Trajectory:traj; Voronoi-Map: map
    Output: Time-Matrix[CellNum][CellNum]: matrix
 1: Point p=traj.next();//Fetch a new GPS point
 2: SemiList tmpList=new SemiList();
 3: while (P!=null) do
 4:    SemiPoint Sn=M.semi(P); //Get the semi-point
       based on physical location
 5:    if (Sn is different from temList.last() ) then
 6:       for each SemiPoint Si ∈ temList do
 7:          Matrix[Si.ID][Sn.ID]=StatisticFuction();
 8:       end for
 9:       tmpList.add(semiPoint);
10:    end if
11:    p=traj.next();
12: end while
13: return matrix;
```

The GPS trajectory contains a detailed log of the vehicle's travel history. Each GPS point includes the latitude and longitude coordinates, as well as the corresponding time stamp. Taking use of these data we can directly get the travel path and time cost between two locations. To match the semantic city map, the cell-ID of each GPS is allocated to it as the Semantic Point. After that, we have mapped millions of POI points to tens of thousands of stops, largely decrease the s-torage and computing cost. Considering how the semantic city map is generated, we can confirm that the timing error, which were introduced by such a mapping strategy, is no more than the time cost of vehicle passing through a cell. Take Beijing as an example, the travel time of buses between any two stops in Beijing is strictly less than 10 minutes; that means the final found shortest time cost will include an error less than 10 minutes, which is acceptable in the scenario of itinerary planning. Practically, the difficulty is that, there are not enough trips (a trip, as defined before, is a complete trajectory from start to the end) for any pair of Semantic Points to calculate the statistical driving time. Therefore, we propose that, for all the cells passed by, even they are not the start or end point for the path, we still treat them as statistic source. Thus, if we denote a trip as $A_s - A_e$, then after we take into consider the pass-by cells, it can be denoted as:

$$< A_s, A_1, A_2, A_3, A_4, \ldots, A_e >$$

As Algorithm 1 shows, when calculating the path statistics between cell A_1 and A_3, the sub-trip $A_1 - A_2 - A_3$ of $A_s - A_e$, which runs through both A_1 and A_3, is also taken part into statistics. Especially, this method can find potential shortest paths lying behind a longer trip. After entering a new cell, the algorithm will traverse over all the historical semantic points in the trip, store the travelling time between each historical semantic point and the new semantic point, and update the statistics. (line 5-8)

4. CATEGORY BASED ITINERARY RECOMMENDATION ALGORITHM

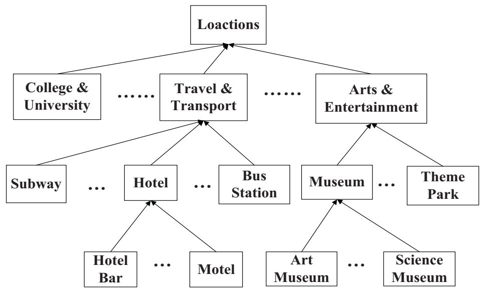


Figure 4: Hierarchy of the location points

4.1 Hierarchy of the location points

There are many hierarchical categorization methods of locations. This paper takes in use the categorization of Foursquare, one of the earliest and largest LBSN websites today. Fig. 4 partly shows the hierarchy of this categorization in the form of a tree. As shown in Algorithm 2, when a user gives his/her demanded location category, all the locations belong to that category and the corresponding subcategories are taken into consideration.

4.2 Least-time itinerary planning recommendation

This subsection proposes the itinerary planning recommendation algorithm that uses a cost function considering time cost, aiming at giving recommendations with least time cost.

4.2.1 BaseLine: Single-object optimization

We implement the Baseline recommendation algorithm with three steps. The algorithm enumerates all the possible locations, and traverses all the feasible trips and tries to find the best one to recommend. In previous section, we have received needs of user(***needs***),built traffic graph matrix(***matrix***) with each elements present cost from two semi-points(***Cost*** gotten by ***CostFunction()***),gotten category tree from Foursquare(CategoryTree,***ct***)and built the index of POIs(POIsIndex,***PI***)with ***CategorySet*** to describe which categories one POI belongs. As shown in Algorithm 2, in the first step (line 3),the k most popular geographical locations are given to each activity instance as candidates; in the second step (line 5-8), pick one from the k locations for each activity instance, and find the path connecting them by checking the semantic traffic map; then a recommendation item is generated; in the third step (line 10), rank the set of all recommendation items according to the cost of each item, and generate the final recommendation list.

4.2.2 Challenges of the Baseline algorithm

The baseline algorithm is simple and instinctive, but faces the following challenges:

1. Combinatorial explosion: the algorithm needs to traverse all the possible combinations of all the activities' candidate locations, which is at the level of k^{dim} (dim is the number of activities). When k increases, the computing cost will increase immensely. For example, when dim is 6 and k is set to 10, the time cost of the baseline algorithm reaches 2328 ms, which cannot meet the real-time needs of online service.

Algorithm 2 Category-based Activity Scheduling

```
    Input: needs; ct; PI; matrix;k
    Output: Tuple(POI[n], Path[n-1],Cost):scheduling
 1: for all need needs[i] do
 2:     CategorySet cs[i]=ct.subCategory(needs[i].category);
 3:     get top-k famous POIs POIs[i][k] belongs to cs[i];
 4: end for
 5: for all  composition POI[n] built by geting one elements from each column of POIs[i][k]  do
 6:     for all  $i < n - 1$ do
 7:         build Path[i] using POI[i],POI[i+1];
 8:         cost=CostFunction(matrix.pathCost[i]) ;
 9:     end for
10:     scheduling.add(POI[n],Path[n-1],cost);
11: end for
12: scheduling.sortBy(cost);
13: return scheduling;
```

2. Over-fitting of single object: when optimization object is to get least time cost and k is set to a large value (which means more candidate locations), the algorithm may be over-fit to the single object, causing that the popularity of locations are ignored, and the final recommendation will be a group of locations geographically assembling in a small area.

Thus we conclude that the itinerary planning recommendation problem should be treated as a multi-objective optimization problem, which do not strictly require the optimized result; the approximate optimization result is acceptable. In the following section, we propose a multi-objective optimization algorithm, which aims at finding approximate optimizations in a short time.

4.3 Ant Colony Optimization (ACA) based multi-objective itinerary planning

Multi-objective optimization problem is very common in scientific research and engineering practice. In general, it has a final object which is composed of several objects. This kind of optimization is usually high-dimensional and with large scale, and need to consider the weight allocation for different sub-objects. However, we cannot precisely design the weight allocations for different sub-objects of the problem in this paper. For example, it's hard to determine which of the two, the travel time and popularity of the locations, is more important.

Fortunately, ACA, Simulated Annealing, Genetic Algorithm and Neural Network and other intelligent algorithms have been developed and applied extensively. This paper proposed a modified version of ACA to solve the multi-objective optimization of itinerary planning recommendation. The modified ACA mainly focus on the weight allocation for different sub-objects. Furthermore, it is convenient to adjust the number of ants and running times according to the system load, improving the service of the system.

4.3.1 Build up the ant colony system

ACA is a bionic algorithm [4]; its main idea is to simulate the food seeking behavior of the ants. It initially put a large number of ants that randomly wander in the search space, once an ant found the "food", it put some pheromone along the path to increase its attraction to other ants; besides the pheromone evaporates with time goes on. Such a positive feedback strategy leads the algorithm to a global optimization. ACA is intrinsically parallel, which makes it to be easily programmed in parallel. This paper introduces in ACA and makes the following modification: keep the global optimization object unchanged, i.e. finding the shortest path; introduces in a heuristic rule, that merges the popularity of locations into the pheromone updating phase, making the ants tend to choose the locations with higher popularity. Activity scheduling is denoted as a ordered list with elements from n non-intersecting sets.

$$path = \langle S_1, S_2, S_3, \dots S_n \rangle$$

P_{ij} is the $j - th$ POI of $i - th$ set. Here we support every sets include K POIs. Every POI in a set existing a edge, $E(P_{ij}, P_{(i+1)l})$, between all POIs of neighborhood sets.
In ant colony system, we use follow annotations:

$A^t(k)$ —describe the state of the kth ant in t times visit.

$D(P_{ij}, P_{(i+1)l})$—describe the minimal delay between two POIs.

$I^t(P_{ij}, P_{(i+1)l})$—describe the information between two POIs of t times visit.

$P^t_k(P_{ij}, P_{(i+1)l})$—describe the probility of the kth ant transfer form P_{ij} to $P_{(i+1)l}$.

$H(P_{ij})$—describe the famous value of a POI.

4.3.2 *Itinerary planning based on modified ACA*

The procedure of modified ACA.

1. Initialization, randomly put each ant on locations of the first set. (Algorithm 3, line 1-4)

2. The ants traverse the n sets in order, the transfer probability between two cities is as below. (Algorithm 3, line 6-10)

$$P^t_k(P_{ij}, P_{(i+1)l}) = \frac{I^t(P_{ij}, P_{(i+1)l}) H(P_{(i+1)l})}{D(P_{ij}, P_{(i+1)l})} \tag{1}$$

3. After all the ants have completed their travel, calculate the total cost of each path, save the one with the smallest cost. In the meantime, evaluate the current state and determine whether the ending condition is satisfied. If it is satisfied, return the best path; else, update the pheromone of each edge. When the ant completes a round, the pheromone of each edge is changed according to the following equation. (Algorithm 3, line 11-18)

$$\Delta I^t_k(P_{ij}, P_{(i+1)l}) = \begin{cases} \frac{Q}{L^k_t} & if A^t(k) cover E(P_{ij}, P_{(i+1)l}) \\ aa & else \end{cases} \tag{2}$$

$$I^{t+1}(P_{ij}, P_{(i+1)l}) = I^t(P_{ij}, P_{(i+1)l})(1-\gamma) + \sum_{k=1}^{m} \Delta I^t_k(P_{ij}, P_{(i+1)l}) \tag{3}$$

4. Re-put all the ants, start a new period. (Algorithm 3, line 19)

According to the procedure described above, the pseudo code of the modified ACA is given below.

Algorithm 3 Ant Colony based Activity Scheduling

```
1: for all edge, ant do
2:    $I^1(P_{ij}, P_{(i+1)l}) = I_{initial}$;
3:    Place ant on a randomly choose POI of $S_1$ ;
4: end for
5: Let $Path_{minDelay}$ be the best path and $L_{min}$ its delay;
6: for $t = 1$ to $t_{max}$ do
7:    for $k = 1$ to $m$ do
8:       Build tour $Path^t_k$ by applying n-1 times the following step;
9:       Choose next POI with probability computed by Formula (1);
10:   end for
11:   for $k = 1$ to $m$ do
12:      Compute the delay $L^t_k$ of $Path^t_k$ produced by ant k;
13:   end for
14:   if an faster path found then
15:      Update the $Path_{minDelay}$ and $L_{min}$;
16:   end if
17:   for all edge do
18:      Update the $I^{(t+1)}(P_{ij}, P_{(i+1)l})$ by Formula (3);
19:      Replace ant on a randomly choose POI of $S_1$ ;
20:   end for
21: end for
22: Return the $Path_{minDelay}$ and $L_{min}$;
```

5. DEMO SYSTEM AND EVALUATION

Foursquare is the largest location based social network, which has more than 20 million active users. We have built an experiment system based on Foursquare and T-drive [12] dataset for Beijing. Information about the experiment platform and dataset is listed as below:

- System configuration: Our experiments are conducted on a quad-core 2.27GHz Intel i3 CPU with 2G RAM and a 320G 5400 RPM disk driver. Disk page size is 4KB (4096Byte). OS is Ubuntu 11.04 (Linux 2.6.16).

- Public transportation stops data: We collected 10684 bus stops from the Beijing public transport network, which covers all the urban and suburb area.

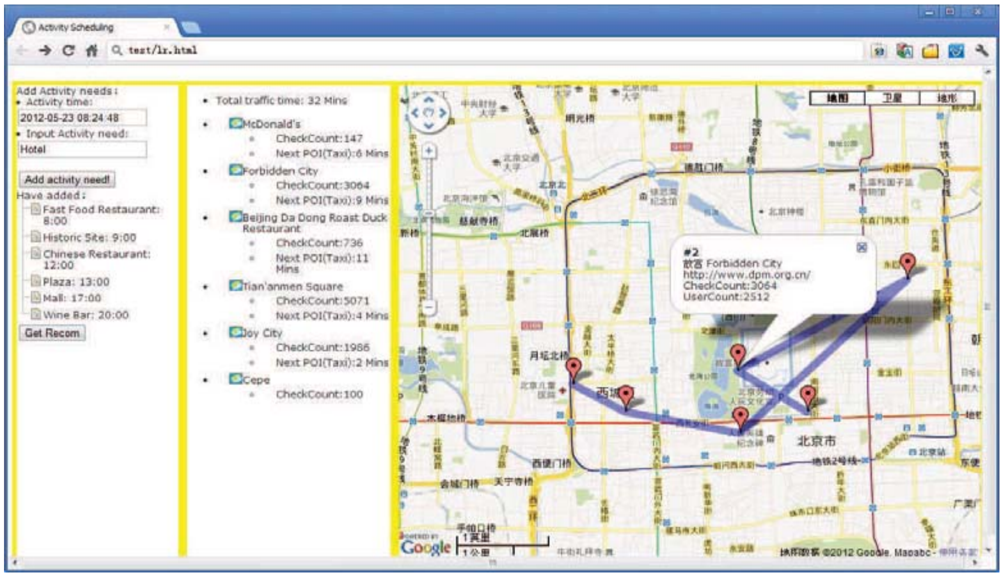
- Foursquare dataset: We collected 30,784 effective POIs and category information by calling open API provided by Foursquare Company. Every POI has a variety of information, such as total "sign in" times, number of historical visitors, user submitted reviews, website address and so on. In order to find all the candidate POIs the belongs to the same categorie, we built Inverted index [7] for every category and sort each POI list by "sign in" times.

- GPS trajectory dataset: Real dataset is provided by the T-drive project of Microsoft Research Asia, which includes all the trajectories of 10357 Taxis from 2008-02-02 to 2008-02-08 in Beijing.

Table 1 lists the recommendation results, computed by the baseline algorithm and modified ACA with k=3 and k=6 respectively, for example 1. The user interface is as shown in

Table 1: Itinerary recommendations for Example 1 in Beijing City

Algorithms	Activity scheduling	Time	Evaluation
k=3 (Baseline)	McDonald's(143)− >(6 Mins)Forbidden City(2914)− >(9 Mins) Beijing Roast Duck Restaurant (720) − >(5 Mins)Jianwai SOHO(277) − >(4 Mins)Sanlitun Village(5688) − >(5 Mins)CJW The Place(167)	25 Mins	8.1
k=6 (Baseline)	McDonald's(143)− >(3 Mins)East Gate: Temple of Heaven(193)− >(3 Mins) Duck de Chine(287) − >(3 Mins)Jianwai SOHO(277) − >(2 Mins)Silk Street Market(1017)− >(2 Mins)CJW The Place(167)	11 Mins	7.4
k=3 (Ant colony)	McDonald's(143)− >(6 Mins)Forbidden City(2914)− >(9 Mins) Beijing Roast Duck Restaurant(720) − >(11 Mins)Tian'anmen Square(4908) − >(4 Mins)Joy City(1933)− >(2 Mins)Cepe(94)	32 Mins	8.9
k=6 (Ant colony)	McDonald's(143)− >(6 Mins)Forbidden City(2914)− >(9 Mins) Beijing Roast Duck Restaurant(720) − >(5 Mins)Jianwai SOHO(277) − >(0 Mins)The Place(1798)− >(5 Mins)Enoterra(338)	25 Mins	8.2



Add Activity needs:
- Activity time:
2012-05-23 08:24:48
- Input Activity need:
Hotel

Add activity need!

Have added:
- Fast Food Restaurant: 8:00
- Historic Site: 9:00
- Chinese Restaurant: 12:00
- Plaza: 13:00
- Mall: 17:00
- Wine Bar: 20:00

Get Recom

- Total traffic time: 32 Mins
- McDonald's
  - CheckCount:147
  - Next POI(Taxi):6 Mins
- Forbidden City
  - CheckCount:3064
  - Next POI(Taxi):9 Mins
- Beijing Da Dong Roast Duck Restaurant
  - CheckCount:736
  - Next POI(Taxi):11 Mins
- Tian'anmen Square
  - CheckCount:5071
  - Next POI(Taxi):4 Mins
- Joy City
  - CheckCount:1986
  - Next POI(Taxi):2 Mins
- Cepe
  - CheckCount:100

#2
故宫 Forbidden City
http://www.dpm.org.cn/
CheckCount:3064
UserCount:2512

Figure 5: User interface.

Fig. 5. The recommendation result includes the location of each activity and the popularity indicated by "sign in" number (shown after POI name). What's more, users can click the link to browse homepage of every POI and information on foursquare website, which contains reviews and evaluation of other users. At the same time, recommendation results are drawn in GIS map and the shortest driving time between two activities has been given (the number in brackets, in minutes). From the above information, user can get a full understanding of the recommended itinerary.

The evaluation value in table 1 is the average value of all the evaluations (a value within 1-10 is used to indicate their degree of satisfaction) made by the 14 volunteers belong to class A and B in the questionnaire. The satisfaction degree of all the four recommendations are above 7, indicating that the results have good user acceptance and important reference value. At the same time, we find that recommended results of ant colony algorithm, though with larger time cost, are better accepted by users than the baseline algorithm. Something interesting is that, when there are more candidate locations (larger k) taken for recommending, the satisfaction degree decreases. It shows that, compares to several minutes' additional cost, the users care more about the popularity. Next step, we will provide online services, which is more sensitive to the recommendation algorithm's performance. In the next subsection, the time cost of the two algorithms are evaluated.

6. RELATED WORKS

This section lists the related studies related to our work and points out their difference with our work:

- GeoLife project of Microsoft Research Asia collects 165 users' trajectory data from 2008 to 2010. They have developed many interesting applications, such as the traffic navigation system [13], and studied the relations between frequently visited locations and the users. Their discovery can answer questions such as "What's the most popular tourist attractions of Beijing city?", "where a tourist, who has gone to the Imperial Palace, will be?", "which is the most frequent route to a certain place?" and so on[15]. They also provides smart itinerary recommendation services,which use simplified query composed of start point, end point and duration to get a complete set of itinerary is automatically generated based on real user-generated GPS trajectories[10]. At the same time,they also implements a personalized recommendation that provides an individual with locations matching her travel preferences[14].However,they have not provided a intention oriented itinerary recommendation service.

- The research of [5]tries to find out trajectory pattern and analyze the life style of people. The work associates each of the social network users with a specific geographic location to build a spatial social network graph. Their main contribution is that they proposed a serial of operators to the query social networks and spatial networks together, and have implemented them based on both relational and graph databases. The work has inspired us to combine spatial and social networks for exploring new application and technology.

- The classic scene of spatial keyword query is to give a physical location and a set of keywords to find a single object which best matches the input keywords with minimal distance [1, 6]. The research of [2] expands spatial keyword search, proposes and realizes the search for a set of objects, which together matches the input keywords with minimum space distance among all the objects. The applications' scenarios of this kind of query are restricted for not fully using all kinds of background knowledge from other information sources. Our work can use collective intelligence to find out more useful locations and give a reasonable travel path for the recommended itinerary.

- The work of [8] supports the trajectory retrieval using k query points. This work requires the user to input k accurate location points, and then finds out trajectories pass through or is nearest to all the input points. However, it doesn't support category based queries and can not guarantee that the result trajectory is shortest or with minimum travel time.

7. CONCLUSION

By jointing the data from social network with physical trajectories, and take usage of hierarchical categories of geographical locations, this paper realizes recommendation for itinerary planning. Based on the sematic traffic information contained in the historical trajectories, the recommendation algorithm gives the best path along multiple points, which satisfies the user's demands better than the shortest path searching algorithm. Meanwhile, the recommendations are presented with related reviews about the recommended locations on social network, effectively help the user to understand the final recommendations. Therefore, the recommending results are meaningful for the users to plan their itinerary.

8. ACKNOWLEDGMENTS

Thank Dr. Xing Xie from Microsoft Research Asia (MSRA) who gives me a lot of guidance and his team, who provides me with the Taxi trajectory data of Beijing.

9. REFERENCES

[1] X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestige-based relevant spatial web objects. *PVLDB*, 3(1):373–384, 2010.

[2] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In *Proceedings of ACM International Conference on Management of Data, (Athens, Greece, June 12-16,2011)*, pages 373–384, 2011.

[3] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications(Third Edition)*. Springer-Verlag., Heidelberg, 2008.

[4] M. Dorigo and L. M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evolutionary Computation*, 1(1):53–66, 1997.

[5] Y. Doytsher, B. Galon, and Y. Kanza. Querying geo-social data by bridging spatial networks and social networks. In *GIS-LBSN*, pages 39–46, 2010.

[6] I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE*, pages 656–665, 2008.

[7] M. Patil, S. V. Thankachan, R. Shah, W.-K. Hon, J. S. Vitter, and S. Chandrasekaran. Inverted indexes for phrases and strings. In *SIGIR*, pages 555–564, 2011.

[8] L. A. Tang, Y. Zheng, X. Xie, J. Yuan, X. Yu, and J. Han. Retrieving k-nearest neighboring trajectories by a set of point locations. In *SSTD*, pages 223–241, 2011.

[9] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer. Semitri: a framework for semantic annotation of heterogeneous trajectories. In *Proceedings of the 14th International Conference on Extending Database Technology, (Uppsala, Sweden, March 21-24, 2011)*, pages 259–270, 2011.

[10] H. Yoon, Y. Zheng, X. Xie, and W. Woo. Smart itinerary recommendation based on user-generated gps trajectories. In *UIC*, pages 19–34, 2010.

[11] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (San Diego, CA, USA, August 21-24, 2011)*, pages 316–324, 2011.

[12] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *ACM-GIS, (San Jose, CA, USA,November 3-5, 2010)*, pages 99–108, 2010.

[13] Y. Zheng, Y. Chen, X. Xie, and W.-Y. Ma. Geolife2.0: A location-based social networking service. In *Mobile Data Management*, pages 357–358, 2009.

[14] Y. Zheng and X. Xie. Learning travel recommendations from user-generated gps traces. *ACM Transactions on Intelligent Systems and Technology*, 2(1):2, 2011.

[15] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from gps trajectories. In *WWW*, pages 791–800, 2009.

Mining Regular Routes from GPS Data for Ridesharing Recommendations

Wen He[1,2], Deyi Li[1,3], Tianlei Zhang[1], Lifeng An[1], Mu Guo[1], Guisheng Chen[3]
[1] Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2] Xi'an Communication Institute, Xi'an, China
[3]Chinese Institute of Electronic System Engineering, Beijing, China
he-w09@mails.tsinghua.edu.cn, lidy@cae.cn, { ztl2004,anlifeng,guom08 }@gmail.com

ABSTRACT

The widely use of GPS-enabled devices has provided us amount of trajectories related to individuals' activities. This gives us an opportunity to learn more about the users' daily lives and offer optimized suggestions to improve people's trip styles. In this paper, we mine regular routes from a users' historical trajectory dataset, and provide ridesharing recommendations to a group of users who share similar routes. Here, regular route means a complete route where a user may frequently pass through approximately in the same time of day. In this paper, we first divide users' GPS data into individual routes, and a group of routes which occurred in a similar time of day are grouped together by a sliding time window. A frequency-based regular route mining algorithm is proposed, which is robust to slight disturbances in trajectory data. A feature of Fixed Stop Rate (*FSR*) is used to distinguish the different types of transport modes. Finally, based on the mined regular routes and transport modes, a grid-based route table is constructed for a quick ride matching. We evaluate our method using a large GPS dataset collected by 178 users over a period of four years. The experiment results demonstrate that the proposed method can effectively extract the regular routes and generate rideshare plan among users. This work may help ridesharing to become more efficient and convenient.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – *data mining.* H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Clustering, retrieval model.* J.4 **[Social and Behavioral Sciences]***:* Sociology.

General Terms

Algorithms, Experimentation.

Keywords

GPS mining, regular route, ridesharing, frequency-based mining, grid-based route table.

1. INTRODUCTION

Nowadays, traffic congestion has become a worldwide problem especially in the metropolitan areas. Lots of measures have been adopted to relieve this problem, such as improving traffic signal timing methods, adding road lanes, or constructing new streets. However, few of these measures worked well under the deterioration caused by the massive increase of cars. In Beijing, China, where the traffic condition is considered to be one of the worst in the world, the government has to require private cars to keep off the road for one day a week, and to restrict car purchases to combat the serious traffic problems. But the traffic condition is still not satisfied especially during rush hours. The main reasons are largely because the large number of vehicles on the roads. However, if you look closer at traffic, you may easy to find that too many people drive long distances alone every day. And there is no doubt that many of them are heading the same way. Thus, effectively use of empty car seats by ridesharing is definitely a quick and effective way to reduce number of vehicles on the road, thus improving traffic conditions and reducing green house gas emissions.

In recent years, although lots of websites and projects[1] have attempted to promote ridesharing, successful ridesharing systems are still in short supplies [1]. A rideshare system can be widely accepted only if it is easy, safe, flexible, and efficient. In a typical rideshare system, drivers will first issue their trips in advance, and riders need to search if there are any routes that match his/her request. Most of the time, riders need to do several searches to find a satisfied match both in timing and in route. With the unfamiliar individual who will share a trip with us, we may actually be worried about the "stranger danger" and also the journey reliability. The increasing complexity of work and social schedules and the related increase in vehicle trip complexity, is assumed to have made ridesharing less desirable [2].

On the other hand, with the prevalence of GPS-enabled devices, more and more people are likely to record their daily trajectory logs and to share them with others. These logs contain not only their life experiences but also their life modes. This provides us an opportunity to learn more about the users' daily lives and offer optimized suggestions to improve people's trip styles. For example, we could find out the daily commute route of each user, and provide ridesharing recommendations to a group of people who have similar routes. In this way, ridesharing of regular routes can be implemented automatically. Furthermore, with days of

[1] http://www.liftsurfer.com/, http://www.rideshareonline.com/

trajectory records, the reliability of the route and route provider can also be qualified.

However, there are still several challenges to realize the idea proposed above. Firstly, a regular route (*RR*) is not a frequent or a personal route [3][4]. A frequent route is consisted of route segments where a user frequently passes through. But *an RR* is a complete route where user frequently passes through in similar times of day. Thus, *an RR* is a part of a frequent route, but not all the frequent routes are *RRs*. Secondly, users don't always start *an RR* at the same time each day. The variation of the start time may be quite different for different users. Thirdly, because of uncertainty factors such as traffic conditions or traffic signals, the durations of *an RR* are also different on different days. Users may reach a same region at different times, even while having the same starting time. Fourthly, *an RR* may also have different trajectory sequences on different days. This happens probably due to a GPS signal drift or obstacles on the road *etc.* Finally, *an RR* may consist of several different transportation modes, like bus->walking->bus, or car->walking->subway. There is no explicit boundary to divide a trajectory into each individual route. We should not give a recommendation to users who all travel by public transportation.

In this paper, we aim to mine regular routes from a user's historical GPS trajectories, and made ridesharing recommendations according to a group of users' regular routes. In our method, we construct user's GPS data into grid-based directed edges, and divide a user's GPS trajectories into each individual route. A group of routes which occur at similar times of day are grouped together. Based on each route cluster, we perform a frequency-based regular routes mining method to infer the *RR*. And the main travel mode of each *RR* is recognized. Finally ridesharing recommendations are made based on the discovered regular routes and the recognized travel modes.

The main contributions of this paper are listed as follows:

- We propose a method to split the mixed user trajectories into each individual route.
- We propose a frequency-based regular routes mining method to infer users' *RRs*, which could significantly distinguish the *RRs* from frequent or personalized routes.
- We identify a new feature to improve the accuracy in distinguishing travel modes between public transports (bus and railway) and private driving (taxi and private car).
- We evaluate our method using a large GPS dataset which is provided by GeoLife [5][6][7]. This dataset contains 178 realistic user GPS trajectories over a period of four years.

2. RELATED WORKS

2.1 Ridesharing Recommendation

In the past few years, some methods have been proposed to provide effective and efficient route matching between drivers and riders[8]. In [9], a location-based cab-sharing service was proposed to help reduce cab fare costs and effectively utilize available cabs. A comprehensive consideration of users' preference when creating a new match was proposed in [10]. In paper [11], an approach that assigns users to form ridesharing groups according to their routes and fees was proposed. In their work, authors tried to improve the quality of ridesharing by increasing the driver's income. In [12], an optimized route for multiple riders was proposed based on the Bee Colony Optimization Metaheuristic method. However, most of these works are based on groups of given routes. Users need to manually arrange their routes.

2.2 Mining Route History

In [13], a method was proposed to mine long and sharable route from vehicles' GPS data. There are several differences between their work and ours. First, we do not only mine the sharable routes from data generated by cars, but all trajectory logs from users. This means we give recommendations not only to driver users, but also to users who take public transport as one of their common travel modes. Second, our sharable routes are not directly generated based on one day's trajectory log. *RRs* are firstly mined which are more steady and reliable for a ridesharing. Finally, compared to a strictly time alignment in regions, we give a more flexible time interval because in real traffic, it's difficult to find two cars that are always keep synchronized, even they started at the same time and were running on the same road. They may pass the next road point at a different time due to the complexity of traffic condition.

Some other similar works are trying to mine various interesting knowledge from users' historical GPS data, such as transportation modes [6], interesting locations and classical travel sequences [5], and a faster way for driving [14]. Chen *et al.* proposed a method to mine personal routes from GPS data [3], and to predict the destination and future route. The difference between a personal route and a regular route is that, a personal route does not consider the time factor, and is not a complete route.

3. ARCHITECTURE

Figure 1 shows the architecture of our system, which is consisted of three components: Routes Processing, Regular Routes Mining, and Ridesharing Recommendation. The first two components are processed based on each user's trajectory logs, and the third is running on the database of extracted regular routes. The user-based components only need to be preformed once while a user submitting his/her logs to the system.

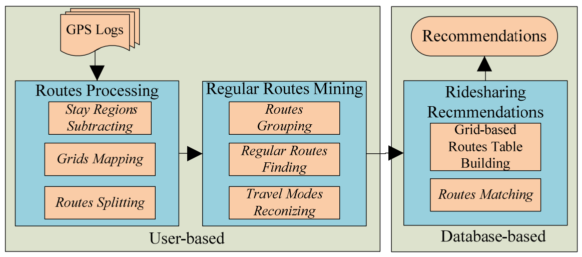


Figure 1. Architecture

Routes Processing: This component processes users' GPS logs into individual routes. The main steps contain: 1) *Stay Regions Subtracting*: A stay regions is definitely not a part of a regular route. And if there is a stay region within a successive GPS log, two routes can be extracted before and after the stay regions respectively. Thus, we first need to find and subtract these regions from the original GPS logs. 2) *Grids Mapping:* Instead of directly mapping points into geographical grids, we combine the time information with grids, and a series of temporal grids is built. 3) *Routes Splitting*: Finally we segment the trajectory into each individual route and pass into *RRs* mining components.

Regular Routes Mining: This component responsible for mining *RRs* from users' route sets. There are also three steps. 1) *Routes*

Grouping: We first group the routes which happened at similar times of a day together. 2) *Regular Routes Finding*: Then *RR*s are mined from each route set. For an *RR*, it must have been passed through by a number of routes which happened at similar times as each other. We call these routes as support routes. For a support route, most parts of it are also frequently traverses by users. Based on this thought, a frequency-based regular routes mining algorithm is proposed. 3) *Travel Modes Recognizing:* A feature of fixed stop rate (*FSR*) is used to recognize the different travel modes of an *RR*. We notice that, a public transport, not only stops frequently, but also stops regularly at fixed positions. Therefore, having obtained a series of support routes of each *RR*, we could compare the stop rate at fixed regions. Experiment results show that, *FSR* could achieve a better performance than existing stop rate feature only. Finally time properties are added to each *RR* for future use.

Ridesharing Recommendations: In this component, two steps are made to recommend ridesharing among similar *RRs*.1) *Grid-based Routes Table Building*：We first construct a grid-based route table. In this table, each record contains a grid identifier and regular routes identifiers which passes through this grid. For a regular route generated by public transport, we only record it at the starting and ending grids. Otherwise, we record each part of the route on the table. 2）*Routes Matching*: With the grid-based routes table, we only need to search two routes which appeared in pairs and also have similar time properties.

4. THE MINING OF REGULAR ROUTES

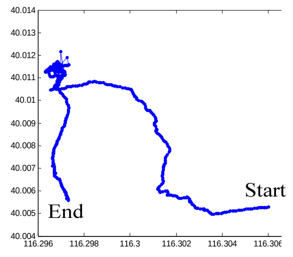
4.1 Routes Processing

The raw trajectories uploaded from GPS devices may contain only one route in a short period of time or a whole log during a day. In order to discover the *RR*s of a user, we first need to segment these data into individual routes. There are two cases that a sequence of GPS points should be split: 1) the time that a user stayed in a region exceeding a threshold of ST_{threh}; 2) the time gap between two temporally adjacent GPS points is longer than a threshold of GT_{threh}. Where ST_{threh} and GT_{threh} are two thresholds defined by user. In the first case, user may arrive at his destination, and when he left, a new route will begin. The second case is mostly because the GPS device was shut down or lost satellite signal over a certain time.

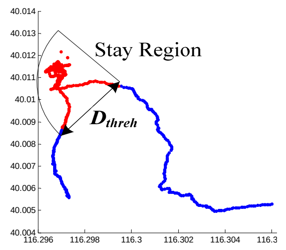
STEP 1: Stay Region Subtracting

Given a raw trajectory $T\{P_1,P_2,...P_k\}$ from GPS device (where each point is formed as $P(lat, lngt, t)$, here *lat, lngt* and *t* are the latitude, longitude and timestamp of a point respectively), we first extract stay regions based on the movement range within a certain time, which is similar to the work of Yu Z. *et al.* [5]. One difference is that, we do not denote this region by a single point, but by a pair of indicators (P_m, P_n), where P_m and P_n are the beginning and ending points of the stay region. And then, points between P_m and P_n will be deleted from the trajectory. The new trajectory *P* after stay regions subtracting becomes $T\{P_1, P_2,... P_m, P_{n+1},..., P_k\}$. This means we simply use P_m instead of the whole stay region. We could do this because that we do not care about the details within a stay region (different of the work by Yu Z., in which stay region is the interesting region), but the routes to enter into or depart from stay regions. Well then, P_m is just the ending point of the route which enters into a stay region, and P_{n+1} is the starting point when user departs from the stay region.

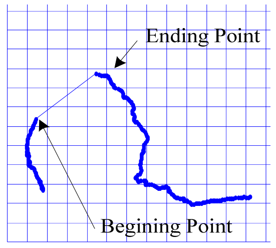
Figure 2(a) shows a GPS trajectory from user data. And a stay region is denoted in Figure 2(b). In Figure 2(c), the result after stay region subtracting is shown.



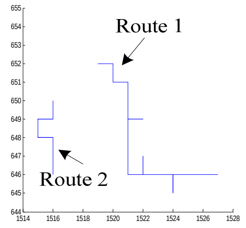
(a) A sample Trajectory Sequence



(b) Stay Region Finding



(c) Trajectory sequence after stay region subtracting



(d) the two routes after routes spilitting

Figure 2. An example of routes processing

STEP 2: Grids Mapping

To reduce the huge amount of data, trajectory points are then mapped into equally distributed geographical grids (see Figure 2(c)) as $\{(g_1, t_1), (g_k, t_k)....(g_k,t_k)\}$, where g_i is the grid identifier. A set of consecutive points which mapped into the same grid can be grouped together as a temporal gird.

Definition 1. (*Temporal Grid, TG*): A temporal grid $TG(g, t_a, t_l)$ stands for a geographical grid g where a user entered at time t_a and left at time t_l. The spatial range of the gird is fixed beforehand. If there is only one point $(lat_i, lngt_i, t_i)$ projected into grid g, we set $t_a = t_l = t_i$.

STEP 3: Routes Splitting

Since stay regions have been detected and subtracted from trajectories, we need only to consider the time gap between two adjacent temporal grids when extracting individual routes.

Definition 2. (*Route*): A Route is a sequence of temporal grids which denoted as $R\{TG_1,TG_2,...,TG_n\}$, where $\forall$ $1 \le i < n-1$, $TG_{i+1}.t_a - TG_i.tl < GT_{threh}$. The start and end time of a route *R* is represented as *R.st* and *R.et*, obviously $R.st = TG_1.ta$ and $R.et = TG_n.t_l$. Figure 2(*d*) shows the result after grids mapping and routes splitting.

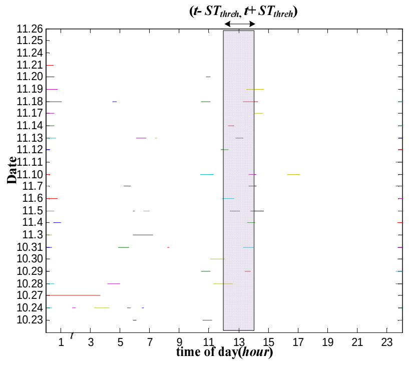
4.2 Regular Routes Mining

Definition 3. (*Regular Route, RR*): An *RR* is a complete route where a user frequently passed through in approximately the same time of day. There are two parameters when determining an *RR*. The first is F_{threh}, which is used to decide the frequency of a route, and the second is $SimT_{threh}$, which is used to decide a similar time.

STEP 4: Routes Grouping

A user may generate lots of routes on a single day. The sheer number of these routes is enormous. Although an *RR* should happen usually, the number is still small compared to the total number. Therefore it's difficult to extract *RRs* from all routes directly. But an *RR* should always happen at a similar time of day.

Two routes which occurring separately at 11:00 AM and 3:00 PM should not be viewed as an *RR*, even though they have the same route. Therefore, a sliding time window of fixed duration (*t-* $SimT_{threh}$, *t*+ $SimT_{threh}$) is used to cluster routes by their occurrence time. The time window moves along time axis with a step of $SimT_{threh}$, routes dropped in the window will be grouped together as a set of *t-Routes* $tR\{R_m,\ldots R_n\}$, where $(R_i.st,\ R.et)\cap(t-\ SimT_{thre},\ t+\ SimT_{thre})\neq\Phi,\ R_i\in tR$. The *t* here represents the center of the time window. Since most of the *RRs* happen during weekdays, such as going to work or sending kids to school. And in weekends, there may be other kinds of *RRs* such as going to the supermarket and so on. Thus, we group routes not only based on the time of day but also the day of the week. Figure 3 shows the distributions of a user's travel time during weekdays between 20/11/2008 to 20/12/2008 where each line represents a route and the *x*-axis and y-axis represent the occurrence time and occurrence date, respectively. Groups of *t-Routes* with a $SimT_{threh}$=60 *min* are shown in table beside it.



t	*Total number*	*t-Routes (t-60min,t+60min)*
1	18	3, 4, 8, 19, 22, 24, 28, 35, 37, 42, 44, 47, 56, 58, 62, 64, 76, 78
2	2	4, 8
3	2	5, 8
4	5	5, 8, 17, 25, 73
5	8	1, 5, 6, 17, 25, 38, 45, 73
6	9	1, 6, 7, 25, 29, 38, 39, 45, 59
7	6	1, 7, 29, 39, 59, 60
8	2	26, 60
9	1	29
10	5	2, 20, 48, 74, 79
11	8	2, 18, 20, 23, 48, 57, 74, 79
12	10	2, 18, 23, 40, 43, 28, 57, 61, 63, 79
13	14	18, 21, 27, 36, 40, 41, 43, 46, 49, 57, 61, 63, 75, 77
14	7	21, 27, 36, 41, 46, 49, 61, 65, 75, 77
15	6	41, 46, 49, 65, 75, 77
16	5	42、58、133、145、153
18	0	
…	…	…

Figure 3. A case of routes grouping

STEP 5: Regular routes Finding

Definition 4. (*Directed Edge*, *DE*): A directed edge is a link of *TG*, denoted as *DE*($TG_m\rightarrow TG_n$). The velocity on a *DE* is defined as:

$$DE.v=\frac{Distance(TG_m,TG_n)}{TG_n.t_l-TG_m.t_l} \tag{1}$$

Given a route *R* $\{TG_1,TG_2,\ldots,TG_n\}$, there are *n*-1 directed edges $\{DE_1\ (TG_1\rightarrow TG_2),\ DE_2(TG_2\rightarrow TG_3),\ldots,\ DE_n(TG_{n-1}\rightarrow TG_n)\}$. The velocity of the route is separated into $\{DE_1(R).v,\ DE_2(R).v,\ldots DE_n(R).v\}$. We denote route *R* as a support route of these *DEs* and record as $DE_i.sup=\{R\}$.

Definition 5. (*Frequent Directed Edge, FDE*): In a set of *t-Routes*, the frequency of a *DE* is denoted as *DE.num*. Therefore, a *DE* will have *DE.num* support routes like $DE.sup=\{R_1R_2,\ldots R_{DE.num}\}$. We say a *DE* is a *FDE* if *DE.num* is larger than threshold of f_{threh}.

Definitions in a route are represented in Figure4. There are three trajectories in Figure4 (a). After grids mapping, the trajectories are formed as R_1, R_2 and R_3 in Figure4 (b). Each arrow in R_i is a *DE*. DE_m(see Figure) is one of the *DEs*. There are 2 support routes of DE_m, which are R_1 and R_3. The velocities of R_1 and R_3 passes DE_m are denoted as $DE_m(R_1).v$ and $DE_m(R_3).v$, respectively.

As is described above, an *RR* is a route which is frequently visited by a couple of complete routes, but not some parts of a route. This means we should not directly use *FDEs* to represent an RR. In a set of *t-Routes*, *FDEs* may exist without *an RR.* But if there is *an RR,* the *RR* will have large common parts with *FDEs*. Figure 5 shows two groups of *t-Routes*, each group of *t-Routes* is consisted of five routes. The f_{threh} is set to 2, and *FDEs* are denoted by red broad lines. In Figure 5(a), there are 3 routes, which have a lot of common *FDEs,* accordingly, there will be an *RR*. But in Figure 5(b), although there is no *RR*, there are still some *FDEs*.

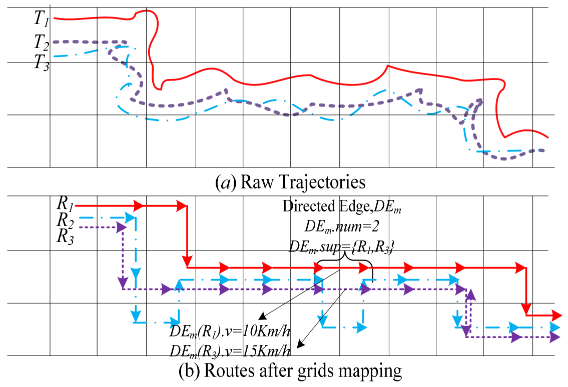


Figure 4. Definitions in a route

To find specific *RRs*, a frequency-based regular route mining method is proposed as following:

1. *Calculate frequent coefficient* (*FC*) *of each route*

The frequent coefficient is defined as $f_c(R)=m/n$, where *n* is the number of *DEs* in the route *R*, *m* is the number of *FDEs* in the route *R*. The frequent coefficient can reflect the integrated frequency of a route. The higher the *FC* the more parts of the route are frequently traversed by the user.

2. *Find frequent routes*

A route with $f_c(R)>fc_{threh}$ will be deemed as a frequent route. If we set fc_{threh} to 0.8, there are 3 frequent routes in Figure5 (a), which are R_1,R_2 and R_3.

3. *Calculate regular coefficient (RC) of each FDE*

The regular coefficient of a *FDE* is defined as

$$DE.rc=\sum_{i=1}^{DE.num}(f_c(DE.\sup_i)>fc_{threh}) \tag{2}$$

The regular coefficient is used to measure how many frequent routes had visited the *FDE*. If a *FDE* is a part of a regular route, it must be visited by a lot of frequent routes, not just some usual routes.

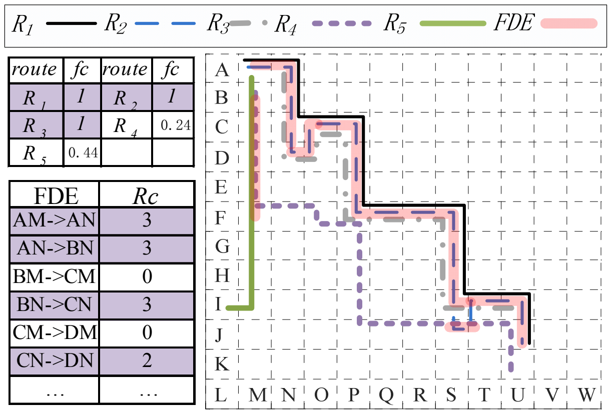
4. *Find Regular FDEs* (*RFDE*)

Then a *FDE* has $DE.rc>F_{threh}$ is extracted as an *RFDE,* and an *RR* is the collection of *RFDEs*, which is denoted as $RR\{DE_{m,\ldots,}DE_n\}$, where $DE_i\in RFDEs,\ i=m,\ldots n.$ If we set F_{threh} to 2, the *RRs* in Figure 5(a) is $RR\{AM\rightarrow AN,\ BN\rightarrow CN,\ CN\rightarrow DN,\ldots,\ IU\rightarrow JU\}$ which are colored in table of *FDEs*. We call all the frequent routes passing through the *RR* as the support routes of the *RR*, and denoted as $RR.sup=\{R_1,R_2,..R_n\}$. The support routes of the *RR* in Figure 5(a) are R_1, R_2 and R_3.

5. *Use RFDEs instead of FDEs to repeat step 2 to 4.*

To improve the precision, we could repeat the steps from 2 to 4, and use *RFDEs* instead of *FDEs* in step 2. This could help filtering the wrong support routes of an *RR,* which are just frequently visited by *FDEs* but not *RFDEs*.

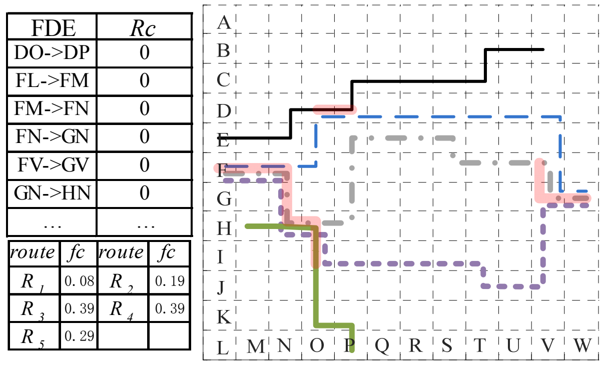
With the method proposed above, we are able to ensure that an *RR* has been at least visited by F_{threh} frequent routes. And fc_{threh} of a frequent route has been visited multiple times than f_{threh}. A common route may also pass through some *FDEs*, but it will make no contribution to an *RR*. Just like in Figure5 (a), both R_2 and R_4 passed the *DE* (JS->JT), but since R_4 is not a frequent route, it has no contribution to an *RR*, *DE*(JS->JT) cannot be an *RFDE*.



route	fc	route	fc
R_1	1	R_2	1
R_3	1	R_4	0. 24
R_5	0. 44		

FDE	*Rc*
AM->AN	3
AN->BN	3
BM->CM	0
BN->CN	3
CM->DM	0
CN->DN	2
…	…

(a) An *RR* is mined from the group of *t-Routes*



FDE	*Rc*
DO->DP	0
FL->FM	0
FM->FN	0
FN->GN	0
FV->GV	0
GN->HN	0
…	…

route	fc	route	fc
R_1	0. 08	R_2	0. 19
R_3	0. 39	R_4	0. 39
R_5	0. 29		

(b) NO *RR* is mined

Figure 5. Two cases of mining regular routes

Once we have extracted *RRs* from historical trajectories, we add time property (*ts*, *td*) for each *RR*, where *ts* and *td* denote the start and the duration time of the route respectively. Since some support routes of *RRs* are not recorded from the beginning, like R_3 in Figure 5(a), we firstly fill the missing time interval using the average time of the other support routes. Then *ts* and *td* can be calculated as

$$ts = \sum_{i=1}^{n} RR.sup_i.st / n \tag{3}$$

$$td = \sum_{i=1}^{n} (RR.sup_i.et - RR.sup_i.st) / n \tag{4}$$

where *n* is the number of the support routes of an *RR*.

STEP 6: Travel Modes Recoging

Besides the trajectory of each *RR*, we also need to know its travel modes. Since, we should not recommend two users who both went to work by bus to share a car. Although walking exists highly likely between and/or after a bus transfer, with the aim to make a recommendation for ridesharing, we only distinguish the main transport modes of an *RR*, which are public transport and private driving.

In [6], a stop rate is used to distinguish different transport modes. Most of the time, a bus is likely to stop more times than a car. On the other hand, another feature we observe is that public transportation would stop more frequently at fixed regions like bus stops or subway stations. Since we have already discovered the support routes of each *RR*, we could observe the stop rate at fixed regions between support routes.

***Definition 6**. (Stop Probability, SP):* Stop probability is used to measure the likelihood that each user passed an *RFDE* with a velocity below a certain threshold.

For an *RFDE*, there were *DE.rc* frequent routes passed it, we denote these frequent routes as:

$$DE.fr_{=}\{DE.sup_n\} \;\; if\,(fc(DE.sup_n) > fc_{thre}) \;\; n=1,\ldots DE.num$$

Then the stop probability is defined as

$$SP(DE) = \frac{\sum_{i=1}^{DE.rc} ((DE(DE.fr_i).v < V_{thre})}{DE.rc} \tag{5}$$

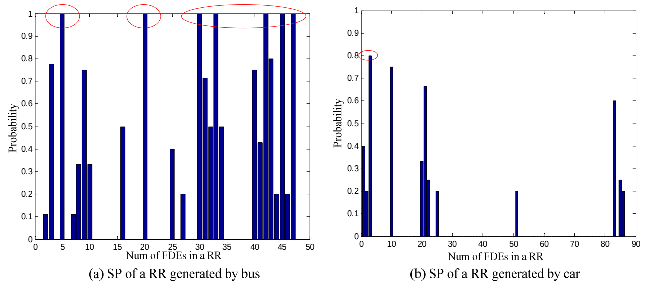
Then an *RFDE* with *SP* lower than P_{stop} is a stop region.

***Definition 7**. (Fixed Stop Rate, FSR):* The Fixed Stop rate of an *RR* is the number of stop regions within a certain distance. We could calculate *FSR* by:

$$FSR = \sum_{i=1}^{n} (SP(DE_i) > P_{stop}) / RR.distance \tag{6}$$

where *n* is the number of *RFDE* in an *RR*.

Figure 6 provides two examples of the stop probability of two regular routes. The travel mode in Figure 6(a) is public transport, and in Figure 6(b) it is driving. We could see clearly that, a bus may always pass some regions with a low velocity. While for a car, it may pass uncertainty regions with a low velocity.



(a) SP of a RR generated by bus (b) SP of a RR generated by car

Figure6. Fixed stop rate

5. RIDESHARING RECOMMENDATIONS

STEP 7: Grid-based Routes Table Building

To accelerate the matching rate between users, a grid-based route table (GRT) is built. In this table, each record contains a grid identifier and regular routes which passes through the gird, like $g_i(RR_{m,\ldots}RR_n)$. Given an *RR,* if it is generated by private driving, it will be recorded in each grid it had passed, but if it is generated by public transportation, it will only be recorded in its origin and destination grids. Table 1 shows an example grid-based route table of the *RR* in Figure 5.

STEP 8: Routes Matching

Ordinarily, there are two kinds of car sharing. The first kind is, one of the users usually goes to work by public transportation, and the other user usually goes to work by private driving. Then the first user can be a rider of the second user. The second kind is, both of the two users usually go to work by car, and both of them can be set as a driver or a rider. Therefore, if a query route is generated by public transport, only routes by driving modes could be recommended.

Table 1. A Gird-Based Route Table

Grid Identifier	***RR***
AM->AN	R$_1$, R_2, R_3
AN->BN	R$_1$, R_2, R_3
...	...

Given a query route, we first map the origin and destination of the *RR* into *GRT*. We use g_o and g_d to represent the origin and destination grids, respectively. Then all the grids within the distance D_{threh} of g_o and g_d will be set as the search regions. We select *RRs* from the *GRT* where the grid identifiers equal to search regions. If there is an *RR,* which both exist in the origin and destination regions, the *RR* will be selected as a candidate route.

The second step is travel mode filtering. If the travel mode of RR_i is public transport, then only the candidate routes with mode of private driving will be extracted.

In the third step, we examine the time property of each candidate route. Only those routes with starting time in the range of $SimT_{thre}$ of the starting time of the query route will be reserved.

Finally, to make a recommendation, we need to sort all the selected routes. There are two kinds of sort methods. The first is to sort by Common Rate (*CR*), which stands for the common extent of two route lines. The *CR* is calculated as:

$$CR(RR_n) = \frac{Distance(RR_n)}{Distance(RR_i)} \tag{7}$$

where RR_n is the candidate route, and RR_i is the query route. The second method is to sort by duration time of each candidate route. A route with less duration will be recommended first. The flowchart of the process is shown in Figure 7.

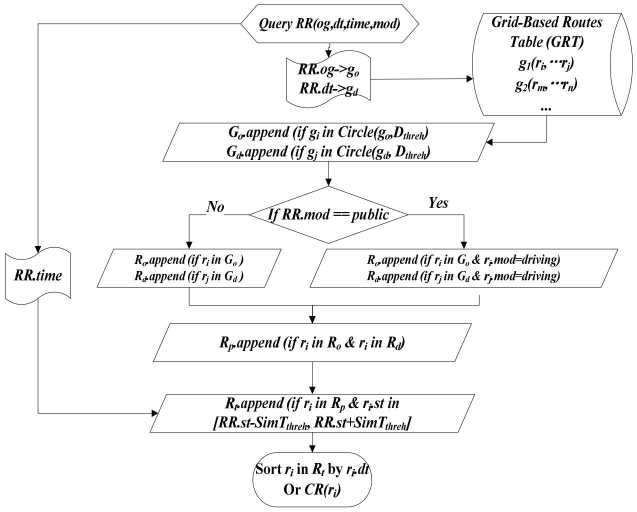

Query RR(og,dt,time,mod)
Grid-Based Routes Table (GRT)
$g_1(r_i, \cdots r_j)$
$g_2(r_m, \cdots r_n)$
...
RR.og->g_o
RR.dt->g_d
G_o.append (if g_i in Circle(g_o,D_{threh})
G_d.append (if g_j in Circle(g_d, D_{threh})
No
If RR.mod == public
Yes
RR.time
R_o.append (if r_i in G_o)
R_d.append (if r_j in G_d)
R_o.append (if r_i in G_o & r_i.mod=driving)
R_d.append (if r_j in G_d & r_j.mod=driving)
R_p.append (if r_i in R_o & r_i in R_d)
R_t.append (if r_i in R_p & r_i.st in [RR.st-$SimT_{threh}$, RR.st+$SimT_{threh}$]
Sort r_i in R_t by r_i.dt Or CR(r_i)

Figure7. Flowchart of finding similar routes

6. EXPERIMENTS DISCUSSION

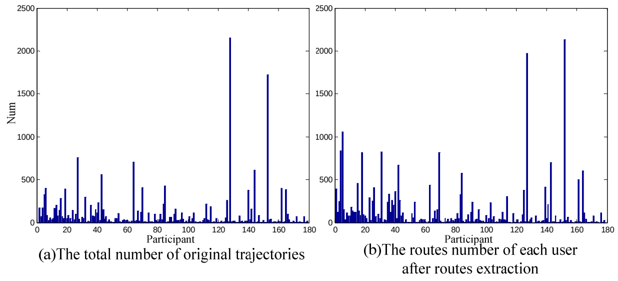
6.1 Testing Data

We test our method based on the GPS trajectory dataset collected by Geolife project. This dataset is consisted of 178 users' realistic trips over a period of 4 years (from 2007 to 2011). More information about this dataset can be found in [15].

We extracted every user's regular routes from this dataset monthly. Some users may have similar *RR*s during consecutive months, but some other users may have different *RRs* in different months. This may happens since a job transfer or the GPS device had changed owner. Most of the time, we see all mined *RRs* as different users' *RRs*, only if we may make a recommendation between a same user. Moreover, to enhance the matching probability, only the occurrence time of an *RR* has been taken into account, without the consideration of the date of the route. This means that we may make a ridesharing recommendation between two *RRs*, one happened in 2007, and the other happened in 2011

6.2 Experiment Result

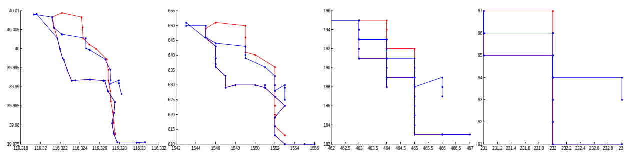
Figure 8(a) illustrates the total number of original trajectories in the dataset. The number of routes after routes processing is shown in Figure 8(b). The threshold ST_{threh} is set to 300*m*, and GT_{threh} is set to 30*min*. Since most of these data are created in Beijing, China, the other data out of Beijing is too few to support a ridesharing and have been filtered. That's why most of the route numbers are bigger than origin, but some of them become smaller after routes processing.



(a)The total number of original trajectories (b)The routes number of each user after routes extraction

Figure8. The number changing after routes extraction

In Figure 9, we compare the influence of the grid size in step 2. Figure 9(a) shows two original trajectories, In Figure9 (b-d), points are mapped into girds of size 3*sec*, 10*sec*, and 20*sec* respectively. Obviously, the smaller the grid size, the larger the storage space is needed (just like Figure8 (b)). And the processing speed will also be affected by the large data in the following steps. But too large a grid size will lose some details of the trajectory (just like in Figure8 (d). The ideal size of a grid should be able to distinguish two different routes, but also as small as possible. In our experiment, we use 10*sec* as final grids size.



(a) original trajectories (b) gird size with 3 *sec* (c) gird size with 10 *sec* (d) gird size with 20 *sec*

Figure9. Comparison with different size of grids

Figure10 illustrates the process of *RR*s mining. Figure 10 (a) is a group of *t-Routes* which is consisted of 9 routes occurred at similar time. Figure 10(b) is the result after grids mapping. The *RR* is extracted in Figure 10(c), and the 3 routes in Figure 10(d) are the support routes of the extracted *RR*. From the results, we could see that, our method is robust to slight disturbances in trajectory data. This is benefited from the *DE*-based matching, which did not need a complete matching on whole routes. In another hand, the method can also distinguish between the different routes, as noted in Figure 10(b).

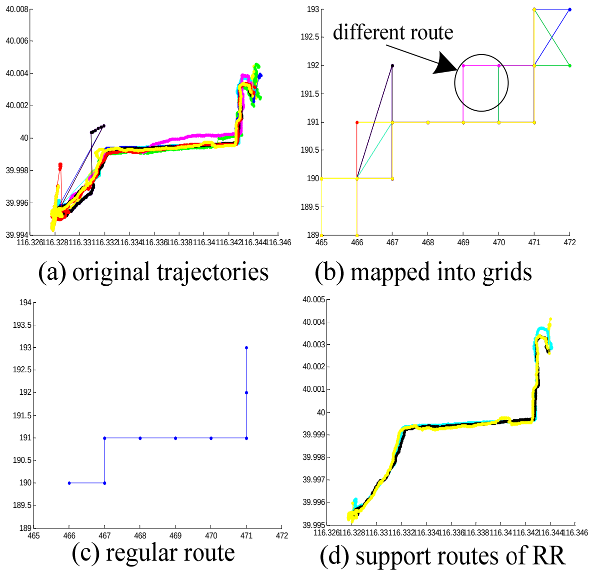


Figure 10. Example of mining *RR*s

In Figure 11, we give another example, where all the trajectories are generated by bus. From Figure11 (a), we could see there are three buses passing the stop stations near starting and ending points. But the three buses had different bus routes, and user may take different buses randomly. Accordingly, we mined two regular routes within the three bus routes. Only from the result, we may think there are two regular routes of the user. But after travel mode mining, we find that, both the two routes were generated by bus (the *FSR* of this example has shown in Figure 6(a)). Thus we only restore the starting and ending points in grid-based route table for future matching. And this means there is only one regular route for the user.

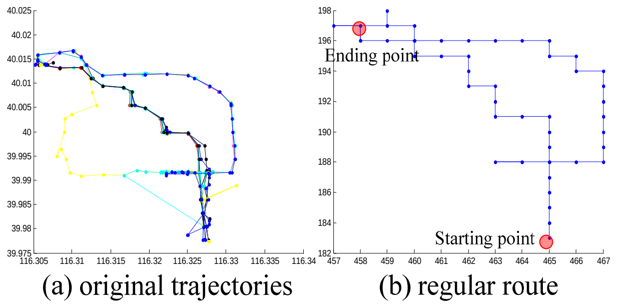


Figure11. Example 2 of mining *RR*s

Figure 12 gives all the mined regular routes when $F_{thre}=3$ and $F_{thre}=8$. There are 389 regular routes when we set F_{thre} to 3, and only 31 routes, when set F_{thre} to 8. Routes short than 2*Km* has been filtered, since they were too short to make a ridesharing. From Figure 12(a) we could see the extracted routes match the main street in Beijing city well, which is consistent with our commonsense. And we could observe that, the regular routes are dense in the north of Beijing, especially around the Zhichun Road, Haidian District (which has been noted in Figure), where is the location of the Microsoft Research Asia. And this is also consistent with the owner of the dataset.

Figure13 shows the effect of using *FSR* to distinguish traffic modes between public transportation and driving. There are only 69 users who have labeled their trajectories with transportation mode in the dataset. Among these users, we mined 89 regular routes. Therefore, we only tested our method on this small subset. From the result we could see that, when we set the threshold V_{thre} *to* 17, the accuracy could reach 0.876, which is quite better than the accuracy of 0.6 which is obtained by only use the feature of *SR* in [5]. Note that, the velocity is a little higher than our common sense of a stop velocity. That's because this velocity is not an instantaneous velocity, but having a different average speed on each *RFDE*.

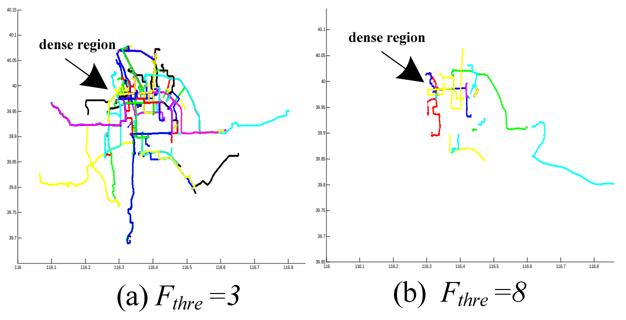


Figure12. Extracted regular routes

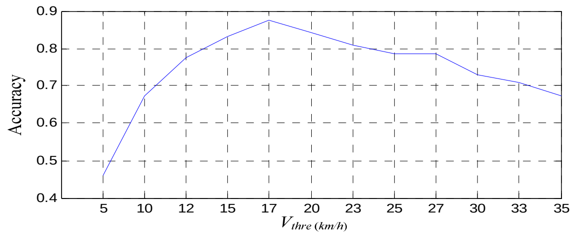


Figure13. Selecting threshold for *FSR*

Figure 14 gives several results on routes matching. The ride requester has been noted on each figure. The travel mode of the two requesters in Figure 14(a) and Figure14 (c) are public transport. Thus the recommend routes are all generated by driving mode. In Figure 14(b) there are three users who have almost same regular routes. Figure 14(d) gives a result, where the travel mode of the requester is driving. And we find 3 riders for the driver. We noticed that, these riders have different origins and destinations. Only one of them will have a whole trip with the driver, and the other two riders will either take on or take off at the middle of the trip. To our surprise, when set $F_{thre}=3$ we successfully find 232 routes among the 389 routes, which could have a match with others. This passed half of the number of the mined regular routes. From this point, we also demonstrate that a significant increase of the road availability could be made, while ridesharing becomes popularized.

In Table 2, we give the storage requirement of the proposed method. The first row is the number of records, and the second row is the storage ration between the numbers of the original dataset. Obviously, the final gird-based route table is a tiny subset of the original dataset. And since regular route mining is independent between users, we don't need to store the data during preliminary steps. The storage requirement of this method is quite lightweight.

7. CONCLUSION

This paper presents an approach to mine regular route from a user's historical GPS trajectories for ridesharing recommendations. In this method, we build GPS data into grid-based directed edges, and divide trajectories into individual routes. A sliding window is used to group routes which occurred at similar time of day. To discover every user's regular routes, a frequency-based regular route mining algorithm is proposed. This algorithm is considered from the following three aspects. Firstly, each part of a regular route must be visited frequently. Secondly, a regular route should be frequently visited by some complete routes called support routes. Finally, most parts of a support route must pass through the frequently visited regions. The other contributions of this paper contain: A new feature is identified to distinguish travel modes between public transportation and individual driving; a grid-based route table is established for a fast ridesharing recommendation. The proposed method is evaluated on a real-world GPS dataset, which is consisted of 178 users over a period of 4 years. The experiment results demonstrated the effectiveness and robustness of the proposed method.

Ridesharing recommendations from GPS trajectories can be seen as a kind of personal optimizing service, which could further motivate users to record and upload GPS data, and may help to improve users experience on ridesharing. But in this work, we only considered the situation, in which driver and rider both pass through a nearby origin and destination region. In fact, there are many other types of ridesharing. For example, for some users, maybe we could not provide a complete matched ride route, but we could find a route which reaches at his/her nearest subway station. More flexible ridesharing strategies will be considered in our future work.

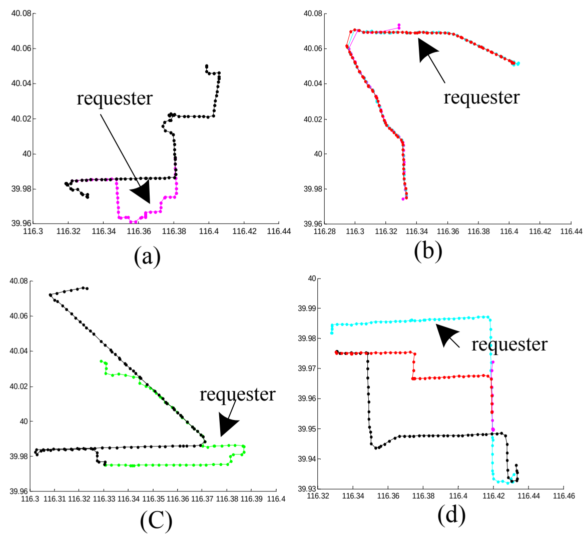


Figure14. Examples in similar routes matching

Table 2. Storage Cost of the Mehod

Orig. points	After stay points delete	After points grouping	After *RR* extraction	Grid-based routes table
23667828	16013562	1050313	21544	3674
1	0.676	0.044	0.0009	5x10^{-8}

8. ACKNOWLEDGMENTS

This paper is supported by the key project of National Science Foundation of China (No, 61034005).

9. REFERENCES

[1] Keivan G., Ali H. and Masoud H. 2011. *Real-Time Rideshare Matching Problem*. Technical Report. University of Maryland at College Park.

[2] Andrew A., Attanucci J., and Rabi M. 2011. Real-Time Ridesharing. T*ransportation Research Record: Journal of the Transportation Research Board* .2217: 103-110.

[3] Chen, L., M. Lv, and Qian Y. 2011.A personal route prediction system based on trajectory data mining. Information Sciences 181(7), 1264-1284.

[4] Chang, K.-P., L.-Y. Wei, M.-Y. Yeh, and W.-C. Peng. 2011. Discovering personalized routes from trajectories. In *Proc. of the 3rd ACM SIGSPATIAL International Workshop on LBSN'11*. Chicago, Illinois, ACM, 33-40.

[5] Zheng Y., Zhang L., Xie X., and Ma, W.-Y. 2009. Mining interesting locations and travel sequences from GPS trajectories. *In Proc. of* WWW'09. Madrid Spain. ACM Press, 791-800.

[6] Zheng Y., Li Q., Chen Y. Xie X. and Ma W.-Y. 2008. *Understanding Mobility Based on GPS Data. In Proc. of UbiComp'08.* Seoul, Korea. ACM Press: 312-321.

[7] Zheng Y., Xie X., and Ma W. 2010. GeoLife: A Collaborative Social Networking Service among User, location and trajectory. Invited paper, in *IEEE Data Engineering Bulletin*. 33, 2, 2010, 32-40.

[8] Ece Kamar and Eric Horvitz.2009. Collaboration and Shared Plans in the Open World: Studies of Ridesharing. IJCAI 2009, 187-194.

[9] Gidofalvi, G. and T. B. Pedersen. 2007. Cab-sharing: An Effective, Door-to-Door, On-Demand Transportation Service. *Proceedings of the 6th European Congress on Intelligent Transport Systems and Services, ERTICO.*

[10] Kammerdiener, T. and H. Zhang. 2011. Classification of ride-sharing partners based on multiple constraints. J. Comput. Sci. Coll. 26(4), 95-101.

[11] C.-W., Cho, Y.-H. Wu, C. Yen, and C.-Y. Chang. 2011. Passenger Search by Spatial Index for Ridesharing. *In TAAI* 2011, 88-93.

[12] Teodorović, D. and M. Dell' Orco. 2008. Mitigating Traffic Congestion: Solving the Ride-Matching Problem by Bee Colony Optimization. Transportation Planning and Technology 31(2): 135-152.

[13] Gidófalvi, G. and T. Pedersen. 2009. Mining Long, Sharable Patterns in Trajectories of Moving Objects. GeoInformatica 13(1): 27-55.

[14] J. Yuan, Y. Zheng, C. Zhang, W. Xie, G. Sun, H. Yan, and X. Xie. 2010. T-drive: Driving directions based on taxi trajectories. Proc.GIS. ACM, 2010.

[15] http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/default.aspx

Efficient distributed computation of human mobility aggregates through User Mobility Profiles

Mirco Nanni, Roberto Trasarti, Giulio Rossetti, Dino Pedreschi
KDD Lab - ISTI CNR
Pisa, Italy
name.surname@isti.cnr.it

ABSTRACT

A basic task of urban mobility management is the real-time monitoring of traffic within key areas of the territory, such as main entrances to the city, important attractors and possible bottlenecks. Some of them are well known areas, while while others can appear, disappear or simply change during the year, or even during the week, due for instance to roadworks, accidents and special events (strikes, demonstrations, concerts, new toll road fares). Especially in the latter cases, it would be useful to have a traffic monitoring system able to dynamically adapt to reference areas specified by the user.

In this paper we propose and study a solution exploiting on-board location devices in private cars mobility, that continuously trace the position of the vehicle and periodically communicate it to a central station. Such vehicles provide a statistical sample of the whole population, and therefore can be used to compute a summary of the traffic conditions for the mobility manager. However, the large mass of information to be transmitted and processed to achieve that might be too much for a real-time monitoring system, the main problem being the systematic communication from each vehicle to a unique, centralized station.

In this work we tackle the problem by adopting the general view of distributed systems for the computation of a global function, consisting in minimizing the amount of information communicated through a careful coordination of the single nodes (vehicles) of the system. Our approach involves the use of predictive models that allow the central station to guess (in most cases and within some given error threshold) the location of the monitored vehicles and then to estimate the density of key areas without communications with the nodes.

1. INTRODUCTION

In the context of urban mobility management, a basic task required by administrators is the monitoring of traffic within a variety of key locations: main gateways to the city, important attractors and possible bottlenecks. Some such locations ar well known, and therefore a monitoring environment can be set up by means of road-side sensors (including cameras), although set up and maintenance costs might be significant for large cities. Other key areas can appear, disappear or simply change with time, due to seasonality or special events. For instance roadworks, accidents or events such a strikes, demonstrations, concerts, new toll-road fares can change the status of the city, and make some areas more critical than usual. In these cases, it would be useful to have a traffic monitoring system able to dynamically adapt to reference areas specified by the user.

A solution can come from recent, growing trends in the deployment of on-board location devices in private cars mobility. Such devices continuously trace the position of the vehicle, and periodically communicate it to a central station, that stores it. Such vehicles provide a statistical sample of the whole population, and therefore can be used to compute a summary of the traffic conditions for the mobility manager. The analytical power of detailed and massive GPS trajectory in unveiling the patterns of human mobility behavior data has been shown in [1]. However, the large mass of information to be transmitted and processed to achieve that might be too much for a real-time monitoring system, the main problem being the continuous communication from each vehicle to a unique, centralized station. In this paper, we use a massive trajectory dataset consisting of approx. 1.5 million travels, sampled at a high rate from more than 40,000 private cars tracked for a month in a 50 square km area in Tuscany, Italy — a dataset which clearly illustrates the computational and economic challenge of continuous transmission to a central server.

Recently, *safe zones* were introduced as a principled mechanism for the efficient distributed computation and monitoring of a global aggregate function, consisting in minimizing the amount of information communicated through a careful coordination of the individual nodes (vehicles, in our domain) [2, 3]. The basic idea is that each node is instructed on how to check locally whether its changes of position can have a relevant impact on the global function, or not. In the negative case, no communication is needed. Of course, that implies a reasonable definition of *relevant impact*, as well as some computational capability at the node level to check it. The safe zone idea, realized through clever computational geometric methods, has the potential of drastically reducing the number of communications between the distributed nodes and the central station, and we checked empirically that this is the case also in our urban mobility setting.

In this paper, we ask the following question: can the amount of needed data transmissions from distributed cars to central station be further reduced by taking into account the regularity of human mobility? We know that the way people move is highly predictable: we tend to follow daily routines, dictated by our social constraints, so that the degree of entropy of our whereabouts very small, as shown by many recent empirical studies on large scale data on human mobility patterns and profiles [4, 5, 6]. Our idea is consequential: if human travel is often systematic and repetitive, we can exploit such regularity to avoid transmitting data whenever we follow our routines, and instead transmit when we are movements are outside our typical behavior. In this sense, our aim is to exploit the fact that the distributed system of cars and central station is *techno-social*, and therefore it follows not only general laws dictated by geometry and mathematics, but also statistical laws dictated by human behavior. We want to use both properties to optimize the distributed computation, and empirically measure the obtained results over realistic scenario. We describe in this paper how to achieve this goal based on mining different kinds of mobility profiles from the GPS trajectory data, and show how this novel data-driven approach significantly improves over the safe-zone approach.

2. RELATED WORKS

The topic of this paper lies at the crossroad of two research fields: the distributed computation of global functions (a specific instance of which is treat in this work), and the computation of predictive models for mobility.

The global function considered in this paper is essentially a sum of variables, each of them derived from the location of an object. Existing works in literature provide solutions for this case, for instance [2] deals with the problem of checking whether a linear sum of variables crosses a given threshold, and develops conditions that allow the central node to correctly test the threshold check even if some node does not communicate its latest values. More recently, also some general approach for very general classes of functions have been proposed in [3], essentially allowing any function that can be expressed as $f(\bar{\mathbf{x}})$, where $\bar{\mathbf{x}}$ is the average of the individual vectors of variables (one vector for each node of the network) and f is any function. The latter is based on the concept of *Safe Zones*, i.e. sets of values that an individual vector of variables can assume without affecting the global function significantly, i.e. as long as the vector lies within its Safe Zone, the global function is guaranteed not to cross the threshold even if the coordinator still uses older values of the vector. The Safe Zone approach works particularly well when the individual vectors are expected not to change too much in time, while they might be less effective when significant variations are common. In the specific context considered in this paper, the individual vectors are locations of vehicles and derived quantities, which typically can have large variations during the day, therefore the Safe Zone approach is expected meet efficiency issues. Similar considerations have been performed in [7], in the area of distributed query tracking. Their basic idea consists in combining data compression methods for limiting the size of the transmitted data (namely, *sketches*) with a predictive model that allows to avoid communication whenever a node *behaves as expected*. In our work we try to merge the Safe Zones ideas (though limited to the simplest case, since we deal with a linear function) with the use of predictive models suitable for mobility data.

The kind of predictive model required by our application should describe the expected mobility of a moving object throughout a typical day. Therefore, we are interested in extracting periodic patterns of movements, that link the routes followed with their time within the period (e.g. the hour of the day). In literature, the work in [8] approaches this problem by partitioning the period (e.g. the day) into time slots (e.g. hours), and defining a periodic pattern as a description of a representative location for each time slot (or * if no such representative can be found). Another approach, described in [5], consists in looking for typical trips, i.e. trips that repeat themselves approximately several time in the history of an individual, thus considering whole routes. The basic analytical methods derive from trajectory mining tools previously developed and combined into the M-Atlas framework [1]. Both approaches – location-based and route-based – are considered in this paper, adapted to our context and experimentally compared.

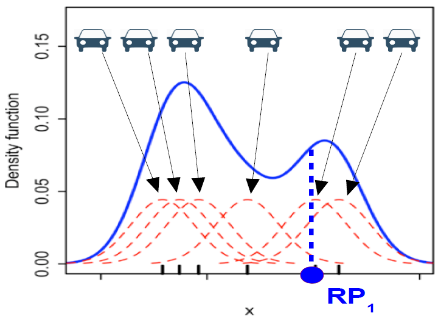


Figure 1: Example of vehicle density estimation for a reference point RP_1, on a single dimension, with a Gaussian kernel.

3. PROBLEM DEFINITION

Our reference application consists in evaluating the density of vehicles in correspondence of a given set RP of n_{RP} points in space, called *reference points*. In particular, density is estimated through a kernel-based approach, i.e., the density in a point is computed by counting all vehicles in space, yet weighted according to their distance from the point.

The application involves a central controller that computes (or estimates) the vehicle densities, and a set of nodes, each representing a vehicle. Each node receives a stream of location updates (coming from the on-board GPS device) and communicates the new location to the controller whenever needed to keep the global density estimates correct.

Definition 1 (DMP: DENSITY MONITORING PROBLEM). *Given a set $RP = \{RP_1, \ldots, RP_{n_{RP}}\}$ of n_{RP} reference points, a set $V = \{V_1, \ldots, V_{n_V}\}$ of vehicles and a kernel function $K(.)$, the* density monitoring problem *consists in computing, at each time instant, the function $f_{DMP}(V)$, defined as $f_{DMP}(V) = [K_1, \ldots, K_{n_{RP}}]^T$, where:*

$$\forall 1 \le i \le n_{RP}. \quad K_i = \sum_{j=1}^{n_V} K(V_j^{xy} - RP_i^{xy}) \qquad (1)$$

Here, $V_j^{xy} \in \mathcal{R}^2$ and $RP_i^{xy} \in \mathcal{R}^2$ represent, respectively, the actual position of vehicle V_j and the position of reference point RP_i.

In this paper the kernel function used is a Gaussian as shown in Figure 1 where the the DMP for a single reference point is represented as sum of the contributions given by six vehicles.

Whenever the number n_V of vehicles or their location update frequency (or both) reach high values, it is necessary to trade the exactness of the estimation defined above with a reduction of information exchange and processing. The loss of precision, in our context, is bounded by a parameter ϵ, that represents the deviation from the exact output for the DMP.

Definition 2 (ADMP: APPROXIMATE DMP). *Given a DMP with reference points $RP = \{RP_1, \ldots, RP_{n_{RP}}\}$, vehicle set $V =$*

$\{V_1, \ldots, V_{n_V}\}$ *and kernel function* $K(.)$*, and given an error tolerance parameter* ϵ*, the* approximate density monitoring problem *consists in computing, at each time instant, a function* $f_{ADMP}(V)$ *that approximates* f_{DMP}*. In particular, given the following error function:*

$$error(K^A, K) = \max_{i=1}^{n_{RP}} |K_i^A - K_i|$$

where $K = f_{DMP}(V)$ *and* $K^A = f_{ADMP}(V)$*, it always holds that* $error(f_{ADMP}(V), f_{DMP}(V)) \leq \epsilon$.

In other words, given an error threshold ϵ we require that the density estimate of each single RP provided by f_{ADMP} differs at most of ϵ from the corresponding value provided by f_{DMP}.

Solving a DMP or a ADMP consists essentially in defining a process able to satisfy their requirements in every possible status and evolution of the overall system. The latter aspect can be modeled by a stream of status changes that each node senses during the monitoring period; the “process”, then, basically defines a protocol used by nodes and controller to communicate only the essential information needed to satisfy the requirements of the (A)DMP.

In this paper several different ADMP solutions will be explored, in order to evaluate the impact of applying different levels of intelligence (in particular, learning from history) and usage of background knowledge.

4. BASIC APPROACHES FOR DISTRIBUTED DENSITY MAP MONITORING

Level 0: Communicate all

The trivial solution to the ADMP problem consists in having all the nodes sending an update to the controller for each update they receive. Obviously, that allows the controller to produce a perfect estimate of the global function (it actually yields a solution for the DMP problem, equivalent to $\epsilon = 0$), since it always knows the actual value of the variables it involves, at the price of communicating everything.

Level 1: static Safe Zones

This solution follows strictly the ideas based on Safe Zones [3], and therefore assumes that most objects are static or most of the time they move around some specific points in space, such as the home or work location. The basic idea, then, is to define a *default location* for each object v, and when no update arrives to the controller, it assumes that v is inside its default location.

More concretely, through analysis of historical data each node can be assigned to an optimal location that is used as its default position; then, basically the controller computes densities assuming that each node lies in its default position. Each node has assigned a geographical area such that, as long as it moves within that area the value computed by the controller is still a good approximation (w.r.t. the error threshold ϵ provided as parameter of the application). When the node moves outside its given area, it communicates the location update to the controller, which will use it to compute a correct estimation.

However, the context of mobility is characterized by massive and rapid changes in the data, since locations are highly dynamic, making this approach inadequate. For this reason, we will not further consider it, and instead will propose a variant that (in principle) better fits our scenario.

Level 2: adaptive Safe Zones

The basic assumption behind this approach is that the objects are not necessarily static, yet their movements are relatively slow. As an effect, when an object visits a given location, its associated region (see description of static Safe Zones above) will most likely contain several of the next locations of the object, yet no single location is able to capture a large part of the mobility of the object.

The protocol works as for static Safe Zones, but when an update must be communicated, the node is assigned to a new default location and to its corresponding geographical area, computed around its most recent measured location. Recomputing a new region (essentially, a new Safe Zone) is made possible and easy by the linearity of the global function to monitor (a sum of contributions), which enables to modify the Safe Zone of a node without compromising those of other objects. This kind of approach is much more problematic in contexts where the global function is more complex, since in those cases a change to a single object might involve changes to several other objects to reach an overall balance.

5. DISTRIBUTED DENSITY MAP MONITORING BASED ON PREDICTIVE MODELS

Since recent studies on human mobility claim that the latter is dominated by regular movements and therefore is highly predictable [4], here we analyze a segment of recent history of each node, in order to identify its regularities and use them as models to predict their locations in the next future. In particular, two variants of this idea are considered:

Most Frequent Location (MFL): we assume that the average user tends to visit (or cross) everyday the same places at the same hour of the day, therefore we look for single spatio-temporal locations that occur frequently, i.e., in many different days of the period. In this approach we do not try to link the frequent locations of a node, therefore the predictive model might contain consecutive sequence of locations that do not form consistent trajectories.

Mobility Profiles: here we make a stronger assumption, i.e. that the user tends to repeat the same trips everyday (home-to-work, and vice versa, for instance), thus involving an higher level concept of *frequent trip*, that requires a coherent sequence of spatio-temporal locations.

Both approaches create a typical daily schedule of each user, possibly with gaps for those moments of the day where the historical data did not reach a consensus on which location/trip to associate to them. The protocol, then, consists in letting the controller use at each instant the location predicted by the predictive model. In case of gaps (therefore no suggestion is provided by the predictive model) a *default model* is applied whose prediction is equal to the last known real location of the object. This is equivalent to adopt an *adaptive Safe Zones* solution limited to the gaps.

We remark that the Mobility Profiles approach implicitly adds a coherence constraint in the predictive model generation, therefore it will tend to produce predictive models with more gaps than the Most Frequent Location approach, yet the predictions provided are more likely to be reliable. Essentially, here we trading coverage for accuracy (to use information retrieval terms), and it is not clear a priori which solution might yield the best trade-off.

We collectively name the approaches mentioned above as protocols of the family *Level 3: predictive models*. In the following we describe the extraction and usage of the two variants.

5.1 Level 3.1: Most Frequent Location

5.1.1 MFL definition

In order to exploit the mobility habits of people, we start to build *schedules* of expected behaviors by using their most frequent visited locations. To do so, we need to acquire mobility information during a *training* period where the learning of habits will take place, then define frequency thresholds and build for each user a schedule that associates each time slot of the day to the most frequent location that occurred in that time slot throughout the training period, filtering out those locations that have an insufficient frequency w.r.t. the given threshold. The kind of model built with this approach is similar to the one described in [8].

Definition 3 (MFL USER DAILY SCHEDULE). *A MFL schedule is defined as the time-ordered set of the most frequent locations visited by an user within a specified observation periods. The daily schedule is discretized in time slots of equal durations.*

In order to identify what are the most frequent locations, defined by their GPS coordinates, we impose two constraints: (i) a time constraint, (ii) a spatial constraint. These two information are needed to build and align the daily schedule of a user.

Definition 4 (TIME SLOTS). *To determine to which time interval belong each single location we split each day in several slots of the same size. We define Δt as the time span that identifies the width of time frame reserved to each slot.*

Once defined a set of time slots we assign to each of them the set of locations visited by the user during the time slot. From this set we want to obtain a single representative location. A wide set of alternatives are possible to decide which location to choose as representative of the set (compute the center of mass, took the centroid, etc.). In this paper we choose to maintain the most dense location, i.e. the location that has the largest number of observations close to itself.

Definition 5 (SPATIAL RADIUS, NEIGHBORS). *Given a threshold Δs, called* spatial radius*, two locations A and B are considered* neighbors *if $||A - B||_\infty \leq \Delta s$, i.e. all their coordinates differs at most by Δs.*

5.1.2 MFL extraction

Once we have for a specified user a complete schedule of the visited locations during different days we need to synthesize a general schedule. In order to do it, we align all the daily schedules by collecting for each time slot all the observed locations that correspond to the time slot over the whole period; then, we calculate the most frequent location for each time slot. To avoid situations where the most frequent location appears only in a small fraction of the analyzed period, we impose a minimum support threshold.

Once this model is built we can use it as a proxy for the user mobility behaviour to the extent of predicting the location in which the user will be at a given time.

5.1.3 MFL-based prediction

The prediction phase rely on a direct query to the MFL schedule for the desired user. Given a query, defined as couple (u, t) composed by the user u and a timestamp t, we map t into the relative time slot and retrieve the MFL for the user u in that time slot, if it is defined. If a MFL for timestamp t does not exist, we apply a *default model*, that always suggests the last known location (i.e., the last one communicated to the controller).

5.2 Level 3.2: Mobility Profiles

5.2.1 Mobility profiles definition

We recall the concepts introduced in [5] where the user's history is defined as ordered sequence of spatio-temporal points $H = \langle p_1 \dots p_n \rangle$ where $p_i = (x, y, t)$ and x, y are spatial coordinates and t is an absolute timepoint. This history contains different trips made by the user, therefore in order to distinguish between them we need to detect when a user stops for a while in a place. This points in the stream will correspond to the end of a trip and the beginning of the next one:

Definition 6 (USER'S TRIPS). *Given the history H of a user and the thresholds $th^{stop}_{spatial}$ and $th^{stop}_{temporal}$, a* potential stop *is defined as a maximal subsequence S of the user's history H where the points remain within a spatial area for a certain period of time: $\overline{S} = \langle p_m \dots p_k \rangle | 0 < m \leq k \leq n \wedge \forall_{m \leq i \leq k} Dist(p_m, p_i) \leq th^{stop}_{spatial} \wedge Dur(p_m, p_k) \geq th^{stop}_{temporal}$. Finally we define a trip as the subsequence T of the user's history H between two consecutive stops in the ordered set $\overline{S}$ or between a stop and the first/last point of H.*

where $Dist$ is the Euclidean distance function defined between the spatial coordinates of the points, and Dur is the difference in the temporal coordinates of the points. Our objective is to use the user's trips in order to find his/her routine behaviors, this can be done grouping together the trips using a spatio-temporal distance function and extracting the medoid trip:

Definition 7 (ROUTINE). *Given a trip group g with at least th_{supp} elements and the distance function δ used to compute it, its* routine *is defined as the medoid of the set, i.e.:*

$$routine(g, \delta) = \arg\min_{t \in g} \sum_{t' \in g \setminus \{t\}} \delta(t, t')$$

where th_{supp} is the minimum size threshold used to reeve small groups which are not considered useful. Now we are ready to define the users mobility profile as the set of routine discovered over the history of the user:

Definition 8 (MOBILITY PROFILE). *Given a set of trip groups G of a user and the distance function δ used to compute them, the user's* mobility profile *is defined as his/her corresponding set of routines:*

$$profile(G, \delta) = \{routine(g, \delta) \mid g \in G\}$$

The mobility profile in other word represents a summarization of the movements of the user discarding the small variations which appear occasionally in his history.

5.2.2 Mobility profiles extraction

The extraction of mobility profiles from the user history is implemented as a sequence of modules which realizes the steps described above: Stop detection, Trip generation, T-Clustering equipped with a spatio-temporal function called Synch Route Similarity. The first module analyzes the user's history checking if the spatial distance between two consecutive points is lower than the threshold $th^{stop}_{spatial}$, when this happens the modules incrementally checks, and eventually stores, the following points until the constraint is not satisfied anymore. At the end of this process the module checks if the the sequences found satisfy also the temporal constraint using the $th^{stop}_{temporal}$ threshold, if this is satisfied the sequence will be considered as a stop for the user. The second module builds the trip

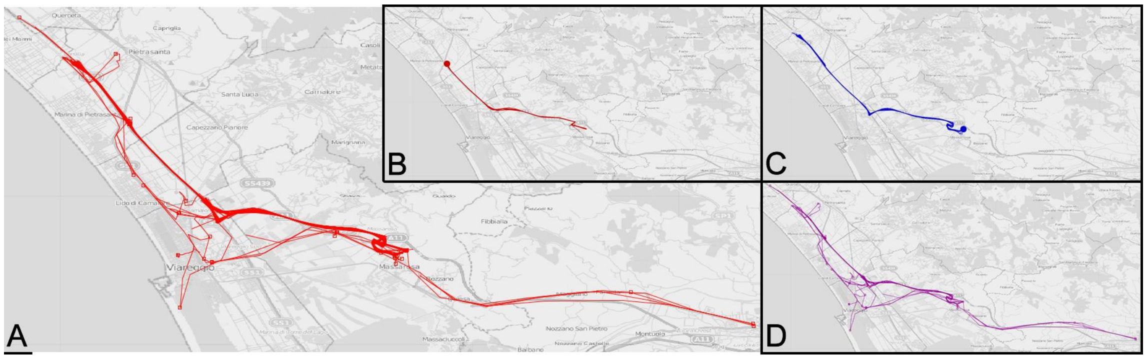


Figure 2: An example of mobility profile extraction: (a) The entire set of trips of a user, (b,c) the two clusters extracted, and (d) the remaining trips which are not periodic.

as sub-sequences of points between the begin and the first stop, each two consecutive stops and between the last and the end of the history. The last module runs a density-based algorithm called *T-Clustering* [9] using a spatio-temporal distance D_{SRS} which is a slight modification of the *Route Similarity*. This distance function starts comparing the initial timepoints of the two trips and it the temporal distance between the two are more than a give threshold (i.e. one hour) it returns an infinite distance without any further computation, otherwise it returns the distance computed as the route similarity. To obtain the clusters the T-Clustering algorithm checks if the following predicate is satisfied:

$$D_{SRS}(t1, t2) \leq (t1.lentgh + t2.length) * c_{PRadius}$$

where $c_{PRadius}$ is a the *Spatial Profile Radius* representing the tolerance used in the profile construction.

At the end of the process the clusters are filtered by their size, defined as number of trips, using the th_{supp} threshold. Finally, from each survived clusters a medoid is extracted and grouped obtaining the mobility profile of the user. In Figure 2 a real example is presented: here the user's trips are shown (a) including both the systematic and occasional ones, in (b) and (c) the two clusters extracted are presented showing a group of trips which are similar and synchronous, and in (d) the other trips which are the occasional movements that will be not considered. It is important to notice how the two clusters are very similar but reversed in the direction, this is usual due the fact that a big percentage of the users have two main reasons to move: going from home to work and viceversa.

5.2.3 *Mobility profile-based prediction*

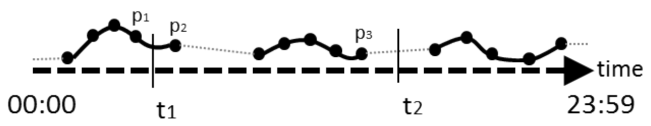


Figure 3: A profile composed by three routines. Only part of the day is covered, while *holes* are filled by the default model

Having extracted the user's mobility profile, we want to use it to predict the user's position at a certain time. It is important to notice that a mobility profile does not necessarily cover the whole daily schedule of a user. Let consider the two possible cases shown in Fig.3: (i) the prediction is made for the time instant t_1, corresponding to a period of the day where the profile is *defined*, and (ii) the prediction is made for the time instant t_2 corresponding to a period of the day where the profile is *not defined*.

In the first case the prediction will be the spatial interpolation between the two temporally closest points which surround t_1, namely p_1 and p_2. In the other case the prediction will be the last known point of the routine preceding temporally t_2, namely p_3. This corresponds to adopt a *default model* that always suggests the last known location, as done for MFL.

6. EXPERIMENTS

In this section we evaluate the different approaches presented in the paper, measuring the communications they save over the trivial protocol ("communicate all").

6.1 Dataset description

The dataset used in the following experiments is produced by a set of 40,000 cars, which represents the 2% of circulating cars in the coastal area of Tuscany. These points were tracked using GPS receivers with a sampling rate of 30s and a positioning system error of 10-20m in normal conditions over a period 5 weeks. The area covers a large territory with mixed land usages (residential areas, industrial zones, countryside, suburbs, etc.). The dataset was collected by Octotelematics S.p.A.[10], and a small sample is shown in Figure 4. Previous experiences on this data source (e.g. [1]) provided strong evidence of its validity and representativeness.

6.2 Experiment setup

A crucial aspect of the application is the position of the RPs in space. In order to test the effectiveness of the methods on a real scenario, we decided to use the positions of a set actual sensors used by the mobility agency in Tuscany, placed on the main gates of the city of Pisa plus one over the main bridge of the city center and two on two important neighboring towns. In Fig.5 the complete set of sensors is shown, and in Fig.6 a detail of Pisa is shown where each entrance of the city is monitored. The physical devices placed on the territory are permanent sensors based on laser technology, which can count the number and estimate the speed of cars passing nearby.

The testing of the methods presented in this paper requires to consider the following kinds of parameters:

- data-dependent parameters: in particular, we consider the sampling rate of the input GPS data in terms of average time

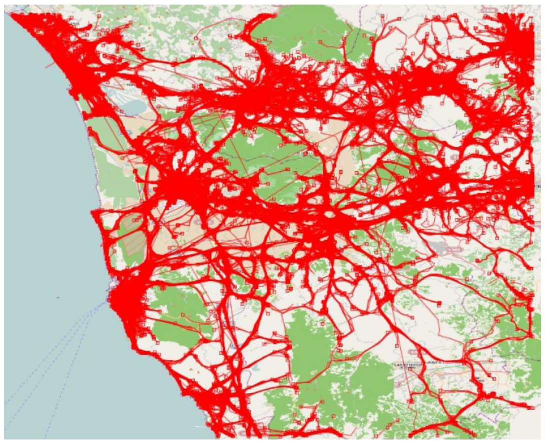


Figure 4: Sample of the dataset used for experiments

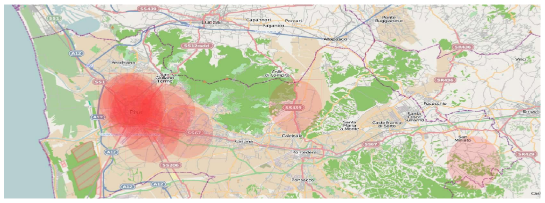


Figure 5: Location of RPs adopted in the experiments, and buffers representing kernel widths for the density computation

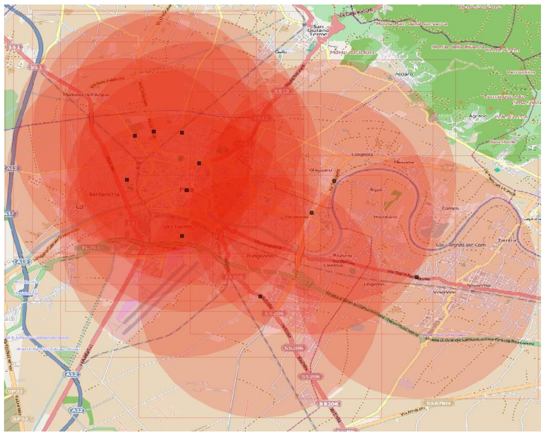


Figure 6: A detailed view of the sensors and the focal area in the city of Pisa

gap between consecutive location updates received by the vehicles. Where not explicitly mentioned, the sampling rate will be set to the default value of one point every 5 minutes (average);

- application-dependent: beside the set of RPs, which was chosen and fixed above, the application requires to specify (i) the width of the kernel adopted in computing the density over each RP, and (ii) the maximum (absolute) error tolerated in computing such densities. The width of the kernel is expressed as the distance for the vehicle at which its weight in the density computation is equal to 0.1. Where not explicitly mentioned, such width is set to 4 km. The error threshold, instead, is set to 5% of the overall average density of all RPs;

- predictive model-dependent: each predictive model is built on the base of its own parameters. In particular, we will explore the impact of the model spatial radius used, which defines how accurate must be the model. The lower is the radius, the higher is the accuracy but also the higher is the number of gaps in the model, since it is more difficult to find satisfactory predictive models. Where not explicitly mentioned, the spatial radius for the Most Frequent Location model is set to 1 km, and the Profile spatial radius is set to 0.3. Moreover, the temporal granularity adopted in MFL (i.e. the Δt used to define time slots) is set to the double of the data sampling rate, in order to have on average two points for each time slot.

In the following sections we will study the impact of the approaches proposed, and provide some spatial exploration of the results.

6.3 Data sampling rate

In this section we present an overall study of the performances of the system using the different methods discussed in the paper, compared against the trivial protocol *Level 0* ("communicate all"). In Fig.7 we show the communication rates varying the sampling rate of the data:

Adaptive SZ : the increasing trend shows that the adaptive Save Zones solution is affected by the sampling rate. Indeed, longer temporal gaps between location updates means an higher spatial distance between them, rising the probability of crossing the actual Safe Zone, and therefore requiring to communicate and update the Safe Zone more frequently;

MFL : we can see that the communication rate increases very quickly, due the fact that with an higher sampling rate the method cannot find (dense) groups in the time intervals. This affects mostly the MFL models of users with a small number of points, which become less stable or disappear completely;

Profiles : the Profiles-based solution appears extremely stable while changing the sampling rate, thanks to the fact that it tries to find a systematic whole trip of the user, with the result that the profiles which are extracted with different sampling rates are composed by less points but maintain their semantics, i.e. they still describe the same trip (though less accurately).

6.4 Application-dependent parameters

The impact of these parameters is very regular. Therefore, due to space limitation, we simply summarize their overall effect.

The width of the kernel (expressed as a distance, as described above) was studied in the range of values between 1 km and 10 km. In all methods applied, the communications increase monotonically with the kernel width.

Similarly, the density error threshold was studied in the range of values between 1% and 10% of the overall average of densities over all RPs. In all methods applied, the communications decrease monotonically with the error.

6.5 Model-dependent parameters: MFL

Fig.8 shows the number of MFL models created while changing the spatial radius parameter, hence the number of users which has a

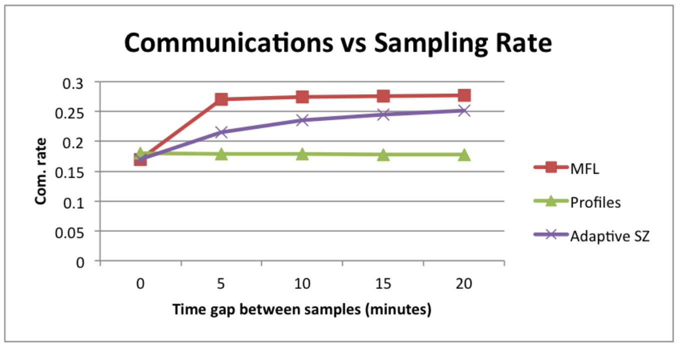


Figure 7: Overall analysis of the four methods varying the sampling rate of the data.

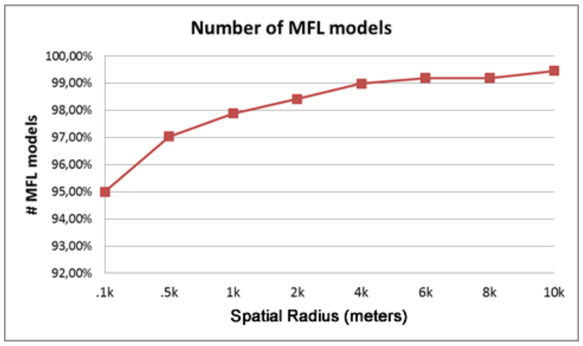


Figure 8: Number of MFL models created by the nodes using different tolerance values.

frequent behavior in at least a time interval. The trend is clearly increasing and tends to reach saturation. However, as shown in Fig.9, this does not mean that the performances of the method increase as well, in fact the total communication rate slightly increases with increasing tolerances, highlighting the fact that the MFL models created are not good in the prediction. The figure also shows the ratio of updates for which MFL could be applied (i.e. MFL provided a prediction) with success, thus saving a communication. Similarly, it shows the ratio of updates for which MFL could not apply, yet the default model successfully avoided the communication. We can see that the two ratios are rather symmetric, therefore resulting in overall very stable communication savings.

6.6 Model-dependent parameters: Profiles

In this section we present the performances obtained using the Profiles approach. As described in Section 5.2 here during the initial phase each node builds a profile and sends it to the controller. Then, when the system starts, the nodes check if their actual positions are coherent with their profiles. If not, they communicate to the controller, otherwise nothing is communicated, since the controller can predict the position using the profiles. In Fig.10 we show how the Profile spatial radius value changes the number of profiles extracted during the initialization phase: increasing the radius the number of profiles increases, i.e. the number of nodes who have a profile. Indeed, higher radii make the similarity between the user's trips less strict, thus making the formation of groups and profiles easier. It is interesting to notice how the number have a big increasing when the radius passes from 0.3 to 0.5 detecting a crucial

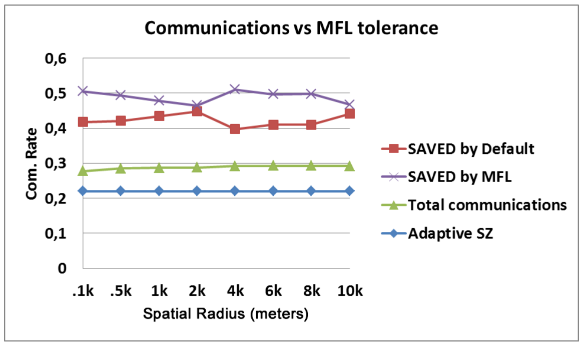


Figure 9: MFL performances compared to adaptive Safe Zones approach and the communication saved by MFL and default model.

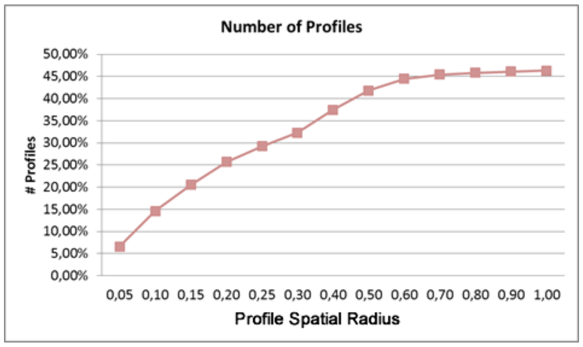


Figure 10: Number of profiles created by the nodes using different tolerance values.

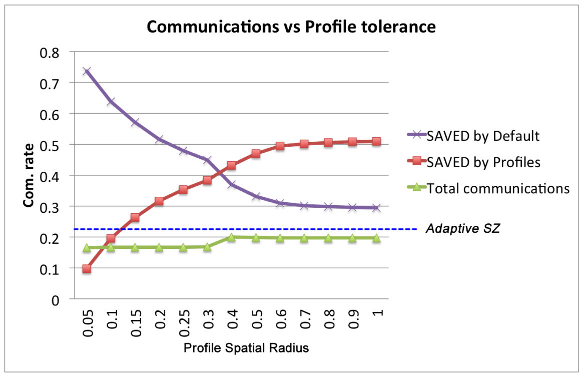


Figure 11: Profiles and adaptive Safe Zones performances and communication saved by profiles and default model.

point for the profile construction. Having more profiles does not mean to have better performances. Indeed, loose profiles lead to loose predictions. This can be seen in the Fig.11 where the performances of the system remain almost the same even if the number of nodes with a profile increases. Moreover, with radius equal to 0.3 the performances decrease, meaning that a critical point is reached and the profiles become too loose and the errors in profile prediction becomes higher. If Fig.11 we compare the performances of the profile approach against the adaptive Safe Zones. As we can see, Profiles gain a saving of 6.5%, meaning that the concept of profiles actually produces significant benefits. More in detail, analyzing the communications saved by the profiles and the default model (used when there is no profile to apply) we can see that profiles tend to replace the default model, improving the overall performances.

6.6.1 Spatial exploration of results

In this section, we provide an exploration of the performance results on the map. Fig.12 shows where the relative errors occur during the execution of the system. The color scale goes from red, representing a big percentage of errors, to blue which represents a small percentage of errors. Clearly, the error percentage is affected by the proximity to the RPs, in fact the error threshold is more likely exceeded in proximity of each focal point where the kernel function reaches the maximum. In other words, in those areas a small error in the location prediction leads to a big error on the density computation, therefore causing more likely a communication from the node to the controller. Fig.13 shows a detailed view of the city at

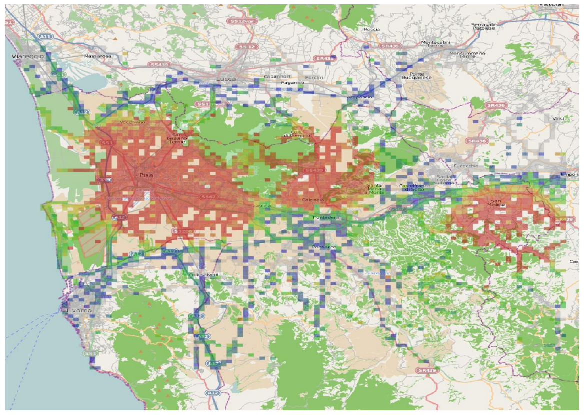


Figure 12: Distribution of relative errors occurred during the system execution.

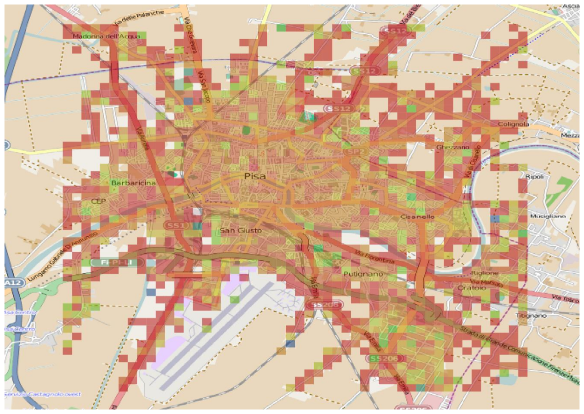


Figure 13: A detailed view of the relative error distribution in Pisa city using a smaller granularity.

a finer granularity, which is covered by several RPs. It is clear how the distribution of the errors in space is not homogeneous, in fact the city center (where an RP is placed) seems to be less affected by errors than the main gates and their relative roads.

7. CONCLUSIONS

In this work we developed and compared several approaches to the problem of computing population density over key areas in a distributed context, trying to reduce as much as possible the communication required. The approaches mainly differed for the way they tried to exploit the recent history of the moving objects involved, in some cases by estimating an optimal static default location for each object, in other cases by learning their mobility habits and exploiting them as prediction means. The experimental comparisons performed provided several insights on the effectiveness of each approach, in many cases with surprising outcomes.

Several new questions and open issues arose during the development of this work. We mention three of them: (i) since the computation involves potentially sensible information about individuals, can the proposed framework be made privacy-preserving? (ii) are there parts of the map more difficult to "learn"? E.g. highways are expected to be difficult, due to the high presence of occasional trips and occasional passers-by; (iii) if taken collectively, individual non-systematic behaviors might form typical paths, e.g. vehicles on the highways: how to integrate them in the framework? Aspects to consider on this way include the fact that typical paths cannot be associated to the vehicle ID (therefore there must be a different way to choose a "model" for a given model-less vehicle, such as prefix match), and to mine typical paths it is needed a centralized computation, therefore nodes might send to the controller, for instance, all trips not described by a profile.

8. REFERENCES

[1] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, C. Renso, S. Rinzivillo, and R. Trasarti, “Unveiling the complexity of human mobility by querying and mining massive trajectory data,” *The VLDB Journal*, vol. 20, no. 5, pp. 695–719, 2011.

[2] M. Dilman and D. Raz, “Efficient reactive monitoring,” in *INFOCOM*, 2001, pp. 1012–1019.

[3] I. Sharfman, A. Schuster, and D. Keren, “A geometric approach to monitoring threshold functions over distributed data streams,” *ACM Trans. Database Syst.*, vol. 32, no. 4, Nov. 2007.

[4] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, “Limits of predictability in human mobility,” *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[5] R. Trasarti, F. Pinelli, M. Nanni, and F. Giannotti, “Mining mobility user profiles for car pooling,” in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 1190–1198.

[6] D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A.-L. Barabási, “Human mobility, social ties, and link prediction,” in *KDD*, 2011, pp. 1100–1108.

[7] G. Cormode and M. Garofalakis, “Sketching streams through the net: distributed approximate query tracking,” in *Proceedings of the 31st international conference on Very large data bases*. VLDB, 2005, pp. 13–24.

[8] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D. W. Cheung, “Mining, indexing, and querying historical spatiotemporal data,” in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 236–245.

[9] G. Andrienko, N. Andrienko, S. Rinzivillo, M. Nanni, D. Pedreschi, and F. Giannotti, “Interactive visual clustering of large collections of trajectories,” in *IEEE VAST*, 2009, pp. 3–10.

[10] “Octotelematics s.p.a.: http://www.octotelematics.it/.”

Discovering Urban Spatial-Temporal Structure from Human Activity Patterns

Shan Jiang
Massachusetts Institute of Technology 77 Mass. Ave. E55-19E Cambridge, MA 02142 USA
+1 (857) 654-5066
shanjang@mit.edu

Joseph Ferreira, Jr.
Massachusetts Institute of Technology 77 Mass. Ave. 9-532 Cambridge, MA 02139 USA
+1 (617) 253-7410
jf@mit.edu

Marta C. Gonzalez
Massachusetts Institute of Technology 77 Mass. Ave. 1-153 Cambridge, MA 02139 USA
+1 (617) 715-4140
martag@mit.edu

ABSTRACT

Urban geographers, planners, and economists have long been studying urban spatial structure to understand the development of cities. Statistical and data mining techniques, as proposed in this paper, go a long way in improving our knowledge about human activities extracted from travel surveys. As of today, most urban simulators have not yet incorporated the various types of individuals by their daily activities. In this work, we detect clusters of individuals by daily activity patterns, integrated with their usage of space and time, and show that daily routines can be highly predictable, with clear differences depending on the group, e.g. students vs. part time workers. This analysis presents the basis to capture collective activities at large scales and expand our perception of urban structure from the spatial dimension to spatial-temporal dimension. It will be helpful for planers to understand how individuals utilize time and interact with urban space in metropolitan areas and crucial for the design of sustainable cities in the future.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: data mining, spatial databases and GIS*;* I.5.3 **[Clustering]**: clustering algorithms and similarity measures.

General Terms

Algorithms, Design, Measurement, Human Factors, Performance

Keywords

Urban spatial-temporal structure, Human activity intensity, Kernel density estimation, Time-cumulative spatial activity density, Computational social science

1. INTRODUCTION

Cities, home to billions of people, are complex systems [1, 2]. For decades, urban spatial structure, measured by the degree of spatial concentration of population and employment, has been studied by urban scholars to describe the structure and organization of cities, and their function and role in people's life [3-5]. On the one hand, with the improvement of transportation systems, cities have evolved from monocentric to polycentric forms in their spatial configurations [3, 6-8]. On the other hand, with the advances in information and communication technologies (ICT), cities have been racing to be "smarter", in terms of their human, social, and environmental capital profiling [9, 10]. As a result, today swelling cities have become incidentally data repositories of human activities (gained from the emerging massive urban sensors such as GPS, mobile phone, and online user-generated social media). These facts, plus the spectacular ability of researchers to collect and analyze data, have helped us understand the nature of human mobility (e.g., high predictability in daily routine [11, 12]) and the dynamics of cities [13, 14], and provided a great potential for planners to optimize the value of existing infrastructures in the city [15].

While celebrating opportunities these massive urban sensing data bring to us, researchers have also realized challenges of adopting them, due to privacy and legal restrictions and economic constrains. With little or no information about either the social demographic characteristics of the individuals or the types of activities they are conducting, our understanding of the causes and underlying reasons of human behavior are still inadequate [16]. For example, in the study by Becker et al. [17], although it is promising to see plausible estimates of the spatial distribution of residence by users of different phone usage patterns (e.g., classified "workers" or "partiers") based on call detail records (CDR), we still cannot get a complete picture of human activities in non-home/work categories, as it is hard to infer or validate those types of activities from the CDR data. Yet since cities have been playing increasingly important roles as consumption centers [18], urban planners are pressed to know how cities are used by different types of people for different types of activities. Given the nature of these urban sensing data (such as CDR), similar challenges are faced by other studies (e.g., Eagle et al. [19]), where non-home/work activities are hard to be differentiated. Meanwhile, it is also unclear how knowledge gained from targeted groups of communities (e.g., universities [19]) may apply when the scale is enlarged to metropolitan area and beyond.

Activity-based travel surveys collected by planners to develop transportation and activity models for cities, on the other hand, have the potential to complement the new insights in human activities and mobility gained from massive urban sensing data as discussed above. If responded accurately, travel surveys can inform us about "who, what, when, where, why and how of travel for each person in a surveyed household" [20]. These questions have been in the center of urban planning, geography and

UrbComp'12, August 12, 2012. Beijing, China.
Copyright 2012 ACM 978-1-4503-1542-5/08/2012 …$15.00.

transportation fields for decades [21-26], due to their importance in helping us understand the complexity and dynamics of cities.

In this paper, we concentrate on numerical methods to mine the spatial-temporal activity patterns of individuals obtained from a recent travel diary survey in the Chicago Metropolitan Area. The advances of this study lie in three folds.

First, we expand the traditional understanding of urban structure from spatial dimension based on static density of population to spatial-temporal dimension, which we call urban spatial-temporal structure. We measure this structure by defining a spatial distribution of activity intensity with temporal information extracted from the travel survey data.

Second, we cluster individuals according to their daily activity patterns and spatial-temporal traces. By doing so, it enables us to go from the traditional classification of individuals as students, workers and other types to more diverse groups, and enriches our understanding about human activities in the city. Combining the clusters of individuals with the spatial dimension, we identify sub-regions of the metropolitan area with inherent connections to the performed activities.

Finally, we combine the above measures together to analyze and visualize the time-cumulative spatial densities of various activity types by sub-regions of Chicago for different types of individuals. By comparing the spatial distributions of the intensity of various activities, we demonstrate that enormous information about urban spatial-temporal structure can be quantitatively analyzed and vividly illustrated. It will be very useful for urban planners to understand how urban areas have been used in space and time by different types of individuals, and will be crucial for them to propose solutions for sustainable cities.

2. DATA SOURCE AND STUDY AREA

To understand the urban spatial-temporal structure, in this research, we study the Chicago Metropolitan Area as an example. We employ a publicly available large-scale "Travel Tracker Survey"[1] conducted by the Chicago Metropolitan Agency for Planning (CMAP) in 2008 [27]. As this survey is designed to estimate the regional travel demand, the carefully planned sampling strategy ensures a good representation of the total population in the region, which includes eight counties—Cook, Du Page, Grundy, Kane, Kendall, Lake, McHenry, and Will—from the Northeastern Illinois Region, and two counties—Porter and LaPorte—from the Northwestern Indiana Region.

In this study, since we focus on the urban spatial-temporal structure on weekday, we use the survey records from Monday through Thursday as a sample to represent an average weekday, which contains daily activities of 23,527 distinct individuals. For each distinct individual, the survey records every activity destination, arrival and departure time, location (the longitude and latitude pair), activity type (such as home, work, school, shopping, recreation, etc.), and duration at the destination, in 24-hours of the surveyed day. To facilitate the analysis of this study, we transform the individuals' survey records as described above to minute-by-minute records with information of latitude and longitude location, time and activity type[2]. This data transformation allows us to explore the spatial-temporal structure of the Chicago metropolitan area in this study.

3. URBAN SPATIAL-TEMPORAL STRUCTURE

Urban spatial-temporal structure (USTS) extends the traditional concept of urban spatial structure by incorporating the temporal information. Specifically, it contains information of time stamps of individuals' activities, activity locations, activity types, and optionally, personal attributes (which are not indispensible, but may provide rich socioeconomic context). In this section, we use kernel density estimator (KDE) to estimate several measurements of USTS and present empirical findings of the Chicago urban spatial-temporal structure.

3.1 Measurements of Urban Spatial-Temporal Structure

In order to capture the urban spatial-temporal structure, we propose to measure a "spatial-temporal activity density" at each time instant of the day and analyze it through KDE. For the purpose of visualization and demonstration, we integrate this density with respect to time, and get a "time-cumulative spatial activity density", which measures s*patial distribution of activity intensity*— a normalized sum of individuals' activity duration in a study area during a time period of interest. Human activity intensity can be crucial for many purposes, such as the analysis and prediction of energy consumption, infrastructure usage and business opportunities, etc.

3.1.1 Preliminary

We use a 3-dimensional space $\boldsymbol{S}$ to describe time and 2-dimensional activity locations as follows:

$$\boldsymbol{S} \triangleq \{\boldsymbol{s} = (t, x, y) | t \in [t_0, t_1], (x, y) \in \boldsymbol{A}\} = [t_0, t_1] \times \boldsymbol{A} \qquad (1)$$

where t, x, and y are the time, longitude, and latitude, respectively, $[t_0, t_1]$ defines the time period of interest (e.g., one or several minutes, hours, days, weeks, etc.), and $\boldsymbol{A}$ is the set of (longitude, latitude)-pairs for the study area.

3.1.2 Spatial-Temporal Activity Density

We use the distribution density π on $\boldsymbol{S}$ of a particular activity in a study area $\boldsymbol{A}$ during a time period of interest ($[t_0, t_1]$) as the measurement of urban spatial-temporal structure. We call the measurement π the *spatial-temporal activity density*, which is defined as the probability density of individuals' presence at destination[3] (x, y) at time t for the corresponding activity purpose, and the density π satisfies

$$\int_{\boldsymbol{S}} \pi(\boldsymbol{s})\, d\boldsymbol{s} = \int_{[t_0,t_1]\times \boldsymbol{A}} \pi(t, x, y)\, dt dx dy = 1. \qquad (2)$$

[1] Source: http://www.cmap.illinois.gov/travel-tracker-survey. Retrieved on June 12, 2012.

[2] Since we do not know individuals' exact travel path between destinations, and the movements usually amounts to a small portion of people's daily life within 24-hours (around 5%), in order to fill in the gap between two consecutive destinations, we make a simplified assumption that people move in a straight line with constant speed when they travel. It is worth noting that this filled-in information is only used in subsection 4.2.2 *Clusters of Daily Traces*, and due to the small proportion of time spent in travel, the estimation error caused by this approximation is very limited. Readers should also note that since travel is not considered as activities at destinations, therefore this treatment of data transformation will have little effect on the estimation of urban spatial-temporal structure discussed in the paper.

[3] We define *destination* as a place where people go due to the need of committing a certain type of activity (e.g., work, school, shopping, recreation, etc.). Here, we do not consider individuals' movement en route while measuring activity density. Therefore our treatment of traces will not affect the accuracy of the estimation of spatial-temporal activity density and the time-cumulative spatial activity density.

To understand the urban spatial-temporal structure, it would be helpful to visualize the spatial-temporal activity density. However, as it is defined on a 3-dimensional space $\boldsymbol{S}$, a direct static visualization becomes impossible. To circumvent the visualization barrier of spatial-temporal activity density, we explore two alternatives. One is to examine the distribution density $\tilde{\pi}_{\bar{t}}$ on $\boldsymbol{A}$ at a fixed time instant $\bar{t}$.[4] The other is to consider a time-cumulative spatial activity density.

3.1.3 *Time-cumulative Spatial Activity Density*

A *time-cumulative spatial activity density* $\pi_{a,b}$ on $\boldsymbol{A}$ is defined as follows

$$\pi_{a,b}(x,y) = c_{a,b} \int_{[a,b]} \pi(t,x,y)dt \qquad (3)$$

where $[a,b] \subseteq [t_0,t_1]$, and $c_{a,b}$ is a positive constant to make $\pi_{a,b}$ satisfy the restriction that the integral of density $\pi_{a,b}$ on $\boldsymbol{A}$ equals 1, i.e., $c_{a,b}$ is a constant of normalization.[5] We use this measurement to understand the spatial distributions of intensity of various activity types and explore urban spatial-temporal structure quantitatively, which will be presented in section 3.2.

3.1.4 *Kernel Density Estimation*

In this study, we use the kernel density estimator to estimate the time-cumulative spatial density of a certain set of activities. Suppose that we have n observations of a set of activities $\{\boldsymbol{s}_i = (t_i, x_i, y_i)\}_{i=1}^{n} \subseteq [a,b] \times \boldsymbol{A}$, then the corresponding time-cumulative spatial density is estimated by

$$\hat{\pi}_{a,b}(x,y;H) = \frac{1}{n}\sum_{i=1}^{n} K_H\left(\begin{pmatrix}x\\y\end{pmatrix} - \begin{pmatrix}x_i\\y_i\end{pmatrix}\right),$$

where the *bandwidth matrix* H is a 2×2 symmetric positive definite matrix,

$$K_H\left(\begin{pmatrix}x\\y\end{pmatrix}\right) = |H|^{-1/2} K\left(H^{-1/2}\begin{pmatrix}x\\y\end{pmatrix}\right),$$

and K is a 2-variate kernel function that satisfies

$$\int_{\boldsymbol{A}} K\left(\begin{pmatrix}x\\y\end{pmatrix}\right) dxdy = 1.$$

Here we follow the popular practice to take K to be the standard 2-variate normal density, a spherically symmetric kernel,[6]

$$K\left(\begin{pmatrix}x\\y\end{pmatrix}\right) = \frac{1}{2\pi}\exp\left(-\frac{1}{2}(x^2+y^2)\right),$$

which implies that $K_H\left(\begin{pmatrix}x\\y\end{pmatrix} - \begin{pmatrix}x_i\\y_i\end{pmatrix}\right)$ is the $N\left(\begin{pmatrix}x_i\\y_i\end{pmatrix}, H\right)$ density in vector $\begin{pmatrix}x\\y\end{pmatrix}$. To simplify the selection of H, we assume that $H = h^2 I$, where $h > 0$ and I is the 2×2 identity matrix. It is clear that a larger bandwidth parameter h leads to a smoother estimation and a smaller h gives more fluctuations. In this study, the bandwidth h is chosen to minimize the mean integrated squared error (MISE). For detailed discussion about nonparametric kernel density estimation and the MISE criterion, and the applications of KDE on topics of interests in the urban realm, readers may refer to Wand and Jones [28], Kwan & Hong [29], and Yuan et al.[14], respectively.

3.2 Chicago Metropolitan Area General Spatial-Temporal Structure

By using kernel density estimation method described above, we estimate the activity intensity in the Chicago Metropolitan Area over 24 hours to understand the Chicago Urban Spatial-temporal structure. Figure 1 shows the time-cumulative spatial activity densities in the Chicago metropolitan area by activity categories of home, work, school, shopping/ errands, and recreation/ entertainment.

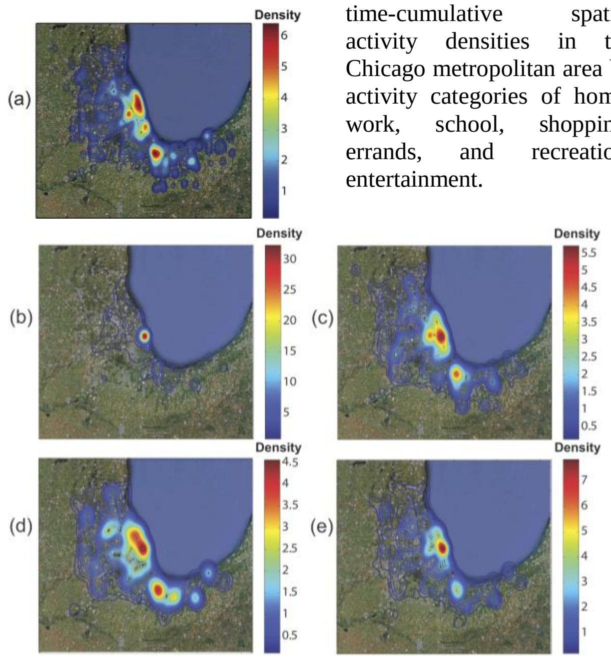


Figure 1 Chicago time-cumulative spatial densities of different activity types (a) home, (b) work, (c) school, (d) shopping and errands, and (e) recreation and entertainment, on an average weekday.

We can see from the figure that areas with intensive person-hours for home activities are along the north lakeshore, south lakeshore, and around Oak Park. Downtown Chicago alone has extremely intensive work activities—very high total person-hours for working—compared with the rest of the region. School activities in the region are mainly concentrated in the city of Chicago (where major universities and schools are clustered), and the southeastern part of the region (where Purdue University is located). Several regions with intensive person-hours for shopping activities exist, including the downtown and northern part of the City of Chicago, the City of Evanston, Schaumburg, southeast area of the region (cities of Hammond and Schererville). Areas that have been visited and used intensively for entertainment and/or recreational activities is narrower than their counterparts for shopping activities in space, but with some overlaps, including the downtown and northern part of the City of Chicago, City of Evanston, Oak Park, and the southeastern part of the Chicago metropolitan region.

As discussed in the introduction section, these results are interesting and informative; however, we want to further explore how the urban spaces are utilized by different types of individuals with different activity patterns. In other words, we want to ask "do students, workers, or other types of individuals use urban spaces in similar or different ways?" "Are there differences in activity

[4] The notation $\tilde{\pi}$ means that it is obtained from π via normalization.

[5] Note that $\tilde{\pi}_{\bar{t}}$ and $\pi_{a,b}$ are closely related: If we assume that π is continuous in t (which is a very reasonable assumption), then we have $\pi_{a,b} \to \tilde{\pi}_{\bar{t}}$, as $a, b \to \bar{t}$.

[6] Normal density is in the class of admissible kernel functions, and people often choose it for the simplicity of interpretation. For detailed discussion about choice of kernel functions, please refer to [28] Wand, P. and Jones, C. *Kernel Smoothing*. Chapman & Hall, 1995.

intensities by similar types of individuals across different subareas of the region?" "In what ways are their activity intensity and usage of urban areas in space and time similar or different?"

4. CLUSTERING DAILY ACTIVITY PATTERNS AND TRACES

In order to answer these questions and to further understand urban spatial-temporal structure, clustering individuals both by their daily temporal-activity patterns and by their spatial-temporal trace becomes important and natural components of this study.

4.1 *K*-Means Clustering via PCA

The *K*-means algorithm has been widely applied to partition datasets into a number of clusters [30]. It performs well for many problems. However, the computational cost can be very high, when the dimension of the data is large [31]. As the principle component analysis (PCA)/eigen decomposition method is widely employed for dimension reduction, it is a common practice to utilize the *K*-means algorithm via PCA and obtain successful applications [32, 33]. In an earlier study [34], we explain in detail the processes and validity of applying the *K*-means algorithm via PCA method to cluster individuals into different groups according to their activity patterns in the Chicago Metropolitan Area. For details about the *K*-Means clustering via PCA method, readers may refer to Section 4 of Jiang et al. [34].

4.2 Daily Activity Pattern and Trace Clusters in Chicago

In this section, we first review the clustering results of activity patterns in the Chicago Metropolitan Area on an average weekday [34], and then we cluster individuals' daily spatial-temporal traces using a similar process.

4.2.1 Clusters of Daily Activity Patterns

Figure 2 summaries the analysis results of individuals' daily activity pattern clusters on an average weekday in the Chicago Metropolitan Area.

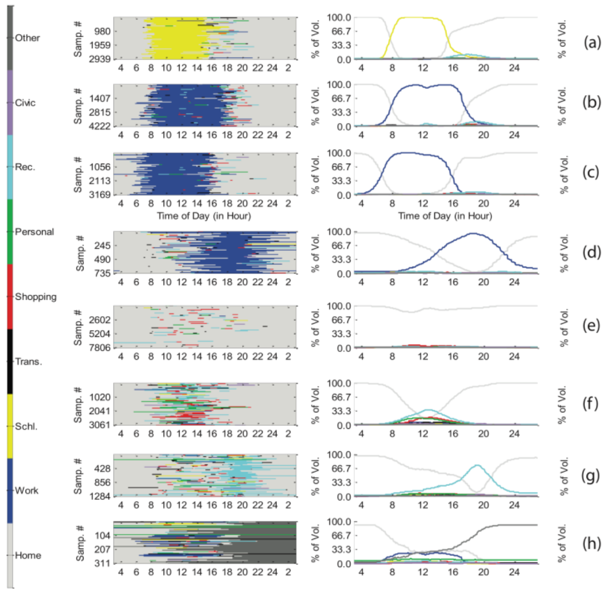


Figure 2 Clustering of individuals' weekday activity patterns in Chicago, with clusters (a) students, (b) regular workers, (c) early-bird workers, (d) afternoon workers, (e) stay-at-home, (f) morning adventurers, (g) afternoon adventurers, and (h) overnight adventurers. Adapted from [34] Jiang et al.

The first column of Figure 2 displays individuals' daily activity sequences for each cluster. The second column shows the aggregated volume of different types of activities in the metropolitan area during a specific time interval over 24 hours. The results contains 8 types of personal activity patterns, including students (12.50%), regular workers (17.90%), early-bird workers (13.50%), afternoon workers (3.10%), the stay-at-home (33.20%), the morning adventurers (13.00%), the afternoon adventurers (5.50%) and the overnight adventurers (1.30%). Notice that this represents much richer information about urban groups than the traditional classification in which only 47% of the individuals can be categorized as students or workers based on their daily activity patterns.

4.2.2 Clusters of Daily Traces

We cluster individuals' spatial-temporal traces according to their space-time similarities by applying the *K*-Means algorithm via PCA in a very similar procedure to the study we conducted earlier [34].

First, we sample the individuals' locations and activities every five minutes, using the transformed data described in Section 2. Let $n_{\boldsymbol{T}}$ denote the total number of individuals in the sample, then for each individual i =1, …, $n_{\boldsymbol{T}}$, we have an 576 (=288×2) dimensional vector $\boldsymbol{T}_i = (x_{i,1}, \dots, x_{i,288}, y_{i,1}, \dots, y_{i,288})'$ to describe his/her trace, where $x_{i,j}$ and $y_{i,j}$ is individual *i*'s longitude and latitude in the *j*-th time instant being sampled.

Second, we deal with $\boldsymbol{T}_i$ in the same way as we deal with $\boldsymbol{a}_i$ in Section 4 of Jiang et al [34]. We arrange the thus obtained 576 eigenvalues in descending order, i.e., $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{576} \geq 0$, and the eigenvector $\boldsymbol{v}_k$ that corresponds to the *k*-th eigenvalue is called the *k*-th *eigentrace*.

Third, we reconstruct the individual *i*'s spatial-temporal trace $\boldsymbol{T}_i$, by using a subset of eigentraces as follows. Let $\boldsymbol{d}_i = \boldsymbol{T}_i - \overline{\boldsymbol{T}}$, where $\overline{\boldsymbol{T}} = \frac{1}{n_T}\sum_{i=1}^{n_T} \boldsymbol{T}_i$ is the sample mean, and suppose the projection of $\boldsymbol{d}_i$ onto the first h eigentraces $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_h\}$ are $(z_1, \dots, z_h)'$. According to formula

$$\widehat{\boldsymbol{T}}_i = \overline{\boldsymbol{T}} + (\boldsymbol{v}_1, \dots, \boldsymbol{v}_h)(z_1, \dots, z_h)' \tag{4}$$

we obtain a vector $\widehat{\boldsymbol{T}}_i = (\hat{x}_{i,1}, \dots, \hat{x}_{i,288}, \hat{y}_{i,1}, \dots, \hat{y}_{i,288})' \in \mathbb{R}^{576}$. We define the *reconstruction error* $e(\boldsymbol{T}_i)$ for $\boldsymbol{T}_i$ as the *average reconstruction deviation*, namely,

$$e(\boldsymbol{T}_i) = \frac{1}{288}\sum_{j=1}^{288} \operatorname{dist}\left((x_{i,j}, y_{i,j}), (\hat{x}_{i,j}, \hat{y}_{i,j})\right) \tag{5}$$

where $\operatorname{dist}\left((x_{i,j}, y_{i,j}), (\hat{x}_{i,j}, \hat{y}_{i,j})\right)$ is the distance (in kilometers) between two locations $(x_{i,j}, y_{i,j})$ and $(\hat{x}_{i,j}, \hat{y}_{i,j})$. Given any $\varepsilon > 0$, we can find some $h > 0$, so that the *average reconstruction error*, $\frac{\sum_{i=1}^{n_T} e(\boldsymbol{T}_i)}{n_T}$, caused by neglecting the projections onto the ignored eigentraces $\{\boldsymbol{v}_{h+1}, \dots, \boldsymbol{v}_{576}\}$ is no greater than ε. Let $\varepsilon_0 > 0$ be the acceptable error level, and define $h(\varepsilon_0)$ to be the smallest h such that the average reconstruction error induced by using the first h eigentraces is no greater than ε_0. We then call $h(\varepsilon_0)$ the *appropriate number of eigentraces*. In this study, we take $\varepsilon_0 = 0.5$ kilometers, and we get $h(\varepsilon_0) = 33$. Thus, eigentraces $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_{33}\}$ are used in the reconstruction of $\boldsymbol{T}_i$ and in the *K*-means clustering of individuals' spatial-temporal traces.

Lastly, according to Dunn's Index (DI), which maximizes inter-cluster distances while minimizing the intra-cluster distances, our analysis shows that with a clustering number of 2, it provides the most stable partition of individuals (i.e., the higher the index, the better the clustering results) (see Table 1). However, since we are more interested in a sophisticated clustering of individuals than the dichotomy grouping, we find that when the clustering number is 5, it provides a second-best alternative yet much richer partition of individuals' daily traces.

Table 1. Cluster numbers (*k*) and the Dunn's Index (DI)

k	**2**	**3**	**4**	**5**	**6**
DI	2.431	1.501	1.853	2.017	1.253
k	**7**	**8**	**9**	**10**	**11**
DI	1.288	1.365	1.38	1.387	1.230
k	**12**	**13**	**14**	**15**	**16**
DI	1.370	1.059	1.063	1.088	1.092

Figure 3 shows the clustering results of individuals' spatial-temporal traces with k=5 in space. The left panel of the figure shows a 2-D view of the clustering results with each color representing a cluster, and x-, y-axes representing longitude and latitude; while the right panel of the figure shows a 3-D view of the same clustering results, with an additional vertical z-axis representing the time dimension. This type of 3-D view of individuals' traces is called spatiotemporal prism by geographers [35], and is visually helpful to understand people's movement in space and time.

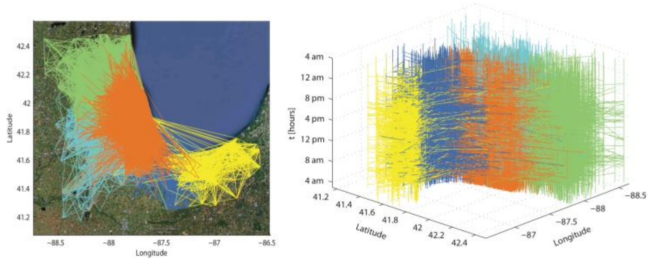


Figure 3 Clustering of individuals' daily traces on an average weekday in Chicago (cluster number=5).

As we know, in people's daily life, their travel from one destination to another in space and time is not usually confined by administratively defined boundaries such as county boundaries. Therefore clustering a region into subdivision by using individuals' spatial-temporal traces can reveal underlying inherent connections in the region better than using administratively defined areas such as counties.

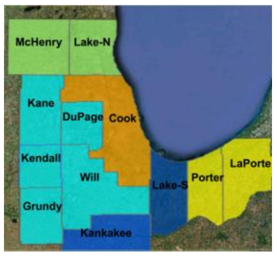


Figure 4 Counties in the Northeastern Illinois Region and Northwestern Indiana Region

Figure 4 shows the geographic location of counties that define the Chicago Metropolitan. From the geographical coverage of the clustered traces in Figure 3, we find that Cluster #1, depicted in dark blue, covers the Lake County (south to Cook) and the southeast corner of Cook County. Cluster #2, depicted in cyan, covers most area of Du Page, the north part of Will and Grundy, and eastern side of Kane and Kendall counties, and the northern part of Grundy County. Cluster #3, depicted in green, covers the north part of Cook County, the Lake County (north to Cook), and the McHenry County. Cluster # 4, depicted in yellow, covers the Porter County and LaPorte County (in the State of Indiana). Cluster # 5, depicted in orange, covers the center of the Cook County, or the City of Chicago. It is interesting to analyze further the intrinsic reasons of this regional clusters defined by the user traces. It may be possible to find some demographic or economic reasons inherent in these aggregations.

5. URBAN SPATIAL-TEMPORAL STRUCTURE BY REGION, ACTIVITY PATTERN AND TYPE

With the previous analysis we obtained clusters of individuals by their daily activity patterns and their spatial-temporal traces (in Section 4). Together with the proposed measurement (i.e., time-cumulative spatial activity density) to estimate urban spatial-temporal structure (in Section 3), we are able to further explore the detailed urban spatial-temporal structure for individuals with different daily activity types defined as students, regular workers, early workers, afternoon workers, stay-at-home, morning adventurers, afternoon adventurers and overnight adventurers. We can observe now how these types of individuals distribute across different clustered sub-regions based on their time-cumulative spatial activity density.

Due to limited space of the paper, and the importance of the City of Chicago, in terms of intensity of various activities (shown in subsection 3.2), as a demonstration, in this section we further explore the spatial temporal structure for individuals whose daily activities are heavily concentrated in this sub-division of the region (i.e., Cluster # 5 obtained from the trace clustering result in subsection 4.2.2), and focus on five types of activities for the eight types of individuals (obtained from the activity pattern clustering results in subsection 4.2.1). We first discuss the spatial distribution patterns of activity intensities on home, work and school activities for individuals engaged in a fixed daily activity, namely: students and workers (including early workers, early-bird workers and afternoon workers) in subsection 5.1. Then discuss the spatial distribution patterns of intensity of the same five types of activities for stay-at-homes and the three types of adventurers (i.e., morning, afternoon and overnight adventurers). The latter group includes individuals that are not traditionally classified by activity patterns and are much harder to model in urban simulators.

5.1 The Students and Workers

As depicted in Figure 5, we find that the spatial distribution of home activities for students and workers are quite similar—heavily concentrated in the downtown area and northern part of the city of Chicago (Figure 5, row 1). It also shows similar patterns in the work activity intensity for the three types of workers, but quite different for the students (with multiple centers in the west part of the city) (Figure 5, row 2).

The spatial distribution of school activity intensity for students matches in a very similar way to that of their home activity intensity. While for regular workers and early-bird workers, there are two centers—one in the north part of the city, and one in the downtown area of the city. For the afternoon workers, the school activity intensity is very weak, but mostly concentrated in the far northern and western edge of the city (Figure 5, row 3).

Shopping activity intensity for the regular and early-bird workers are similar—concentrated in downtown Chicago; and with two additional sub-centers for the students—one in the north, and one in the northwest (around the O'Hare airport). And there are two centers hand-by-hand in the center of Chicago for the afternoon workers.

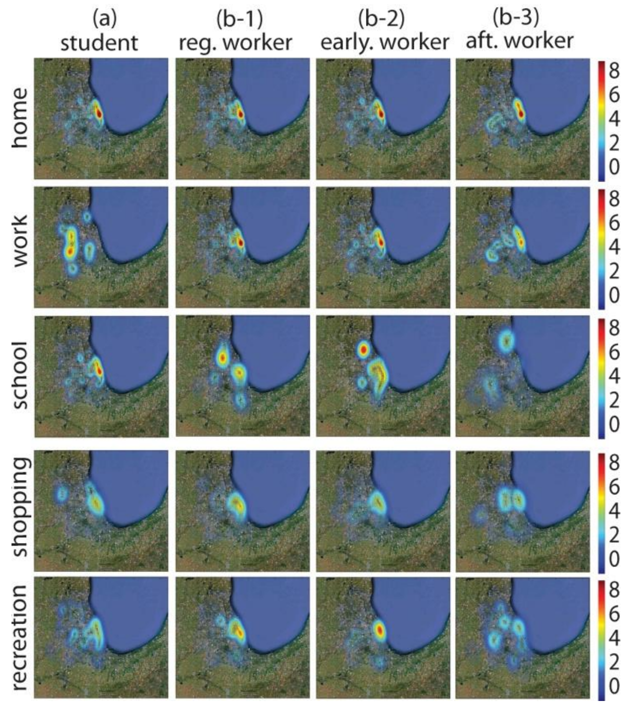


Figure 5 Time-cumulative spatial densities of home, work, and school activities in the Chicago sub-region by (a) students, (b-1) regular workers, (b-2) early-bird workers, and (b-3) afternoon workers on an average weekday.

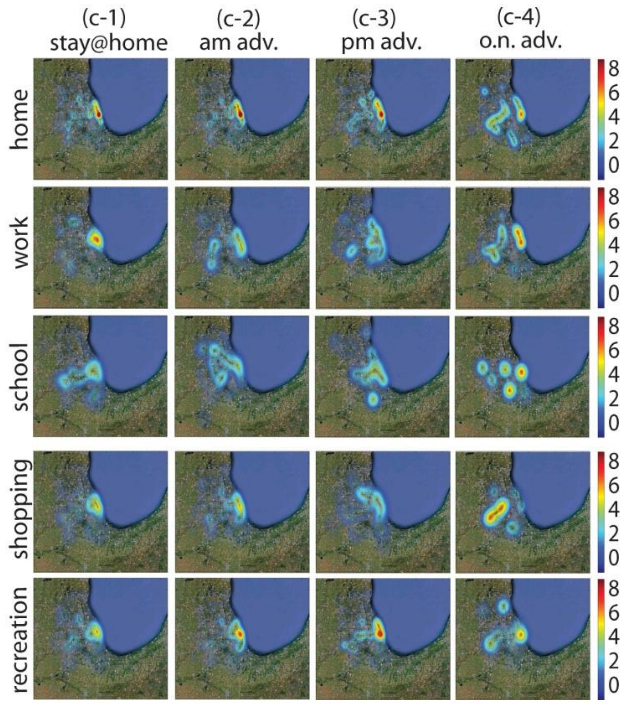


Figure 6 Time-cumulative spatial densities of shopping/ errands & recreation/entertainment activities in the Chicago sub-region by (c-1) stay-at-home, (c-2) morning adventurers, (c-3) afternoon adventurers, and (c-4) overnight adventurers.

Interestingly, we see that the intensity of recreational activity for regular and early-bird workers are similar in geographical location (concentrated in downtown Chicago) compared to their shopping activity, but with stronger intensity— meaning that for these two types of workers, there are either more of them doing recreational (e.g., eating out with friends)/entertainment activities and/or they spend longer time in these activities than in shopping on weekday in the downtown area (Figure 5, row 4). The spatial distribution of recreation/entertainment activity intensity for afternoon workers are more dispersed in space and time than their counterparts of regular and early-bird workers. For students, their recreational activity centers are distributed along the lake shore from north to downtown Chicago, and also in the western part of the city.

To summarize, similar spatial distribution patterns of intensity exist for students' home and school activity, as well as for workers' home and work activity. In terms of the spatial distribution of shopping and recreational activity intensities, they are quite similar for regular and early-bird workers (concentrated in the city center), but exhibit diverse and multiple centers for the afternoon workers (which can be explained by their space-time flexibility during the day compared with regular and early-bird workers). For students, their spatial distribution of recreational activity intensity is similar to that of their school activity, but with lower density, and their shopping activity are more concentrated in the center and northern part of the city, as well as in the northeastern corner of the city near the O'Hare airport.

5.2 The Stay-at-Homes and Adventurers

Different from the students and workers, who spend most of their day on (spatially and temporally constrained) activities of school and work, adventurers have more time flexibility; and the stay-at-homes are more limited in terms of their spatial flexibility.

We can see from Figure 6 that the spatial distributions of activity intensities of various types (except for home activity) are very diverse in space for the adventurers, but very uniformly concentrated for the stay-at homes. For the morning and afternoon adventurers, the second most intense activity right after the home activity is recreation/entertainment, and they are heavily concentrated in the city center, and moderately concentrated in both the north and south parts of the city for morning adventures, and only in the north for the afternoon adventurers.

Shopping activities for the overnight adventures are heavily concentrated along the southwestern corridor, and their recreational activities are also distributed in the far north part of the city.

The school and work activity intensity distribution in space for morning and afternoon adventurers are even more diverse and dispersed compared with their shopping and recreational activities, which means that very few adventurers conduct work/school activities, and their spatial distribution is more spread in the city than concentrated in the downtown area.

Interestingly, the spatial distributions of home and work activity intensities for the overnight adventures are very similar. But that of their school activity intensity is very diverse and spreads over different parts of the city.

6. CONCLUSIONS

This paper introduces a new concept of urban spatial-temporal structure, which expands the traditional theories of urban structure, and offers a framework to study and analyze it using activity-based travel survey data. We discuss the advantages of travel survey data in terms of their richness in revealing individuals' activity types, space and time presence, and we provide a new perspective on how to mine survey data as complements to the recently emerging massive urban sensing data.

With the introduction of spatial distribution of activity intensity, measured by time-cumulative spatial activity density, and the employment of the kernel density estimator, we provide an approach to analyze the proposed urban spatial-temporal structure.

In order to discover similarities and differences in human activity patterns and spatial-temporal traces, we apply *K*-means algorithm via PCA method to cluster individuals into groups. By estimating and visualizing the time-cumulative spatial densities of various activities by person types (obtained from the clustering of daily activity patterns) in one of the sub-regions of the Chicago Metropolitan Area (identified from the clustering of individuals' traces), we are able to explore the diverse spatial-temporal structure of Chicago. Due to space limit, we do not demonstrate the results for the rest of the four sub-regions in the Chicago metropolitan area in this paper. However, abundant information obtained from other sub-regions will help planers gain solid understandings on "how different sub-regions of the metropolitan area have been utilized by different types of individuals for different activity types".

Answering these questions will be essential for urban planners and scholars to understand the dynamics and complexity of the polycentric city, and this paper offers new insights for urban planning through the estimates of spatial distributions of various activity intensities for different types of individuals. As activity intensity is very closely related to energy consumption, business opportunity and infrastructure usage, the measurements and approaches proposed here will be helpful for urban planners to design sustainable cities in the future.

7. ACKNOWLEDGMENTS

This research was funded in part by the MIT Department of Urban Studies and Planning, by the US Department of Transportation Region One University Transportation Center, and by the Singapore National Research Foundation (NRF) through the Singapore-MIT Alliance for Research and Technology (SMART) Center for Future Mobility (FM).

8. REFERENCES

[1] Batty, M. *Cities as Complex Systems: Scaling, Interactions, Networks, Dynamics and Urban Morphologies* Springer, Berlin, DE, 2009.

[2] Portugali, J., Meyer, H., Stolk, E. and Tan, E. *Complexity Theories of Cities Have Come of Age: An Overview With Implications to Urban Planning and Design*. Springer, Heidelberg, 2012.

[3] Anas, A., Arnott, R. and Small, K. A. Urban Spatial Structure. *Journal of Economic Literature*, 36, 3 1998, 1426-1464.

[4] Horton, F. E. and Reynolds, D. R. Effects of Urban Spatial Structure on Individual Behavior. *Economic Geography*, 47, 1 1971, 36-48.

[5] Florida, R., Gulden, T. and Mellander, C. The rise of the mega-region. *Cambridge Journal of Regions, Economy and Society*, 1, 3, November 1, 2008, 459-476.

[6] Greene, D. L. Recent Trends in Urban Spatial Structure. *Growth and Change*, 11, 1, 1980, 29-40.

[7] McMillen, D. P. and McDonald, J. F. A Nonparametric Analysis of Employment Density in a Polycentric City. *Journal of Regional Science*, 37, 4, 1997, 591-612.

[8] Gordon, P., Richardson, H. W. and Wong, H. L. The distribution of population and employment in a polycentric city: the case of Los Angeles. *Environment and Planning A*, 18, 1986, 161-173.

[9] Mahizhnan, A. Smart cities: The Singapore case. *Cities*, 16, 1 1999, 13-18.

[10] Ratti, C. and Townsend, A. The Social Nexus. *Scientific American* 305, 16 August 2011, 6.

[11] Song, C., Qu, Z., Blumm, N. and Barabási, A.-L. Limits of Predictability in Human Mobility. *Science*, 327, 5968, February 19 2010, 1018-1021.

[12] Gonzalez, M. C., Hidalgo, C. A. and Barabasi, A.-L. Understanding individual human mobility patterns. *Nature*, 453, 7196, 2008, 779-782.

[13] Calabrese, F., Colonna, M., Lovisolo, P., Parata, D. and Ratti, C. Real-Time Urban Monitoring Using Cell Phones: A Case Study in Rome. *IEEE Transactions on Intelligent Transportation Systems*, 12, 1 2011, 141-151.

[14] Yuan, J., Zheng, Y. and Xie, X. *Discovering Regions of Different Functions in a City Using Human Mobility and POIs*, Beijing, 2012.

[15] Zheng Y., Zhang L., Xie X. and Y., M. W. *Mining interesting locations and travel sequences from gps trajectories*. Madrid, 2009.

[16] Nature Editorial A flood of hard data. *Nature*, 453, 7196 2008, 698-698.

[17] Becker, R. A., Caceres, R., Hanson, K., Loh, J. M., Urbanek, S., Varshavsky, A. and Volinsky, C. A Tale of One City: Using Cellular Network Data for Urban Planning. *Pervasive Computing, IEEE*, 10, 4, 2011, 18-26.

[18] Glaeser, E. L., Kolko, J. and Saiz, A. Consumer city. *Journal of Economic Geography*, 1, 1, January 1, 2001, 27-50.

[19] Eagle, N. and Pentland, A. Eigenbehaviors: identifying structure in routine. *Behavioral Ecology and Sociobiology*, 63, 7 2009, 1057-1066.

[20] Gandia, R. *City outlines travel diary plan to determine future transportation needs* Sun Media, Calgary, Alberta, Canada, 2012.

[21] Hägerstrand, T. Reflections on "what about people in regional science?". *Papers in Regional Science*, 66, 1, 1989, 1-6.

[22] Yu, H. and Shaw, S.-L. Exploring potential human activities in physical and virtual spaces: a spatio-temporal GIS approach. *International Journal of Geographical Information Science*, 22, 4, 2008, 409 - 430.

[23] Harvey, A. and Taylor, M. Activity settings and travel behaviour: A social contact perspective. *Transportation*, 27, 1, 2000, 53-73.

[24] Hanson, S. and Hanson, P. Gender and Urban Activity Patterns in Uppsala, Sweden. *Geographical Review*, 70, 3, 1980, 291-299.

[25] Hanson, S. and Kwan, M.-P. *Transport: Critical Essays in Human Geography*, Ashgate, Aldershot, 2008.

[26] Kwan, M.-p. and Lee, J. *Geovisualization of Human Activity Patterns Using 3D GIS: A Time-Geographic Approach*. Oxford University Press, Oxford, 2003.

[27] Chicago Metropolitan Agency for Planning *Chicago Travel Tracker Household Travel Inventory*, 2008.

[28] Wand, P. and Jones, C. *Kernel Smoothing*. Chapman & Hall, 1995.

[29] Kwan, M.-P. and Hong, X.-D. Network-based constraints-oriented choice set formation using GIS. *Geographical Systems*, 5, 1998, 139-162.

[30] Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., Zhou, Z.-H., Steinbach, M., Hand, D. and Steinberg, D. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14, 1, 2008, 1-37.

[31] Huang, Z. Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. *Data Mining and Knowledge Discovery*, 2, 3, 1998, 283-304.

[32] Ding, C. and He, X. K-means clustering via principal component analysis. In *Proceedings of the Proceedings of the twenty-first international conference on Machine learning*, ACM, 2004. Banff, Alberta, Canada.

[33] Zha, H., Ding, C., Gu, M., He, X. and Simon, H. Spectral relaxation for K-means clustering. *Advances in Neural Information Processing Systems*, 14 (NIPS'01) 2001, 1057–1064.

[34] Jiang, S., Ferreira, J. and González, M. Clustering daily patterns of human activities in the city. *Data Mining and Knowledge Discovery*, April 23, 2012, 1-33.

[35] Miller, H. J. A Measurement Theory for Time Geography. *Geographical Analysis*, 37, 1, 2005, 17-45.

User Oriented Trajectory Similarity Search

Haibo Wang
School of Information Technology & Electrical Engineering, The University of Queensland
uqhwan15@uq.edu.au

Kuien Liu
The Institute of Software
Chinese Academy of Sciences
kuien@iscas.ac.cn

ABSTRACT
Trajectory similarity search studies the problem of finding a trajectory from the database such the found trajectory most similar to the query trajectory. Past research mainly focused on two aspects: shape similarity search and semantic similarity search, leaving personalized similarity search untouched. In this paper, we propose a new query which takes user's preference into consideration to provide personalized searching. We define a new data model for this query and identify the efficiency issue as the key challenge: given a user specified trajectory, how to efficiently retrieve the most similar trajectory from the database. By taking advantage of the spatial localities, we develop a two-phase algorithm to tame this challenge. Two optimized strategies are also developed to speed up the query process. Both the theoretical analysis and the experiments demonstrate the high efficiency of the proposed method.

1. INTRODUCTION
Trajectory similarity search is a hot research topic in recent years due to its broad range of applications, such as friend recommendations, trip planning, traffic analysis and carpooling. It studies the problem of finding a trajectory from the trajectory database such that the found trajectory is most similar to the query trajectory. A fair amount of research works were involved in the past decades on this topic, some [1, 15, 11, 16, 9, 4, 3, 13] focused on shaped-based similarity search, in which each trajectory consists of a sequence of equally important sample points; while others [17, 10, 20, 19, 12, 14] focused on the semantic aspects, in which each trajectory is represented as a sequence of meaningful entities, such as POIs, locations or regions.

The similarity for the shape-based search depends on how many common parts the trajectories share, while for the semantic-based search it depends on how many *significant* common parts shared. Clearly, the second kind of similarity is more reasonable as it considers more of those significant parts instead of treating each part equally. To measure the second kind similarity, it employes variuos huristic methods to identify which parts are important and meaningful and this, inherently, has two disadvantages:

- Due to the quality of the training data or the imperfection of the mining method, it cannot gurantee to find all of the important parts. Some significant places may be missed.
- Among all those important parts, they are not equally important to all users. In other words, this method cannot provide personalized service to different users.

To overcome these disadvantages, we propose a new type of query: *user-oriented trajectory similarity search*, in which each user can specify the relative importance of each part in the query trajectory. This new query has the advantages that:

- It can provide personalized query without missing those important parts, as whose importance have been designated by the user.
- It supports flexible and even highly complex query patterns. For instance, the relative importance of all parts in the query trajectory follows standard normal distribution.

The key challenge here is the efficiency issue. More specifically, given a user specified query, how to efficiently find out the most similar trajectory from the trajectory database which may contain huge number of trajectories.

Unfortunately, no existing solutions can readily be used to conquer this challenge. Vlachos et al. in [13] explore discovering similar multidimensional trajectories by building a cluster-based hierarchical indexing tree. This method, however, suffers from finding good clusters and representing points to build the hierarchical indexing tree on which the performance heavily depends. Besides, it considers shape similarity allowing spatial shifting between trajectories, which is totally different from our settings: no spatial shifting is allowed. Chen et al. in [3] exploit edit distance to measure trajectory similarity and provides three pruning techniques to efficiently retrieve the most similar trajectory. But under the user specified query in which the relative importance of every sample points are considered, these pruning techniques are no longer hold. A typical example is that one pruning technique works by bounding the number of *common Q-grams* for two sequences within *edit distance* k, where k is supposed to be an integer, while in our settings, due to the relative importance involved, this condition is hard to be guaranteed.

In this work, our major contributions are:

1. We propose a new type of trajectory similarity search with the merits of considering users' preferences and supporting personalized query.

2. We carefully define a new data model for this query and develop an efficient method to answer the similarity query. Two optimized strategies are also developed to speed up the query process.

3. We do theoretical analysis, carry out experiments on real dataset and both demonstrate the high efficiency of our method.

The remainder of this paper is organized as follows. In section 2, we define the data model and the problem. We then develop a two-phase algorithm and two optimized strategies in Section 3. Theoretical analysis and the experiments are conducted in Section 4 and Section 5, respectively. We discuss related work in section 6 and conclude this paper in section 7.

2. DATA MODEL AND PROBLEM

With the advancement of the modern GPS technologies, it is not unreasonable to assume that all trajectories in the database have similar sampling rate. If not, we can interpolate the trajectory data to make them satisfied the assumption condition.

Let $T = \{p_1, p_2, \cdots, p_n\}$ be a data trajectory, where $|T|$ denotes the size of sample points in T. Each sample point $p_i = \langle lon_i, lat_i\rangle$ is a pair of real values, where lon_i and lat_i correspond to longitude and latitude respectively. A user trajectory database is a set of data trajectories $DB = \{T_1, T_2, \cdots, T_N\}$, where N is the number of trajectories in DB. Let $Q = \{q_1, q_2, \cdots, q_m\}$ be the query trajectory, in which each sample point q_i is a triple of real values, namely $\langle lon_i, lat_i, w_i\rangle$, where $w_i \geq 0$ is the user assigned weight indicating the importance of the sample point. We say sample points q_i and p_j *matched* if $|lon_i - lon_j| \leq \epsilon$ and $|lat_i - lat_j| \leq \epsilon$; here the ϵ is the matching threshold.

There are several methods to measure trajectory similarity: Euclidean-based methods [1, 15, 11], DTW [16, 9, 4], ERP [2], EDR [3] and LCSS [13]. Among these methods, the first three (Euclidean-based methods, DTW and ERP) are sensitive to noise; the EDR concerns more about the dissimilarities between trajectories, as it only penalizes the gaps while ignores their common parts; the LCSS, on the other hand, is more robust to noise and more accurate to compute the similar parts, because just like its name (Longest Common Subsequence), it measures how many common parts two trajectories share and a larger value implies a better similarity. Here in this paper we use the same idea behind LCSS to measure the similarity between a data trajectory and a query trajectory. Specifically, given the aforementioned T and Q, we define the similarity measure *Heaviest Common Subsequence (HCSS)* between them as the weighted sum of their longest common subsequence and denote it as $HCSS(T, Q)$. The specific value can be derived from the following recursive computation:

$$\begin{cases} 0 & \text{if } n = 0 \text{ or } m = 0, \\ w_1 + HCSS(Rest(T), Rest(Q)) & \text{if } p_1,\ q_1 \text{ are matched}, \\ max \begin{cases} HCSS(Rest(T), Q), \\ HCSS(T, Rest(Q)) \end{cases} & \text{otherwise} \end{cases}$$

where $Rest(.)$ denotes the rest part of a trajectory with the first sample point removed.

Problem. Given a query trajectory Q, find T from the trajectory database DB such that $HCSS(T, Q) > 0$ and

$$HCSS(T, Q) \geq HCSS(T', Q), \forall T' \in DB \text{ and } T' \neq T.$$

In the extreme case that $\forall T \in DB$, $HCSS(T, Q) = 0$, we would say no trajectory in the database is similar to the query trajectory and therefore have no obligation to return any trajectory.

3. QUERY PROCESSING

One naive solution for the problem is to compute the similarity values with every trajectory in the database, then choose the one with the largest $HCSS$ value. The cost of this method is prohibitively expensive simply because it has to load every data trajectory from the external memory into internal memory in order to compute the $HCSS$ value, which would introduce tons of IOs, not mention the computation cost is quadratic to trajectory length.

To reduce the IO as well as the computation cost, we have the following observation.

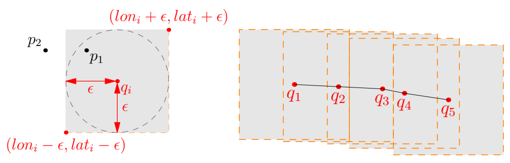
Observation 1 *Trajectories in the database are spatially scattered and the query trajectory is only within some limited area.*

Based on this observation, instead of retrieving all trajectories from the external memory, we only need to retrieve those trajectories having at least one sample point included by a small range that covers the query trajectory as a candidate set and it is highly likely that the candidate set contains the most similar trajectory. The key issue here is how large the *covering range* should be to make the candidate set surely containing the most similar trajectory. If it is too large, it may retrieve excessive number of trajectories at the price of a fair amount unnecessary IOs; on the other hand, if it is too small, it may miss the most similar trajectory as well. Therefore a proper covering range should be provided so that it dose not produce too many unnecessary IOs while at the same time it can still guarantee the correctness of the candidate set without introducing false dismissals.

Definition 1. *(ϵ-buffer)* Given a query trajectory Q, we define its *ϵ-buffer* $\mathcal{B}_\epsilon(Q)$ as the *union* of rectangular areas of all sample points, where the *rectangular area* of a sample point q_i is $[(lon_i - \epsilon, lat_i - \epsilon), (lon_i + \epsilon, lat_i + \epsilon)]$ and ϵ is the threshold (Figure 1).

THEOREM 1. *The ϵ-buffer of query trajectory Q can serve as a covering range without introducing false dismissals.*

PROOF. Let T be the most similar trajectory. From the problem definition we know $HCSS(T, Q) > 0$, which implies T must have at least one its sample point included by



$(lon_i+\epsilon, lat_i+\epsilon)$
p_2
p_1
ϵ
q_i
ϵ
$(lon_i-\epsilon, lat_i-\epsilon)$
q_1
q_2
q_3
q_4
q_5

(a) (b)

Figure 1: Examples of rectangular area and $\mathcal{B}_\epsilon(Q)$

the covering range $\mathcal{B}_\epsilon(Q)$ and therefore is included in the returned candidate set. □

Theorem 1 indicates us that we only need to search the query trajectory's ϵ-buffer to find the most similar trajectory, and in terms IO cost, it is a dramatic improvement over the naive method in which the correctness is achieved by searching the whole data space. Inspired by the above analysis, we get our first algorithm to find the most similar trajectory.

Before presenting the algorithm, we claim that every trajectory T in the database has a unique id called *tid*, and each sample point in T contains the *tid* information. If we get a sample point, we then know which trajectory it belongs to. Also we use an R-tree [7] to index all sample points in the trajectory database; by doing this, we can conduct range search on the R-tree and thus reduce the search cost.

Algorithm 1: *MST(TR, Q, ϵ)*

```
Input: R-tree root $TR$, query trajectory $Q$, threshold $\epsilon$
Output: The unique id of the most similar trajectory
1  $id \leftarrow \infty$;
2  /* Filter phase */
3  $List_\mathcal{C} \leftarrow \phi$; // the candidate set
4  foreach $q_i$ in $Q$ do
5      $rect \leftarrow [(lon_i-\epsilon, lat_i-\epsilon), (lon_i+\epsilon, lat_i+\epsilon)]$;
6      $List_\mathcal{P} \leftarrow TR$.rangeSearch($rect$);
7      foreach sample point $p$ in $List_\mathcal{P}$ do
8          $tid \leftarrow p.getTid()$;
9          if $List_\mathcal{C}.contain(tid)$ is not ture then
10             $List_\mathcal{C}.add(tid)$;
11 /* Refinement phase */
12 $distance \leftarrow -\infty$;
13 foreach $tid$ in $List_\mathcal{C}$ do
14     $T \leftarrow$ retrieve the $tid$th trajectory from external memory;
15     $hcss \leftarrow HCSS(T, Q)$;
16     if $distance < hcss$ then
17         $distance \leftarrow hcss$;
18         $id \leftarrow tid$;
19 return $id$;
```

The algorithm adopts a two phases strategy: filter and refinement. In the filter phase (lines 2-10), we generate a candidate set based on R-tree range search (i.e., search the area covered by $\mathcal{B}_\epsilon(Q)$) without introducing false dismissals; in the refinement phase (lines 11-19), we retrieve each trajectory appearing in the candidate set from the external memory, compute the exact similarity values, choose the one with the largest *HCSS* value and return it as the most similar trajectory. The correctness of the algorithm is guaranteed by Theorem 1.

3.1 Adaptive Filter Strategy

If we look carefully at the filter phase, we will find that Algorithm 1 conducts range search for every sample point in the query trajectory. This, however, may not necessary, as indicated by the following observation.

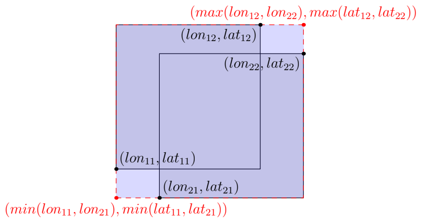
Observation 2 *The query trajectory consists of a series of sequential sample points which are usually spatially close.*

As a result, there are many overlaps between the range searches conducted in Algorithm 1. In Figure 1(b), we can clearly see the overlaps between consecutive range searches (rectangular areas). The multiple searching of these overlapping areas is obviously undesired as it will introduce unnecessary IO cost. To remedy this, we introduce the concept of *grouping consecutive query points*.

Definition 2. ***(Grouped area)*** For any two consecutive points q_i, q_{i+1} in the query trajectory, let their corresponding *rectangular areas* are $[(lon_{i,1}, lat_{i,1}), (lon_{i,2}, lat_{i,2})]$ and $[(lon_{i+1,1}, lat_{i+1,1}), (lon_{i+1,2}, lat_{i+1,2})]$, respectively. We group them together and define their *grouped area* $\mathcal{G}(q_i, q_{i+1})$ is as $[(lon_1, lat_1), (lon_2, lat_2)]$, where

$$\begin{aligned} lon_1 &= min(lon_{i,1}, lon_{i+1,1}) \\ lat_1 &= min(lat_{i,1}, lat_{i+1,1}) \\ lon_2 &= max(lon_{i,2}, lon_{i+1,2}) \\ lat_2 &= max(lat_{i,2}, lat_{i+1,2}) \end{aligned}$$

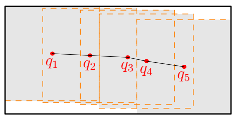
In the same way, we define grouped area for $k+1$ consecutive points q_i, q_{i+1}, $\cdots$, q_{i+k} as $\mathcal{G}(q_i, q_{i+k})$. Figure 2 shows an example, in which two smaller rectangular areas are grouped together to form the larger one with red dashed edges.



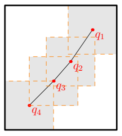
$(max(lon_{12}, lon_{22}), max(lat_{12}, lat_{22}))$
(lon_{12}, lat_{12})
(lon_{22}, lat_{22})
(lon_{11}, lat_{11})
(lon_{21}, lat_{21})
$(min(lon_{11}, lon_{21}), min(lat_{11}, lat_{21}))$

Figure 2: Grouped area $\mathcal{G}(q_1, q_2)$

To overcome the drawback caused by redundant IOs, one solution is to group certain number of consecutive points together so that we can have just one range search to cover the union of their rectangular areas. As illustrated in Figure 3(a), instead of conducting 5 range searches, we group the 5 query sample points together and only search once the grouped area $\mathcal{G}(q_1, q_5)$ bounded by the black box. This method works in the reason that it covers more than the original searching area (i.e., the *ϵ-buffer* of the query trajectory) and therefore guarantees no false dismissals, while the cost saved from overlapping search is *quite enough* to compensate the extra cost from searching new *dead space* (the white space between the shaded area and the black bounding box in Figure 3(a)).



(a)



(b)

Figure 3: Points distribution vs. dead space

A problem with this solution is how many consecutive points shall be grouped together so that the extra cost introduced by searching new dead space will not cancel out the saved cost. If we group too few points, although the extra cost is small, it may not effective to reduce the redundant IO cost; if we group too many points, although the saved cost may be a lot, it may also introduce too much new dead space and the search cost on which would turn back canceling out the benefit earned from the saved cost. Therefore, in pursuit of keeping the total range search cost as small as possible, a proper grouping strategy shall be developed to ensure the balance between saving redundant IO cost and introducing new cost.

One strategy is simply grouping fixed number of consecutive points. This method, however, has the risk of bringing in too much dead space if the consecutive points are distributed as Figure 3(b) showing. To get rid of the risk, we need to take the grouping process under control and develop an *adaptive* grouping strategy. More specifically, we need to develop a *bound* for the search space so that after grouping, the grouped area will not surpass the boundary, thereby *bounding* the introduced dead space. Under this condition, we can aggressively group those consecutive points.

Definition 3. *(**α-boundary**)* Given a polyline $L = \{p_1, p_2, \cdots, p_n\}$ and $\alpha \geq 0$, let every line segment (p_i, p_{i+1}) be moved α distance along both directions that perpendicular to it to get the upper and lower lines, we define the **area** *between all upper and all lower lines* plussing *the two outer half circles centered at the end points of L with radius α* as the ***α-boundary*** of L, or $\mathcal{D}_\alpha(L)$, as illustrated in Figure 4.

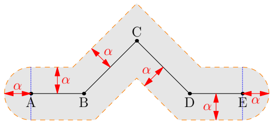


Figure 4: $L = \{A, B, C, D, E\}$ and $\mathcal{D}_\alpha(L)$

We can easily see that the *ϵ-buffer* of the query trajectory Q are covered by Q's *$\sqrt{2}\epsilon$-boundary*, namely $\mathcal{B}_\epsilon(Q) \subseteq \mathcal{D}_{\sqrt{2}\epsilon}(Q)$, because the perpendicular distance from every point in $\mathcal{B}_\epsilon(Q)$ to the polyline Q is less or equal than $\sqrt{2}\epsilon$. However, we cannot use this boundary as a bound to do grouping work owning to the fact that it is too tight to group any consecutive points. In other words, we need a more looser boundary.

Line simplification technique like Douglas-Peucker [6] algorithm may drop us a hint on this. Given a distance threshold $\delta > 0$ and a polyline specified by a sequence of n points $\{q_1, q_2, \cdots, q_n\}$, the goal of Douglas-Peucker algorithm is to derive a simplified polyline to which the perpendicular distance of every point in the original polyline is at most δ. The algorithm initially constructs the line segment (q_1, q_n). It then identifies the point q_i furthest to the line. If this point's perpendicular distance to the line is within δ, it returns (q_1, q_n) and terminates. Otherwise it recursively applies the same process on the two sub-polylines $\{q_1, \cdots, q_i\}$ and $\{q_i, \cdots, q_n\}$.

After simplification, the original polyline is completely lying within the *δ-boundary* of the simplified polyline, and this implies if query trajectory Q is the original polyline, then Q's *ϵ-buffer* is completely within the *$(\sqrt{2}\epsilon + \delta)$-boundary* of the simplified polyline Q', or formally, $\mathcal{B}_\epsilon(Q) \subseteq \mathcal{D}_{\sqrt{2}\epsilon+\delta}(Q')$.

$\mathcal{D}_{\sqrt{2}\epsilon+\delta}(Q')$ can thus serve as an acceptable boundary for grouping work. In specific, for every simplified segment, we start grouping all those consecutive points from its start point aggressively until the next to be grouped area surpasses the boundary. After grouping, if there are still some points between the simplified line segment left ungrouped, we then start the next grouping process until all points end up within their own grouped areas. Algorithm 2 shows this adaptive grouping strategy.

Algorithm 2: *generateGroupedAreas(Q, ϵ, δ)*

```
Input: query trajectory $Q$, threshold $\epsilon$, threshold $\delta$
Output: A list of grouped areas
1  $List_{\mathcal{G}} \leftarrow \phi$;
2  $Q' \leftarrow DouglasPeucker(Q[1:m], \delta)$;
3  $j \leftarrow 2$;
4  $rect \leftarrow [(+\infty, +\infty), (-\infty, -\infty)]$;
5  for $i \leftarrow 1$ to $m-1$ do
6      if $i < Q'[j].subscript$ then
7          /* Grouping in the same simplified segment*/
8          $rect_{curr} \leftarrow$ rectangular area of $Q[i]$;
9          $rect_{temp} \leftarrow$ grouped area of $rect_{curr}$ and $rect$;
10         if $rect_{temp}$ within $\mathcal{D}_{\sqrt{2}\epsilon+\delta}(Q')$ then
11             $rect \leftarrow rect_{temp}$;
12         else
13             $List_{\mathcal{G}}.add(rect)$;
14             /* Start the next grouping process */
15             $rect \leftarrow$ rectangular area of $Q[i]$;
16     else
17         $List_{\mathcal{G}}.add(rect)$;
18         /* Start grouping the next simplified segment */
19         $j \leftarrow j+1$;
20         $rect \leftarrow$ rectangular area of $Q[i]$;
21 $List_{\mathcal{G}}.add$(rectangular area of $Q[m]$); // The last one
22 return $List_{\mathcal{G}}$;
```

In Algorithm 2, line 10 checks whether the *next to be grouped area $rect_{temp}$* is within the *$(\sqrt{2}\epsilon + \delta)$-boundary* of Q'. It does so by checking whether the perpendicular distances of the four vertices in $rect_{temp}$ to the current simplified line segment $(Q'[j-1], Q'[j])$ are all within $\sqrt{2}\epsilon + \delta$. If yes, then it returns *true*; otherwise it returns *false*.

As an example, for $Q = \{q_1, \cdots, q_9\}$, Figure 5 shows the grouping results, namely $\mathcal{G}(q_1, q_5), \mathcal{G}(q_6, q_8)$ and $\mathcal{G}(q_9, q_9)$, represented by the three red-edged rectangles. Also, the black line $\{q_1, q_6, q_9\}$ in the figure represents the simplified polyline Q' while the black dashed boundary depicts the corresponding $(\sqrt{2}\epsilon + \delta)$*-boundary*.

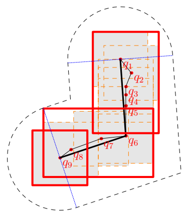


Figure 5: An example of grouped areas

In the best case, we can group all points between a simplified line segment into one grouped area (such as the $\mathcal{G}(q_1, q_5)$ in Figure 5); while in the worst case, we cannot group any two consecutive points at all. Fortunately, the worst case is rare due to the loose boundary derived from line simplification. Overall, the grouping effectiveness is influenced by a variety of factors, including consecutive points distribution, underline trajectory data distribution, and in particular, the value of the distance threshold. Heuristically, setting δ to $\epsilon/2$ achieves good grouping effectiveness as confirmed by the experiments.

According to the result in Figure 5, we now only need to conduct 3 times of range search instead of the originally 9 times to produce the no-false-dismissal candidate set. This, obviously, greatly reduces the search cost as the introduced dead space is much smaller than the repetitive searched overlapping areas. Since we solve the problem caused by overlapping search, we can slightly modify Algorithm 1 to adopt the new *adaptive filter strategy*. Specifically, we call Algorithm 2 between lines 3 and 4 to produce a list of grouped areas $List_{\mathcal{G}}$, replace line 4 with "**foreach** *rect in* $List_{\mathcal{G}}$ **do**" and delete line 5 to get the new algorithm.

3.2 A Better Refinement Strategy

For the refinement phase in Algorithm 1, it simply retrieves every candidate trajectory from the external memory and conducts the quadratic $HCSS$ computation w.r.t. Q to get the most similar trajectory, which, in some extent, bears heavy IO and computation cost. Also, comparing with the original filter strategy, the adaptive filter strategy although successfully reduces the overall range search cost, but it also raises the chance to return more candidate trajectories owing to search the introduced dead space. This may further increase the cost of the refinement phase. Is it possible to reduce this cost?

Observation 3 *Among the large number of returned candidate trajectories in filter phase, only* one *or* a few *trajectories have the chance to be the most similar trajectory.*

This observation suggests us designing a better refinement strategy is not impossible. Such as if we develop an upper bound for $HCSS$, we then can prune most of the unrelated trajectories.

From the adaptive grouping strategy in previous section, after grouping, we know every point in the query trajectory belongs to one and only one grouped area, although between grouped areas some small overlaps may exist. For one grouped area, let k be the number of grouped consecutive points, and let w be the largest weight among the k corresponding weights. Then the product value $k * w$ is called the *group weight* or denoted as $\mathcal{W}$. Each grouped area carries its own *group weight*. For example, the first grouped area $\mathcal{G}(q_1, q_5)$ in Figure 5 grouped five points, if we assume the largest weight among the five corresponding weights is 0.7, then we have $k = 5$, $w = 0.7$, and $\mathcal{G}(q_1, q_5)$ carries the group weight $\mathcal{W} = 5 * 0.7 = 3.5$.

Definition 4. ***(HGSS)*** For any T in the trajectory database, given Q and the corresponding grouped areas $\{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_K\}$, we define the *Heaviest Grouped Subsequence* between them

$$HGSS(T, Q) = \sum_{i=1}^{K} F(i)$$

where $F(i) = \mathcal{G}_i.\mathcal{W}$ if T has at least one sample points in grouped area $\mathcal{G}_i$, otherwise $F(i) = 0$.

LEMMA 1. *$HGSS$ is an upper bound of $HCSS$, that is, $HGSS(T, Q) \geq HCSS(T, Q)$.*

PROOF. Let $Q = \{q_1, q_2, \cdots, q_m\}$, then $HCSS(T, Q) = \sum_{i=1}^{m} f(i)$, where $f(i) = w_i$ if during the computation process q_i contributes w_i to the final $HCSS$ value, otherwise $f(i) = 0$. Consider any grouped area $\mathcal{G}_j(q_s, q_e)$, if T has one sample point within this area, then

$$F(j) = \sum_{i=s}^{e} max\{w_s, w_{s+1}, \cdots, w_e\} \geq \sum_{i=s}^{e} w_i \geq \sum_{i=s}^{e} f(i)$$

otherwise $F(j) = 0 = \sum_{i=s}^{e} f(i)$. Thus, we have $\sum_{i=1}^{K} F(i) \geq \sum_{i=1}^{m} f(i)$. $\square$

Since the $HGSS$ can be obtained in the filter phase with negligible cost, we now can design a better strategy for the refinement process.

As shown in Algorithm 3, in the refinement step, we sort all returned candidate trajectories in descending order by $HGSS$ value (line 12), and then visit the first element and compute the exact $HCSS$ value (lines 15-16). If it no less than the $HGSS$ of the next unvisited trajectory (lines 20-22), we return the one with largest $HCSS$ value among all visited trajectories as the most similar trajectory and then terminate, as the $HCSS$ values of the rest trajectories are no larger than the current $HCSS$ value; otherwise we continue this process with the next unvisited trajectory.

Integrated with the adaptive filter strategy and the better refinement strategy, Algorithm 3 is the final optimized algorithm to our user oriented trajectory similarity search.

4. THEORETICAL ANALYSIS

Our upcoming experiments show that our method can efficiently handle the query process. In this section, we do a theoretical analysis to demonstrate the complexity of the proposed method.

Let the database contain N trajectories and every trajectory on average has the length of n sample points. Let the query

Algorithm 3: *OptimizedMST(TR, Q, ϵ, δ)*

```
Input: R-tree root $TR$, query trajectory $Q$, threshold $\epsilon$,
       threshold $\delta$
Output: The unique id of the most similar trajectory
1  $id \leftarrow \infty$;
2  /* Filter phase */
3  $List_{\mathcal{C}} \leftarrow \phi$; // the candidate set
4  $List_{\mathcal{G}} \leftarrow generateGroupedAreas(Q, \epsilon, \delta)$;
5  foreach $rect$ in $List_{\mathcal{G}}$ do
6      $List_{\mathcal{P}} \leftarrow TR$.rangeSearch($rect$);
7      foreach sample point $p$ in $List_{\mathcal{P}}$ do
8          $tid \leftarrow p.getTid()$;
9          if $List_{\mathcal{C}}.contain(tid)$ is not ture then
10             $List_{\mathcal{C}}.add(tid)$;
11 /* Refinement phase */
12 Sort $List_{\mathcal{C}}$ in descending order by $HGSS$;
13 $distance \leftarrow -\infty$;
14 for $i \leftarrow 1$ to $List_{\mathcal{C}}.length$ do
15     $T \leftarrow$ retrieve the $List_{\mathcal{C}}[i].tid$th trajectory from external
       memory;
16     $hcss \leftarrow HCSS(T, Q)$;
17     if $distance < hcss$ then
18         $distance \leftarrow hcss$;
19         $id \leftarrow tid$;
20     if $hcss \geq List_{\mathcal{C}}[i+1].HGSS$ then
21         /* Prune the rest trajectories */
22         break;
23 return $id$;
```

trajectory have the length of m. Let the average cost retrieving one trajectory from the database be C. As the similarity value $HCSS$ is derived in a dynamic programming manner, its computation cost is $O(mn)$. For the naive method, as it has to retrieve every trajectory from the database and compute the exact $HCSS$ value, its complexity consists IO cost $O(NC)$ as well as computation cost $O(Nmn)$, namely $O(N(C + mn))$.

Let C_R be the average range search cost on R-tree; let ξ_1 be the candidate trajectory rate. The complexity of Algorithm 1 consists two parts: filter cost $O(mC_R)$ and refinement cost $O(\xi_1 N(C + mn)$, therefore it is $O(mC_R + \xi_1 N(C + mn))$.

Let ξ_2 be the actually processed trajectory rate after pruning; let m' be the number of grouped areas. Obviously, $\xi_2 < \xi_1$ and $m' < m$. As in the adaptive filter strategy we bound the introduced dead space, we expect the average range search in Algorithm 3 is almost the same with C_R, or at most constant times of C_R, therefore the filter cost of Algorithm 3 is $O(m'C_R)$, and the complexity of it is $O(m'C_R + \xi_2 N(C + mn))$.

To sum up, the complexities of the three algorithms are shown in Table 1.

Generally, the range searching cost is far less than the cost of retrieval all trajectories from database, i.e., $mC_R \ll NC$, and the rates ξ_1 and ξ_2 are also expected to be small, such as less than 10%, then Algorithm 1 and 3 are far more efficient

Table 1: Algorithm complexity

Algorithm	Complexity
Naive	$O(N(C + mn))$
MST	$O(mC_R + \xi_1 N(C + mn))$
OptimizedMST	$O(m'C_R + \xi_2 N(C + mn))$

than the naive method, as confirmed by our experiments.

5. EXPERIMENTS

In this section, we refer Algorithm 1 as MST, Algorithm 3 as $OMST$, and the naive method as $Naive$. We then design experiments to answer the following questions:

1. Compared with the $Naive$ method, what are the performance of our methods: MST, $OMST$?

2. In terms of saved IO cost, what is the performance of the adaptive filter strategy?

3. In terms of saved cost (both IO and computation), what is the performance of the better refinement strategy?

We measure the IO cost in question 2 with the number of disk blocks that the algorithm visits during the range search; meanwhile, we measure the saved cost in question 3 with the number of trajectories retrieved from disk and processed during the refinement phase.

Since the matching threshold ϵ is application dependent [13], we run several probing programs on each dataset and choose the one close to human observations. Also, we set the distance threshold δ as $\epsilon/2$.

5.1 Settings

Dataset: We use Beijing dataset for the experiment. The dataset is a three-day taxi trajectory dataset whose distribution is shown in Figure 6. After cleaning, the dataset contains 6176841 sample points and consists of 30284 trajectories with lengths varying from 20 to 1400.

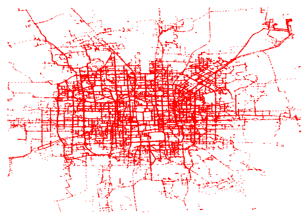


Figure 6: The Beijing Dataset

To mimic the scenario that the dataset is too large to resident in main memory, we put the dataset in external memory and use an R-tree to index it. In the R-tree, we set the page size 4096 bytes, the capacities and the fill factors (both node and leaf) 100 and 70%, respectively.

The query trajectories are divided into six groups whose lengths are 40, 80, 160, 320, 640, 1280, respectively. Each group consists of 50 trajectories and the cost is obtained from the average of the corresponding items. For each query trajectory, we use a random function to weight its sample points.

Environments: All these algorithms are implemented in Java and examined on a Windows XP platform with Intel Core i7 CPU (2.93GHz) and 3.5GB memory.

5.2 Performance

Figure 7(a) shows the query time of the three algorithms in histogram. From it we can see the $Naive$ method takes the longest time for each query. As a matter of fact, even for queries as short as 40, it still takes about 20 seconds to get the answer. While for our proposed method MST, the answering time is significantly shorter. Even for the longest queries, it only takes about 20 seconds to get the answer. The optimized $OMST$ algorithm is the most efficient algorithm. Its answering time is significantly shorter than the corresponding time of MST and drastically shorter than the corresponding time of $Naive$.

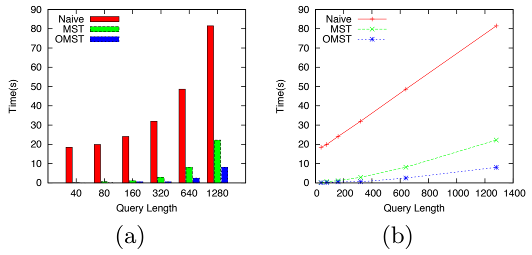


Figure 7: The query time

Figure 7(b) shows the query time in the real length scale. As the computation cost of the similarity value is quadratic to the query length, if we only count on the computation cost, the query time should be quadratic to the length. But the real time is nearly linear. This implies that the IO cost plays an important part in the query process.

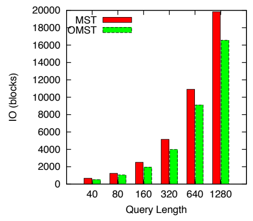


Figure 8: Range query efficiency

Figure 8 illustrates the number of disk blocks that the two algorithm visits during the filter phase with the range queries. Since algorithm $OMST$ adopts the adaptive filter strategy, from the figure, we can see its IO cost is significantly smaller than the corresponding cost in MST, which implies the adaptive filter strategy is quite effective.

In Figure 9, the MST denotes the number of trajectories in the candidate set in Algorithm 1 and also the number of processed trajectories in the refinement phase; the $OMST$ denotes the number of trajectories in the candidate set with the adaptive filter strategy in Algorithm 3 and the $OMST$-*prune* denotes the number of actually processed trajectories in the refinement phase after pruning. We can see the number of trajectories of $OMST$ is significantly higher than that

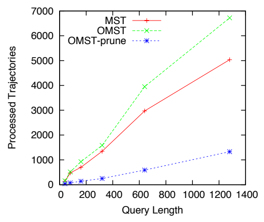


Figure 9: Processed trajectories

of MST due to the introduced dead space in the adaptive filter strategy, while after pruning, the number of actually processed trajectories is much smaller than that of MST and $OMST$, which implies the pruning ability of the upper bound is quite good.

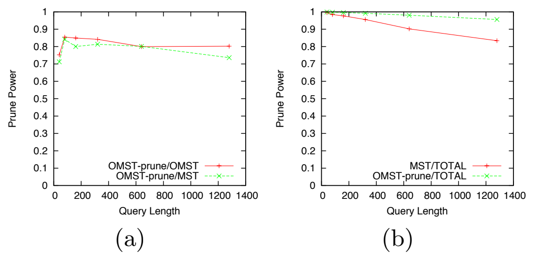


Figure 10: The prune power

Figure 10(a) shows the relative prune power of the pruning techniques over MST and $OMST$. The relative prune power is defined to be the fraction of candidate trajectories being pruned in the refinement phase. We can see both the relative pruning powers in MST and $OMST$ are around 80%, which from another perspective demonstrates the effectiveness of our pruning techniques. Figure 10(b) shows the absolute prune power of Algorithm 1 and Algorithm 3, where the absolute prune power is the fraction of trajectories being pruned in the refinement phase w.r.t. all trajectories in the database. For Algorithm 1, the prune power is above 83%; while for Algorithm 3, the prune power is above 95%. Having the ability to prune most trajectories in the database is the main reason that Algorithm 1 and Algorithm 3 are far more efficient than the Naive method.

6. RELATED WORK

Trajectory similarity search has been studied for decades. From the earliest similarity measures like Euclidean-based distance [1, 15, 11], Dynamic Time Warping (DTW) [16, 9, 4] to the more recently similarity measures like Edit Distance on Real sequence (EDR) [3], Longest Common Subsequence (LCSS) [13], a consider amount of methods have been proposed. Among these methods, Euclidean-based distance and DTW are sensitive to noise, while the EDR and LCSS are more robust and accurate. However, none of these work considers user oriented similarity search, which is exactly what we study in this paper.

In [13] Vlachos et al. suggest using LCSS as the similarity measure, which matches two sequences by allowing them to stretch, without rearranging the sequence of elements but allowing some elements to be unmatched. As a result, the LCSS measure can efficiently handle outliers (or noise) that often exist in trajectories due to sensor failures, error in detection techniques and disturbance signals. A cluster-based indexing is proposed to improve the retrieval efficiency using LCSS. The performance of this indexing method depends on the clustering results. However, due to LCSS not following triangle inequality, it is hard to find good clusters and representing points in the data set [8]. Besides, [13] considers shape similarity allowing spatial shifting between trajectories, which is quite different form our settings: the spatial shifting is strictly prohibited.

Chen et al. in [3] propose EDR as a similarity measure which considers spatial shifting as well as assigning penalties according to the sizes of gaps in between similar shapes. Three pruning techniques are also developed to speed up the query process. But under the condition of user specified query where the relative importance of each sample point is considered, the pruning techniques are no longer hold.

In the query trajectory, if we set all the significant points as weight 1, and other points as weight 0, then this work bears some resemblance to [5] in which Chen et al. explore searching trajectories by specifying a series locations. However,as the number of specified locations may be a few, in our work, we study a more general problem with arbitrary number of sample points (theoretically) where each sample point may pertain to arbitrary weight.

There are also some interesting work [20, 19, 18, 17, 10] using data mining techniques to find the semantic aspects of trajectories. In these work, each trajectory is firstly transformed into a semantic trajectory, then based on semantic representing the system does friends recommendation. Different from these work, our work towards providing personalized recommendation in which the preference (or significance) is designated by users.

7. CONCLUSION

In this paper, we study a new problem of user-oriented trajectory similarity search, in which each user can specify their own important parts in the query trajectory to get the personalized searching results. We identify the efficiency issue as the key challenge and develop a two-phase algorithm taking advantage of the spatial localities to conquer this challenge. As we observe that there are some redundant IOs in the filter phase and only few trajectories have the chance to be returned in the refinement phase, we develop two optimized strategies to speed up the query process. The theoretical analysis and the experiment results demonstrate the effectiveness of the two optimized strategies and justify the advantage of the proposed methods over the naive method for at least an order of magnitude. Furthermore, the proposed methods are easy to implement and incorporate into trajectory databases.

8. ACKNOWLEDGMENTS

This paper is partially supported by NSFC project under grant number 60970030.

9. REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. *FODO*, pages 69–84, 1993.

[2] L. Chen and R. Ng. On the marriage of lp-norms and edit distance. In *VLDB*, pages 792–803, 2004.

[3] L. Chen, M. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, pages 491–502. ACM, 2005.

[4] S. Chen and R. Kashyap. A spatio-temporal semantic model for multimedia database systems and multimedia information systems. *TKDE*, 13(4):607–622, 2001.

[5] Z. Chen, H. Shen, X. Zhou, Y. Zheng, and X. Xie. Searching trajectories by locations: an efficiency study. In *SIGMOD*, pages 255–266. ACM, 2010.

[6] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[7] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57. ACM, 1984.

[8] D. Jacobs, D. Weinshall, and Y. Gdalyahu. Classification with nonmetric distances: Image retrieval and class representation. *PAMI*, 22(6):583–600, 2000.

[9] E. Keogh. Exact indexing of dynamic time warping. In *VLDB*, pages 406–417, 2002.

[10] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W. Ma. Mining user similarity based on location history. In *GIS*, pages 34:1–34:10. ACM, 2008.

[11] B. Lin and J. Su. Shapes based trajectory queries for moving objects. In *GIS*, pages 21–30. ACM, 2005.

[12] A. Palma, V. Bogorny, B. Kuijpers, and L. Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 863–868. ACM, 2008.

[13] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *ICDE*, pages 673–684. IEEE, 2002.

[14] X. Xiao, Y. Zheng, Q. Luo, and X. Xie. Finding similar users using category-based location history. In *GIS*, pages 442–445. ACM, 2010.

[15] Y. Yanagisawa, J. Akahani, and T. Satoh. Shape-based similarity query for trajectory of mobile objects. In *MDM*, pages 63–77. Springer, 2003.

[16] B. Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE*, pages 201–208. IEEE, 1998.

[17] J. Ying, E. Lu, W. Lee, T. Weng, and V. Tseng. Mining user similarity from semantic trajectories. In *LBSN*, pages 19–26. ACM, 2010.

[18] Y. Zheng and X. Xie. Learning travel recommendations from user-generated gps traces. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(1):2, 2011.

[19] Y. Zheng, L. Zhang, Z. Ma, X. Xie, and W. Ma. Recommending friends and locations based on individual location history. *TWEB*, 5(1):5, 2011.

[20] Y. Zheng, L. Zhang, X. Xie, and W. Ma. Mining interesting locations and travel sequences from gps trajectories. In *WWW*, pages 791–800. ACM, 2009.

Towards Fine-Grained Urban Traffic Knowledge Extraction Using Mobile Sensing

Xuegang (Jeff) Ban
JEC 4034, CEE, Rensselaer Polytechnic Institute
110 8th St, Troy, NY 12180
banx@rpi.edu

Marco Gruteser
WINLAB, Rutgers University
671 Route 1 South
North Brunswick, NJ 08902
gruteser@winlab.rutgers.edu

ABSTRACT
We introduce our vision for mining fine-grained urban traffic knowledge from mobile sensing, especially GPS location traces. Beyond characterizing human mobility patterns and measuring traffic congestion, we show how mobile sensing can also reveal details such as intersection performance statistics that are useful for optimizing the timing of a traffic signal. Realizing such applications requires co-designing privacy protection algorithms and novel traffic modeling techniques so that the needs for privacy preserving and traffic modeling can be simultaneously satisfied. We explore privacy algorithms based on the virtual trip lines (VTL) concept to regulate where and when the mobile data should be collected. The traffic modeling techniques feature an integration of traffic principles and learning/optimization techniques. The proposed methods are illustrated using two case studies for extracting traffic knowledge for urban signalized intersection.

Keywords
Urban Traffic Knowledge, Mobile Sensing, Location Traces, Traffic Theory

1. INTRODUCTION AND MOTIVATION
The recent proliferation of Global Positioning System (GPS) equipped vehicles and devices have led to the emergence and rapid deployment of *mobile traffic sensors*, which move with the traffic flow they are monitoring. Mobile sensors can collect detailed location traces of individual persons or vehicles, information that promises great advances in many science and engineering fields, including public health monitoring/diagnostics [19], extraction of personal or social behaviors [7] and mobility patterns [8], and transportation [12, 2].

In the transportation area, mobile sensing has recently motivated two important investigations, namely, *city-scale* transportation knowledge extraction and *fine-grained* urban traffic knowledge extraction. The former concerns with large scale (i.e., city-scale) traffic congestion patterns such as travel times [26], routing [26, 25], social activity patterns [16], urban planning [28], land use [23], human mobility patterns [8], among others. Chapter 5 in [29] provides a summary of city-scale transportation knowledge extraction, focusing on possible patterns that can be extracted from location traces (trajectories). Other related critical issues are also discussed in [29] such as privacy concerns. On the other hand, fined-grained urban traffic knowledge extraction emphasizes on detailed, smaller-scale descriptions of urban traffic flow, such as traffic states and performances, as well as associated traffic operations and control. Applications for fine-grained urban traffic knowledge extraction can be broadly categorized as those for highways and urban arterials. Highway traffic can be generally modeled as continuous flow [11, 12, 24]. Arterial traffic however is often disrupted, e.g., by traffic signals, resulting in discontinuities and kinks in arterial traffic flow states. Such unique features can actually be utilized to reconstruct arterial traffic flow patterns such as delay (or travel times) [2] and queue lengths [1, 6], as well as signal timing information [9].

City-scale and fine-grained urban knowledge extraction represent, respectively, the macro-level and micro-level modeling of urban transportation systems. They are thus equally important for better understanding, describing, and managing the urban transportation systems. They share many commonalities, e.g., their methodologies require integration of data mining tools and some domain knowledge, privacy seems to be a concern to both areas, etc. Their also have distinctive differences, in terms of their specific application domains and the detailed study methodologies. In particular, city-scale applications require wide-area data coverage but may tolerate coarse resolution accuracy in time or space. For example, Taxi GPS data reported in a 2-5 minute interval can be used to mine routing and city-scale traffic conditions [26, 25]. Fine-grained urban knowledge extraction however requires much finer resolution especially in the time domain (second-by-second location traces is usually desirable) and relatively high penetration [1]. On the other hand, fine-grained applications usually need only smaller-area data coverage (e.g., signal performance modeling based on mobile data only needs location traces that cover a single intersection or several intersections).

In this paper, we focus on fine-grained urban traffic knowledge extraction to estimate real time traffic signal performances using mobile sensing especially location traces. We show that the traffic knowledge that can be mined from mobile sensing is much richer than the traffic congestion and mobility pattern knowledge that has been the focus of most research so far for city-scale urban knowledge extraction. For example, it has been a long-standing challenge to collect traffic data in urban environment such as arterial signalized intersections. The primary source of data collection has been traditionally via fixed-location sensors such as loop detectors. However, the deployment of such detectors are limited and the cost to expand their coverage is prohibitive. For example, the New York City has over 95% of its total 12,225 traffic signals as pre-timed and no detector is deployed at most of these intersections [1]. As a result, recently the National Traffic Signal Report Card assigned the grade "F" (the lowest) to the detection and data collection system for traffic signals in the United States (NTOC, 2007). The wide deployment of mobile sensors will eventually allow collecting traces at low cost from a significant fraction of the population, which enables the estimation of detailed traffic system states, i.e., fine-grained urban traffic knowledge extraction, from individual drivers sharing their location/driving information via mobile sensors. This in turn can greatly benefit existing and emerging applications in urban traffic knowledge extraction, such as performance measurement of traffic signals and arterial networks.

Fine-grained urban traffic knowledge extraction using mobile sensors can be considered as a special form of Human Centric Sensing which needs to address a set of challenges [20]. In this paper we focus on two of them that are particularly important to transportation modeling applications: (i) the development of novel modeling methodologies to utilize the unique format of mobile data; and (ii) privacy protection. First, compared with fixed-location sensor data, data from mobile sensors has distinct features. Figure 1 illustrates the difference of the two data types in a time-space coordinate system at a signalized intersection. We show the red time duration and green time duration at the location of the intersection. Short line segments represent the fixed-location sensor data collected by the loop detector system which consists of a presence detector at the stop line and an advanced detector in the upstream link. Two long thick curves in the figure represent the trajectories obtained from the mobile sensors. As shown in the figure, fixed-location sensors collect traffic flow measures, such as volume, density, and speed, for all vehicles, but only at spatially discrete locations where sensors are deployed. Mobile sensors on the other hand can reveal detailed behaviors and provide (almost) spatially continuous trajectories of vehicles, but only for a sample of the traffic flow. Since part of the traffic flow is hidden, we cannot obtain accurate aggregated measures, such as traffic volume or density, from mobile sensors. As a result, existing modeling methods that work well for fixed-location sensors may not be directly applied to mobile data.

The unique characteristics of mobile data thus call for novel modeling approaches. Unfortunately, existing research on using mobile data (often called "probe data") in transportation is limited by very low penetration of such data. As a result, mobile data have been used mainly as a supplement to fixed location sensor data. Pure statistical analysis has been the main method to deal with mobile data [14], focusing on

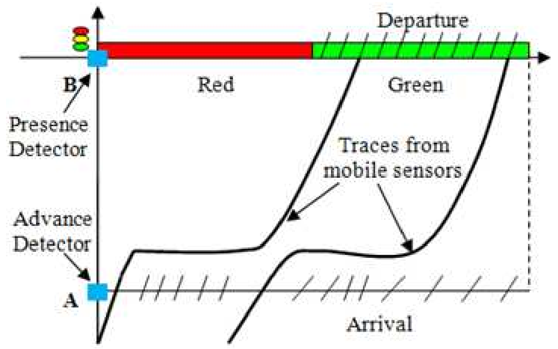


Figure 1: Comparison of mobile data and fixed-location-sensor data

applications to estimate average traffic flow or travel times [18]. Those previous approaches cannot fully capture and utilize the unique features of mobile data shown in Figure 1. In this paper, we focus on relatively large penetration of mobile data (e.g., larger than 10%) since we believe this will be the future trend. Under relatively high penetration, mobile data will play a dominate or at least equally important role (compared with fixed-location sensor data) for traffic data collection and modeling. This has the potential to transform current practice of urban traffic knowledge extraction. Privacy violation is another issue to concern. As mobile sensors can potentially reveal the complete traces of travelers that could contain sensitive location related information, the medical conditions, political affiliations or commercial secret may be inferred. Today the privacy issues of mobile devices are well realized by the public; see e.g., [22].

These two challenges call for an integrative framework to simultaneously consider modeling data needs and privacy preservation, i.e., to *co-design* privacy algorithms and modeling techniques so that the needs for urban traffic knowledge extraction and privacy protection can be both satisfied. This paper summarizes the authors' recent work in this area. We present the privacy protection scheme first in the next section. The new traffic modeling methods using privacy preserving mobile data are then presented and illustrated using case studies for fine-grained urban traffic knowledge extraction. We conclude the paper by discussing several important future research directions.

2. PRIVACY PROTECTION SCHEME

Our research pursues a privacy-by-design approach [5] to balance traffic modeling data needs and privacy protection when dealing with mobile data. This approach seeks to minimize the collection of personally identifiable information. To this end, we anonymize information by omitting any identifiers (such as names, equipment serial numbers, etc.) and further restrict the collection of location traces to limit re-identification risks. One aspect of restricting collection is to only record data in those locations where it is actually required for providing the service. To achieve this, we have introduced together with other colleagues Virtual Trip Lines (VTLs) [13], which define point locations on roadways where data should be collected, and VTL zones, which are areas along roadways where data is needed. Figure 2 shows an example VTL zone for an intersection monitoring application.

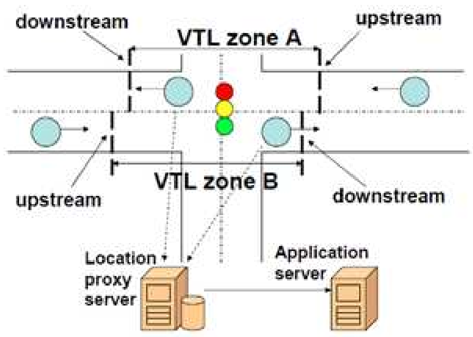


Figure 2: VTL mobile data collection system (source: [22])

It shows the VTL locations for the east-west street through dashed bold lines. Not shown are the VTL locations for north-south street, which would be part of a general solution. These VTL locations can be stored on the mobile sensing device. The device monitors its own location using GPS, for example, and will only report its speed and location trajectory to the location server between the upstream and downstream VTLs that demarcate the VTL zones. Notice that the exact VTL locations can be adjusted depending on application requirements. In this example VTL1 is placed farther away from the intersection than VTL2 in order to capture the queuing process of vehicles approaching the intersection. Since anonymous data can be often re-identified by correlating it with other data sources, the application server can apply further cloaking algorithms that filter the data to reduce risk as described in [13] or [27].

The use of these different types of privacy filtered data has been tested for both freeway modeling [11, 24] and urban arterial modeling [2, 1]. Sun et al. [21] further showed that the mobile data (such as travel times, short trajectories) collected via such a system can be properly used for traffic modeling applications such as real time estimation of vehicular queue lengths at a traffic signal. In this paper, we will illustrate this with the following two case studies that use the travel times of vehicles equipped with mobile sensors (called sample vehicles).

3. CASE STUDY I: DELAY PATTERN

Delay caused by traffic signals is the major source of delay in urban environment. Intersection delay pattern here refers to the experienced delay of a vehicle arriving at the intersection at any time. As shown in Figure 3, what we can actually measure in real world is the discrete delays or travel times from individual vehicles (shown as the circles on the piecewise linear curve in the bottom of the figure). Delay pattern is thus a continuous approximation of such discrete measurements. In fact, when people talk about intersection delays at a certain time (say 8 am), they never care if there is a vehicle actually arriving at the intersection at that particular time; indeed, they refer to the delay pattern of the intersection. To show how delay pattern can be estimated using mobile data, we show in Figure 3 a signalized intersection with two VTLs deployed upstream (VTL1) and downstream (VTL2). Under certain assumptions, we can use the bold solid triangles (or trapezoids) in the figure

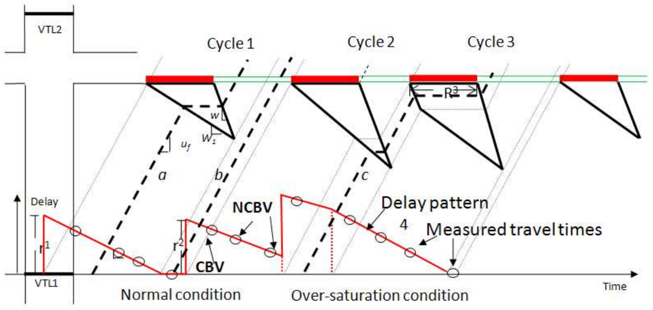


Figure 3: Intersection delay pattern

to represent how queue forms and dissipates based on the traffic shockwave theory [15, 17, 2]. The horizontal part of a triangle represents the duration of red time. As shown by the trajectories of vehicles (dashed lines), if a vehicle approaches the intersection in red time or if the queue length is not zero (e.g., trajectory *a* in the figure), the vehicle will join the end of the queue first and thus be delayed. The delay encountered by the vehicle is the horizontal part of the trajectory. Otherwise, if a vehicle arrives during green time and there is no queue (e.g., trajectory *b*), the vehicle will pass the intersection with no delay. By analyzing the geometry of the triangles, we can construct the theoretical delay pattern curve as shown in the bottom of Figure 3. The curve is piecewise linear and contains *critical points*, i.e., discontinuities and non-smoothness. Discontinuities indicate traffic signal changes (such as the start of the red time) and non-smoothness indicate traffic state changes (such as a queue is fully discharged).

Mobile data however cannot be used to construct directly the delay pattern; rather they provide samples of intersection delays, shown as circles along the delay curve in the figure. These sample delays, under proper penetration, can be used to identify the critical points of and further to estimate the delay pattern curves. In [2], this is done via a least square estimation algorithm to fit the sample travel times to the piece wise linear curves after grouping the samples into different cycles. Figure 4 shows the results of applying the estimation algorithm to a field test in the Bay Area in California [1]. In the figure, the asterisks along the piecewise lines (i.e., the estimated delay pattern) are the observed travel times (delay plus a constant minimum traverse time from VTL1 to VTL2) and the plus signs at the bottom are the errors. It is clear that the delay pattern can match well the observed samples. Knowing the pattern will help identify traffic conditions, e.g., over-saturation (i.e., vehicles cannot be fully discharged within a cycle) as indicated in the figure, or to estimate real time queue lengths [1].

The above analysis underlines the most salient feature of the new modeling method, i.e., a proper integration of traffic principles and learning/optimization techniques. For the delay pattern estimation here, the *knowledge* is based on the traffic flow theory that describes the delay pattern as piecewise linear curves whose critical points (discontinuities and non-smoothness) have clear physical meanings. On the other hand, the *learning/optimization techniques* is least square estimation that helps estimate the key parameters of such

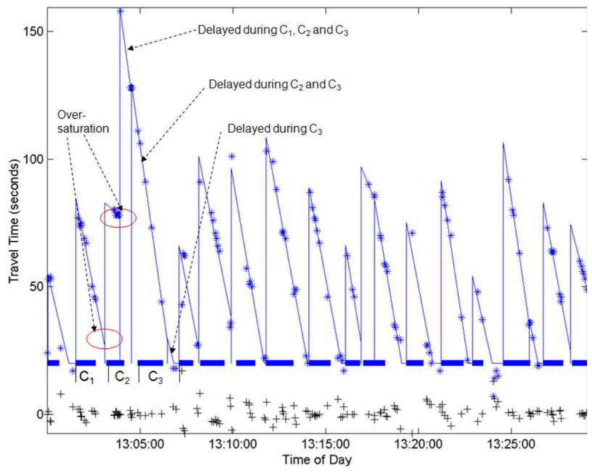


Figure 4: Delay pattern estimation of a field test (Source: [2])

patterns from mobile data and ultimately reconstruct the patterns. These two components need to be integrated in a holistic manner, which however may be different for different applications, as shown in the next case study.

4. CASE STUDY II: SIGNAL TIMING

Mobile data, especially intersection travel times, can be used to estimate the timing parameters of urban traffic signals (such as the cycle length, number of phases, and cycle by cycle red and green times), which are important for traffic signal operations and signal/arterial performance measurement. It has been for long assumed that such parameters should be available input, e.g., from transportation management agencies such as departments of transportation, to traffic models. In fact, collecting signal timing parameters directly from the agencies is probably trivial for small scale data collection (such as for a few signals). However, collecting such information this way for large areas (such as a region or nation-wide) can be very challenging and time consuming due to many possible technical and institutional hurdles. On the other hand, many traffic information providers have started to collect increasingly large amount of mobile data. Therefore an alternative way is to infer the signal timing information directly from the data that have already been collected such as travel times, probably with the help of limited (and easily obtained) knowledge about traffic signals.

[9] developed a robust signal timing estimation method based on intersection travel times. The method is again featured by a combination of traffic flow theories and learning/ optimization methods, which can estimate the exact cycle start/end times. The method contains three major steps: cycle breaking, exact cycle boundary detection, and effective red (or green) time estimation. Cycle breaking determines whether a new cycle starts by applying the support vector machine (SVM) to identify travel time samples that indicate the starts of red times. The exact cycle boundary estimation detects the exact cycle start/end times. It can be formulated as a nonlinear program by assuming that the cycle length is constant (the effective red and green times may

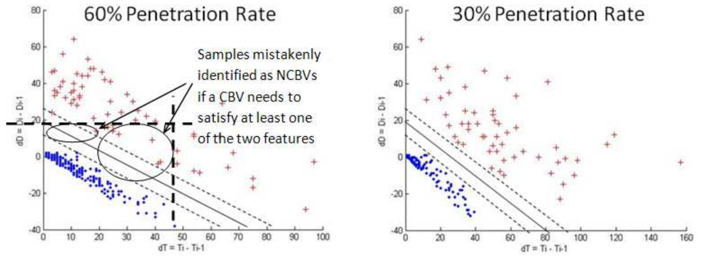
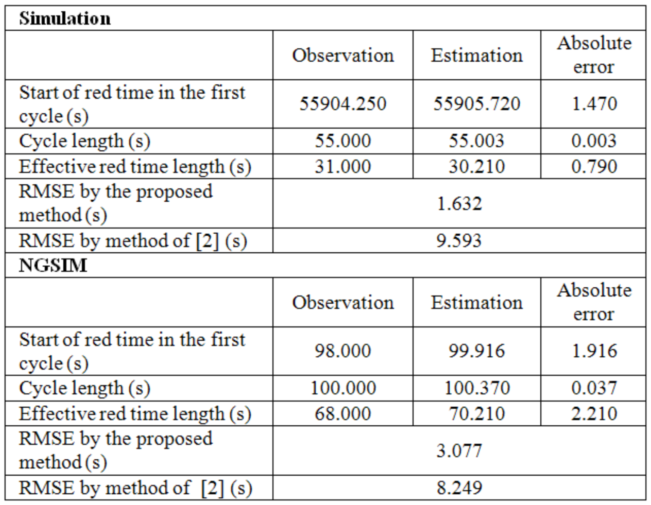


Figure 5: Cycle breaking (source: [7])

vary from cycle to cycle, which covers a large portion of existing traffic signals, e.g., in the US). The method can further detect the number of missing cycles using sample delays and the SVM results. The effective red (or green) time estimation calculates the duration of effective red (or green) times. This is done via using delay patterns by investigating when non-smoothness in the delay pattern happens. Due to space limitation, we only present in this paper how the cycle breaking is done via the use of SVM.

We first define, as shown in Figure 3, a cycle breaking vehicle (CBV) as the first sample vehicle in a cycle. The other vehicles in this cycle are defined as non-cycle breaking vehicle (NCBV). Note that the CBV of a cycle is not necessarily the first vehicle actually arriving at the signal in the cycle if the penetration is not 100% (in this case, the first vehicle may not be sampled). The CBVs usually have higher delays as shown in Figure 3. In [2], this feature is used to detect whether a new cycle starts by defining a threshold: if the delay increase from one vehicle to the next vehicle exceeds this threshold, a new cycle starts. The results however are not reliable due to oscillation and noise in measurements, and especially low penetration rate of mobile data. In [9], the SVM model applies two features: the arrival time difference $t_i - t_{i-1}$ and the delay difference $\tau_i - \tau_{i-1}$ between two consecutively sampled vehicles. Here t_i is the ith sample vehicle's arrival time at VTL1 and τ_i is the intersection delay of the ith sample vehicle. The second feature is exactly what was used in [2]. Figure 5 depicts these two features for a field test under 60% and 30% penetration rate of travel time data [9]. In the figure, dots are for NCBVs and plus signs are for CBVs. We can see that there is a clear margin of separation between CBVs and NCBVs using these two features. Using either feature or a simple combination of the two features however is not effective. In Figure 5, the vertical dashed bold line indicates the threshold in delay increase; the horizontal dashed bold line indicates the threshold in arrival times. The figure shows that even both measures are used (e.g., a CBV needs to satisfy at least one of the two measures), there will be still large errors for mis-identification, as those indicated by the circles.

SVM can combine the two features in a more intelligent way. To show how the SVM model can be developed, let the historical travel time data be denoted by $(x_i, y_i), i = 1, \ldots, M$, where $x_i = (t_i - t_{i-1}, d_i - d_{i-1})^T$ is a data point and $y_i = 1$ is the corresponding label ($y_i = 1$ for CBV and $y_i = -1$ for NCBV). SVM divides the data set into two groups: one for $y_i = 1$ and the other for $y_i = -1$. It can further produce two support planes (lines in the R^2



Simulation			
	Observation	Estimation	Absolute error
Start of red time in the first cycle (s)	55904.250	55905.720	1.470
Cycle length (s)	55.000	55.003	0.003
Effective red time length (s)	31.000	30.210	0.790
RMSE by the proposed method (s)	1.632		
RMSE by method of [2] (s)	9.593		
NGSIM			
	Observation	Estimation	Absolute error
Start of red time in the first cycle (s)	98.000	99.916	1.916
Cycle length (s)	100.000	100.370	0.037
Effective red time length (s)	68.000	70.210	2.210
RMSE by the proposed method (s)	3.077		
RMSE by method of [2] (s)	8.249		

Figure 6: Cycle breaking results: Simulation and NGSIM datasets (source: [7])

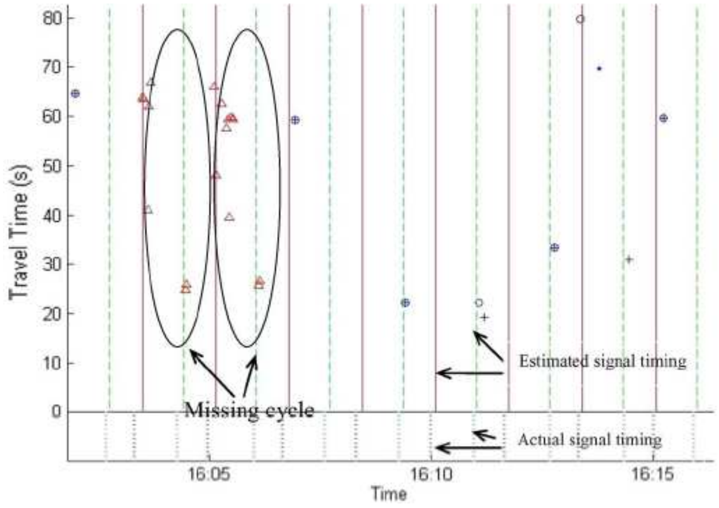


Figure 7: Signal timing estimation results: NGSIM dataset (source: [7])

space) for such separation as depicted in Figure 5. Let $w = (w_1, w_2)^T \in R^2$ and b be a scalar. If w and b are properly selected, we will have $wx_i - b \geq 1$ for $y_i = 1$ and $wx_i - b \leq -1$ for $y_i = -1$. Then the two support lines are $wx_i - b = 1$ and $wx_i - b = -1$. The distance between these two lines can be shown as $2/||w||$ with $||w||$ denoting the norm of w. If we aim to maximize the distance between these two support planes, (w, b) can be determined by solving the following SVM problem [9]:

$$min_{w,b} \quad 1/2||w||^2 + G(\sum_{i=1}^{M} \varepsilon_i) \tag{1}$$

$$\text{subject to} \quad y_i(wx_i - b) \geq 1 - \varepsilon_i, i = 1, \ldots, M, \tag{2}$$

$$\varepsilon_i \geq 0, i = 1, \ldots, M. \tag{3}$$

Here ε_i is the error term for cases where the classes cannot be perfectly divided. G is the weight factor assigned to the error terms in the overall objective function; a larger G assigns a larger penalty to the error. Solving the above SVM model is usually not computationally demanding since it is a convex, quadratic program; see [4, 3]. After solving the SVM model, the (w, b) pair, in particular, the two planes (lines): $wx_i - b = 1$ and $wx_i - b = -1$, can be used to identify whether a given data sample x_j is a CBV or NCBV. Figure 6 shows the results of applying the SVM model to break cycles using the datasets of simulation and NGSIM (next generation simulation, widely used for traffic research) [9], based on which to estimate the cycle boundaries and lengths. Compared with the previous method in [2], the root mean square error (RMSE) of the results is much improved: from 9.6 seconds to 1.6 seconds for the simulation data, and from 8.2 seconds to 3.0 seconds for the NGSIM data.

Figure 7 shows the signal timing estimation results by applying the three step method for the NGSIM dataset. There are in total 9 cycles in the dataset and is no sample for the second and third cycles. The figure shows that the two missing cycles can be correctly detected by the three step method. The estimated start times of red are plotted using vertical solid lines and the estimated start times of green are plotted using vertical dashed lines. The actual starts of red and green times are also shown in the figure using dotted lines. We can see that the estimated cycle boundaries and red and green times match well with those from field observations, indicating that the method works reasonably well. Same conclusion can also be obtained for the simulation data which is omitted here.

The above analysis underlines again the unique feature of the new modeling method that integrates traffic principles (in this case, how delay changes within a cycle and across cycles) with learning/optimization methods (in this case, SVM and nonlinear programming). It also highlights the importance of learning techniques in such a process: the knowledge (i.e., the two features for cycle breaking) has to be used intelligently since simple use of them may not work well (as illustrated by the plot to the right in Figure 5). On the other hand, traffic knowledge (such as the signal cycle length is fixed and the vehicle delay after signal turns red will "jump") is also critical. After all, we are dealing with a physical traffic system with control devices and vehicles/drivers. Traffic knowledge usually represents some important physical phenomena or characteristics of the system and thus needs to be properly respected and integrated into the learning/optimization models to produce meaningful results.

5. DISCUSSIONS AND FUTURE RESEARCH

We presented in this paper our recent work on co-designing privacy protection algorithms and traffic modeling methods for fine-grained urban traffic knowledge extraction using location traces from mobile sensoring. The privacy protection scheme is based on the VTL concept to regulate the collection of mobile data. The traffic modeling method is featured by a combination of traffic principles and learning/optimization techniques. Two case studies were also presented to demonstrate the proposed methods.

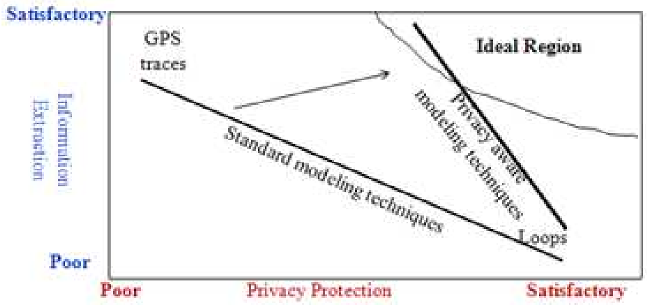


Figure 8: Privacy-aware modeling paradigm

The development of the VTL concept is based on a close collaboration between location privacy experts and transportation researchers, with the aim to properly balance the needs of both traffic modeling (i.e., traffic knowledge extraction) and privacy protection. It clearly indicates that for many traffic/transportation applications, data needs for modeling do not necessarily have to be compromised to ensure privacy protection. An application-aware design of privacy algorithms can retain features important for the application, while still achieving privacy by removing features that are less important. The key is the close collaboration between the two research communities of privacy protection and transportation to simultaneously consider privacy protection and application needs: being aware of the effects of applying privacy schemes to data when developing modeling methods, and being aware of data needs when designing privacy preserving mechanisms. As the future paradigm will likely shift from traditional sensors (such as loops) to mobile sensors as shown in Figure 8, such collaboration is critical to transform the current modeling techniques with no or little privacy consideration to a new paradigm for privacy-aware modeling techniques to satisfy requirements of both data needs for modeling and privacy protection.

The proper integration of transportation principles and data-mining tools is important to develop new mobile-data-based traffic modeling methods, especially when the penetration of mobile data reaches certain "critical mass." In this case, the mobile data can present patterns that reveal traffic system control or state changes (such as red time starts or queue disappears, as we discussed in the delay pattern estimation section). Knowledge about these patterns (many of which are indeed well-known in the application domain) are crucial since they can provide guidance to the learning/optimization models to focus on the most relevant, important features. Applying advanced learning/optimization techniques on the other hand is also critical to obtain accurate, robust estimation of the patterns, and further system control and states.

Fine-grained urban traffic knowledge extraction using mobile sensoring is an emerging area and many challenges still remain. Below we summarize the limitations of our current research and highlight some challenges for future research.

(a) Our current focus in on urban intersections which are a crucial component for urban traffic. The next step is to expand it to model arterial corridors (consist of a number of intersections) and networks. The important issue for such expansion is to capture the interactions among different intersections, which can be revealed by vehicle platooning. The key challenge therefore is to study how vehicle platoons form and disperse at intersections using mobile data.

(b) The presented models in this paper are deterministic. Since traffic is random in nature, we need to capture such sotchasticity in the modeling process. This is particularly important when modeling urban corridors or networks (such as to study the vehicle platooning process). We recently developed Bayesian Network based statistical learning methods to model arterial traffic flow [10]. The method will be expanded for large scale arterial applications.

(c) The VTL based privacy protection algorithm is suitable for modeling urban intersections. Our research also indicates that specific privacy techniques may need to be developed for different urban applications such as origin-destination demand estimation, urban congestion pricing, among others. For this, a comprehensive framework is needed. It should contain a suite of privacy techniques that can be used/tailored depending on specific urban applications. The authors are working on this topic and results will be reported in subsequent papers.

6. ACKNOWLEDGMENTS

The authors thank the comments of two anonymous reviewers which improve the quality of an early version of this paper. The authors greatly appreciate their collaborators and students who worked with them on this topic over the last several years. Many technical details related to the case studies presented in this paper can be found in the joint papers with those collaborators. This research is partially funded by National Science Foundation grants: CMMI-1031452, CMMI-1055555, CMMI-1031400, CNS-0845896; US Department of Transportation (USDOT); California Department of Transportation (Caltrans); and New York State Department of Transportation (NYSDOT). Any opinions, findings, and conclusions or recommendations in this paper are those of the authors and do not necessarily reflect the views of any of the above funding agencies.

7. REFERENCES

[1] X. Ban, P. Hao, and Z. Sun. Real time queue length estimation for signalized intersections using sample travel times from mobile sensors. *Transportation Research Part C*, 19(6):1133–1156, 2011.

[2] X. Ban, R. Herring, P. Hao, and A. Bayen. Delay pattern estimation for signalized intersections using sampled travel times. *Transportation Research Record*, 2130:109–119, 2009.

[3] K. Bennett and C. Campbell. Support vector machines: Hype or hallelujah? *SIGKDD Explorations*, 2(2):1–13, 2000.

[4] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[5] A. Cavoukian. Privacy by design. Technical Report Internet Link: http://www.privacybydesign.ca/content/uploads/2010/03/PrivacybyDesignBook.pdf

(Accessed on June 20, 2012), Information and Privacy Commissioner of Ontario Canada, 2009.

[6] Y. Cheng, X. Qin, J. Jin, and B. Ran. An exploratory shockwave approach to estimating queue length using probe trajectories. *Journal of Intelligent Transportation Systems*, 16(1):12–23, 2012.

[7] T. Choudhury, G. Borriello, and et al. The mobile sensing platform: an embedded system for activity recognition. *IEEE Pervasive Magazine*, 7(2):32 – 41, 2008.

[8] M. Gonzalez, C. Hidalgo, and A. Barabasi. Understanding individual human mobility patterns. *Nature*, 453(Jun 5):779–782, 2008.

[9] P. Hao, X. Ban, K. Bennett, Q. Ji, and Z. Sun. Signal timing estimation using intersection travel times. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):792–804, 2012.

[10] P. Hao, D. Guo, X. Ban, Q. Ji, and Z. Sun. Vehicle index inference for signalized intersections using sample travel times. In *Proceedings of the 91st Transportation Research Board Annual Meeting*, Washington, DC, U.S.A., January 2012.

[11] J. Herrera and A. Bayen. Incorporation of lagrangian measurements in freeway traffic state estimation. *Transportation Research Part B*, 44(4):460–481, 2009.

[12] J. Herrera, D. Work, R. Herring, X. Ban, and A. Bayen. Evaluation of traffic data obtained via gps-enabled mobile phones: the mobile century field experiment. *Transportation Research Part C*, 18(4):568–583, 2010.

[13] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J. Herrera, and A. Bayen. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *The Sixth Annual International conference on Mobile Systems, Applications and Services (MobiSys 2008)*, Breckenridge, U.S.A., June 2008.

[14] A. Krause, E. Horvitz, A. Kansal, and F. Zhao. Toward community sensing. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, St. Louis, MO, U.S.A., 2008.

[15] M. Lighthill and J. Whitham. On kinematic waves. i: Flow movement in long rivers; ii: A theory of traffic flow on long crowded roads. In *Proceeding of Royal Society A*, volume 229, pages 281–345, 1955.

[16] S. Networks. http://www.sensenetworks.com/products/macrosense-technology-platform/citysense.

[17] P.I.Richards. Shock waves on the higway. *Operation Research*, 4:42–51, 1956.

[18] C. A. Quiroga and D. Bullock. Travel time studies with global positioning and geographic information systems: an integrated methodology. *Transportation Research Part*, 6(1-2).

[19] J. Sriram, M. Shin, T. Choudhury, and D. Kotz. Activity-aware ecg-based patient authentication for remote health monitoring. In *Proceedings of the International Conference on Multi-Modal Interfaces (ICMI-MLMI)*, pages 297–304, November 2009.

[20] M. Srivastava, T. Abdelzaher, and B. Szymanski. Human-centric sensing. *Phil Trans.R.Soc*, 370, ser. A:176–197, 2012.

[21] Z. Sun, B. Zan, X. Ban, M. Gruteser, and P. Hao. Evaluation of privacy preserving algorithms using traffic knowledge based adversary models. In *Proceedings of the 14th IEEE conference on Intelligent Transportation Systems*, Washington, DC, U.S.A., 2011.

[22] N. Y. Times. http://www.nytimes.com/2011/04/21/business/21data.html.

[23] J. Toole, M. Ulm, D. Bauer, and M. Gonzalez. Inferring land use from mobile phone activity. In *In Proceedings of the ACM SIGKDD International Workshop on Urban Computing*, 2012.

[24] D. Work, S. Blandin, O. Tossavainen, B. Piccoli, and A. Bayen. A traffic model for velocity data assimilation. *Applied Mathematics Research eXpress*, 2010(1):1–35, 2010.

[25] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–324, 2011.

[26] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: Driving directions based on taxi trajectories. In *In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 99–108, 2010.

[27] B. Zan, Z. Sun, M. Gruteser, and X. Ban. Vtl zone-based path cloaking algorithm. In *Proceedings of the 14th IEEE conference on Intelligent Transportation Systems*, Washington, DC, U.S.A., 2011.

[28] Y. Zheng, Y. Liu, J. Yuan, and X. Xie. Urban computing with taxicabs. In *In Proceedings of the 13th international conference on Ubiquitous computing*, pages 89–98, 2011.

[29] Y. Zheng and X. Zhou. *Computing with spatial trajectories*. Springer, 2011.

Exploration of Ground Truth from Raw GPS Data

Huajian Mao
National University of Defense Technology
State Key Laboratory of High Performance Computing
Hunan 410073, China
huajianmao@nudt.edu.cn

Wuman Luo, Haoyu Tan, Lionel M. Ni
Hong Kong University of Science and Technology
Hong Kong
{luowuman, hytan, ni}@cse.ust.hk

Nong Xiao
National University of Defense Technology
State Key Laboratory of High Performance Computing
Hunan 410073, China
nongxiao@nudt.edu.cn

ABSTRACT

To enable smart transportation, a large volume of vehicular GPS trajectory data has been collected in the metropolitan-scale Shanghai Grid project. The collected raw GPS data, however, suffers from various errors. Thus, it is inappropriate to use the raw GPS dataset directly for many potential smart transportation applications. Map matching, a process to align the raw GPS data onto the corresponding road network, is a commonly used technique to calibrate the raw GPS data. In practice, however, there is no ground truth data to validate the calibrated GPS data. It is necessary and desirable to have ground truth data to evaluate the effectiveness of various map matching algorithms, especially in complex environments. In this paper, we propose truthFinder, an interactive map matching system for ground truth data exploration. It incorporates traditional map matching algorithms and human intelligence in a unified manner. The accuracy of truthFinder is guaranteed by the observation that a vehicular trajectory can be correctly identified by human-labeling with the help of a period of historical GPS dataset. To the best of our knowledge, truthFinder is the first interactive map matching system trying to explore the ground truth from historical GPS trajectory data. To measure the cost of human interactions, we design a cost model that classifies and quantifies user operations. Having the guaranteed accuracy, truthFinder is evaluated in terms of operation cost. The results show that truthFinder makes the cost of map matching process up to two orders of magnitude less than the pure human-labeling approach.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

System, Measurement

Keywords

Ground truth, map matching, GPS data

1. INTRODUCTION

Smart transportation is expected to play an important role to meet the growing demand of various transportation-related services from citizens [16] and government officers [2], especially in modern cities. A fundamental requirement to smart transportation is to collect the dynamic vehicular location data to form the basis to build an effective traffic information system [4]. The collected large-scale vehicular dataset is subject to further analysis, such as traffic estimation [15], hot spot detection [9], driving pattern recognition [8], traffic mining [5], and similar routes discovery [3], before the goals of smart transportation can be achieved. Real-time vehicular data collection is the first step toward smart transportation. For example, in Shanghai Grid (SG) project [7], most of the public vehicles are equipped with a GPS and a GPRS wireless communication module. Each of these vehicles periodically sends GPS reports to a data center. In the current implementation of the SG project, a very large volume (about 3-4 million records per day) of vehicular GPS trajectory data has been collected with different techniques.

The collected raw GPS data unfortunately suffers from two major errors. First, due to the limitation of the GPS technology, the vehicle location coordinates are not necessary precise mainly due to environmental factors. Second, a vehicle's location trajectory is reported in discrete samples for cost concern of GPRS communication. Even worse, the reporting or sampling interval may be adjusted by the driver in the SG project. Thus, it is challenging to estimate a vehicle's location during the sampling interval. Consequently, it is inappropriate to use the raw GPS dataset directly, which may lead to inaccurate conclusions or decisions for the potential smart transportation applications.

Due to the problems introduced above, raw GPS data calibration or recovery is the next important step toward smart transportation. An intuitive approach to correct the raw GPS data is to align the data onto the corresponding road network to find out a sequence of road segments that a vehicle has traveled along. This process is usually referred to as *map matching* [13] [10]. Typically, a good map matching should possess the desirable property of high accuracy which is evaluated by a complete validation. In practice, however, there is no ground truth data to validate the calibrated GPS data. Although map matching has been studied for many years, there still exist several challenging problems due to the lack of ground truth data. First, to validate the accuracy of a map matching algorithm, a ground truth path is required to compare with the output of the algorithm. Very few existing map matching algorithms provide a meaningful validation technique due to the aforementioned reason. Second, it is necessary and desirable to have ground truth data to tune and evaluate the effectiveness of various map matching algorithms. Since most map matching algorithms are heuristic, their accuracy is strongly related to the tuning and selection of various design parameters. However, the parameters should be tuned with the ground truth data. Wrong parameters will lead the algorithm inaccurate. Therefore, finding a complete trajectory ground truth is critical to the map matching research.

We observe that most of the ground truth path of the trajectories can be correctly identified by human-labeling on the historical raw GPS dataset. It is believed that human-labeled data can be almost 100% accurate and it is widely used to explore ground truth dataset to evaluate map matching algorithms [10] [17]. In general, a human labeling process involves both cognitive works (e.g., determining the road segment for a particular GPS report) and manual works (e.g., recording the sequence of road segment identifier). Since this process involves too much human intelligence and action, it is usually not feasible to apply pure human-labeling to large GPS datasets.

To solve this problem, in this paper, we propose truthFinder, an interactive map matching system for ground truth data exploration. The goal of truthFinder is to minimize the human involvement. Specifically, we try to let the user interact with the system as little as possible. Formally, the goal of truthFinder is defined as follow: *For a given trajectory T and a road network $G(V, E)$, we want to explore the ground truth path P with a small cost C in terms of operations.* For this purpose, there are several challenges. First, it is difficult to quantify the cost of human interaction. For this challenge, we propose a cost model for truthFinder to measure the efficiency of the method. Second, using the visualization of the trajectory and the digital map is not trivial. For example, the trajectory may contain the same road segment twice or more. We should avoid such overlapping in visualization and allow the user to select anyone of them. With this issue, we introduce several techniques (e.g., multi-layer presentation for showing trajectories and paths, and multi-color notation for the candidate roads) to make it convenient to explore the ground truth. Third, the existing map matching algorithms should be modified to be stable. As such, we propose an interactive map matching system, that is, taking the users interaction into account, the trajectory generated at the next round should be more accurate than the current one.

ThruthFinder incorporates traditional map matching algorithms and human intelligence in a unified manner. The accuracy of truthFinder is guaranteed by the observation that a vehicular trajectory can be correctly identified by human-labeling with the help of a period of historical GPS dataset. To measure the cost of human interactions, we design a cost model that classifies and quantifies user operations. Having the guaranteed accuracy, truthFinder is evaluated in terms of operation cost. The results show that truthFinder makes the cost of map matching process up to two orders of magnitude less than the pure human-labeling approach. To sum up, our contributions are as follows:

- We design a cost model that classifies and quantifies user interactions. Our model avoids absolute measurements of human behaviors. Instead, we define several operations with regard to our system and use the number of each operation in cost analysis.
- We propose the architecture and implementation issues of truthFinder in detail. We are arguably the first to offer an interactive map matching system. Our design can be easily generalized for similar purposes.
- We provide a method to explore the ground truth path data from raw GPS trajectory data while guarantying the accuracy for different situations at the same time. In this way, the issues of map matching algorithm validation can be overcome by using truthFinder.
- Our system is evaluated in terms of operation cost. The experimental results show that truthFinder significantly outperforms traditional method of exploring ground truth data from scratch.

The rest of this paper is organized as follows. Section 2 describes the prior related work in detail. Section 3 shows the system architecture design. Section 4 puts forward the cost model of our interactive map matching system for ground truth exploration. Section 5 gives the evaluation of our work based on our implemented prototype system. We conclude our paper and present the future directions in Section 6.

2. RELATED WORK

The truthFinder system shares its design and consideration with several recent efforts of data calibration work. We categorize the related works into two groups as the map matching algorithms and the methods of ground truth path exploration.

2.1 Map Matching

Map matching has been studied in many litterateurs [13] [10] [1] [11]. Different map matching algorithms have different strategies varying from those using simple search techniques to those using more advanced techniques. In [13], the authors present an in-depth literature review of map matching algorithms. Generally, the existing algorithms are classified into four classes: 1) *geometric analysis*, which makes use of the geometric information of the spatial road network data by considering only the shape of the links [6]; 2) *topological analysis*, which makes use of the geometry of the links as well as connectivity and contiguity of the links [14]; 3) *probabilistic map matching algorithms* [12] and 4) *advanced map matching algorithms*, which use more refined concepts such as a Kalmam Filter or a fuzzy logic model or a Hidden Markov Model [11].

2.2 Ground Truth Exploration

Generally, according to the dataset used in the evaluation of the aforementioned map matching works, ground truth path exploration methods can be classified into three classes:

Datasets collected by driving vehicles. The researchers of [1] [11] drive around the city, and periodically record the GPS positions together with the roads where they drive on. At the end of the travel, they will get a sequence of the raw GPS reports, along with the path they have passed. Each of the reports will be assigned with a road segment to indicate where the vehicle is at the time it is reported. After the assignment, the GPS data contains both the reports information and the topological information. Then, the GPS reports are used as the input of map matching, while the paths are treated as the ground truth data. This approach is widely used because the GPS reports and the ground truth paths are well matched as the paths are constructed by the actual driving route. However, because this approach is highly time-consuming, it is not likely to collect a large such dataset in this way.

Human labeled datasets. This method has been used in [10] [17]. The researchers start with a set of raw GPS trajectories without any prior knowledge of the actual paths. Then they find the most likely road segment for each of the GPS records in the trajectory to represent the GPS record is reported from. After assigning all of the records, a path will be created. As the path is assessed by human intelligence for each of the records, the accuracy is guaranteed at a very high level (almost 100%). Therefore, the path is considered as a ground truth path. As it is easy to collect a large set of raw GPS trajectories and the corresponding road network, this method is capable of generating a large dataset of trajectories with ground truth paths. However, simply generating the ground truth path based on the raw GPS data is always expensive and inefficient.

Synthetic datasets. Some works[10] also generate ground truth data synthetically. They pick up a path from the road network, periodically select some points on the path, and introduce some errors with normal distribution to generate the synthetic data. Afterward, the paths generated are used as the ground truth data. This is presumably the most inexpensive way to generate a dataset containing both raw GPS trajectories and their ground truth paths. However, there exist differences between the synthetic and the real world dataset, e.g., example, the driving pattern, the reports sampling interval, the GPS position error distribution, and etc. Thus it is always not suitable to use only the synthetic dataset to evaluate the performance and accuracy of a map matching algorithm.

3. DESIGN OF TRUTHFINDER

Motivated by the issues of map matching and ground truth exploration, truthFinder is proposed to interactively match the raw GPS trajectories onto a road network with the help of both traditional map matching algorithm and human intelligence to explore the ground truth data efficiently. It should be noticed that the goal of truthFinder is different from that of the traditional map matching algorithms. For traditional map matching algorithm, they are always used as the tool of calibrating a large volume of data to the road network with a high accuracy and low latency. However, because of several reasons (e.g., the complexity road network, the outlier of the trajectory data), it is impossible for the map matching algorithm to keep its result consistently with high accuracy all the time in all of the situations. So it can not be used as a tool of exploring the ground truth data from the raw GPS data. While the truthFinder is an interactive system which has human effort involved, so it is observed that the explored data can be used as ground truth. However, as human label is involved in truthFinder, it is not possible to explore a very large volume of GPS data, for example trajectory data collected in two years. However, truthFinder can be used to explore as much ground truth data as possible, for example, ground truth of enough trajectories data which can cover whole of the Shanghai, for example, data collected in two days. In summary, map matching algorithm is used to calibrate a large volume of GPS data, while truthFinder is used to explore a complete dataset for other aims, like map matching algorithm validation and parameter tuning.

3.1 Design Issues

Generally, it is a good idea to explore the ground truth data by interactively matching the GPS reports which incorporates traditional map matching algorithms and human intelligence in a unified manner. This method not only keeps the accuracy to be almost 100%, but also has a low cost for ground truth exploration. However, to explore the ground truth from the raw GPS data by human-labeling is very challenging, especially in the environment where the road network is complex. There are mainly four challenging issues:

- To find the right position from mass of candidate roads effectively by the users is a challenge. There are always many roads around each of the GPS positions, especially in the environment where the road network is very complex. It is difficult for human to find the right road where the GPS record was reported from.
- To select the intermediate roads between two roads in a complex road network is difficult. After the roads are identified for two consequent GPS reports, it still costs human much effort to recognize the path between the two roads, especially for the situations where the two positions are far away.
- To correct all of the reports onto the right roads in one time is always impossible. For example, in the complex road network, roads may overlap, which may leads the user map the report onto a right position but wrong road. Then an unreasonable path may be explored.
- To explore ground truth data from a long trajectory (thousands of GPS reports are included) is very time costly. For example, if we use the trivial method like exploring ground truth data from scratch, the cost is always linear to the record number, which is always very large, like hundreds to thousands. It will cost the users a lot of operations to add the road segments and the mediate road segments between each adjacent pair of the GPS reports.

3.2 System Overview

Motivated by the previous discussed design issues, we design truthFinder with several considerations. The architecture of our proposed interactive system is shown in Fig. 1. It composes of four major components: *Recommendation Preparation*, *Information Presentation*, *Candidates Assessing and Tuning*, and *Ground Truth Data Exploring.*

Generally, the work flow of truthFinder can be summarized as follows: First, given a sequence of raw GPS reports, the recommendation preparation component generates a sequence of road segments (potential ground truth path) using a selected map matching algorithm, like STM [10], HMM [11], etc. Second, the information presentation component visualizes the path along with the original trajectory onto the digital map. If the user accepts the accuracy of the path,

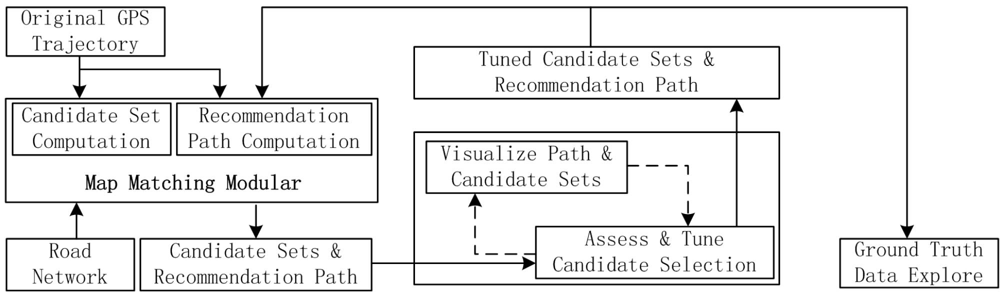

Original GPS Trajectory
Candidate Set Computation
Recommendation Path Computation
Map Matching Modular
Road Network
Candidate Sets & Recommendation Path
Tuned Candidate Sets & Recommendation Path
Visualize Path & Candidate Sets
Assess & Tune Candidate Selection
Ground Truth Data Explore

Figure 1: Overview of truthFinder system architecture

then system will jump to the last step. Third, if the accuracy is failed to achieve a high accuracy, the system asks the user to adjust the path by adding and removing a number of road segments. Fourth, based on the adjustment, the selected map matching algorithm takes the tuned path as the input and generates a new path that is supposed to be more accurate than the previous one. After a new recommended path is generated, truthFinder goes on its process from the step 2, and iteratively runs the step 2, 3 and 4 until an accurate enough path is found. Finally, truthFinder saves the path which is supposed to be accurate enough, and uses this path as ground truth for the original inputted trajectory. In the following sections, we present the details of our truthFinder one component by one component.

3.3 Recommendation Preparation

Based on the observation that human-labeled data can achieve a high accuracy, ground truth can be trivially explored by evaluating the possibility for each of the roads near the report position one by one for each of the records in the trajectory, and finding out the ground truth path by adding all of the most likely road segments to form a ground truth path. In this paper, we call this method *EPScratch*.

However, the cost of EPScratch is always linear to the record number, which is always very large, like hundreds to thousands. It will cost the users a lot of operations to add the road segments, which means it will take the users a very long time to explore the ground truth path from scratch. So how to significantly reduce the number of human operations needed to operate on the ground truth exploration is the key of the interactive map matching system. For this reason, truthFinder uses recommendation preparation component to generate a potential good path which is calculated by the traditional map matching algorithms to reduce the operations needed.

First, Recommendation Preparation component will generate a potential better path based on the original trajectory. Given a GPS trajectory T: $r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_n$, truthFinder runs a traditional map matching algorithm basing on the related road network $G(V, E)$ to generate a recommendation path P: $e_1 \rightarrow e_2 \rightarrow ... \rightarrow e_m$. As this component is iteratively called by truthFinder, both the original trajectory and the human tuned path can be the input of this component. Besides, truthFinder also retrieves the possible candidate roads $CandRoadSet_i = \{(Road_1, score_1), ..., (Road_j, score_j)\}$ for each of the GPS records r_i $(1 \leq i \leq n)$ on the trajectory to assist the user to select a better path. In the candidate road sets, each element consists of a road e_i and a score value $e_i.score$ which represents the possibility. After that, the recommendation preparation component will pack the trajectory T, the recommendation path P, and the candidate roads sets $CandRoadSet_i$ $(1 \leq i \leq n)$ together, and then sends them to the information presentation component.

3.4 Information Presentation

This component is used to visualize the information packed from the preparation component including: the trajectory T, the candidate road sets $CandRoad_i$ $(1 \leq i \leq n)$, and the path P. However, as there are always many candidate roads for each GPS position, especially in the environment where the road network is very complex, how to visualize the information from recommendation component is a big challenge. The challenge of visualization mainly comes from the mass of candidate roads, and the intermediate edges between two GPS positions. To overcome these challenges truthFinder uses the following techniques:

First, multi-layer is introduced in truthFinder. As we calculate the most likely path, the recommended path, we show them on the top layer. So that the user can assess the accuracy of the path easily. Also as there always exist wrong road segments in the recommended path, the top-level showing of the path makes user easier to find which roads should be removed from the path.

Second, multi-color notation is used for the candidate roads showing. Finding the most accurate road from the mass candidate road sets with a same color is challenged. The truthFinder system proposes a probability based color notation for the candidate road selection. The probability is calculated by the distance of the observed position to the road and the route between the consequent two candidate roads. The roads with a color representing higher probability will be more likely to be selected to add into the path.

Third, real time path showing is proposed. The user should assess their selection of the candidate roads both of the positions and the route between the two positions. And a good method to show the path between two selected candidate roads is needed. The truthFinder system designs a real time path showing technique. When a new candidate road is selected, truthFinder automatically calculates the intermediate roads to next selected position, and shows them with different colors. With our experience of using truthFinder, this makes the user convenient to select the most likely path.

The information representation component makes the ground truth data exploring from the historical GPS data much easier and more efficient with high confidence.

3.5 Assessing and Tuning

The existing map matching algorithms are always sensitive to the map context and thereby the accuracy is not guaranteed for situations other than their experiment settings. To improve the accuracy and make the accuracy stable, human effort is needed for assessing whether the result is good enough to be exported as a ground truth data. When there are some of the GPS positions which can find a better candidate, the user should tune them and compose a better ground truth path. After tuning the positions, a new recommendation path will be created, which will be iteratively treated by the recommendation preparation component.

Generally, the user operations for tuning include: 1) deleting an unreasonable edge from the recommended path, 2) recognizing a better road segment and adding it into the path, 3) adding the intermediate edges between two GPS positions.

With the information preparation component, deleting the unreasonable edges from the recommended path is efficient. For example, as both the trajectory of raw GPS trajectory data and the recommended path are showed in truthFinder, users can compare them by their shape, potential route path, and other factors. The differences between the trajectory and the path can be easily found. Therefore, deleting unsuitable roads can be done by comparing between these candidate roads.

While for recognizing and adding operations, after a better candidate road is recognized, the user should add it into the recommended path, and consensually, adding the route path between two candidate roads to the recommended path. When these steps are finished, a better path would be found. However, errors may still exist in the recommend path. Users should iterate these steps until a ground truth data is found.

3.6 Ground Truth Data Exploring

After interactively visualized, assessed and tuned, a final recommended path will be generated. As the path is assessed by human effort, it is guaranteed to have a high accuracy, and reasonable to be treated as a ground truth data. Then the ground truth will be explored.

The truthFinder system explores the ground truth path information together with the original GPS data. Both the position and the road id where the position located on the path will be exported. For example, we use truthFinder to explore the ground truth path for the GPS trajectory T: $r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_n$, suppose after map matching with truthFinder, a path P: $e_1 \rightarrow e_2 \rightarrow ... \rightarrow e_m$ is found, where each position r_i $(1 \leq i \leq n)$ in T is mapped onto an edge e_{ij} at position r_{ij}, where $e_{ij} \in e_1, e_2, ..., e_m$. We not only explore the original information included in trajectory T, but also explore the longitude and latitude of r_{ij} together with the $e_{ij}.id$.

With this last phase, truthFinder generates a recommended path with high accuracy. To the best of our knowledge, it is the best method to explore the ground truth path data from the original GPS data with human intelligence involved. As the result is assessed with human intelligence, the explored dataset can be used as ground truth data. As such, it is widely used to generate ground truth dataset to evaluate map matching algorithms [10][17].

4. COST MODEL FOR TRUTHFINDER

To define the cost model, we first give some preliminaries, and then define a weighted cost model based on these preliminaries for the cost in each of the iterations. After that, we will give the cost model for the total cost of truthFinder in terms of operation per record.

Table 1: Main variables used in cost model.

Var	Description
r_i	the i_{th} position in the trajectory, $1 \leq i \leq n$
S_j	error positions set in the j_{th} iteration
N_j	number of error positions in set S_j
E_t	threshold of error positions in ground truth path
A_m	accuracy of the map matching algorithm
A_h	human ability to correct error position
w_d	cost of deleting an edge from the path
w_a	cost of adding a road segment into the path

4.1 Preliminaries

Viewing from the process of truthFinder, the phases involved include preparing a recommended path, visualizing the recommend data and interactively tuning them with human assessing, and finally exploring the ground truth. In these phases, human effort is mainly involved in the phase of tuning and assessing the path with several types of operations, like deleting an edge, adding a better road segment, etc. As different operations may have different cost (for example, deleting an edge from the recommended path always costs less than the operation of recognizing a better candidate road and adding it into the path), the cost of human aid is calculated from this phase in term of weighted operations.

To discuss our cost model conveniently, we give several definitions of the variables used in the cost model. Table 1 summarizes the main variables. Adding a road into a path needs to find and assess the suitable ones from a bundle of candidate roads. While deleting a road from a path can be done directly. Generally, w_a is always much bigger than w_d.

4.2 Cost Model

The cost model for truthFinder is composed by each of the iterations involved in the assessing and tuning phase. First, we give the cost of truthFinder in one of the iterations. Let's consider the cost of the j_{th} iteration. As we notated in Table 1, there are N_j error positions in the recommended path, and the positions are r_{jk}, where $r_{jk} \in S_j$. For each error position, user has to tune it to a better road by deleting it from the recommended path, and adding a better position, and the intermediate edges. So the cost for the error position r_{jk} in the j_{th} iteration is

$$C_{jk} = N_{jkd} * w_{jkd} + N_{jka} * w_{jka} \qquad (1)$$

for each $1 \leq k \leq N_j$. So the total cost for the j_{th} iteration goes to

$$C_j = \sum_{k=1}^{N_j} (C_{jk}) \qquad (2)$$

In this model for the j_{th} iteration, the cost is directly impacted by the number of operations and the corresponding cost weight. As such, to minimize the cost of adding and deleting operations, we should keep the weight of the operation cost at a low level. Empirically, the value of deleting a road from the ground truth candidate path is always stable. So the way to reduce the cost would be minimizing the weight of adding a road segment as low as possible.

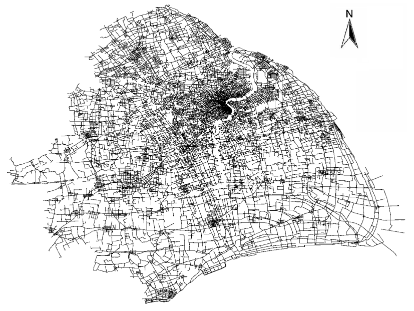


Figure 2: Road network of Shanghai, China.

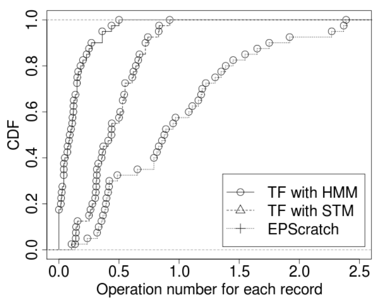


Figure 3: CDF of cost.

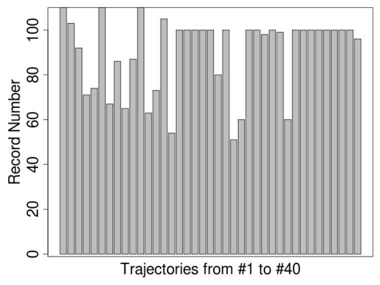


Figure 4: Record numbers of the trajectories.

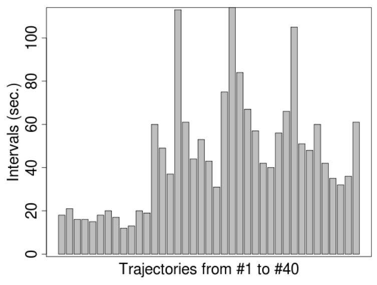


Figure 5: Average sampling intervals of the trajectories.

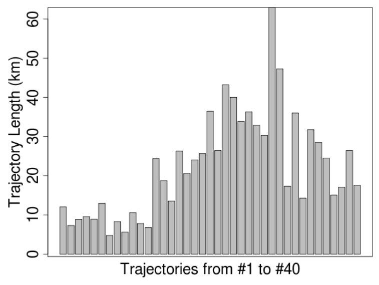


Figure 6: Lengths of the trajectories.

The weight of adding a road segment into the path, w_a, is always affected by the map context. In a complex road network, where roads are very dense, it always costs more to find a better road to be added into the ground truth path. So this operation of adding will cost more than that of adding a road into the truth path in a situation where only few roads are included in the spare region. As shown by the cost model, reducing the value of w_a makes the cost of ground truth exploration less, it is necessary to reduce the cost of adding a road into the ground truth path. For this reason, truthFinder selects the top k candidate roads for each report with the highest possibilities. This makes the adding a road by selecting it from the candidate road set at a low cost. However, one situation should be considered, that if the road segment in the ground truth path is not included in the selected top k candidate roads, then the user has to find a road from the original map, and add it into the path. This will make the cost very high as a penalty, which means the weight of the adding operation w_a very high.

As exploring a ground truth path from a given trajectory is done by iteratively assessing and tuning the recommended path, the total cost for truthFinder to explore the path should be iteratively added. In every iteration, the cost is calculated by Eq. (2), so the total cost for truthFinder is to add up the cost in every iteration. As we supposed, the accuracies of the algorithm and human are A_m and A_h, truthFinder has to iterative the assessing and tuning phase I iterations, where

$$I \leq \frac{\log(\frac{E_t}{1-A_m})}{log(1-A_h)} \tag{3}$$

Then, in average, for each of the reports, the total cost of exploring the ground truth data becomes the total cost of each iteration divided by the number of reports, which is

$$\bar{C} = \sum_{j=1}^{I} \sum_{k=1}^{N_j} (N_{jkd} * w_{jkd} + N_{jka} * w_{jka})/n \tag{4}$$

As the model shows, the average cost per report depends on the weighted cost in each iteration of the ground truth exploration process which we have discussed the previous section, and the iteration numbers. Relatively, the iteration number is decided by the accuracy of map matching algorithms and the ability of the user to correct the wrong selected roads. However, it is difficult to define a metric and fairly evaluate the ability. But empirically, the user can always reduce the iteration number at a very small value (like two to three iterations) for most of the case we did in the evaluation. Actually, it is an interesting and important work, we will study it in future work.

5. EVALUATION

The objective of the experiments are to evaluate the cost of truthFinder under our defined cost model, and find out how the map matching algorithms impact the cost of truthFinder. In this section, we present representative results for truthFinder. We first state the description of the experimental settings. After that, we give the results of our experiments including (i) the cost comparing to EPScratch, and (ii)the impact of map matching algorithms.

5.1 Experimental Settings

We present the experimental settings in this section, including the dataset we used and the road network of the city where we collected our dataset. The truthFinder system is deployed on an IBM server which has 4G memory and a

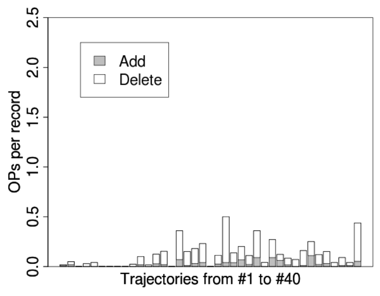


Figure 7: Cost of truthFinder using HMM.

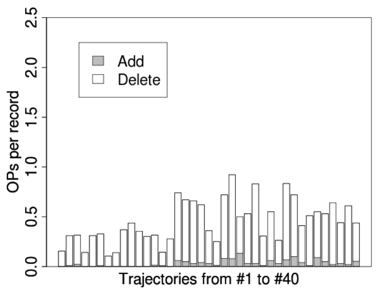


Figure 8: Cost of truthFinder using STM.

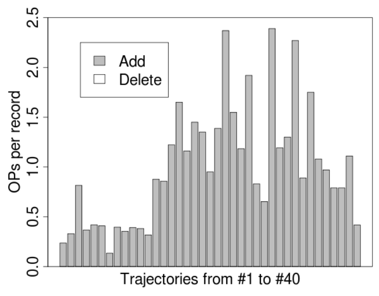


Figure 9: Cost of EPScratch.

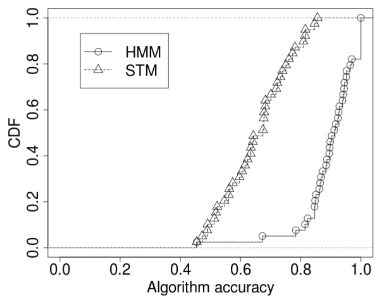


Figure 10: CDF of accuracy of the map matching algorithms.

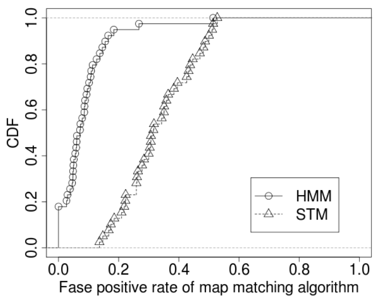


Figure 11: CDF of the false positive rate.

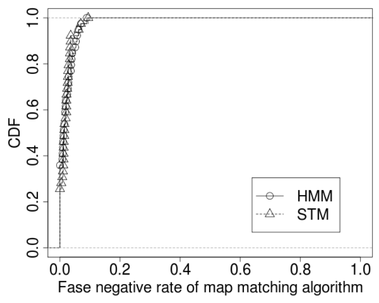


Figure 12: CDF of the false negative rate.

CPU of Intel Xeon with 8 processors (E5405 @ 2.00 GHz).

Road Network Since 2005, we have collected a large volume of GPS reports from Shanghai Grid. As our trajectory dataset was collected from Shanghai, we use the road network of Shanghai as shown in Fig. 2 in our evaluation experiments. There are 22180 vertexes and 65510 directed roads in this network. So for the road network $G(V, E)$ of Shanghai, $\#V = 22180$ and $\#E = 65510$.

Trajectory Dataset To explore the ground truth with truthFinder, we randomly selected a dataset with 40 trajectories in our experiments for comparing truthFinder to the method of EPScratch. The characteristics of the trajectories in the dataset are presented in the figures. Fig. 4 shows the record number for each trajectory in dataset. As it shows, the number varies from 50 to 120. We calculate trajectory length by adding the direct distance between every two adjacent reports. The distance ranges are showed in Fig. 6 (about 10km to about 60km). The sampling intervals of the reports in each trajectory are shown in Fig. 5.

5.2 Cost of truthFinder

In this section, we present the experiment result on the dataset to compare operation cost of truthFinder and EPScratch. We have done experiments on both the HMM [11] and STM [10] map matching algorithms in truthFinder for the recommendation path generating. We compare them in term of operation numbers, where operations of adding road and deleting road are separated.

As demonstrated by Fig. 7, truthFinder keeps the operation cost very low, which is about 0.2 operations for each record using HMM algorithm. While for the situation where truthFinder uses the STM algorithm, the cost is a little high, which come to 0.5 for every record (Fig. 8). The reason is that the accuracy of STM algorithm is a little lower than HMM algorithm in our dataset and map context. When the map matching algorithm has a high accuracy, truthFinder will significantly reduce the cost of human operations for matching the raw GPS trajectories to its ground truth path. Meanwhile from Fig. 3, we find that, about 98% of the ground truth path of the trajectories can be explored within 0.5 human operations for each record, and about 80% of them can be done within 0.2 operations.

While the method of EPScratch costs much higher than truthFinder, especially the adding roads operations. The cost for finding the ground truth for most of the trajectories are always very high, compared to our truthFinder based method. It is almost two orders of magnitude of that of truthFinder based on HMM map matching algorithm. The reason is that, not only the roads where the records are reported from should be added into the found-out ground truth path, but also the intermediate roads should be added. As discussed in Section 4.1, w_a is much larger than w_d. The total weight cost of EPScratch will be very large. The truthFinder system reduces the cost of map matching process up to two orders of magnitude less than the EPScratch approach. With this comparison, we are confirmed that, truthFinder will reduce the total cost significantly and can explore more trajectories than EPScratch.

5.3 Impact of Map Matching Algorithms

Next, we concern the impact of the traditional map matching algorithms for truthFinder. We first present the accuracy of the map matching algorithms using our dataset and the explored ground truth path. Then we present the impact of the map matching algorithms.

We have implemented two map matching algorithms, including STM [10], and HMM [11]. We use each of the trajectories in the dataset as the input of each map matching algorithm, which will generate a path (a sequence of roads), as its output. Then we calculate their accuracy. We measure the accuracy of the map matching algorithms with the metric defined in Eq. (5), where $Set_i.ground$ is the set of road IDs in ground truth path, and $Set_i.alg$ is the set of road IDs in algorithm calculated path.

$$A_i = \frac{\#of(Set_i.ground \cap Set_i.alg)}{\#of(Set_i.ground \cup Set_i.alg)} \qquad (5)$$

Fig. 10 shows the accuracy of different map matching algorithms in our road network with our dataset. From the result, we can find that, in our experiment environment, the accuracies of HMM algorithm, most of which are between 80% and 93%, are always higher than that of STM algorithm, whose values changes frequently and always are lower than 80%. Together with the results in the previous experiments, we are confirmed that the higher accuracy the map matching algorithm has, the less operations the truthFinder costs.

As demonstrated by Eq. (4), the cost of truthFinder depends on the number of adding and deleting a road operations, as well as the weight of these operations. The operation of adding a road is always caused by the situation that a correct road in the ground truth path not included in the recommended path (false negative), while the operation of deleting a road is caused by the reason that wrong roads are included in the recommended path (false positive). So we calculated the false negative (Fig. 12) and false positive (Fig. 11) rate for both of these two map matching algorithms. From the results, we can find that, for the false negative rates, HMM and STM algorithms share similar trends, so relatively, as demonstrated in figures of the system cost, the adding operations of truthFinder are also similar. However, the false positive rates of HMM are always less than 0.2 which is less than that of STM algorithm whose rate changes frequently ranging from 0.15 to 1. Together with results of cost, we can find that the higher the false positive rate is, the more the deleting are needed. Similarly, the higher the false negative rate is, the more the adding are needed.

6. CONCLUSION

In this paper, we propose truthFinder, a system of interactive map matching the collected raw GPS trajectory data. The truthFinder system characterizes itself with several unique features. It employs human intelligence to aid the map matching algorithms to explore ground truth data from raw GPS data. To measure the cost, we define a cost model and evaluate our prototype system with this model. The result shows that truthFinder significantly reduces the cost. The truthFinder system would be an efficient way to solve the validation issue map matching problem.

7. ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments and kindly suggestions. This work is supported by the National Natural Science Foundation of China under Grant No. 60736013, Grant No. 61120106005, Grant No. 61025009 and Grant No. 60903040.

8. REFERENCES

[1] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proceedings of the VLDB'05*, pages 853–864. VLDB, 2005.

[2] Z. Chen, H. Shen, and X. Zhou. Discovering popular routes from trajectories. In *Proceedings of the ICDE'11*, pages 900–911. IEEE, 2011.

[3] Z. Chen, H. Shen, X. Zhou, Y. Zheng, and X. Xie. Searching trajectories by locations: An efficiency study. In *Proceedings of the SIGMOD'10*, pages 255–266. ACM, 2010.

[4] P. Cudre-Mauroux, E. Wu, and S. Madden. Trajstore: An adaptive storage system for very large trajectory data sets. In *Proceedings of the ICDE'10*, pages 109–120. IEEE, 2010.

[5] H. Gonzalez, J. Han, X. Li, M. Myslinska, and etc. Adaptive fastest path computation on a road network: A traffic mining approach. In *Proceedings of the VLDB'07*, pages 794–805. VLDB, 2007.

[6] J. Greenfeld. Matching gps observations to locations on a digital map. In *Proceedings of the 81th Annual Meeting of the Transportation Research Board (TRB'02)*, 2002.

[7] M. Li, M. Wu, Y. Li, J. Cao, L. Huang, Q. Deng, X. Lin, C. Jiang, W. Tong, Y. Gui, et al. Shanghaigrid: an information service grid. *Concurrency and Computation: Practice and Experience*, 18(1):111–135, 2006.

[8] Z. Li, M. Ji, J. Lee, and etc. Movemine: mining moving object databases. In *Proceedings of the SIGMOD'10*, pages 1203–1206. ACM, 2010.

[9] S. Liu, Y. Liu, L. Ni, J. Fan, and M. Li. Towards mobility-based clustering. In *Proceedings of the SIGKDD'10*, pages 919–928. ACM, 2010.

[10] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. In *Proceedings of the GIS'09*, pages 352–361. ACM, 2009.

[11] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the GIS'09*, pages 336–343. ACM, 2009.

[12] W. Ochieng, M. Quddus, and R. Noland. Map-matching in complex urban road networks. *Revista Brasileira de Cartografia*, 2(55), 2009.

[13] M. Quddus, W. Ochieng, and R. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312–328, 2007.

[14] M. Quddus, W. Ochieng, L. Zhao, and etc. A general map matching algorithm for transport telematics applications. *GPS solutions*, 7(3):157–167, 2003.

[15] K. Tufte, J. Li, D. Maier, and etc. Travel time estimation using niagarast and latte. In *Proceedings of the SIGMOD'07*, pages 1091–1093. ACM, 2007.

[16] M. Xie, L. Lakshmanan, and P. Wood. Comprec-trip: A composite recommendation system for travel planning. In *Proceedings of the ICDE'11*, pages 1352–1355. IEEE, 2011.

[17] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G. Sun. An interactive-voting based map matching algorithm. In *Proceedings of the MDM'10*, pages 43–52. IEEE, 2010.

Coordinated Clustering Algorithms to Support Charging Infrastructure Design for Electric Vehicles

Marjan Momtazpour[1], Patrick Butler[1], M. Shahriar Hossain[1],
Mohammad C. Bozchalui[2], Naren Ramakrishnan[1], Ratnesh Sharma[2]
[1]Department of Computer Science, Virginia Tech, VA, 24060, USA
[2]NEC Laboratories America, Inc., CA, 95014, USA
{marjan, pabutler, msh}@cs.vt.edu,
mohammad@sv.nec-labs.com, naren@cs.vt.edu, ratnesh@sv.nec-labs.com

ABSTRACT

The confluence of several developments has created an opportune moment for energy system modernization. In the past decade, smart grids have attracted many research activities in different domains. To realize the next generation of smart grids, we must have a comprehensive understanding of interdependent networks and processes. Next-generation energy systems networks cannot be effectively designed, analyzed, and controlled in isolation from the social, economic, sensing, and control contexts in which they operate. In this paper, we develop coordinated clustering techniques to work with network models of urban environments to aid in placement of charging stations for an electrical vehicle deployment scenario. We demonstrate the multiple factors that can be simultaneously leveraged in our framework in order to achieve practical urban deployment. Our ultimate goal is to help realize sustainable energy system management in urban electrical infrastructure by modeling and analyzing networks of interactions between electric systems and urban populations.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: [Database Applications - Data mining - Spatial databases and GIS]; I.5.3 [**Pattern Recognition**]: [Clustering]; I.2.6 [**Artificial Intelligence**]: [Learning]

General Terms

Experimentation, Algorithms, Design, Measurement

Keywords

Data mining, clustering, coordinated clustering, smart grids, electric vehicles, synthetic populations.

1. INTRODUCTION

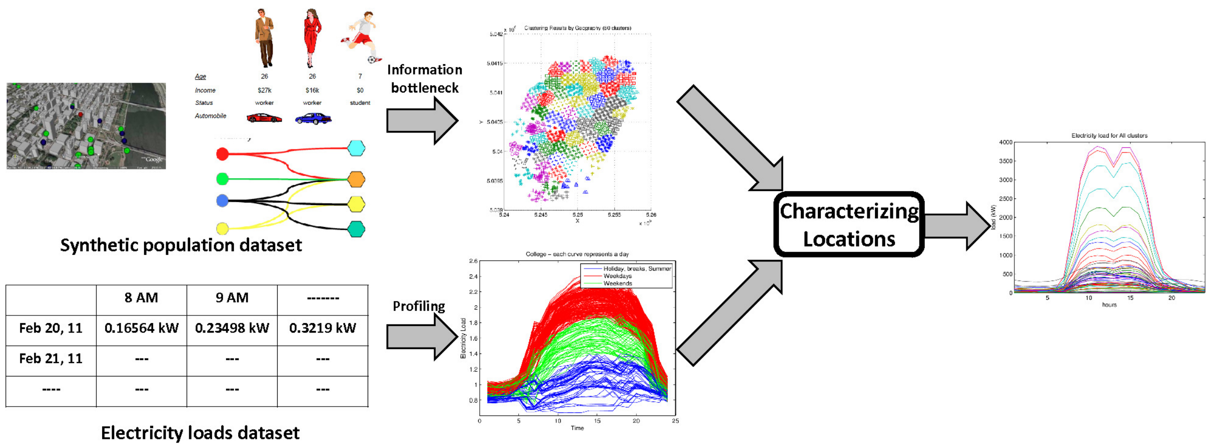
The impending decline of fossil fuels is rapidly ushering an emphasis from traditional methods of energy production, distribution, and consumption to sustainable approaches [11]. The advent of electric vehicles (EVs) is one such promising shift but to prepare for a world laden with EVs we must revisit smart grid design and operation.

One of the key issues in ushering in EVs is the design and placement of charging infrastructure to support their operation. Issues to be taken into account include [11]: (i) prediction of EV charging needs based on their owners' activities; (ii) prediction of EV charging demands at different locations in the city, and available charge of EV batteries; (iii) design of distributed mechanisms that manage the movements of EVs to different charging stations; and (iv) optimizing the charging cycles of EVs to satisfy users' requirements, while maximizing vehicle-to-grid profits.

In this paper, we address the charging infrastructure design problem by adopting an urban computing approach. Urban computing, [8], is an emerging area which aims to foster human life in urban environments through the methods of computational science. It is focused on understanding the concepts behind events and phenomena spanning urban areas using available data sources, such as people movements and traffic flows.

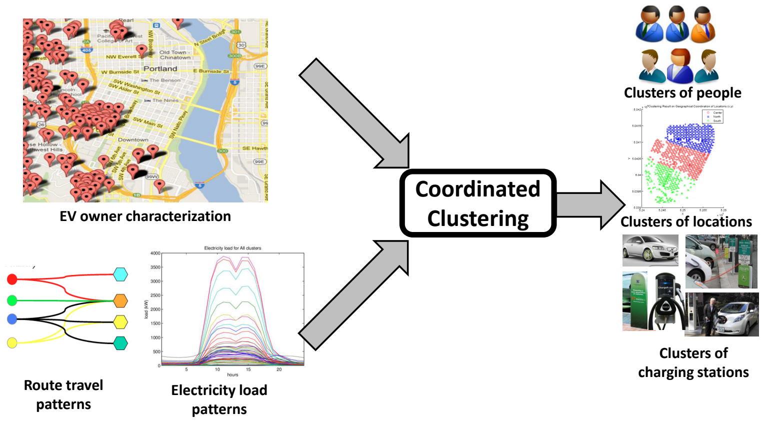
Organizing relevant data sources to solve compelling urban computing scenarios is itself an important research issue. Here, we use network datasets organized from synthetic population studies, originally designed for epidemiological scenarios, to explore the EV charging station placement problem. The dataset was organized for the SIAM Data Mining 2006 Workshop on Pandemic Preparedness [3] and models activities of an urban population in the city of Portland, Oregon. The supplied dataset [1] tracks a set of synthetic individuals in Portland and, for each of them, provides a small number of demographic attributes (age, income, work status, household structure) and daily activities representing a normative day (including places visited and times). The city itself is modeled as a set of aggregated activity locations, two per roadway link. A collection of interoperable simulations—modeling urban infrastructure, people activities, route plans, traffic, and population dynamics—mimic the time-dependent interactions of every individual in a regional area. This form of 'individual modeling' provides a bottom-up approach mirroring the contact structure of individuals and is naturally suited for formulating and studying the effect of intervention policies and considering 'what-if' scenarios.

In more detail, we characterize this dataset with a view toward understanding the behavior of EV owners and to determine which locations are most appropriate to install charging stations. We develop a coordinated clustering formulation to identify a set of locations that can be considered



	8 AM	9 AM	-------
Feb 20, 11	0.16564 kW	0.23498 kW	0.3219 kW
Feb 21, 11	---	---	---
----	---	---	---

(a) Discovering location functionalities and characterizing electricity loads.



(b) Coordinated clustering of people, locations, and charging stations.

Figure 1: Overview of our methodology.

as the best candidates for charging stations.

2. RELATED WORK

We survey related work in two categories: mining GPS datasets and smart grid analytics. GPS datasets have emerged as a popular source for modeling and mining in urban computing contexts. They have been used to extract information about roads, traffic, buildings, and people behaviors [20], [21], [9]. The range of applications is quite varied as well, from anomaly detection [9] to taxi recommender systems [21] that aim to maximize taxi-driver profits and minimize passengers' waiting times. The notion of location-aware recommender systems is a key topic enabled by the increasing availability of GPS data, e.g., recommending points of interest to tourists [22]. We survey these works in greater detail next.

In [20] Yuan et al. proposed a framework to discover regions of different functionalities based on people movements. They adapt algorithms from the topic modeling literature, by mapping a region as a document and a function as a topic so that human movements become 'words' in this model. The focus of [21] and [19] is different: here, GPS data is used to mine the fastest driving routes for taxi drivers. In [21], Yuan et al. mined smart driving direction from GPS trajectory of taxis, and in [19] they consider driver behavior using other metrics such as driving strategies and weather conditions.

Clusters of moving objects in a noisy stadium environment are detected using the DBSCAN algorithm [5] in [12]. This task supports monitoring a stadium for groups of individuals that exhibit concerted behavior. In [14], the authors estimate distributions of travel-time from GPS data for use in routing and route-recommendation.

Our work here is different from the above works in that we use a synthetic population dataset and routes are based on people's travel habits that are mapped using geographical coordinates and road infrastructures. We are also not *per se* interested in mining the routes but to use the route information to better support charging infrastructure placement.

Smart grid analytics has emerged as a promising approach to usher in the promise of smart grid benefits. Researchers have begun to explore the problems concomitant with EV penetration in urban areas, especially unacceptable increases in electricity consumption [11]. A promising way to approach this problem is to understand the interactions between grid infrastructure and urban populations. While smart grids and EVs have been studied previously from technical and AI point of views, there is a limited number of research on smart grids from an urban computing perspective.

In this space, agent-based systems have been proposed to simulate city behavior in terms of agents with a view toward designing decentralized systems and maximizing grid profits as well as individuals' profit [11]. In [2] information from smart meters is used for forecasting energy consumption patterns in a university campus micro-grid, whose results can be used for future energy planning. In our work, we directly study the problem of charging station placement using coordinated clustering algorithms.

3. METHODOLOGY

Our overall methodology is given in Figure 1. We describe each of the steps in our approach next. At a basic level, we integrate two basic types of data to formulate our data mining scenario. The first data, as described earlier, is a synthetic population of people and activities representing the city of Portland and the second data set is electricity consumption profile of each location. Notice that the proposed methodology is a generic approach and can be applied to real-world data and the fact that we use synthetic data here is only due to our lack of access to real-world data to test our proposed methodology.

The synthetic dataset contains 243,423 locations of which 1,779 are selected as belonging to the downtown area and of further interest for our purposes. Each location is represented by geographical [x,y] coordinate adopting the universal transverse mercator coordinate system (UTM) [1]. There are a total of 1,615,860 people in the entire city. Information about them is organized into households, and for each household we have the details of number of people in the household, and the ages, genders, and incomes of each household member. Each person has a unique ID.

We have some information about each person including age, gender, income, and his/her house ID. The typical movement patterns of people in a typical day (27 hour period) are also available. A total of 8,922,359 movements are provided. In addition to starting and ending locations for people's movements, this dataset also provides the *purpose* of the movement, categorized into nine types: {Home, Work, Shop, Visit, Social/Recreational, Serve Passenger, School, College, and Other}. A given person moves from one location to another location at a specific time for a specific purpose (from the nine mentioned above) and stays in that location for a specified period of time. These movement types can thus be utilized for further detailed studies. We also have the ability to map the locations using Google Maps and calculate distances of traveling between locations.

To this dataset, we augment information about electricity consumption of each location and simulate the effects of EVs on its electricity demand profile. Since actual electricity consumption data for each location is not available until all the consumers have smart meters installed and in operation for some time, we approximate electricity load profile using the existing data (organized by NEC Labs, America).

It is clear that the electricity load of each location greatly depends on the functionality of that location and hence our first approach is to utilize an information bottleneck type approach [17] to characterize locations. Our aim is to cluster locations based on geographical proximity but such that the resulting clusters are highly informative of location function. This is thus our first application of a coordinated clustering formulation, and falls in the scope of clustering with side information. Next, we integrate the electricity load information to characterize usage patterns across clusters with a view toward helping identifying locations to place charging infrastructure.

Our next step is to more accurately characterize usage patterns of likely EV owners. A specific set of clusters from the previous pipeline is used and characterized using high-income attributes as the likely owners of EVs. We then bring in additional factors of locations that influence EV charger placement, e.g., residentiality ratio, load on the location, charging needs, and typical duration of stay in the location. Some of these factors (such as distance traveled) are in turn determined by mapping the home-to-work and work-to-home trajectories of EV owners and their stop locations. We use a coordinated clustering formulation to simultaneously cluster three datasets in a relational setting. Our coordinated clustering framework builds upon our previous work [7] which generalizes relational clustering between two non-homogeneous datasets. This problem is a bit non-trivial since one of the relations is a many-to-many relation and another is a one-to-one relation. The final set of coordinated clusters are then used as interpretation and as a guide to charger placement.

After locating the homes of EV owners, we can determine their trajectories and their stop locations. Then, based on this data, we can estimate their travel distances. This helps us to estimate charging requirements of EVs, during a day. With the help of the distribution of electricity load in the city and charging needs of EVs, we determine proper locations for installing charging stations in city with respect to specific parameters.

4. ALGORITHMS

As described above, our methodology comprises the following four major steps to determine candidate locations for charging stations: (i) discovering locations' functionalities using an information bottleneck method; (ii) electricity load estimation and integrating with results of (i); (iii) studying the behavior of EV owners and calculating specific parameters relevant to their usage patterns; and (iv) candidate selection for charging stations using coordinated clustering techniques. Each of these steps are detailed next.

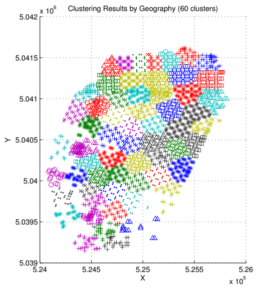
4.1 Discovering Location Functionalities

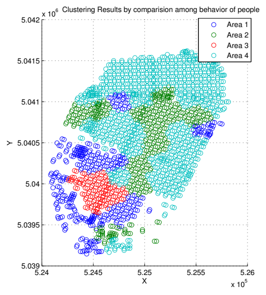
We use information bottleneck methods to characterize locations with a view toward defining the specific purpose of the location. The idea of information bottleneck methods is to cluster data points in a space (here, geography) such that the resulting clusters are highly informative of another random variable (here, function). We focus on 1779 locations in the downtown Portland area whose geographies are defined by (x,y) coordinates and whose functions are given by a 9-length profile vector $P = [p_1, p_2, ..., p_9]$, where p_i is the number of travels incident on that location for the i^{th} purpose (recall the different purposes introduced in the previous section).

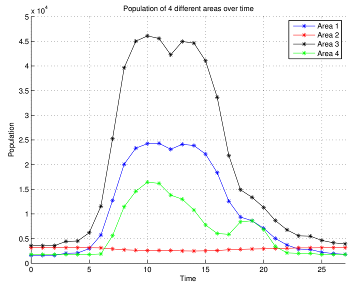
Figure 2 (a) describes the results of a clustering based on euclidean metrics between locations whose results are aggregated in Figure 2 (b) into a revised clustering that also preserves information about activities of people at these locations. The population distribution of these clusters over time is shown in Figure 2 (c) which reveals characteristic changes of crowds around peak hours and lunch times. One final analysis that will be useful is to evaluate each of the discovered clusters with respect to what we term as the *residentiality ratio*. The residentiality ratio for a location is the percentage of people who use that location as a home w.r.t. all people who visit that location (in downtown Portland, many locations have combined home-work profiles, and hence the calculation of residentiality ratio becomes relevant). Figure 2 (d) reveals one cluster with relatively high residentiality ratio among three others.

4.2 Electricity Load Estimation

In order to uncover patterns in electricity load distributions, we now characterize each of the discovered clusters us-


(a)


(b)


(c)

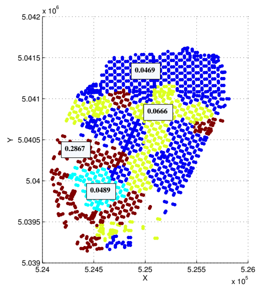
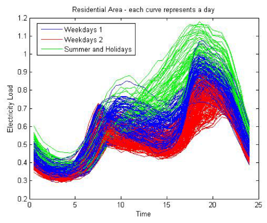
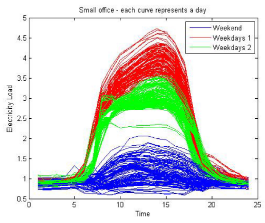
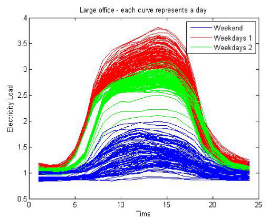

(d)

Figure 2: (a) Clustering downtown locations based on geographic coordinates. (b) Clustering over the previous clustering with people's activities as side-information. (c) Dynamic population of the four discovered clusters over a typical day. (d) Computed residentiality ratio revealing one primary residential cluster.


(a)


(b)


(c)

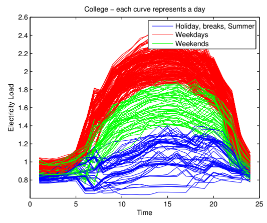

(d)

Figure 3: (a) Electricity usage in residential areas. (b) Electricity usage in small office areas. (c) Electricity usage in large office areas. (d) Electricity usage in college areas.

ing typical profiles gathered from public data sources such as the California End User Survey (CEUS) and other sources of usage information. Figure 3 presents daily electricity consumption profile across large offices, small offices, residential buildings, and colleges for one year. By clustering this data across the year, we can discern important patterns associated with different types of consumption during the year. For instance, in the college setting, we can discern three types of consumption patterns: holiday breaks (including summer), weekdays, and weekends.

Our next step is to compute the electricity load leveraging the above patterns but w.r.t. our network model of the urban environment. Recall that our network model is based on population dynamics but typical electricity load sources are based on square footage calculations. We map these factors using well-accepted measures, i.e., by considering the average square footage occupied by one person in a residential area as 600sft [4], small office as 200sft [18], large office as 200sft [18], college as 50sft [15], retail area as 50sft [15], and other classes as 200. Further, the minimum population for an office to be considered as a large office is set to 300.

Based on some exploratory data analysis, we selected a weekday in the past year (specifically, 18th March, 2011) and used the electricity load data of this day to map to the network model. Consider that in a specific hour, N people go to location l in which n_i of them come for the purpose of p_i while $\sum_{i=1}^{9} n_i = N$. Then the electricity load for that location is computed as

$$E_l = \sum_{i=1}^{9} \frac{n_i A_{p_i} E_{p_i}}{1000}, \qquad (1)$$

where A is the average square footage per person and E_p is electricity consumption of building type p. Observe that a single location can serve multiple purposes and the above equation marginalizes across all uses. For example, if there are 360 people in one location, and 10 of them are in the building for the purpose of home and 350 are for the purpose of office, the total electricity consumption of building would be calculated as $(10 \times 600 \times E_{p_{\text{home}}}/1000) + (350 \times 200 \times E_{p_{\text{office}}}/1000)$ where 600 and 200 are average square footage per person for the different categories, as mentioned earlier. The above methodology enables us to characterize electricity loads in terms of the four location clusters characterized in the previous step (see Figure 4).

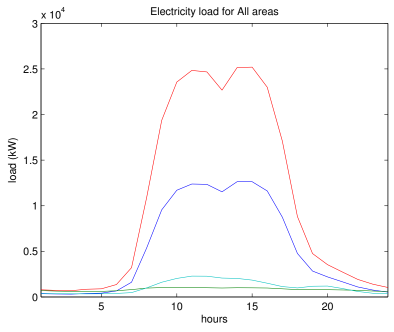


Figure 4: Electricity loads for four characterized location clusters.

4.3 Characterizing EV users

Currently only a small percentage of people use EVs, and this figure is correlated with high income. Based on [10] and [13], only 6 percent of people in the US have income more than 170,000 USD. In our synthetic dataset, 329,218 people make an income greater than 60,000 USD. To explore a hypothetical scenario, we posed the question:

> What if 6.31% of 329,218 people from Portland bought EVs? What charging infrastructure is necessary to support this scenario?

Based on our modeling of these people's movements and patterns, we aim to identify the best locations for charging stations.

Figure 5 (a) gives the distribution of EV users in our potential scenario. We can notice several clusters around high-income neighborhoods. With the aid of Google Maps, we can estimate the amount of time an EV owner drives and how far he/she travels on a regular week day. Figure 5 (b) gives the distribution of distances traveled by these users.

Assuming EV owners charge their cars at their respective homes, our goal is now to identify candidate charging locations during other times. Let us assume that the EV of a person P consumes E_P^C KWh energy per 100 Km. Also, assume that the battery of this vehicle can save E_P^S KWh. Then the estimated total distance that P can travel with his vehicle before he needs to charge its battery is

$$\Delta_P = \frac{100E_P^S}{E_P^C}, \tag{2}$$

As an example, for the Chevrolet Volt [6], with $E_P^S = 16$ KWh and $E_P^C = 22.4$ KWh per 100 Km, the EV can travel 71.43 Km before it needs to be recharged.

If the total traveling distance of P in a day is D_P then the number of times that P needs to charge his vehicle is N_P and is determined as follows:

$$N_P = \left\lfloor \frac{D_P}{\Delta_P} \right\rfloor, \tag{3}$$

As an example, if we assume that an EV's battery can save 16 KWh energy [6], an electric car can go for 71.43 Km before it needs to be charged [16].

Due to the long duration of charging process, we have a constraint to install charging stations only in destinations that people visit. Assume that V_L is the set of EV owners who visited location L during the day. Then $|V_L|$ is the total number of EV owners who have visited location L. However, there is a greater chance for a location to be a charging station if people with higher charge needs visit that location. Hence, the charge needs of location L is determined based on equation 4.

$$W_L = \sum_{P \in V_L} N_P, \tag{4}$$

Figure 5 (c) depicts the histogram of how many times an EV needs to be charged. Also, Figure 5 (d) depicts the charge needs of downtown locations.

On the other hand, each person visit a location for a specific period of time which here we call it duration of stay. In order to put a charging station in one location, we force people to stay for a specific period of time because charging an electric car will take couple of hours. Hence, in locations where people stay longer such as working locations have potential to be charging stations compared to those locations that people stay in them for example half an hour. We use average duration of stay of people in each location as a feature for that location.

It should be noted that the right choice of EV charging stations depends on regular electricity load of each area, the amount of time that each person spends on this location, and number of times that EV owners need to charge their vehicles. Hence, based on EV owners traveling route during peak and off-peak hours, we can come up with a set of candidate regions for charging stations.

4.4 Charging Station Placement using Coordinated Clustering

Since charging EVs is not an instantaneous process, it is helpful to place charging stations at those locations where people visit for an extended period of time. The average duration of stay of people in each location is an important feature in this regard. The right choice of EV charging stations thus depends on the regular electricity load of the area, the amount of time that people spend in the location, and the number of times that EV owners need to charge their vehicles. Hence, based on EV owners' traveling routes during peak and off-peak hours, we can arrive at a set of candidate regions for charging stations.

Let $\mathcal{X}$ be the income dataset and $\mathcal{Y}$ be the locations datasets. $\mathcal{X} = \{\mathbf{x}_s\}, s = 1, \ldots, n_x$ is the set of vectors in dataset $\mathcal{X}$, where each vector is of dimension l_x, i.e., $\mathbf{x}_s \in \mathbb{R}^{l_x}$. Currently, our income dataset contains only one dimension. Similarly, locations dataset $\mathcal{Y} = \{\mathbf{y}_t\}, t = 1, \ldots, n_y, \mathbf{y}_t \in \mathbb{R}^{l_y}$. Locations are denoted by two dimensions (latitude and longitude) in our current database. The many-to-many relationships between $\mathcal{X}$ and $\mathcal{Y}$ are represented by a $n_x \times n_y$ binary matrix B, where $B(s,t) = 1$ if $\mathbf{x}_s$ is related to $\mathbf{y}_t$, else $B(s,t) = 0$. Let $C_{(x)}$ and $C_{(y)}$ be the cluster indices, i.e., indicator random variables, corresponding to the income dataset $\mathcal{X}$ and location dataset $\mathcal{Y}$ and let k_x and k_y be the corresponding number of clusters. Thus, $C_{(x)}$ takes values in $\{1, \ldots, k_x\}$ and $C_{(y)}$ takes values in $\{1, \ldots, k_y\}$.

Let $\mathbf{m}_{i,\mathcal{X}}$ be the prototype vector for cluster i in income dataset $\mathcal{X}$ (similarly $\mathbf{m}_{j,\mathcal{Y}}$). These are the variables we wish to estimate/optimize for. Let $v_i^{(\mathbf{x}_s)}$ (likewise $v_j^{(\mathbf{y}_t)}$) be the cluster membership indicator variables, i.e., the probability that income data sample $\mathbf{x}_s$ is assigned to cluster i in the income dataset $\mathcal{X}$ (resp). Thus, $\sum_{i=1}^{r^x} v_i^{(\mathbf{x}_s)} = \sum_{j=1}^{r^y} v_j^{(\mathbf{y}_t)} = 1$. The traditional k-means *hard* assignment is given by:

$$v_i^{(\mathbf{x}_s)} = \begin{cases} 1 & \text{if } ||\mathbf{x}_s - \mathbf{m}_{i,\mathcal{X}}|| \leq ||\mathbf{x}_s - \mathbf{m}_{i',\mathcal{X}}||, \ i' = 1 \ldots k_x, \\ 0 & \text{otherwise.} \end{cases}$$

(Likewise for $v_j^{(\mathbf{y}_t)}$.) Ideally, we would like a continuous function that tracks these hard assignments to a high degree of accuracy. Such a continuous function for the the cluster membership can be defined as follows.

$$v_i^{(\mathbf{x}_s)} = \frac{\exp(-\frac{\rho}{D}||\mathbf{x}_s - \mathbf{m}_{i,\mathcal{X}}||^2)}{\sum_{i'=1}^{k_x} \exp(-\frac{\rho}{D}||\mathbf{x}_s - \mathbf{m}_{i',\mathcal{X}}||^2)} \tag{5}$$

where ρ is a user-settable parameter and D is the pointset diameter which depends on the data.

An analogous equation holds for $v_j^{(\mathbf{y}_t)}$.

We prepare a $k_x \times k_y$ contingency table to capture the relationships between entries in clusters across income dataset $\mathcal{X}$ and locations dataset $\mathcal{Y}$. To construct this contingency table, we simply iterate over every combination of data entities from $\mathcal{X}$ and $\mathcal{Y}$, determine whether they have a relationship, and suitably increment the appropriate entry in the contingency table:

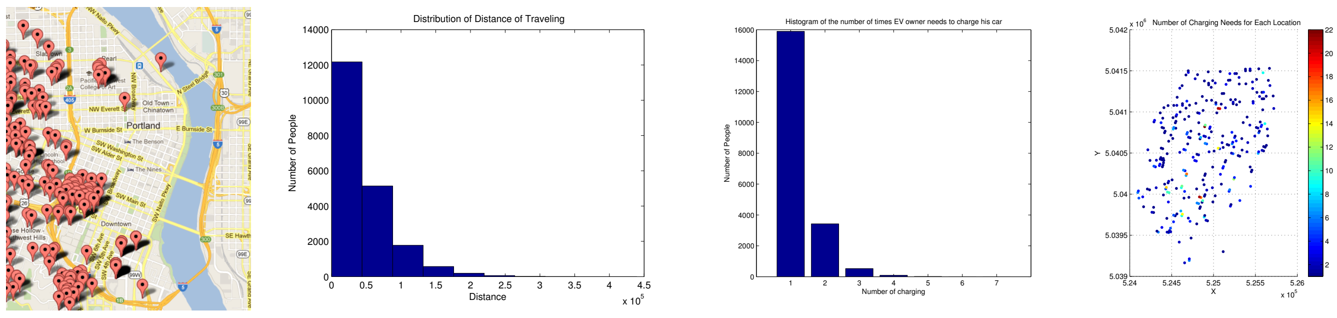
$$w_{ij} = \sum_{s=1}^{n_x} \sum_{t=1}^{n_y} B(s,t) v_i^{(\mathbf{x}_s)} v_j^{(\mathbf{y}_t)}. \tag{6}$$

We also define

$$w_{i.} = \sum_{j=1}^{k_y} w_{ij}, \qquad w_{.j} = \sum_{i=1}^{k_x} w_{ij},$$

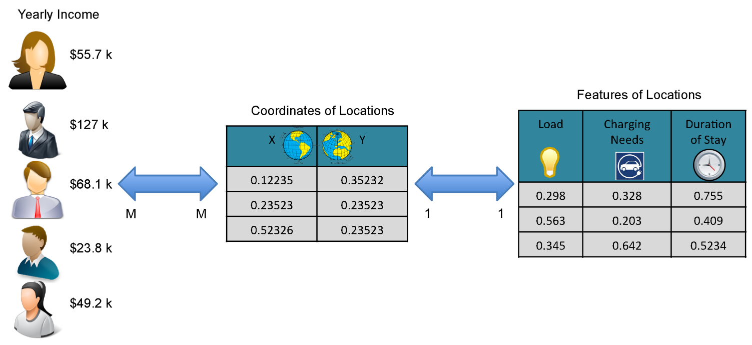
where $w_{i.}$ and $w_{.j}$ are the row-wise (income cluster-wise) and column-wise (locations cluster-wise) counts of the cells of the contingency table respectively.

We also define the row-wise random variables $\alpha_i, i = 1, \ldots, k_x$ and column-wise random variables $\beta_j, j = 1, \ldots, k_y$ with



(a) (b) (c) (d)

Figure 5: (a) EV household locations. (b) Distribution of distances people travel in their EVs. (c) Charging needs for EVs. (d) Number of charging needs (more than zero) per location.



Coordinates of Locations

X	Y
0.12235	0.35232
0.23523	0.23523
0.52326	0.23523

Features of Locations

Load	Charging Needs	Duration of Stay
0.298	0.328	0.755
0.563	0.203	0.409
0.345	0.642	0.5234

Figure 6: Coordinated clustering schema.

probability distributions as follows

$$p(\alpha_i = j) \quad = \quad p(C_{(y)} = j | C_{(x)} = i) = \frac{w_{ij}}{w_{i.}}. \qquad (7)$$

$$p(\beta_j = i) \quad = \quad p(C_{(x)} = i | C_{(y)} = j) = \frac{w_{ij}}{w_{.j}}. \qquad (8)$$

The row-wise distributions represent the conditional distributions of the clusters in dataset in $\mathcal{X}$ given the clusters in $\mathcal{Y}$; the column-wise distributions are also interpreted analogously.

After we construct the contingency table, we must evaluate it to see if it reflects a coordinated clustering. In coordinated clustering, we expect that the contingency table will be nonuniform. We can expect that the contingency table will be an identity matrix when $k_x = k_y$. To keep the formulation and the implementation generic for different number of clusters in two dataset, we need to optimize the variables (cluster prototypes) in such a way that the contingency table is far from its uniform case. For this purpose, we compare the income cluster (row-wise) and locations cluster (column-wise) distributions from the contingency table entries to the uniform distribution.

We use KL-divergences to define our unified objective function:

$$\mathcal{F} = \frac{1}{k_x}\sum_{i=1}^{k_x} D_{KL}\left(\alpha_i || U\left(\frac{1}{k_y}\right)\right) + \frac{1}{k_y}\sum_{j=1}^{k_y} D_{KL}\left(\beta_j || U\left(\frac{1}{k_x}\right)\right), \qquad (9)$$

where D_{KL} is the KL-divergence between two distributions and U indicates the uniform distribution over a row or a column.

Note that the row-wise distributions take values over the columns $1, \ldots, k_y$ and the column-wise distributions take values over the rows $1, \ldots, k_x$. Hence the reference distribution for row-wise variables is over the columns, and vice versa. Also, observe that the row-wise and column-wise KL-divergences are averaged to form $\mathcal{F}$. This is to mitigate the effect of lopsided contingency tables ($k_x \gg k_y$ or $k_y \gg k_x$) wherein it is possible to optimize $\mathcal{F}$ by focusing on the "longer" dimension without really ensuring that the other dimension's projections are close to uniform.

Maximizing $\mathcal{F}$ leads to rows (income clusters) and columns (locations clusters) in the contingency table that are far from the uniform distribution as required by the coordinated clusters. It is equivalent to minimizing $-\mathcal{F}$.

The coordinated clustering formulation presented thus far can have some degenerate solutions where large number of data points in both datasets are assigned to the same cluster leading to a huge overlap of relationships. To mitigate this, we add two more terms with the objective function.

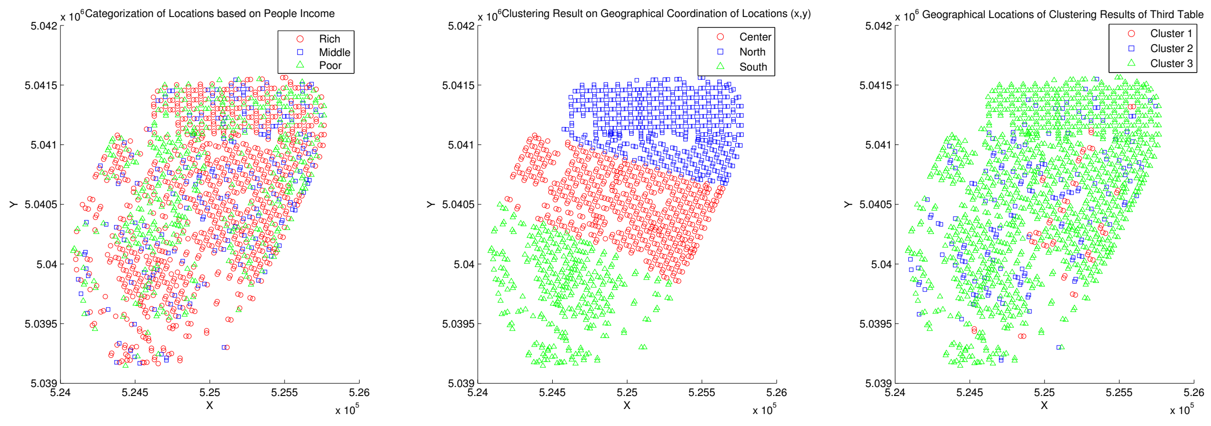
$$\mathcal{F}_{\mathcal{R}} = -\mathcal{F} + D_{KL}\left(p(\alpha) || U\left(\frac{1}{k_x}\right)\right) + D_{KL}\left(p(\beta) || U\left(\frac{1}{k_y}\right)\right). \qquad (10)$$

It should be noted that function $\mathcal{F}_{\mathcal{R}}$ is expected to be minimized. This is the reason why $-\mathcal{F}$ is used in the formula for $\mathcal{F}_{\mathcal{R}}$.

Finally, we describe how to integrate three datasets: income, location, and station properties. Let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ be these three datasets, respectively. There are two sets of relationships, existing between $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Y}$, $\mathcal{Z}$. The objective function for these three datasets and two sets of relationships is defined as follows.

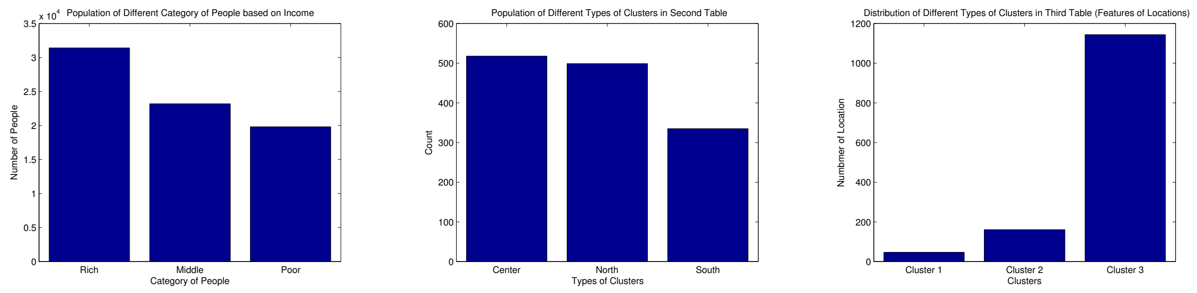
$$\mathcal{F}_{\mathcal{XYZ}} = \mathcal{F}_{\mathcal{R}}(\mathcal{X}, \mathcal{Y}) + \mathcal{F}_{\mathcal{R}}(\mathcal{Y}, \mathcal{Z}). \qquad (11)$$

Here $\mathcal{F}_{\mathcal{R}}(\mathcal{X}, \mathcal{Y})$ refers to the objective function described in Eq. 10 with the income dataset $\mathcal{X}$, and locations dataset $\mathcal{Y}$. $\mathcal{F}_{\mathcal{R}}(\mathcal{Y}, \mathcal{Z})$ refers to the same objective function but input datasets are locations $\mathcal{Y}$, and station property $\mathcal{Z}$. In all



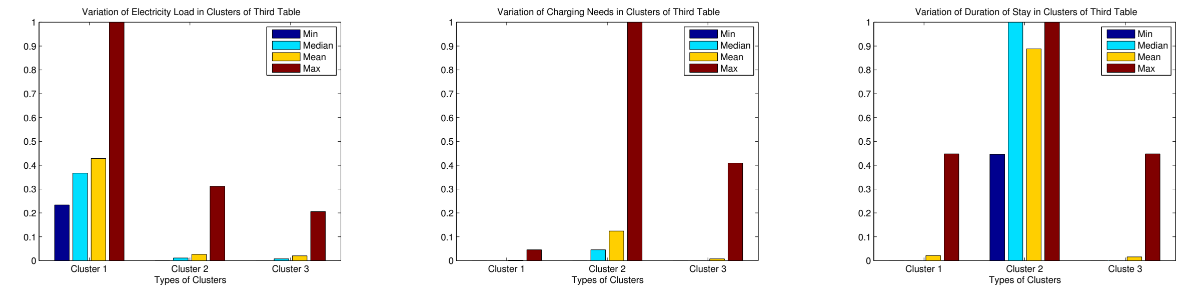
(a) (b) (c)

Figure 7: Results of coordinated clustering (3 clusters) when viewed through the attributes of each domain. (a) Clusters based on income. (b) Clusters based on geographical location. (c) Clusters based on EV charging station attributes.



(a) (b) (c)

Figure 8: Profiles of clusters obtained from coordinated clustering w.r.t. each of the three domains. (a) Income attributes. (b) Location attributes. (c) EV charging station attributions.



(a) (b) (c)

Figure 9: Detailed inspection of clusters for their suitability for locating EV charging stations. (a) Distribution of electricity loads. (b) Distribution of charging needs. (c) Distribution of duration of stay. An ideal cluster should have (low, high, high) values respectively, suggesting that cluster 2 is best suited.

our experiments, we minimize $\mathcal{F}_{\mathcal{X}\mathcal{Y}\mathcal{Z}}$ to apply coordinated clustering between income, locations, and station property datasets.

5. RESULTS

Figure 6 describes the coordinated clustering scenario involving: yearly income, a location's geographical coordinates, and the location's features.

We begin with some preliminary observations about our data. Figure 10 depicts the distribution of people based on their income, indicating that a significant number of people have high income, leading to a large number of EV users. We experimented with coordinated clustering settings involving many settings. Figure 7 depicts three clusters of locations based on each of the attribute sets in our schema. Note that because the clusters are mapped onto (x,y) geographical locations, locality is apparent only in Figure 7 (b).

Profiles of these clusters are described in detail in Figure 8. Of particular interest to us is the view from the perspective of EV attributes, i.e., Figure 8 (c). Details of these clusters are explored in greater detail in Table. 1. Ideal locations for charging stations for EVs must have a relatively low current electricity load (to accommodate the installation of charging infrastructure), high charging needs (population profiles), and high staying duration. As can be seen from Table. 1 cluster 2 fits these requirements. Greater insights into the three clusters from the viewpoint of these

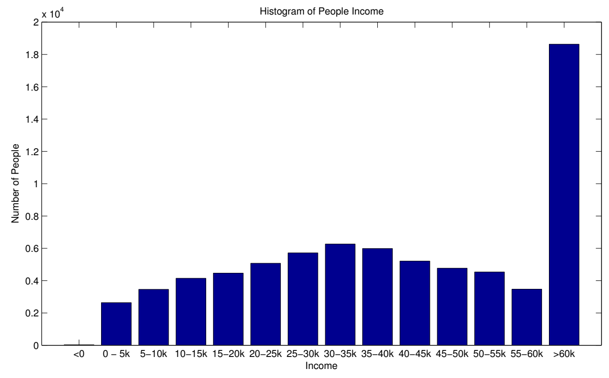


Figure 10: Distribution of income.

Table 1: Characteristics of Clusters in Third Table (Location's Features)

Cluster	Elec. Load	Charging Need	Stay Duration
1	High	Low	Low
2	Low	High	High
3	Low	Low	Low

three attributes is shown in Figure 9 supporting the choice of locations in cluster 2 as the right candidates for locating charging stations.

6. DISCUSSION

Electrical vehicles are only going to become more popular in the near future. We have demonstrated a systematic data mining methodology that can be used to identify locations for placing charging infrastructure as EV needs grow. The results presented here can be generalized to a temporal scenario where we accommodate a growing EV population and to design charging infrastructure to accommodate additional scenarios of smart grid usage and design.

The methodology presented in this paper only incorporates demand data from the electricity infrastructure and future work would incorporate information from the electricity supply side too. Information such loading level of electricity feeders and remaining excess capacity of feeders for EV charging stations can be integrated in presented methodology to improve the placement of EV charging stations. Moreover, this methodology can be used to identify locations of interest for deployment of stationary energy storages to more efficiently utilize existing electricity infrastructure rather than building new expensive transmission capacity to meet the demand of EV charging stations.

7. REFERENCES

[1] Synthetic Data Products for Societal Infrastructures and Proto-Populations: Data Set 1.0. Technical Report NDSSL-TR-06-006, Network Dynamics and Simulation Science Laboratory, Virginia Tech, Blacksburg, VA, 2006.

[2] S. Aman, Y. Simmhan, and V. K. Prasanna. Improving Energy Use Forecast for Campus Micro-grids using Indirect Indicators. In *IEEE Workshop on Domain Driven Data Mining*, 2011.

[3] C. Bailey-Kellogg, N. Ramakrishnan, and M. Marathe. Spatial data mining to support pandemic preparedness. *SIGKDD Explor. Newsl.*, 8(1):80–82, Jun 2006.

[4] K. S. Blake, R. L. Kellerson, and A. Simic. Measuring Overcrowding in Housing, September 2007.

[5] M. Ester, H. Kriegel, J. Sander, and X. Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD '96*, pages 226–231, 1996.

[6] GM-Volt. Chevrolet Volt Specifications. Last accessed May 16 2012 `http://Gm-volt.com/full-specification`.

[7] M. S. Hossain, S. Tadepalli, L. T. Watson, I. Davidson, R. F. Helm, and N. Ramakrishnan. Unifying Dependent Clustering and Disparate Clustering for Non-homogeneous Data. In *KDD '10*, pages 593–602, 2010.

[8] T. Kindberg, M. Chalmers, and E. Paulos. Urban Computing. *IEEE Pervasive Computing*, pages 18–20, July-September 2007.

[9] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xie. Discovering Spatio-Temporal Causal Interactions in Traffic Data Streams. In *KDD '11*, August 2011.

[10] N. Munro. Obama Hikes Subsidy to Wealthy Electric Car Buyers. Last accessed May 16 2012 `http://dailycaller.com/2012/02/13/obama-hikes-subsidy-to-wealthy-electric-car-buyers/`.

[11] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings. Putting the 'Smarts' into the Smart Grid: A Grand Challenge for Artificial Intelligence. *Communications of the ACM*, 55(4):86–97, April 2012.

[12] J. Rosswog and K. Ghose. Detecting and Tracking Coordinated Groups in Dense, Systematically Moving, Crowds. In *SDM '12*, pages 1–11, 2012.

[13] Simply Hired, Inc. Portland Jobs. Last accessed May 16 2012 `http://www.simplyhired.com/a/local-jobs/city/l-Portland,+OR`.

[14] R. Takahashi, T. Osogami, and T. Morimura. Large-Scale Nonparametric Estimation of Vehicle Travel Time Distributions. In *SDM '12*, pages 12–23, April 2012.

[15] The Engineering ToolBox. Common Area per Person in Buildings. Last accessed: May 16 2012 `http://www.engineeringtoolbox.com/number-persons-buildings-d_118.html`.

[16] The official U.S. Government Source for Fuel Economy Information. Fuel Economy. Last accessed May 16 2012 `http://www.fueleconomy.gov/feg/`,.

[17] N. Tishby, F. C. Pereira, and W. Bialek. The Information Bottleneck Method. In *37th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

[18] U.S. General Services Administration. Office Space Use Review, Current Practices and Emerging Trends, September 1997.

[19] J. Yuan, Y. Zheng, and et al. Driving with Knowledge from the Physical World. In *KDD '11*, 2011.

[20] J. Yuan, Y. Zheng, and X. Xie. Discovering Region of Different Functions in a City Using Human Mobility and POI. In *KDD '12*, 2012.

[21] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, and Y. Huang. T-Drive: Driving Directions Based on Taxi Trajectories. In *ACM SIGSPATIAL GIS 2010*, 2010.

[22] Y. Zheng, L. Zhang, X. Xie, and W. Ma. Mining Correlation between Locations Using Human Location History. In *ACM SIGSPATIAL GIS 2009*, 2009.

Avoiding the Crowds: Understanding Tube Station Congestion Patterns from Trip Data

Irina Ceapa
Dept. of Computer Science
University College London
London WC1E 6BT, UK
i.ceapa@cs.ucl.ac.uk

Chris Smith
Dept. of Computer Science
University College London
London WC1E 6BT, UK
c.smith@cs.ucl.ac.uk

Licia Capra
Dept. of Computer Science
University College London
London WC1E 6BT, UK
l.capra@cs.ucl.ac.uk

ABSTRACT

For people travelling using public transport, overcrowding is one of the major causes of discomfort. However, most Advanced Traveller Information Systems (ATIS) do not take crowdedness into account, suggesting routes either based on number of interchanges or overall travel time, regardless of how comfortable (in terms of crowdedness) the trip might be. Identifying times when public transport is overcrowded could help travellers change their travel patterns, by either travelling slightly earlier or later, or by travelling from/to a different but geographically close station. In this paper, we illustrate how historical automated fare collection systems data can be mined in order to reveal station crowding patterns. In particular, we study one such dataset of travel history on the London underground (known colloquially as the "Tube"). Our spatio-temporal analysis demonstrates that crowdedness is a highly regular phenomenon during the working week, with large spikes occurring in short time intervals. We then illustrate how crowding levels can be accurately predicted, even with simple techniques based on historic averages. These results demonstrate that information regarding crowding levels can be incorporated within ATIS, so as to provide travellers with more personalised travel plans.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms, Experimentation

Keywords

Mobility, Public Transport, Predictions

1. INTRODUCTION

With over 75% of the world's population expected to be living in cities by 2050, supporting citizens mobility within the urban environment is a priority for municipalities worldwide. In order to better manage mobility, public multimodal transit systems, coupled with integrated fare management, are being deployed in large cities, including London, UK. However, these transit systems are not able to absorb increasing loads and, especially at peak commuting hours when the system has to cope with a temporary surge in demand, overcrowding creates incredibly high levels of discomfort[1]. As a result, car ownership continues to grow worldwide, despite financial disincentives like costly congestion charges.

How can we overcome the problem of overcrowding in urban public transport systems? Transport operators often attempt to discourage peak-time travel by means of fare differentiation. For example, Transport for London (TfL) uses two price bands for morning-peak and post-morning-peak travel (before and after 9:30AM respectively), with the former being up to 50% more expensive than the latter. However, this has had no observable effect on travellers' journeys [9], with the tube reaching its peak load at around 8:15AM every weekday. This suggests that peoples' behaviour cannot be changed with imposed fare policies that do not take into consideration external constraints (e.g., having to be at work by 9:00AM in the morning). Instead we propose a different method for encouraging travellers to diversify their habits, by providing information about the likely crowding levels.

We analyse underground station usage in London, starting from Automated Fare Collection (AFC) systems data. AFC systems forgo traditional fare media, such as paper tickets or magnetic strip cards, in favour of alternatives such as RFID-based smart cards (e.g., London's Oyster Card, Seattle's Orca Card) or near-field communication on mobile phones (e.g., the Tokyo Metro System). These new payment systems create a digital record every time a trip is made, recording, for example, origin, destination, and travel time of every journey made. By analysing this data we discover that during the working week, crowdedness is a highly regular phenomenon, most probably as a direct result of the home-work commutes that follow rather fixed schedules. Even more interestingly, we show that spikes of crowdedness are concentrated in very short time periods, leaving the transport network under-utilised before and after such spikes. These findings suggest that, if made aware of such crowdedness patterns, travellers could opt to depart slightly earlier or later, thus avoiding congestion peaks while still

[1]`http://www.guardian.co.uk/money/2011/dec/30/worst-train-reading-london-paddington`

making it to work/home on time. We then build a number of crowdedness predictors and compare their accuracy while varying crowdedness thresholds, the size of the prediction window, and amount of data used for training such predictors. In all cases, we find accuracy is very high, even when using very simple techniques.

The remainder of this paper is structured as follows: Section 2 offers background information about the London Underground system and the AFC dataset we have used. In Section 3 we present the results of a spatio-temporal analysis of such data, illustrating the high regularity and short-lived spikes with which crowdedness manifests itself at different stations. Based on this finding, we move from analysis to prediction, and in Section 4 we demonstrate how station crowding levels can be accurately predicted from historical data, even with simple techniques based on historic aggregates. Section 5 positions this work with respect to similar ongoing efforts by the research community. Finally, in Section 6 we present our future work, where we plan to conduct choice experiments with a large population sample, to assess whether travellers' behaviour can be nudged simply by offering them accurate information about the crowdedness levels of the transport network.

2. THE OYSTER CARD DATA

The London Underground network is made up of 11 lines totalling 402 kilometres of track, serving near 270 stations. The transport network is separated into 9 fare zones, with Zone 1 encompassing central London and higher numbers representing regions further away from the centre of London, up to Zone 9, which contains a handful of outlier stations. The zoning system forms part of the fee structure for all rail travel in London, as well as approximating geographical distance from the centre of London. It is estimated that approximately 1,107 million passengers are carried by the tube each year. TfL introduced the Oyster Card in 2003 which now accounts for over 80% of all trips made within London's public transport system (as opposed to traditional paper tickets). Detailed information about each trip is captured when an Oyster card is used to enter or exit the tube network, producing an extensive source of data. For this study, we used a dataset containing all trips made with an Oyster card in the 31 days of March 2010. Each trip in our dataset is recorded as a tuple in the form:

$$\langle u, (o, d), t_o, t_d \rangle$$

Each tuple contains the unique (anonymous) user id (u), the trip (with origin o and destination d stations), the time stamp (to minute precision) t_o when u entered the origin station and the time t_d when u exited from the destination station. Over 70 million trips were recorded in March 2010. After cleaning the raw dataset (e.g., eliminating journeys with an end time that was before the start time and trips where the origin was the same as the destination), approximately 5 million trips ($\approx 8\%$) were removed, leaving us with over 64 million trips in total.

In order to analyse crowdedness at stations, we transformed this data from a per-user basis to a per-station basis as follow: each tuple $\langle u, (o, d), t_o, t_d \rangle$ was split in two, one recording station and time of origin $\langle o, t_o \rangle$ (*touch-ins*), and one recording station and time of exit from destination $\langle d, t_d \rangle$ (*touch-outs*). For each of the 270 stations, raw numbers were aggregated on 2-minute intervals, meaning that,

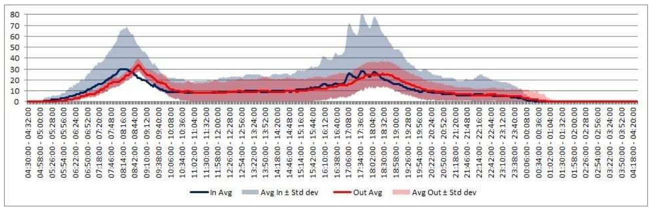


Figure 1: System-wide Weekday View

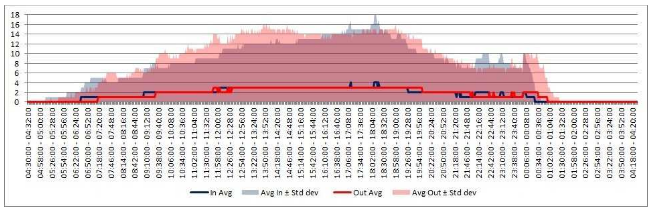


Figure 2: System-wide Weekend View

for each day, 720 observations (i.e., the number of touch-ins or touch-outs in the 2-minute interval) were recorded.

3. CROWDEDNESS ANALYSIS

Having pre-processed the data as described above, we next analyse the aggregate, system-wide usage patterns across all stations, to give a broad perspective of the usage of the system. We then demonstrate the variation in activity patterns exhibited by different by different stations by focusing on a select few. Based on these insights, we then move onto a spatio-temporal analyses of the system, where we use agglomerative hierarchical clustering (a form of unsupervised learning) to classify different patterns which emerge at different stations.

3.1 System-wide Temporal Analysis

To begin with, we look at system-wide aggregate data, only distinguishing between weekdays (Figure 1) and weekends (Figure 2). As expected, the weekday aggregated system behaviour is dominated by the two-pronged commuter pattern, with peaks in the morning at around 8:15AM, and in the afternoon at around 5:30PM, with approximately 35 touch-ins or touch-outs for every 2-minute interval. There are two interesting observations to make about the evening commute period. Firstly, the curve is not as steep as it was for the morning commute. This might be explained by the fact that, in the morning, people have to arrive at work by a certain time, whereas they are not under such an obligation on the return journey in the evening. Secondly, the touch-in curve during the evening peak time displays three smaller prongs, with peaks at roughly every 30 minutes, staring from about 5:10PM. This can also be motivated by people's work behaviour: as the standard working day is from 9:00AM to 5:00PM, some people leave work just after 5:00PM, while others leave later, but organise their schedules around specific time intervals, such as 30 minutes. Note also the large standard deviation (shaded area in the plot), which indicates that the data hides additional information or patterns which we explore next.

The weekend aggregate behaviour, illustrated in Figure 2, is not only much less regular with very high standard deviation, suggesting that crowding levels will not be as easy to predict, but also much less intense (y axis), which suggests that overcrowding is generally not a problem at weekends. For these reasons we thus disregard weekend data, and focus on weekdays only.

3.2 Spatio-Temporal Analysis

The system-wide analysis suggests that the weekday activity is very regular and therefore predictable. However, wide standard deviations suggest that a closer inspection may reveal hidden patterns in the data. For example, we would expect to see a difference between stations in residential areas versus stations in areas that mainly contain office buildings, since the flow of traffic will be in opposite directions during commuting periods. Furthermore, specific patterns should be observed in stations close to sporting locations, or stations close to cultural and entertainment locations, such as theatres. To investigate this we manually pick three stations representative of the different station usages mentioned: a station in a residential area, a station in a busy financial area, and a transport hub station, linking to National Rail.

Residential station – Finchley Central. Finchley Central serves a residential area and we expect this to be reflected in the commute pattern. Weekday activity is illustrated in Figure 3. The morning commute is dominated by touch-ins – people leaving the station and going to work in another area of the city. The maximum number of touch-ins is 60, at around 8:15AM. An important observation is the fact that the standard deviations for this period are relatively low, suggesting regular behaviour. During the evening commute, the situation is reversed, with touch-outs dominating the station usage pattern. As observed in the system-wide analysis, the evening commute is much less steeper than the morning commute. The standard deviations are also much wider, suggesting that the travel patterns in the evening are less regular.

Business station – Canary Wharf. Canary Wharf serves one of the two main financial centers of London, with over 93,000 people work in the area[2], which is mainly served by the Canary Wharf tube station. We expect the travel patterns to be the reverse of those encountered in stations serving residential areas. The day view for weekdays is illustrated in Figure 4. The morning activity consists almost entirely of touch-outs, with people arriving in the area. It increases steadily until it peaks at around 9:00AM, with 500 people leaving the station every 2 minutes. The standard deviation for the morning commute is quite high, reaching almost 100 touch-ins at the peak of the morning commute. As expected, the evening commute exhibits the opposite behaviour, with almost no touch-outs and an extremely regular (indicated by the very small standard deviation) three-pronged spike of touch-ins. The three prongs are spaced approximately 30 minutes apart.

Transport hub station – Waterloo. As our third and final case study, we consider Waterloo, a national railway terminus and busy tube interchange in central London. This is the busiest tube station in the entire TfL network, achieving more than 80 million touch-ins and touch-outs every year. Unlike the previously discussed station profiles, the weekday view, illustrated in Figure 5, shows much smaller

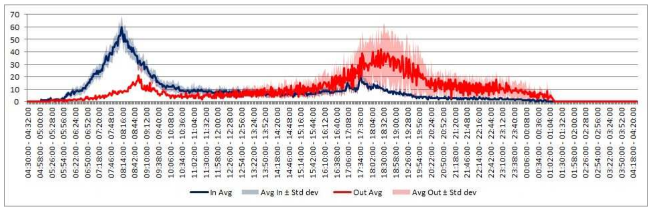


Figure 3: Finchley Central Weekday View

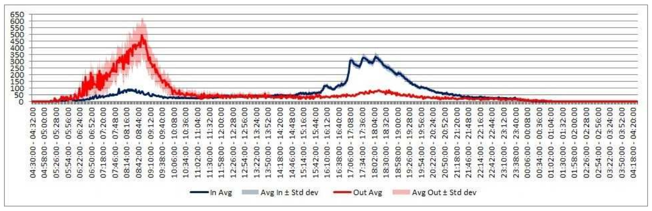


Figure 4: Canary Wharf Weekday View

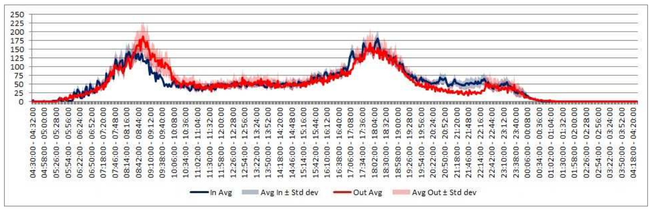


Figure 5: Waterloo Weekday View

differences between touch-ins and touch-outs. Indeed, the peaks for the touch-ins and touch-outs are quite similar in magnitude and only slightly out of phase.

The empirical analysis above supports the supposition that the dominant commute pattern hides individual station usage patterns. We thus proceeded with a more systematic approach, where we used an agglomerative hierarchical clustering technique to automatically group stations based on emerging usage patterns [18]. More precisely, to measure similarity between station usage patterns, we used a technique called dynamic time warping [19], a well-known algorithm used for computing differences between time series. The main strength of the algorithm is that it is extremely efficient as a time-series similarity measure which minimizes the effects of shifting and distortion in time by allowing elastic transformation of time series in order to detect similar shapes with different phases. In our analysis, we used an approximation to Dynamic Time Warping called FastDTW[3], that has linear time and space complexity, proved both theoretically and empirically [17]. The same study also proves that it shows a large improvement in accuracy over two other existing approximate DTW algorithms, Sakoe-Chuba Bands and Data Abstraction.

Using this similarity metric we constructed a hierarchy of clusters using an agglomerative approach. The algorithm works by starting with as many clusters as stations, then iteratively merging the two most similar clusters until a specified halting condition is met. The similarity between clusters

[2]`http://www.canarywharf.co.uk/aboutus/The-Estate/General-Information/`, retrieved 20 March 2012

[3]A Java implementation is provided under an open-source MIT licence at `http://code.google.com/p/fastdtw/`

	C1	C2	C3	C4	C5	C6
Num. of stations	1	23	8	27	3	198
Intra-cluster distance	0.000	0.198	0.008	0.003	0.134	0.011

Table 1: Clusters Information

	C1	C2	C3	C4	C5	C6
C1	0.0	2.316	1.167	1.992	1.205	2.115
C2		0.0	1.745	1.296	1.756	1.206
C3			0.0	1.495	1.395	1.531
C4				0.0	1.546	1.132
C5					0.0	0.550
C6						0.0

Table 2: Inter-Cluster Distances

is defined as the *average linkage*:

$$D_{AB} = \frac{1}{n_A n_B} \sum_{a \in A} \sum_{b \in B} FastDTW(a, b)$$

where clusters A and B have n_A and n_B members (stations) respectively.

The input to the clustering algorithm consists of a vector for each station, containing the difference between touch-ins and touch-outs at each 2-minute time interval. Thus, we characterise each station as either a sink (touch-outs $>$ touch-ins, i.e., more people arrive to this station than leave from this station) or a source (touch-outs $<$ touch-ins). As the size of stations varies significantly, in order to compare congestion levels, we also normalised the data, by dividing the difference between touch-outs and touch-ins by the largest of maximum number of touch-ins and touch-outs. The agglomerative clustering algorithm was then terminated when 6 clusters were produced; this number was found by visual inspection to provide the most informative array of station activity patterns.

A summary of the six resulting clusters is given in Table 1. The majority of the stations were included in cluster 6, while clusters 2 and 4 have between 20 and 30 stations. Clusters 3 and 5 are very small (8 and 3 stations, respectively), while cluster 1 only has 1 station (Wood Lane). The *intra*-cluster distances are very small, suggesting good cluster compactness. Table 2 shows the *inter*-cluster distance. Compared to the intra-cluster distances the inter-cluster distances are large, suggesting a good cluster separation.

The average day view for each of the six clusters is displayed in Figure 6. Sink-like behaviour will result in positive values, while source-like behaviour will display negative values. Cluster 1 is the most visually distinct of the six clusters, with high values and a morning peak which occurs significantly later than the dominating commute pattern. It acts as a powerful sink during the morning, with a peak at around 10:00AM. This behaviour is slowly attenuated until around 5:00PM, when the station becomes a source, although relatively small absolute values suggest that the average number of touch-outs is only slightly larger than the touch-ins. Cluster 2 exhibits opposite behaviour, acting as a source during the morning commute and as a sink during the evening one. Its peaks occur between 7:30AM and 8:00AM, and between 6:30PM and 7:00PM, respectively. Clusters 4 and 6 are very similar to cluster 2, and differ only in the amplitude of the absolute values and in the time when they reach the morning peak (between 8:00AM and 8:30AM). Clusters 3 and 5 are different from all other clusters in the sense that they change their behaviour during the morning commute and again during the evening one. Both start out as sources, until around 8:30AM, when they become sinks before stabilising at a neutral behaviour around mid-day. During the evening, they both return to source behaviour, although cluster 5 turns into a sink between 6:30PM and 9:30PM.

4. CROWDEDNESS PREDICTION

In the previous section we found that tube stations in London are dominated by the commute usage pattern (during weekdays), but with a significant number of stations exhibiting more distinct usage patterns. In all cases, we note that the usage patterns are highly regular during weekdays, so we expect to be able to predict usage levels and, consequently, overcrowding. In this section, we formulate the crowdedness prediction problem as a classification problem (Section 4.1), before reporting the results of an extensive evaluation that compares accuracy as obtained with different classification techniques (Section 4.2).

4.1 Methodology

The Dataset. Recall that we are working with weekdays only. In our dataset, we have 23 days of data (once we exclude weekends), which were divided into two sets: a training set (first 18 days), used to calibrate the parameters of the predictors, and a testing set (last 5 days), used to evaluate the performance of the predictors. While in the analysis section we worked with 2-minute intervals, for prediction we decided to work with 10-minute intervals. This is because an important follow-up of this study is to see whether information on congestion could nudge people to adapt their travel patterns, perhaps by leaving earlier or later. Using too fine-grained intervals (such as 2 minutes) would be too short for people to be able to appreciate the difference and adapt. On the other hand, too coarse-grained intervals (such as 30 minutes) would be impractical and have too much of an impact on peoples' schedules, resulting in people choosing not to adapt. We decided on using 10-minute intervals because they seem the most natural choice to nudge people into changing the time at which they travel. The training and testing data files for each station contain 144 observations per day.

The λ Threshold. To predict whether a station is crowded or not at a given point in time, we first need to define what it means for a station to be crowded. In the absence of official station capacities and congestion thresholds([14]) we define a proxy measure of crowding level and experiment with varying congestion thresholds. We use as a measure of crowding level the proportion of touch-ins (or touch-outs) at the station relative to the maximum number observed in the data. Thus, the maximum crowdedness of 1 indicates the station is at its peak level of crowdedness, and conversely, 0 indicates no touch-ins (or touch-outs) within the measurement interval. Identifying an appropriate congestion threshold, λ, is itself an interesting research question pertaining to individual travellers' preferences. Indeed, crowding tolerance

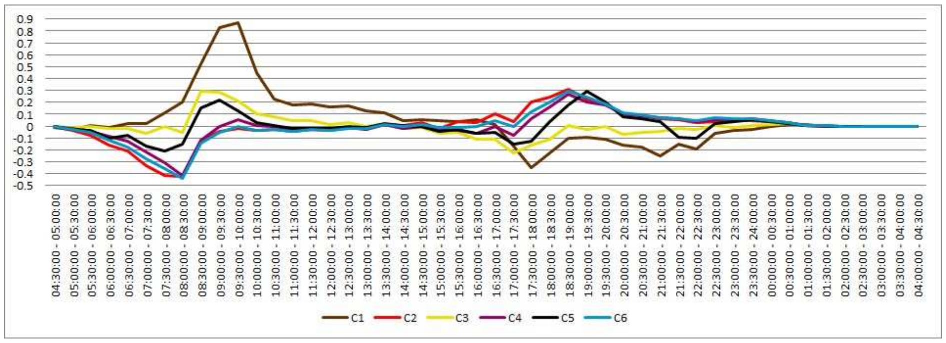


Figure 6: Average Day Views for the Six Clusters

varies greatly among individuals, and even for the same individual in different circumstances (imagine having to get to work in time for a meeting versus planning a trip to go for a walk in Hyde Park). Therefore, we decided to run and evaluate the predictors with different λ values. As we have until now, we will treat touch-ins and touch-outs separately (i.e., crowdedness entering vs leaving a station). The total number of overcrowded intervals for different values of λ is reported in Table 3. Based on these results, we decided to use 3 different values for λ: 0.5, 0.6 and 0.8. As we shall see, as the value of λ increases, and hence the classification problem becomes stricter, the accuracy of the results of our predictors will gradually decline.

λ	Overcrowded (in)	overcrowded (out)
0.4	20.92	25.76
0.5	15.05	17.67
0.6	10.78	12.50
0.7	7.51	8.63
0.8	4.96	5.43

Table 3: The effect of λ on the number of overcrowded intervals

Metrics. We measure results of our classification problem in terms of true positives, true negatives, false positives and false negatives. If both the predicted crowding level and the observed crowding level are greater than or equal to our crowding threshold, λ, the result is a true positive tp. If the predicted crowding level is greater than or equal to λ but the real level is not, the result is a false positive fp. Classifications for negative results (true negative tn and false negative fn) are determined in a similar way. The most undesirable result would be for us to incorrectly predict that an interval is not overcrowded. This would mean that travellers relying on our predictions to avoid congestion would be faced with an overcrowded station. In other words, our predictors should be evaluated primarily with how accurately they avoid false negatives. For this reason, we report results in this paper in terms of *sensitivity*, that is:

$$Sensitivity = \frac{tp}{tp + fn}$$

Sensitivity (also known as *true positive rate* or *recall*) is the proportion of correctly identified positive results from all positive results. For our specific problem, it indicates how often we are right that a station is overcrowded. For a more complete evaluation across other metrics, including precision ($tp/(tp + fp)$), accuracy ($(tp + tn)/(tp + fp + tn + fn)$), specificity ($tn/(tn + fp)$) and F-measure ($2 \cdot (Precision \cdot Sensitivity)/(Precision + Sensitivity)$), the interested reader may refer to [1].

Techniques. We ran three different prediction algorithms, each taking input t, the current time interval, and PW, the prediction window, which varies from 10 (i.e., predict crowding level in the next 10 minute interval) to 120 minutes (i.e., predict crowding level for a 10 minute interval that starts 110 minutes from t). Using the following notation:

- $Train[t]$ – the average crowding level at time interval t in the training set,
- $Test[t]$ – the observed crowding level at time interval t in the test set,
- $\overline{Train[t_1 - t_2]}$ – the average of the crowding levels in the training set during the time intervals from t_1 to t_2, inclusive,

we can formally define our three predictors as follows:

Historic Value - this predictor reports, for all time intervals and for all values of the prediction window, the corresponding value at the interval $t + PW$ in the training set. Such a baseline predictor is expected to perform well if and only if crowdedness is extremely regular (e.g., crowdedness on a Friday at 9:00AM-9:10AM is the same as the average of the recorded crowdedness levels on all weekday slots 9:00AM-9:10AM in the training set).

$$HistoricValue(t, PW) = Train[t + PW]$$

Historic Mean - this predictor also takes advantage of the history available in the training set. However, for a given time interval t and prediction window PW, it reports the average of all values in the training set between t and $t + PW$. This predictor is expected to perform well if crowdedness levels are regular (reliance on historic averages) but temporally shifted within a time window (e.g., crowdedness on a Friday at 9:00AM-9:10AM is the average of the recorded crowdedness levels on all weekday slots from 8:40AM to 9:10AM in the training set).

$$HistoricMean(t, PW) = \overline{Train[t - (t + PW)]}$$

Historic Trend - this technique attempts to improve the Historic Mean predictor by taking into consideration crowdedness level as currently recorded. This means that it can

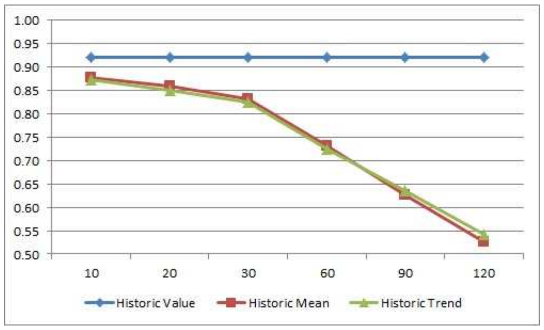


Figure 7: Sensitivity for $\lambda = 0.5$ (touch-ins)

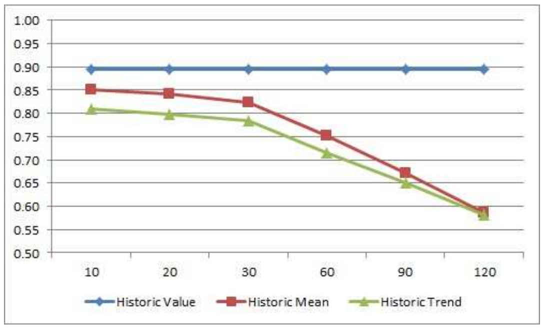


Figure 8: Sensitivity for $\lambda = 0.5$ (touch-outs)

take into consideration anomalies, such as outliers, in the training set that could influence the result. To achieve this, the Historic Trend implementation replaces the value at the current interval in the training set with the corresponding value in the testing set.

$$\begin{aligned} HistoricTrend(t, PW) &= \overline{Train[t-(t+PW)]} \\ &\quad -Train[t] + Test[t] \end{aligned}$$

We also experimented with two other predictors, one based on linear regression with ordinary least squares to estimate parameters, and one based on Kalman Filters. Neither techniques offered improvements over the three techniques above, so we leave them out of this paper in the interest of space. A full report of the findings is available at [1].

4.2 Results

Fixing $\lambda = 0.5$. We begin our evaluation by presenting the results obtained by the three predictors with a congestion threshold of $\lambda = 0.5$. Results in Figures 7 and 8 are aggregated and averaged over all stations in the system, separately for touch-ins and touch-outs. As shown, all predictors perform well when considering short prediction windows. As the prediction window increases from 30 minutes up to 2 hours away, the performance of Historic Mean and Historic Trend worsens, while leaving the performance of Historic Value unscathed; this confirms our hypothesis that crowdedness is highly regular and highly spiked too.

Impact of Varying λ. As previously discussed, the reasons for choosing a particular value for the crowdedness threshold, λ could be explored by an entire user study in its own right, focusing on what crowdedness means for different people or even for the same individual in different circumstances. We intend to do so in our future work. In this paper, we limit ourselves to studying the impact of the crowdedness threshold on the performance of the prediction techniques. In the interest of space, we show results for touch-ins only (results for touch-outs are only slightly worse and can be found in [1]).

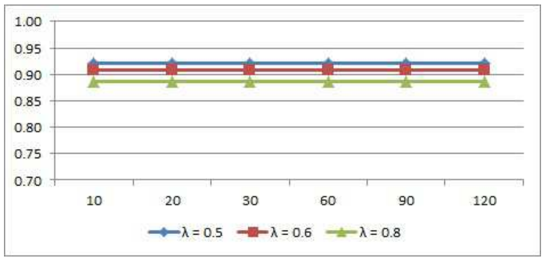


Figure 9: Effect of λ on Sensitivity of Historic Value (touch-ins)

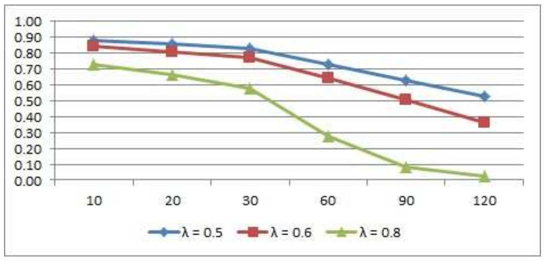


Figure 10: Effect of λ on Sensitivity of Historic Mean (touch-ins)

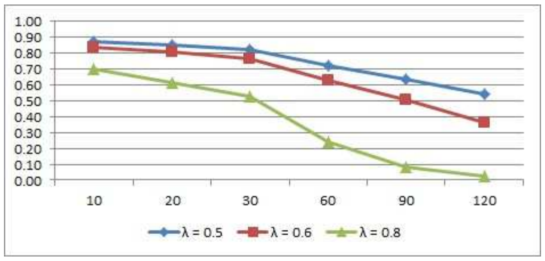


Figure 11: Effect of λ on Sensitivity of Historic Trend (touch-ins)

Figures 9, 10 and 11 illustrate the effect of the congestion threshold values on the performance of the Historic Value, Historic Mean and Historic Trend predictors respectively. As the crowdedness threshold increases, performance decreases (though only marginally so for Historic Value). This result should partly be interpreted in light of the data we have: recall from Table 3 that, as λ increases, the number of crowded time intervals (true positives) significantly decreases. As such, mis-classifying even a single interval results in a quick performance decrease (as measured by the sensitivity metric).

Impact of Training Data. As all our predictors rely on average historic data, an interesting question to answer is how much history is required in order to reach high sensitivity. In other words, what would be the effect of changing the ratio of the training and testing sets. We again evaluate the performance of our three predictors, that is, Historic

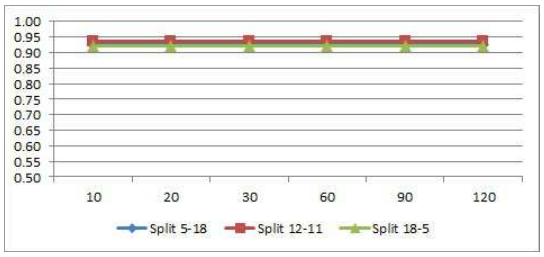


Figure 12: Effect of History Data on Sensitivity for Historic Value (touch-ins)

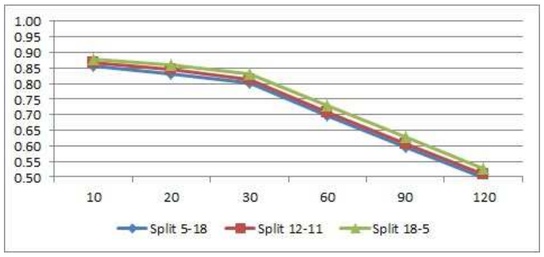


Figure 13: Effect of History Data on Sensitivity for Historic Mean (touch-ins)

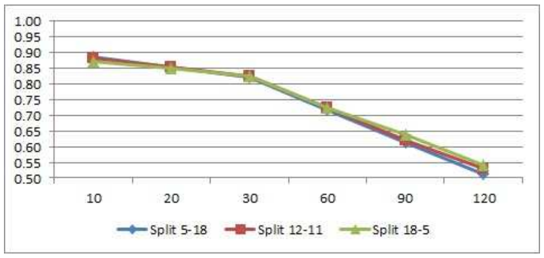


Figure 14: Effect of History Data on Sensitivity for Historic Trend (touch-ins)

Value, Historic Mean and Historic Trend, and results are shown in Figures 12, 13, and 14 respectively. Rather than using 18 days of training and 5 days of testing, we now test two splits: 12 days training - 11 days testing ($12-11$), and 5 days training - 18 days of testing ($5-18$). Once again, in the interest of space, we show results for touch-ins only.

No noticeable difference can be see for Historic Value, with consistently good sensitivity shown throughout. In the case of the Historic Mean predictor, the higher the amount of training data, the better the results, but once again the differences are not very pronounced (no more than 5% improvement). Finally, the Historic Trend predictor also shows better performance with more historical data, but the difference between the 3 splits is very small (about 1-2% on average).

Having only one month worth of data, one has to be cautious with the conclusions being made in terms of prediction accuracy. However, the above analysis would suggest that even short periods of training data (e.g., 2 weeks) are sufficient for the above predictors to learn station usage patterns, as these tend to be very regular. We can thus accurately predict crowdedness at stations, and consequently build more sophisticated journey planner tools that take this factor into account. It is worth noting that, in this work, we are interested in measuring and predicting crowdedness due to seasonal movements, and not exceptional situations due to, for example, unforeseen faults in the system (in other words, such anomalies are smoothed out by our predictors). Anomaly detection would be very useful to offer real-time information during journey execution, so that travellers can adapt on the go. This is however a complementary area of work which we do not investigate in this paper.

5. RELATED WORK

The increasingly wide availability of AFC data has lead to an explosion of research primarily focused on how such data can be used to evaluate and study the performance of the transportation system itself. For example, through demand modelling [3], service reliability measurements [2], average travel time estimation [2], and station transfer analysis [7]. A complimentary line of work has been looking at what the AFC data reveals, not about the transportation system, but about individual traveller behaviour instead: for example, by offering personalised travel time estimations [11], or by recommending what ticket type to purchase [10].

In this paper we start to combine the two: that is, we use AFC data to measure and estimate crowdedness levels of various stations in the transport network (system focus), but with the aim to feed this information back to the traveller, in terms of personalised classifications (station [not] crowded) based on individual tolerances to crowdedness (the λ value). In so doing, this paper adds to the growing body of work on smart cities and urban informatics [4], which is the study of human behaviours and urban infrastructures made possible by the increasingly digitised and networked city. For example, Gonzalez et al., 2008 [6] and Ratti et al., 2008 [15] use mobile phone-based location data to study human mobility patterns; Kostakos et al., 2006 [8] and Sadabadi et al., 2010 [16] rely on distributed Bluetooth receivers to track and predict travel speeds based on the Bluetooth MAC identifiers of passing devices. Most of the cited work, however, continues to focus on aggregate analysis only, rather than attempting to uncover opportunities for personalisation services.

One limitation of our work is that we study 'station' crowdedness only, while it would be useful for travellers to have also information about 'train' crowdedness. Oyster card data prevents us from building trajectory or sub-route-based models (e.g., [12], [5], [20]) since the actual route that a user undertakes between any origin and destination is unknown to us; in many cases, there are a wide variety of candidate routes. Implementing heuristics to derive route choices (for example, minimising the number of interchanges or minimising the hop-count on the tube graph) does not resolve cases where two routes seem equal on the applied heuristic (e.g., they both have one interchange) or when the heuristic derives results where travel time may increase (e.g., in cases where changing line would have reduced travel time). An area of future work will be to incorporate additional sensory information, such as images captured by CCTV cameras installed at station platforms, so to accurately quantify and subsequently predict train crowdedness. So far, CCTV cameras have only been used to detect (and not predict) 'platform' congestions [13]; this is useful for staff members to decide, for example, when to temporarily close stations in entry/exit to alleviate overcrowding.

6. CONCLUSION AND FUTURE WORK

In this paper, we have analysed anonymised AFC data collected for the tube network in London, UK, and have shown that crowding levels at stations is highly regular, and can thus be accurately predicted using simple predictors based on historic data averages. Not only did we show that crowdedness is a highly regular (and predictable) phenomenon during the working week, we also highlighted that big spikes concentrates in rather short time periods. The most important question for us now is what would people do with the information uncovered by this study? Would they change their behaviour, either by adjusting their travel times (by a small interval) to avoid the congestion peak, or by travelling to a different, yet close (geographically or on the same line) tube station? A user study is now required in order to determine what impact, if any, congestion information has on travel patterns. As part of this study, we also need to determine what the congestion threshold λ means for different people and in different situations, so that journey planner tools can leverage this information in computing personalised routes.

Acknowledgements. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7- SST-2008-RTD-1) under Grant Agreement n. 234239.

7. REFERENCES

[1] I. Ceapa. Predicting tube station congestion. Master's thesis, University College London, April 2012.

[2] J. Chan. Rail Transit OD Matrix Estimation and Journey Time Reliability Metrics Using Automated Fare Data. Master's thesis, MIT. June 2007.

[3] K. Chu and R. Chapleau. Enriching Archived Smart Card Transaction Data for Transit Demand Modeling. *Journal of the Transportation Research Board*, (2063), 2008.

[4] M. Foth. *Handbook of Research on Urban Informatics: The Practice and Promise of the Real-Time City*. 2009.

[5] J. Froehlich and J. Krumm. Route Prediction From Trip Observations. In *Intelligent Vehicle Initiative, SAE World Congress*, Detroit, Michigan, 2008.

[6] M. Gonzalez, C. Hidalgo, and A.-L. Barabasi. Understanding Individual Human Mobility Patterns. *Nature*, 453(7196):779–782, 2008.

[7] W. Jang. Travel Time and Transfer Analysis Using Transit Smart Card Data. *Journal of the Transportation Research Board*, (3859), 2010.

[8] V. Kostakos, T. Kindberg, and et al. Instrumenting the City: Developing Methods for Observing and Understanding the Digital Cityscape. In *In Ubicomp*. 2006.

[9] N. Lathia and L. Capra. How smart is your smartcard?: measuring travel behaviours, perceptions, and incentives. In *Ubicomp*, 2011.

[10] N. Lathia and L. Capra. Mining mobility data to minimise travellers' spending on public transport. In *17th ACM SIGKDD*, 2011.

[11] N. Lathia, J. Froehlich, and L. Capra. Mining public transport usage for personalised intelligent transport systems. In *ICDM*, 2010.

[12] H. V. Lint. Empirical Evaluation of New Robust Travel Time Estimation Algorithms. In *89th Annual Transport Research Board*, January 2010.

[13] B. Lo and S. Velastin. Automatic congestion detection system for underground platforms. In *Intl. Symposium on Intelligent Multimedia, Video and Speech Processing,*, May 2001.

[14] Freedom of Information request to TfL. Station capacity and safety policy. `http://www.whatdotheyknow.com/request/station_capacity_and_safety_poli`. answered 17 January 2012.

[15] C. Ratti, R. M. Pulselli, S. Williams, and D. Frenchman. Mobile Landscapes: Using Location Data From Cell Phones for Urban Analysis. *Environment and Planning B: Planning and Design*, 33(5):727–748, 2006.

[16] K. Sadabadi, M. Hamedia, and A. Haghani. Evaulting Moving Average Techniques in Short-Term Travel Time Prediction Using an AVI Data Set. In *89th Transportation Research Board Annual Meeting*.

[17] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, Oct. 2007.

[18] J. L. Schmidhammer. Agglomerative hierarchical clustering methods. `http://bus.utk.edu/stat/stat579`.

[19] P. Senin. Dynamic time warping algorithm review. *Information and Computer Science Department, University of Hawaii at Manoa, Honolulu, USA*, Dec. 2008.

[20] Y. Zheng and X. Zhou. Computing with Spatial Trajectories. Springer 2011.

Using Smart Card Data to Extract Passenger's Spatio-temporal Density and Train's Trajectory of MRT System

Lijun Sun
National University of Singapore
Future Cities Laboratory
Singapore-ETH Centre
sunlijun@nus.edu.sg

Der-Horng Lee
National University of Singapore
dhl@nus.edu.sg

Alex Erath
Future Cities Laboratory
Singapore-ETH Centre
erath@ivt.baug.ethz.ch

Xianfeng Huang
Future Cities Laboratory
Singapore-ETH Centre
huang@arch.ethz.ch

ABSTRACT

Rapid tranit systems are the most important public transportation service modes in many large cities around the world. Hence, its service reliability is of high importance for government and transit agencies. Despite taking all the necessary precautions, disruptions cannot be entirely prevented but what transit agencies can do is to prepare to respond to failure in a timely and effective manner. To this end, information about daily travel demand patterns are crucial to develop efficient failure response strategies. To the extent of urban computing, smart card data offers us the opportunity to investigate and understand the demand pattern of passengers and service level from transit operators.

In this present study, we present a methodology to analyze smart card data collected in Singapore, to describe dynamic demand characteristics of one case mass rapid transit (MRT) service. The smart card reader registers passengers when they enter and leave an MRT station. Between tapping in and out of MRT stations, passengers are either walking to and fro the platform as they alight and board on the trains or they are traveling in the train. To reveal the effective position of the passengers, a regression model based on the observations from the fastest passengers for each origin destination pair has been developed. By applying this model to all other observations, the model allows us to divide passengers in the MRT system into two groups, passengers on the trains and passengers waiting in the stations. The estimation model provides the spatio-temporal density of passengers. From the density plots, trains' trajectories can be identified and passengers can be assigned to single trains according to the estimated location.

Thus, with this model, the location of a certain train and the number of onboard passengers can be estimated, which can further enable transit agencies to improve their response to service disruptions. Since the respective final destination can also be derived from the data set, one can develop effective failure response scenarios such as the planning of contingency buses that bring passengers directly to their final destinations and thus relieves the bridging buses that are typically made available in such situations.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining, Sptial database and GIS*

General Terms

Experimentation

1. INTRODUCTION

Rapid transit systems are increasingly becoming the most important mode of public transportation in many large cities around the world owing to its faster velocity, higher reliability, and larger capacity, as compared with other transport modes. Understanding the demand characteristics of such systems is central to the public transport agencies and operators, so as to manage and improve their services. During a typical weekday, the demand of the bus and rapid transit systems have distinct spatio-temporal characteristics, which have been captured using smart card data, as shown in [7] and [8] respectively. Park et al. [8] studied the demand characteristics of different public transport modes, in particular the rapid transit system, based on the smart card data records in Seoul, South Korea.

The implementation of an automated fare collection system allows public transport agencies to collect large quantities of data, recording passengers activities with detailed time and space information. It has been recognized that there are large potential benefits of using this data to improve public transport planning and operation [9]. As a result, an increasing number of researchers have been using such data to analyze public transport systems characteristics and passenger behaviors. Bagchi and White [3] have demonstrated the feasibility of obtaining turnover rates, trip rates and the proportion of linked trips from smart card data, which can be further used to adjust such services. For some entry-only smart card systems, trip destination information is not recorded but needs to be imputed. Different methodologies haven been proposed to estimate the origin-destination pairs and alighting time [7, 4]. Jang [5] has studied the travel time and transfer activities in Seoul, South Korea using smart card data, which provides a comprehensive travel time map and basic understanding of transit services. By analyzing smart card data collected in Outaouis, Canada, Agard et al. [1] have identified different trip habits based on the predefined user types and variabilities of trips against time. Utsunomiya et al. [12] pointed out that demand pattern varies with day in week, therefore, different operation schedules should be provided for each day. Some researchers also focus on data processing methods and aim to get more meaningful information from smart card data. In [2], different types of analyses are conducted to support further planning purposes. Potential usage and challenges have also been highlighted.

Lee et al. [6] used smart card data from Singapore which contains detailed boarding/alighting activities to conduct an analysis on bus service reliability, including trajectories, occupancy of buses and in particular the headway distribution along the route since bus bunching occurs at times. Based on this approach, different operating strategies can be applied and tested in a simulation environment with passenger demand as inputs.

To data, information dedicated to identify passenger locations within a MRT system based on smart card data remains scant. Identifying trajectories and occupancies of trains is significant to transit agencies in order to improve the service level by designing timetable, adjusting velocity and increasing/decreasing dwell time at stations, however, these information is difficult to obtain from the operators' point of view. This is different from data generated from bus systems, as rapid transit data records do not feature any time information regarding when passengers board or alight from a train, which leads to difficulties in describing the trajectories and occupancies of trains. Fortunately, smart card data provides us the opportunity to extract this information. In this present study, smart card data is used to extract the spatio-temporal demand variation of the MRT(Mass Rapid Transit) system.

In the light of smart card records of passengers' tapping in and tapping out of the system, a model has been proposed to detect different travel time elements. This model can be regressed based on the assumption that the observations with the least duration between each origin and destination pair record over a given day travel through the system has no waiting time. The regressed parameters can then be employed to indicate the most probable location of every passenger, which further results in a realistic description of passengers' spatio-temporal density and trains' trajectories.

The present paper is organized as follows. Firstly, information regarding the featured smart card data and the MRT service which would be used as a case study, are introduced in the following section. In Section 3, a travel time regression model is proposed and estimated with data records from the passengers with minimum travel time. Section 4 presents the methodology to extract the spatio-temporal density of passengers and trajectories of trains based on the proposed travel time model. Finally, conclusions and an outlook on further research and applications are discussed in the concluding section.

2. RELATED WORK

Spatio-temporal distribution of traffic demand provides potential benefits to both transit operators and passengers. To investigate the spatio-temporal demand of urban road network, a number of efforts have been made with different kinds of urban dataset regarding urban road network. For example, taxies play an important role in urban road networks as probe vehicles, in particular, GPS-equipped taxies can generate large quantity of trajectory data. In most cases, urban taxi drivers are quite familiar with the road network they drive on everyday, especially the spatio-temporal distribution of traffic demand for each time slot, in other words, what they have is human sensed real-time traffic information, therefore, they have more intuitive intelligence and experience in finding fastest route for given origin, destination at certain time. Given this assumption, Yuan et al. [13] has designed a two-stage routing algorithm which leads to a smart driving system based on historical GPS trajectory data. With the help of this system, a fastest route can be generated given the departure time, origin and destination of passengers.

The fastest route generating problem is further studied with GPS trajectory data from urban taxies in [14]. Compared with [13], a Cloud-based computing system is developed to generate real-time fastest driving route in this paper. Regarding to estimating real time traffic information on urban road network, the methodology for estimating the travel time and spatio-temporal traffic density on road surfaces was introduced by using GPS trajectory data [13, 14].

The research conducted by Zheng et al. [15] is also based on GPS trajectories data from urban taxis, whereas it focused on the aspect of urban computing and application. In this study, the flaws in the existing urban planning of Beijing was detected with the mentioned dataset, the result of which provided comprehensive view on urban planning problems. This will help the city planners in decision making in conceive future plans to a larger extent.

In this paper, we focus on investigating the passengers' spatio-temporal density of subway system. To meet this objective, this study is conducted with the help of smart card data which record urban transit activities. Furthermore, the density data also provides us the opportunity to extract trains' trajectories, which are quite important in understanding the service level.

3. DATA PREPARATION

The smart card data used in this study was collected by a fare collection system, kindly provided by the Singapore Land Transport Authority (LTA). The smart card is Singapore's single largest contactless stored value smart card system and is mainly used for payments on public buses and Mass Rapid Transit (MRT) trains since April 2002. For this study, only records with both boarding and alighting stops being on the East West MRT line are selected since it is the most busiest rapid transit service in Singapore, as shown in Figure 1.

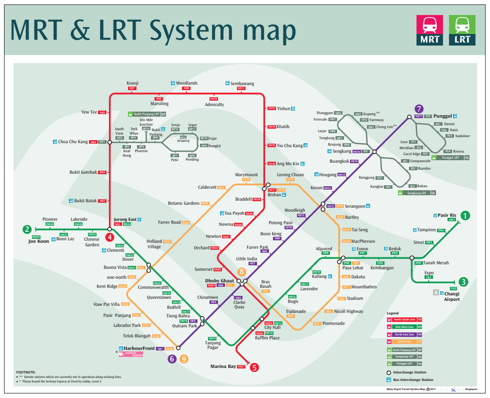


Figure 1: MRT and LRT system map in Singapore [10]

3.1 Smart Card Data

The smart card system was first introduced in Singapore with the aim of providing a convenient, automated fare collection public transportation system. Compared with other smart card data sources stated in [7] and [11], the most significant advantage of the smart card dataset is that it contains precise timing and location information for both boarding and alighting. Hence, transfer information can also be derived. This serves as a basis to generate information on load profile, spatio-temporal variation, and the waiting time of passengers.

In the smart card based fare collection system, the fare charge is calculated based on travel distance, trip mode and different passenger types, so any other information describing these three characteristics can be obtained from the data set.

This present study is conducted based on smart card records of one entire week in April, 2011 provided by the Land Transport Authority (LTA) of Singapore. To test the presented methodology of identifying spatio-temporal density and train trajectories, a one day sample is used.

3.2 Case Study - EW MRT Services

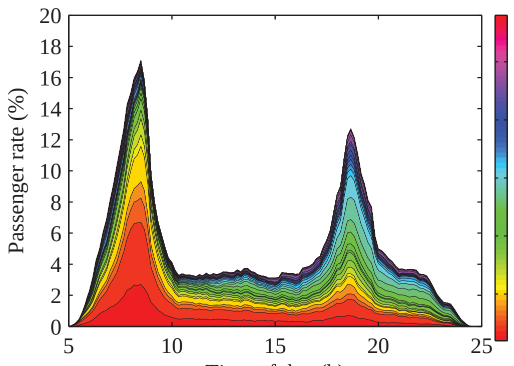
In this study, the East-West (EW) MRT service, which is known as the green line, is chosen to investigate the demand characteristics and test the proposed strategies in order to identify passengers' spatio-temporal density and trains' trajectories. This service has 29 stations moving in both directions. Figure 1 shows the general map of MRT and LRT (Light Rapid Transit) systems in Singapore. The case study examined is service that is on the green line, but the two stations on the extension line leading to Changi Airport are not included [10].

For this study, records of the time taken for passengers to tap in and tap out are used, along with boarding and alighting stations, and passenger types. Other information such as the locations of stations along the routes and characteristics of stations are obtained from supplementary information provided by LTA.

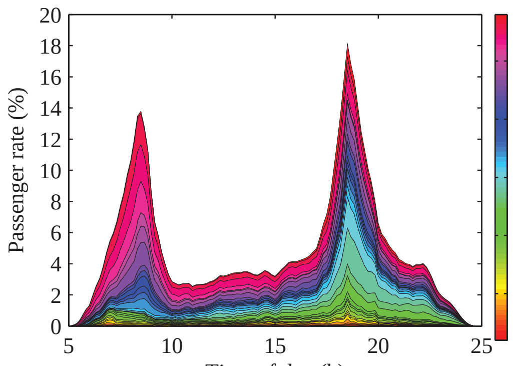
4. DEMAND PATTERN

In this section, travel demand patterns based on the data extracted from smart card data are described. With the help of the smart card data, it is possible to estimate how many passengers are in the MRT system at a given time t, for each station. To this end, records with tapping in time, known as $t_{in} < t$ and tapping out time $t_{out} \geq t$ are identified as passengers in the MRT system.

Figure 2 shows the number of passengers at each station during the course of the day, for train services in both directions, on a Monday in April 2011. It is observed that the demand for each direction has its own characteristics and both have significant morning and evening peaks.



(a) EW1-EW29



(b) EW29-EW1

Figure 2: Demand characteristics on EW line

Figure 2a and Figure 2b show a distinct morning peak at 8:30 am for both directions. Likewise, the evening peak can be observed at 6:30 pm. The different shapes of the two graphs indicate significant commuting in both directions with the morning commute direction from EW1 to EW29 being somewhat more distinctive.

It can be seen as well that in the morning peak, most of the demand originates from the first and last few stops along the line, while in the evening peak, most of the demand departs from the middle section of the line/service. In fact, this pattern maps effectively with land usage in Singapore. The predominant residential locations are located along the outskirts of Singapore and the work locations are centralized at the middle part of the city. During a typical weekday, most of the trips generated in the morning and evening peaks are commuters who travel to their work locations and back home respectively.

Such demand characteristics provide a basic understanding of an MRT service and the travel demand patterns of commuters over a typical weekday. The characteristics can be helpful to fine tune demand responsive train schedules or to define a more reliable strategy regarding the operation of MRT services.

5. TRAVEL TIME AND LOCATION OF PASSENGERS

5.1 Travel Time

Unlike the bus system, the MRT boarding and alighting times of individual passengers cannot be extracted directly. The tapping in and tapping out of their smart card takes place at the ticket gantry of the MRT station which is typically located on another floor of the station, typically one level below the entrance of station. Therefore, we cannot assign passengers to single trains directly. To make this information available to transit operators, a model describing passenger's movement between tapping in and tapping out would be required. In this study, such a model is proposed.

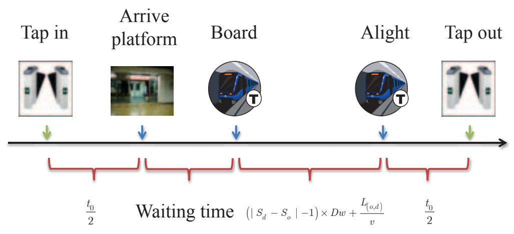


Figure 3: Activity chain of a typical subway trip

Figure 3 shows the typical activities for an MRT train ride. The trip begins with passenger tapping in at the ticket gantry. The passenger then makes his/her way to the platform, boards a train to travel on his/her journey, alights, and ends the journey by tapping out at another ticket gantry at his/her final destination. The waiting time can be calculated as the interval between passenger's arrival at the platform and upon boarding the train. In the smart card dataset, the exact time of tapping in and tapping out are recorded. The interval between these two activities is the total time which a passenger spends in the MRT system. However, as stated, the boarding time and alighting time can not be obtained directly because of the uncertainty in the length of waiting time. This however needs to be imputed.

From all the passengers having the same origin and destination pair, the passenger with the minimum travel time can be located. In this study, the travel velocity of trains is assumed to be constant, therefore, the passenger with the minimum travel time also has the minimum waiting time. Due to the large quantity of data used in this study, the waiting time of these fastest passengers are assumed to be zero, which means that the passengers can board a train immediately upon arriving at the platform.

The time interval between boarding and alighting is assumed to comprise two parts. First, the total running time between every two adjacent stations, and secondly the total dwell time at internal stations. From this, a general travel time model can be formulated as follows:

$$T - T_w = t_0 + (|S_d - S_o| - 1) \times Dw + \frac{L_{(o,d)}}{v} \qquad (1)$$

where T_w is the waiting time while t_0 comprises two parts, the time spent tapping into the station to the time when a passenger arrives at the platform, and the time spent alighting from the train to tapping out of the station. S_o and S_d are the index of the stations, thus $|S_d - S_o 1| - 1$ is the number of stations a passenger has passed, excluding the origin and destination stations. Dw is the average dwell time at each station, which is assumed to be a constant value for all stations without considering the boarding and alighting demand. $L_{(o,d)} = |D\,(d) - D\,(o)\,|$ is the distance from the origin station to the destination station and v is the velocity of the trains. Thus, in this proposed model, only T_w, Dw and v are unknown.

Based on the minimum travel times for each origin-destination pair, this travel time model can be estimated with the fastest passengers who generally have $T_w = 0$. In this regression analysis, the minimum travel time records with an origin same as destination are removed so that the size of the regression data for both directions is $N^2 - N = 841$. The results of this is a travel time model are shown in Figure 4 and Table 1.

Table 1: Regression result of travel time model

Parameters	Value	t stat	p value
$t_0(s)$	109.75	48.5787	0.0000
$Dw(s)$	65.76	61.4119	0.0000
$v(m/s)$	21.63	61.8186	0.0000
R^2		0.9981	

The regression results indicate that dwell time at stations is about $65s$ and that the travel velocity of trains is about $22m/s$, which are in accordance with effective values. Figure 4 shows the observed travel time for the fastest passengers from smart card dataset and the predicted travel time based on the proposed model.

5.2 Determining Location

Given the variability of the platform waiting time and availability of records of both the tapping in and tapping out of the smart card, it would be wise to use the latter for determining the passengers' location. Based on the previous travel time model, a passenger's location L at certain time t for two directions can be described by following Equation 2,

$$T_a - t = \begin{cases} \frac{D(d)-L}{v} + (|S_d - S_n| - 1) \times Dw + \frac{t_0}{2} & \text{if } D\,(d) \geq L \\ \frac{L-D(d)}{v} + (|S_d - S_n| - 1) \times Dw + \frac{t_0}{2} & \text{if } D\,(d) < L \end{cases} \qquad (2)$$

where T_a is the time when a passenger taps out of the station, and n, S_n are the number of stations which the passenger has journeyed through and the location of that station respectively. In Equation (2), t_0 is likewise divided equally into two parts, so only $\frac{t_0}{2}$ is considered for determining the

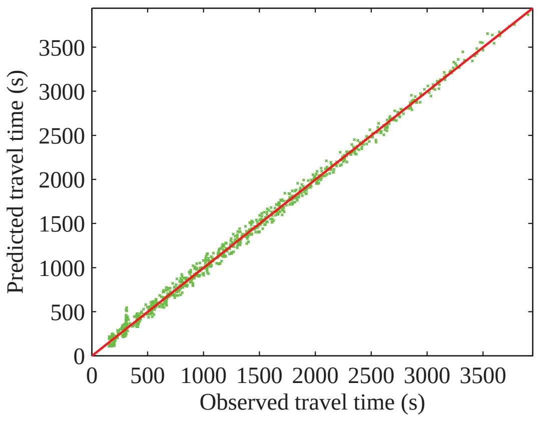


Figure 4: Predicted versus observed travel time for the fastest passengers

passenger's location based on the tapping out smart card record.

The temporary location of any passengers boarding at station k traveling in direction 1 (fulfilling $D(d) > D(o)$) can then be described by Equation 3.

$$L(k) = \left(t - T_a + (|S_d - k| - 1) \times Dw + \frac{t_0}{2}\right) \times v + D(d) \tag{3}$$

To distinguish between passengers waiting on a platform an travelling on a train, Equation (4) is proposed. For all the possible stations in $o, \cdots, k, \cdots, d$, if the first station k^* can be found which satisfies Equation (4), the permanent estimated location of the passenger is $L(k^*)$, where $P(k^*)$ is the location of station k^*.

$$L(k^*) - P(k^*) \geq 0 \tag{4}$$

For the opposite direction, the same method can be applied assuming that the passenger has just passed station k, then the temporary estimated location of this passenger is

$$L(k) = \left(T_a - t - (|S_d - k| - 1) \times Dw - \frac{t_0}{2}\right) \times v + D(d) \tag{5}$$

Then, for all the possible stations in $o, \cdots, k, \cdots, d$, if the first station k^* can be found which satisfies Equ (6), the estimated location of the passenger is $L(k^*)$, where $P(k^*)$ is the location of station k^*.

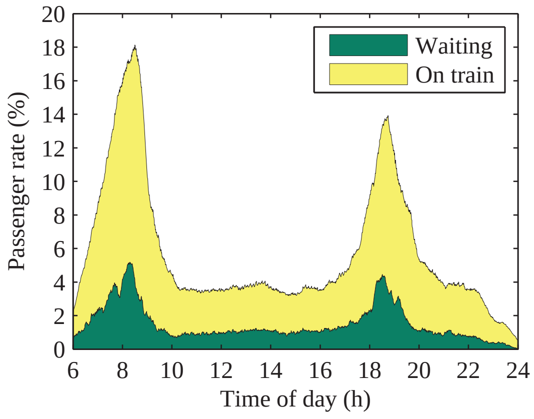
$$P(k^*) - L(k^*) \geq 0 \tag{6}$$

5.3 Waiting Passengers

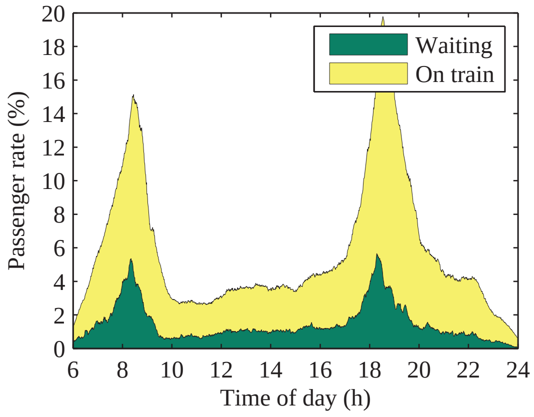
Based on the location model in Section 5.2, if for all the possible stations $o, \cdots, k, \cdots, d$, no station k^* satisfies Equation 4, it must be assumed that the passenger is in the MRT system but not on a train which, according to the travel time model, means that the passengers is either on the way to the platform or waiting there.

In other words, based on location estimation procedure the demand in the subway system can be contiously categorized into two groups: passengers who are on board the trains and passengers who are waiting for their trains.

Figure 5 shows the number of waiting passengers and on board the trains for both directions.



(a) EW1-EW29



(b) EW29-EW1

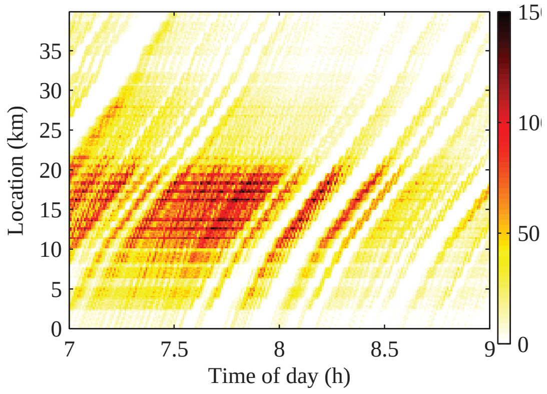
Figure 5: Demand of waiting and onboard passengers on EW line

Compared with Figure 2, Figure 5 provides time-volume relationship for both trains and platforms. This serves as a basis for the spatio-temporal density model presented in the next section. Furthermore, for any point in time and any station, the number of passengers located at the respective station can be derived, which is crucial in the event of train breakdowns or evacuations, in order to determine an effective response strategy.

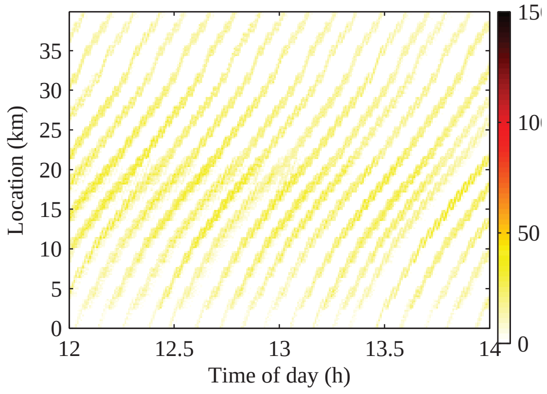
6. SPATIO-TEMPORAL DENSITY AND TRAJECTORIES

Section 5 describes the data processing to determine passengers's locations based on the proposed travel time model. In this section, the results of applying the described method using the smart card data records of a Monday in April 2011. The travel time model has been regressed with the travel time of the fastest passengers with the same data set. These two models make it possible to extract the spatio-temporal density of passengers onboard a train. Furthermore, the trajectories of trains can also be identified based on the spatio-temporal density figure.

Based on the location estimation model, for all the passengers onboard as shown in Figure 5, their locations at any time t can be determined. As a next step, the spatio-temporal density relationship can be constructed using the estimated number of passengers within a certain length interval. Figure 6 shows the spatio-temporal density of passengers, from $7am$ to $9am$ in the morning and $12pm$ to $2pm$ in the afternoon respectively for one direction, in intervals o f $100meters$ and $30seconds$. The colors indicate the passenger density who are onboard a train at a certain time and location. Intuitively, the location estimating model will work better for passengers with less travel time for each origin-destination pair, because there would be more variations for longer travel times for certain origin-destination pair, such as the cumulative difference in dwell time and velocity.



(a) EW1-EW29,7a.m.-9a.m.



(b) EW1-EW29,12p.m.-2p.m.

Figure 6: Spatio-temporal density of passengers on EW line($pax/100m$)

Despite some decentralization in the density figure due to non-observed variability, distinct spatio-temporal relationships can be detected as well. This applies especially to graph (a) which plots the density distribution for midday. However, for peak hours, as depicted in graph (b), the assignment to single trains does not appear to be so straightforward. Here, additional information such as effective train operations on a given day or at least the trains schedule would help to consolidate decentralized density observation to individual train trajectory. After such a procedure, passenger loadings could be determined for every train along the entire Ease-West line. Such information is ideally suited to serve as a basis for developing failure response strategies. Since origin and destination pairs for every observation are known, one could use such data in the event of service disruption for the route planning of contingency buses which would act as a substitute for the disrupted train service. Currently, such buses typically run along the interrupted track section, and serve as bridge services. However, depending on the demand patterns and spatial distribution of the final destination, other strategies such as direct buses to highly frequented destinations might provide better service for affected passengers. Because of the very limited time for replacement service planning after an incident, failure response plans need to be prepared in advance and be readily available in the event of an incident. Compared to a system based on real-time information, the retrospective nature of this study is therefore advantageous. However, given the changing demand patterns over a day, a series of different service dispatch plans would need to be prepared to suit the prevailing demand conditions at a given point in time optimally.

7. CONCLUSION

In this article, the demand characteristics of the case study of one MRT service was investigated using smart card data collected in Singapore, with the objective of identifying effective commuter loadings for every train service.

A travel time model has been proposed by reconstructing a typical MRT trip into segments. The model was regressed using the data collected from the fastest passengers for each origin-destination pair. Based on the regression results, a location estimation model was developed to distinguish between passengers travelling on trains and waiting on platforms.

The location estimation model was then applied to all MRT train passengers. Based on the resulting spatio-temporal density plot, it appears feasible to group observations together to individual train trajectories. Such information in turn, has great potential to improve current disruption response plans. Optimizing demand responsive failure response plans based on origin destination demand data, however, is a complex and extensive problem, especially since a multitude of such plans would need to be prepared given the demand fluctuations over a day. Further research would therefore need to focus on developing heuristics that allow one to generate failure response efficiently.

The proposed model can be improved by accounting for station specific access and egress times t_0 given the different layouts of MRT stations. In terms of applying this to real world scenarios, the scope of the analysis needs to be extended from a single line to the whole MRT network which would require consideration of transfers. We will conduct a further study in the future.

8. ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to the Singapore Land Transport Authority for supporting this research and providing all the required data. Furthermore, the financial support from the National Research Foundation of Singapore, the funding authority of the Future Cities Laboratory, is greatly appreciated.

References

[1] B. Agard, C. Morency, and M. Trépanier. Mining public transport user behaviour from smart card data. In *12th IFAC Symposium on Information Control Problems in Manufacturing - INCOM*, Eaint-Etienne, France, May, 17-19 2006.

[2] K. Alfred Chu and R. Chapleau. Enriching archived smart card transaction data for transit demand modeling. *Transportation Research Record: Journal of the Transportation Research Board*, (2063):63–72, 2008.

[3] M. Bagchi and P. White. The potential of public transport smart card data. *Transport Policy*, 12(5):464–474, 2005.

[4] J. Barry, R. Newhouser, A. Rahbee, and S. Sayeda. Origin and destination estimation in new york city with automated fare system data. *Transportation Research Record: Journal of the Transportation Research Board*, (1817):183–187, 2002.

[5] W. Jang. Travel time and transfer analysis using transit smart card data. *Transportation Research Record: Journal of the Transportation Research Board*, (2144): 142–149, 2010.

[6] D. Lee, L. Sun, and A. Erath. Study of bus service reliability in singapore using fare card data. In *12th Asia-Pacific Intelligent Transpotation Forum*, Kuala Lumpur, Malaysia, April 2012.

[7] M. Munizaga and C. Palma. Estimation of a disaggregate multimodal public transport originĺcdestination matrix from passive smartcard data from santiago, chile. *Transportation Research Part C: Emerging Technologies*, 24:9–18, 2012.

[8] J. Park, D. Kim, and Y. Lim. Use of smart card data to define public transit use in seoul, south korea. *Transportation Research Record: Journal of the Transportation Research Board*, (2063):3–9, 2008.

[9] M. Pelletier, M. Trépanier, and C. Morency. Smart card data use in public transit: A literature review. *Transportation Research Part C: Emerging Technologies*, 19 (4):557–568, 2011.

[10] SMRT. MRT & LRT system map, 2012. `http://www.smrt.com.sg/trains/images/All_CCL_CLE_System_Map_LRT_lines_big.jpg` [Online; accessed May 18, 2012].

[11] M. Trépanier, N. Tranchant, and R. Chapleau. Individual trip destination estimation in a transit smart card automated fare collection system. *Journal of Intelligent Transportation Systems*, 11(1):1–14, 2007.

[12] M. Utsunomiya, J. Attanucci, and N. Wilson. Potential uses of transit smart card registration and transaction data to improve transit planning. *Transportation Research Record: Journal of the Transportation Research Board*, (1971):119–126, 2006.

[13] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 99–108, San Jose, California, 2010.

[14] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–324, San Diego, California, USA, 2011.

[15] Y. Zheng, Y. Liu, J. Yuan, and X. Xie. Urban computing with taxicabs. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 89–98, Beijing, China, 2011.

Where to Wait for a Taxi?

Xudong Zheng
State Key Laboratory of Software Development Environment
Beihang University
Beijing, China
zhengxudong@nlsde.buaa.edu.cn

Xiao Liang
State Key Laboratory of Software Development Environment
Beihang University
Beijing, China
liangxiao@nlsde.buaa.edu.cn

Ke Xu[*]
State Key Laboratory of Software Development Environment
Beihang University
Beijing, China
kexu@nlsde.buaa.edu.cn

ABSTRACT
People often have the demand to decide where to wait for a taxi in order to save their time. In this paper, to address this problem, we employ the non-homogeneous Poisson process (NHPP) to model the behavior of vacant taxis. According to the statistics of the parking time of vacant taxis on the roads and the number of the vacant taxis leaving the roads in history, we can estimate the waiting time at different times on road segments. We also propose an approach to make recommendations for potential passengers on where to wait for a taxi based on our estimated waiting time. Then we evaluate our approach through the experiments on simulated passengers and actual trajectories of 12,000 taxis in Beijing. The results show that our estimation is relatively accurate and could be regarded as a reliable upper bound of the waiting time in probability. And our recommendation is a trade-off between the waiting time and walking distance, which would bring practical assistance to potential passengers. In addition, we develop a mobile application *TaxiWaiter* on Android OS to help the users wait for taxis based on our approach and historical data.

Categories and Subject Descriptors
H.2.8 [**Database Management**]: data mining, spatial databases and GIS

General Terms
Modeling, Statistics, Experimentation

Keywords
Vacant taxi, waiting time, poisson distribution

1. INTRODUCTION
Taxis play an important role in the transportation of cities. For example, Beijing has more than 60,000 taxis to provide services for about 2,000,000 passengers every day. However, there are still many people often annoyed with waiting for taxis. It is not only because of the imbalance between supply and demand, but also due to the lack of the vacant taxis information provided to passengers. For example, if a person does not know that there is another road nearby with more vacant taxis passing by, he/she would spend much more time on waiting for a taxi in current location. Experienced passengers could choose a better road to wait for taxis based on historical experiences. But more people have little knowledge about like how long they would take to wait for taxis here or where is better to wait for taxis, especially in some strange places for them, which may affect the their travels and schedules very much.

In this paper, we propose a method to estimate the waiting time for a vacant taxi at a given time and place, and then provide an approach to make recommendations for potential passengers on where to wait for a taxi. To make this estimation, we establish a model to describe the behavior of vacant taxis. Our model is based on the following observations:

- The higher proportion of time with vacant taxis parking beside the road, the more chances you can take a taxi immediately here. This situation usually occurs during the idle time around some popular places.
- The more vacant taxis leaving a road, the more chances you can take a taxi quickly. The waiting time is affected not only by the number of vacant taxis entering a road, but also by how many people want to take taxis here at that time. Because we do not have the data to directly show the demand for taxis, we think the number of vacant taxis leaving a road approximately reveals the remaining chance for a passenger to take a taxi here after the demand on the road is all met.

Motivated by the two observations above, we adopt the non-homo-geneous Poisson process (NHPP) [11] to model the events of vacant taxis' leaving and derive the probability distribution of the waiting time. Then we could perform estimations and recommendations based on the distribution. We also do some experiments to demonstrate that our approach is practicable and then develop a mobile application to help people wait for taxis.

Our study is built upon the GPS trajectories of taxis in Beijing, China. This data is collected from more than 12,000

[*]Corresponding author.

taxis, which account for about one-fifth of total ones in the city. We select the data between Oct. 2010 and Jan. 2011 to study. Each GPS record contains the identifier of a taxi, current position, timestamp, service status, and some other information. The data sampling interval of each taxi is about 60 seconds.

The major contributions of our work include:

- We employ the NHPP to model the behavior of vacant taxis, which could approximate the real situation well and have a simple form to derive the probability distribution of waiting time.
- We estimate the waiting time for vacant taxis at different time on road segments, analyze the confidence of our estimation, and design a recommender system for the people who want to take taxis.
- We conduct a lot of experiments to evaluate our approach on simulated passengers and actual trajectories of taxis. The results show that our estimation is relatively accurate and our recommendation would be helpful to potential passengers.
- We put our approach into practice by developing a mobile application which could help the user find an appropriate place to wait for a taxi.

The rest of this paper is organized as follows. In Section 2, we give an overview of the related work. Section 3 introduces our model used to estimate the waiting time for a vacant taxi. Section 4 describes the data processing of our work. In Section 5, we analyze the results of our estimated waiting time. Then, we discuss the recommendation for passengers in Section 6. Section 7 shows the experiments and evaluations on our approach. Section 8 introduces an application we developed to help people wait for taxis. Finally, we make a conclusion and propose some future work in Section 9.

2. RELATED WORK

2.1 Recommendations about Taxicabs

Recent years have witnessed the explosive research interest on taxi trajectories [1, 6, 7, 14, 18]. Moreover, many works have also been done to investigate the recommendations for taxi passengers or drivers [2, 5, 9, 13, 17].

Phithakkitnukoon et al. [9] study the prediction of vacant taxis number to provide the information for tourists or taxi service providers. They employ the method based on the naïve Bayesian classifier and obtain the prior probability distribution from the historical data. However their method divides the region of the city into one-kilometer square grids which are too rough to provide practical information for passengers. In addition, their data is only from the traces of 150 taxis in Lisbon, which might not be enough to reveal laws of vacant taxis for the reason of weak statistical significance.

Ge et al. [2] develop a recommender system for taxi drivers which has the ability in recommending a sequence of pick-up points or parking positions so as to maximize a taxi driver's profit. They estimate the probability of pick-up events for each candidate point, and then propose an algorithm to discover a route with minimal potential travel distance before having customer. Li et al. [5] study the strategies for taxi drivers as well. They use L1-Norm SVM to discover the most discriminative features to distinguish the performance of the taxis, and then extract some driving patterns to improve the performance of the taxis. However, all these studies do not concern about the recommendation for passengers.

Yuan et al. [17] propose an approach to make recommendations for both taxi drivers and passengers. They establish a probabilistic model to describe the probability to pick-up passengers, the duration before the next trip, and the distance of the next trip for a vacant taxi. Then they provide some different strategies for taxi drivers, each of which is based on the optimization of one aspect (probability of pick-up, cruising time, or profit). They further extend their work in [16]. Although their methods could also provide recommendations for passengers, their research mainly focuses on drivers. Comparing with them, our research stands on the view of passengers and pays more attentions to estimating the waiting time for vacant taxis.

Yang et al. [13] study the equilibrium of taxi market from the standpoint of economics. They use a bilateral searching and meeting function to characterize the search frictions between vacant taxis and unserved customers. They build a model to describe the relationship between the supply and demand for taxis, and analyze some influencing factors on customer waiting time. But in our study, because of lacking of explicit data for the demand of taxis, we actually estimate the gap between supply and demand through the number of vacant taxis. Moreover, the object of their study is an aggregate taxi market, but our target is to estimate the waiting time for a vacant taxi at a given time and position in a microscopic view.

2.2 Map Matching

Map matching is a main step of data preprocessing in our work. It refers to the process of mapping the GPS points to the road segments to recover a complete path of a trajectory. Quddus makes a survey of map matching algorithms in [10], including geometric, topological, probabilistic, and other advanced algorithms. He also discusses the performances and limitations of them. Lou et al. [8] propose a new algorithm for low-sampling-rate GPS data, which considers the temporal and spatial constraints on the trajectories, then constructs a weighted candidate graph to choose the most appropriate path. Yuan et al. further improve Lou's method in [15] later.

2.3 Non-homogeneous Poisson Process

Poisson process is a stochastic process that is often used to study the occurrence of events. It assumes the arriving rate of events λ is always stable, and has the Poisson distribution of counting and exponential distribution of inter-event time. Non-homogeneous Poisson process [11] is a Poisson process with a time-dependent arriving rate function $\lambda(t)$. This model is more flexible and appropriate to depict the human-related activities because these activities often vary over time and have periodicity. [3] studies the NHPP having cyclic behavior, and [4] introduces a method to estimate the $\lambda(t)$ in NHPP using a piecewise linear function.

3. MODEL

Here we propose a model to describe the waiting time for a vacant taxi, and derive the probability distribution of it. Then we estimate the waiting time using the expectation of the distribution. Finally, we analyze the confidence level of our estimation.

3.1 Motivation

The time to wait for a taxi reflects the availability of taxis on a road. Waiting for a taxi on a road could be divided into the two situations: 1) there are some vacant taxis just stopping beside the road, then you could take the taxi immediately; 2) there are no vacant taxis at hand, you should wait for the coming of next vacant taxi.

We could denote the probability of the first situation as p_{imm}, and the waiting time in the second situation as t_{next}. Then the random variable of actual waiting time t_{wait} could be represented as:

$$t_{wait} = (1 - p_{imm}) \cdot t_{next}$$

Then, we will discuss how to estimate p_{imm} and t_{next}.

3.2 Estimation of Waiting Time

Let's consider p_{imm} at first. We could approximate it by the proportion of time when there are some vacant taxis parking beside the road. We define the *parking time of vacant taxis*, i.e. t_{park}, as the duration with at least one vacant taxi parking on the road to wait for passengers. Therefore, $\hat{p}_{imm}$ for a road r during a timeslot T could be represented as:

$$\hat{p}^{r,T}_{imm} = \frac{t^{r,T}_{park}}{\Delta T}$$

Here ΔT denotes the span of timeslot T. By identifying of some appropriate stops of taxis, we can calculate $\hat{p}_{imm}$ for each road during each timeslot.

For t_{next}, intuitively, the number of vacant taxis leaving a road during a timeslot influences how long you probably spend on waiting for a taxi here. Because vacant taxis departing a road often means that they do not find any passengers on the road and then you have a great chance to take it if you are there. We denote the number of vacant taxis which have left a road as N_{vacant}, and define the *leaving frequency of vacant taxis*, i.e. λ, for a road segment r during a timeslot T as:

$$\lambda^{r,T} = \frac{N^{r,T}_{vacant}}{\Delta T}$$

Human-related activities vary over time, so do taxis. Therefore, we employ the NHPP to model the events of vacant taxis leaving roads. The rate parameter of NHPP is a time-dependent function $\lambda(t)$, and we further assume the rate function has a cycle of 24 hours. For simplicity, we adopt the piecewise linear function as the rate function of NHPP and regard each hour as a timeslot. This model assumes λ for a road is stable during a timeslot and the same timeslot in different days.

To validate our assumptions, we have done the KS-Tests for Poisson distribution on the data of each same timeslot in different days. To avoid the effect of the sparseness, we select the roads with enough data. We conduct the tests on the top 10,000 and top 30,000 roads for comparison[1]. Figure 1 shows the proportion of successful KS-Tests at 95% confidence level. We could see that the proportion for top 10,000 roads is larger than that for top 30,000 roads, and the proportion in the wee hours is rather small. These are because the data in the wee hours and unpopular roads are sparse and more fluctuant. Considering that our approach is mainly related to most of the passengers on popular roads in active time, so our hypotheses basically hold for most cases.

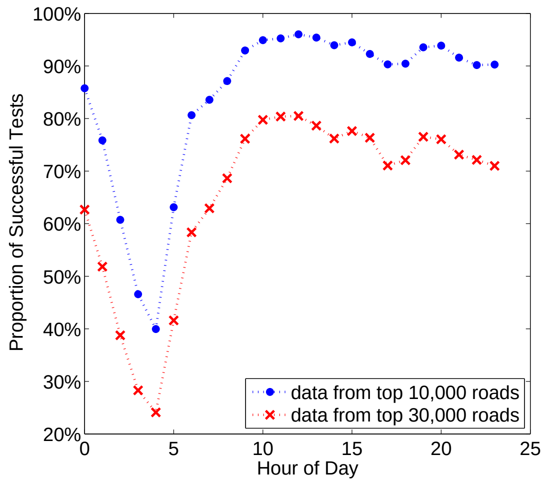


Figure 1: Proportion of successful KS-tests for the hypothesis of Poisson distribution

Under the Poisson hypothesis within a timeslot, we could derive the probability distribution of the waiting time for the next vacant taxi during the timeslot[2]. According to the Poisson process, the probability of the next event occurring within t is [12]:

$$\begin{aligned} P\{t_{next} \le t\} &= 1 - P\{t_{next} > t\} \\ &= 1 - P\{N(t) = 0\} \\ &= 1 - e^{-\lambda \cdot t} \end{aligned}$$

Here $N(t)$ represents the count of the events occurring within t, and $P\{N(t) = k\} = e^{-\lambda \cdot t} \cdot \frac{(\lambda \cdot t)^k}{k!}$. Then the probability density function of t_{next} is:

$$p(t) = \lambda \cdot e^{-\lambda \cdot t}$$

Thus, we can deduce the expectation of t_{next}:

$$\begin{aligned} E[t_{next}] &= \int_0^{\infty} t \cdot \lambda \cdot e^{-\lambda \cdot t} \cdot dt \\ &= \frac{1}{\lambda} \end{aligned}$$

Notice λ in our model denotes the *leaving frequency of vacant taxis*. Therefore with this conclusion, you could realize why the more vacant taxis leaving means the shorter waiting time for taxis. And we could regard the expectation as the estimation of t_{next}.

[1]We select the top roads by the number of pick-up events on it.

[2]For simplicity, we omit the superscript of parameters in the following derivation. It must be noted that the value of the parameter is different for various roads and timeslots.

Then we also need to estimate the λ. Here we employ the maximum likelihood estimation (MLE). If we observe the number of the vacant taxis leaving from a road at the same timeslot T for k days, we denote the count of the ith day is N_i, then the likelihood function is:

$$\mathcal{L}(\lambda) = \prod_{i=1}^{k} \frac{(\lambda \cdot \Delta T)^{N_i}}{N_i!} e^{-\lambda \cdot \Delta T}$$

Setting $\frac{d \ln(\mathcal{L}(\lambda))}{d\lambda} = 0$ and solving λ, we obtain the MLE:

$$\hat{\lambda} = \frac{\sum_{i=1}^{k} N_i}{k \cdot \Delta T} = \frac{\bar{N}}{\Delta T}$$

This conclusion means that we could estimate λ just by counting the leaving of vacant taxis in history.

As a consequence, we could estimate the actual waiting time for vacant taxis as:

$$\begin{aligned} \hat{t}_{wait} &= (1 - \hat{p}_{imm}) \cdot \hat{t}_{next} \\ &= (1 - \hat{p}_{imm}) \cdot \frac{1}{\hat{\lambda}} \end{aligned}$$

3.3 Confidence of Estimation

Now we will analyze the confidence level of our estimation. Let's consider the lower-sided confidence interval of t_{next}. We denote $1 - \alpha$ quantile of the distribution of t_{next} as $t_{next_{1-\alpha}}$, then we could get:

$$\int_0^{t_{next_{1-\alpha}}} t \cdot \lambda e^{-\lambda t} \cdot dt = 1 - \alpha$$

The quantile could be solved as:

$$t_{next_{1-\alpha}} = \frac{\ln(\alpha^{-1})}{\lambda}$$

This result shows that, we have $1 - \alpha$ confidence level of which the waiting time would be no longer than $\ln(\alpha^{-1})$ times of the $\hat{t}_{next}$ we estimated. If we set the upper bound of confidence interval equals to $\hat{t}_{next}$, namely:

$$\frac{\ln(\alpha^{-1})}{\lambda} = \frac{1}{\lambda}$$

We can get $\alpha = \frac{1}{e}$, which means the probability of the waiting time less than our estimation should be $1 - \frac{1}{e}$, which is about 63.21%. These conclusions imply that our estimation could be regarded as a reliable upper bound the possible waiting time in probability.

4. DATA PROCESSING

Our data processing starts with map matching. We have to map the trajectories of taxis to the roads and calculate the entering and leaving time of taxis to the roads. We employ the map matching algorithm proposed by Lou et al. [8]. In addition, we filter some trajectories which seem unusual, such as keeping vacant status too long (5 hours), or staying on the same road too long (2 hours).

Then, according to the model we have established, the processing is divided into two parts. The first part is the calculation of *parking time of vacant taxis*. The key step is to identify the stopping taxis that are waiting for passengers. We should eliminate the situations of waiting traffic lights or other purpose stops. We regard the taxi staying on a road with moderate duration (between 5 minutes and 2 hours) and rather low speed (less than 3.6km/h) as valid. Because too short time of stopping may be caused by traffic lights and too long time of stopping means no desire to take passengers or some unexpected situations.

The second part is to calculate the estimation of *leaving frequency of vacant taxis* λ for each road during each hour. Because the MLE of λ is $\frac{\bar{N}}{\Delta T}$, our task is just to count the number of vacant taxis leaving each road in each timeslot of one hour. And we also filter some outliers before making the average.

We process the trajectories happened during about three month, and calculate the averages $\bar{t}_{park}^{r,T}$ and $\bar{N}_{vacant}^{r,T}$, then the estimated waiting time could be represented as:

$$\begin{aligned} \hat{t}_{wait}^{r,T} &= (1 - \frac{\bar{t}_{park}^{r,T}}{\Delta T}) \cdot \frac{\Delta T}{\bar{N}_{vacant}^{r,T}} \\ &= \frac{\Delta T - \bar{t}_{park}^{r,T}}{\bar{N}_{vacant}^{r,T}} \end{aligned}$$

Here ΔT is the span of a timeslot, i.e. one hour.

However, our estimation of waiting time could not be applied to all roads, because there are some roads forbidding taxis to pick-up passengers. For these roads, there may be many taxis leaving from but few passengers getting on. Due to lack of the data indicating which road forbids the pick-up of passengers, we develop a method to detect these roads through analyzing the trajectories. We define the *pick-up rate*, denoted as θ^r for each road segment r:

$$\theta^r = \frac{\text{number of pick-up on the road segment } r}{\text{number of vacant taxis entering the road segment } r}$$

If there is a road with enough samples (more than 100 vacant taxis entering) and very low pick-up rate (less than 0.03), we will regard it as invalid to wait for taxis. For these roads, we do not make estimations of waiting time.

It is also worthy to be noted that our data is from the taxis which account for 1/5 of the total ones in Beijing. If we assume these 1/5 taxis are randomly distributed in the city, the waiting time would approximately be shortened to 1/5 of our estimation. We also could measure the actual scale factor by in-the-field study. But regardless of what the accurate factor is, the relative order of the waiting time we estimated will be basically kept under the random distribution assumption.

5. ANALYSIS OF THE RESULTS

We apply our approach to the data between Oct. 2010 and Dec. 2010, and then calculate the estimated waiting time for each road and timeslot. Because the data of some roads is very sparse, we only take the top 30,000 road segments with most frequent pick-up events into account[3]. And we also make the estimation of weekday and weekend separately.

Figure 2 gives an overview of the waiting time for vacant

[3]There is only fewer than 1 pick-up event per day in average on each of the remaining road segments.

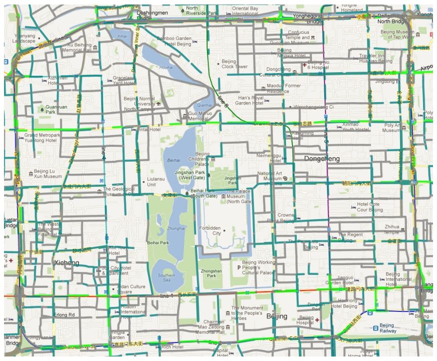


Figure 2: The map of taxi waiting time in Beijing.

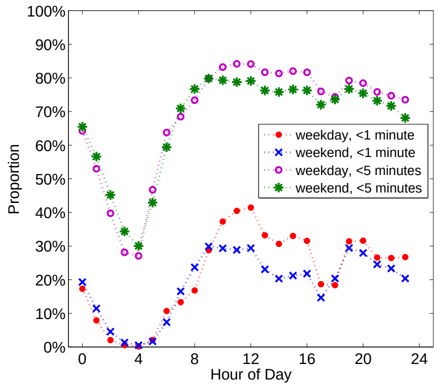


Figure 3: Proportion of roads with estimated waiting time less than 1 minute and 5 minutes.

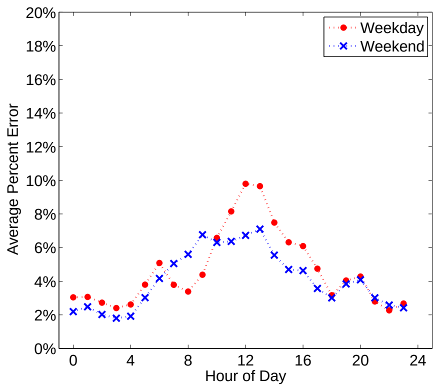


Figure 4: The percent error of the estimation of t_{park}.

taxis at 5 p.m. in a region of Beijing. The light green color denotes the estimated waiting time of less than 1 minute, the dark green color denotes the waiting time between 1 minutes and 5 minutes, and the gray color denotes the waiting time of larger than 5 minutes[4]. As shown in the map, the waiting time may be very different in some roads close to each other, so such information would help people find the appropriate position to wait for a taxi without walking too much.

Then let's analyze the varying of the waiting time in one day. Figure 3 shows the proportions of roads with estimated waiting time less than 1 minute and 5 minutes. From the figure we can see the proportion changes obviously with the time, which indicates the waiting time for taxis varies greatly during a day. The proportion of the roads with short waiting time is really low in the wee hours, because there are only a few taxis providing services. And the proportion reaches the top at noon, which implies that it would be easiest to take a taxi at that time. This is because that the demand of travel is relatively low but most taxis are in the service at noon. We also find that there are some differences between the weekday and weekend. The proportion of roads with short waiting time on weekend is not as high as on weekday in the daytime, the reason of which might be that there are more commercial and entertainment activities during that time on weekend.

6. RECOMMENDATION

With the knowledge we mined from the taxi trajectories, we could provide meaningful information to the people needing to take a taxi. With awareness of the possible waiting time on each road, people could make their schedule better, and avoid wasting time to wait for a taxi on a road with very long possible waiting time.

Furthermore, we also could provide a direct recommendation on where to take a taxi for the person who wants to take a taxi at somewhere and sometime. Considering the speed of pedestrian is slow, we limit the candidate roads to be recommended within a small distance. We denote the candidate roads set as:

$$R_{cand} = \{r : distance(P, r) < d_{max}\}$$

Here P is the position of the person now, r is a candidate road, and d_{max} is the maximal distance people want to walk. Then in the timeslot T, for each road $r \in R_{cand}$, we estimate the total time duration before taking a taxi as:

$$\begin{aligned}\hat{t}^{r,T}_{total} &= \hat{t}_{walk} + \hat{t}^{r,T}_{wait} \\ &= \frac{distance(P, r)}{\hat{v}} + \hat{t}^{r,T}_{wait}\end{aligned}$$

Here $\hat{v}$ is the common speed of the pedestrian. Then we choose the road r in candidates with minimal $\hat{t}^{r,T}_{total}$ as recommendation:

$$r_{best} = \arg\min_{r \in R_{cand}} \hat{t}^{r,T}_{total}$$

In addition, through adjusting the parameters such as d_{max} and $\hat{v}$, we could even control the preference for short waiting time or short walking distance in recommendation.

7. EXPERIMENTS AND EVALUATION

We have conducted comprehensive experiments to evaluate our model. Here we regard the data from Oct. 2010 to Dec. 2010 as the training, and choose three week between Jan. 5th 2011 and Jan. 25th 2011 for testing. We conduct our experiments on the top 30,000 road segments with most frequent pick-up events .

7.1 Validation of Statistics

We first validate two important statistical quantities in our model: *parking time of vacant taxis* t_{park} and *leaving frequency of vacant taxis* λ. Here we use percent error to evaluate the relative accuracy of our estimation. The percent error is defined as:

$$\text{percent error} = \frac{|\text{real value} - \text{estimate value}|}{\text{real value}} \times 100\%$$

Figure 4 shows the average percent error of t_{park}. The total average percent error is 4.52%. The reason of the small average error is that there are a large number of roads rarely having vacant taxis parking on. This result demonstrates the situations of vacant taxis parking beside the road have

[4]This waiting time has already been multiplied by the scale factor 1/5, the same below.

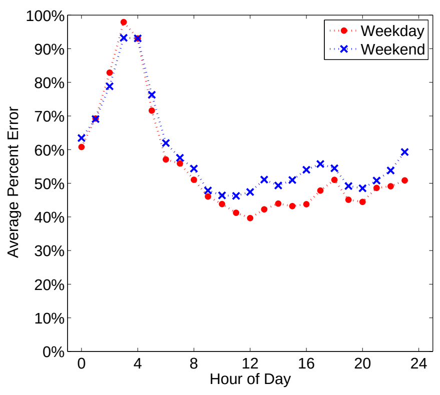


Figure 5: The percent error of the estimation of λ.

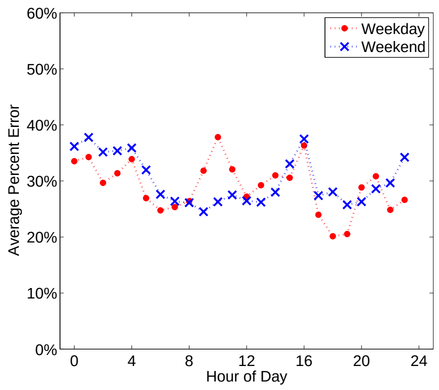


Figure 6: The percent error of the waiting time by simulation.

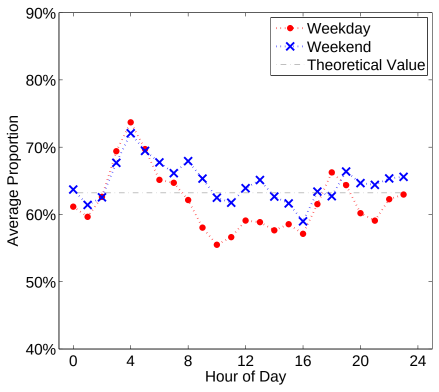


Figure 7: The proportion of tests with simulated waiting time less than the estimation.

little impact on our estimation, but we still keep it to remain the completeness of our model.

Figure 5 shows the average percent error of λ. The total average percent error is 56.12%. We could see the errors are rather larger in the wee hours due to the sparsity and fluctuation of data during that time. But for most time of a day, the percent errors are around 50%. And the weekday has smaller errors, which implies the higher regularity of human-related activities during weekdays.

7.2 Simulation

We also evaluate the estimated waiting time by simulation. For each road, we generate a passenger with a random timestamp, and then calculate how long the passenger should spend on waiting for the next vacant taxi just standing on this road according to the actual testing taxi trajectories. We repeat the simulation 100 times for each timeslot, and compare the average simulated waiting time to our estimated waiting time.

Figure 6 shows the average percent error of the waiting time on all roads at different time in simulation. The total average percent error is 29.37%. This result shows that the estimated waiting time for vacant taxis is relatively accurate and the error of our estimation is acceptable in general. Because the variance of the exponential distribution is relatively large when the λ is small, we could not avoid the errors on the roads with rare vacant taxis passing by.

Figure 7 shows the proportion of tests whose simulated waiting time is less than the estimation. The proportion in all tests is 62.73%. This is very close to the theoretical value 63.21% we have derived from our model (the straight line in the figure), which reflects that our model agrees well with reality from another side. The result also confirms that our estimation could be regarded as a reliable upper bound of the waiting time in probability.

We further evaluate our recommendation about where to wait for a taxi. We randomly generate passengers in a range of the city (no need to be on a road), as well as a timestamp. Then we choose the recommended road according the approach we proposed in section 6[5]. We compare our recommendation with three strategies: 1) *Best strategy* always chooses the road with the best t_{total} according to the actual testing data. This is a virtual strategy because it is based on posterior knowledge of taxi trajectories. It always leads to the best total waiting time, and we regard it as a baseline for comparison of time. 2) *Nearest strategy* always chooses the nearest road nearby, and then stops on it to wait for a vacant taxi. It is a common strategy in reality because people often are reluctant to walk too long. It always leads to the shortest distance to walk, and we also regard it as a baseline to compare walking distance. 3) *Random strategy* just randomly selects a road within the range. It is a possible strategy for the passenger who has no knowledge about the surroundings.

Figure 8 shows the difference of total waiting time compared with the *best strategy*. Our recommendation is obvious better than the *nearest strategy* and *random strategy* in terms of time. And our recommendation is relatively close to the *best strategy*, the total average difference is about 10 minutes. Figure 9 shows the difference of walking distance compared with the *nearest strategy*. Our recommendation is similar to the *best strategy* in terms of distance, and not much different from the *nearest strategy*. The total average difference between our recommendation and the *nearest strategy* is about 100 meters. These results show that, the recommendation made by our approach is a trade off between the waiting time and walking distance, which make the two aspects are all not much different from the best situations.

8. APPLICATION

Based on our approach and actual historical data, we develop a mobile application *TaxiWaiter* on Android OS, which could visualize the waiting time for vacant taxis on roads and also could provide a suggestion on where to wait for a taxi. Figure 10 demonstrates the user interface of the application. The roads are painted with different colors demonstrating the different waiting time on them, which could make the users intuitively understand the availability of taxis on these roads at some time. If the user clicks the *recommend* but-

[5]Here we set d_{max} to 1 km, and $\hat{v}$ to 3.6 km/h in the simulation.

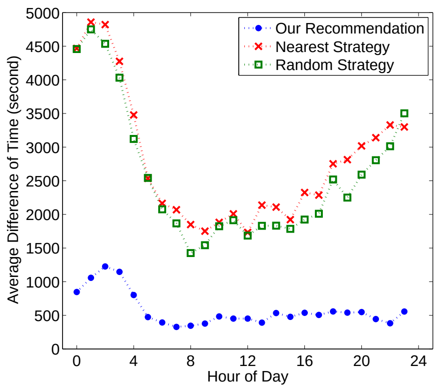


Figure 8: Difference of total waiting time compared with the best strategy.

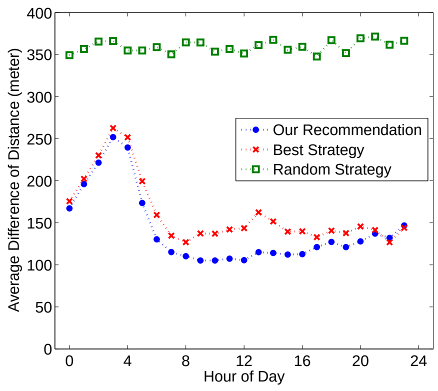


Figure 9: Difference of walking distance compared with the nearest strategy.

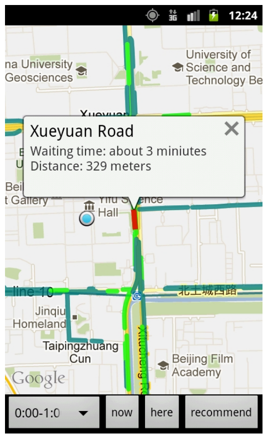


Figure 10: The mobile application *TaxiWaiter* we developed to provide taxi waiting information.

ton, the application could provide the recommended road for the user according to his/her current position (the blue point) and time. The red road in the figure is our recommendation, and the application also shows the possible waiting time and the distance to the recommended road. The user also could adjust the timeslot, walking speed and some other parameters in the application.

With *TaxiWaiter*, the user obtains more information about vacant taxis at different times on road segments and also could get a direct recommendation. These would help he/she make a better decision on where to wait for a taxi.

9. CONCLUSION AND FUTURE WORK

With the model of NHPP to describe the appearance of remaining vacant taxis, we could estimate the waiting time for the next vacant taxis on a road, and then make a recommendation on where to wait for a taxi for potential passengers. The model we established has a concise form and would lead to some meaningful conclusions in theory. The parameters in our model could be estimated from the statistics of historical data directly, which makes our approach practicable.

Through extensive experiments, we could validate that our estimations of taxi waiting time have relatively acceptable errors. The average percent error of the taxi waiting time is about 30%. The result of simulations also shows recommendations made by us would be helpful to the passengers. When the passengers following our recommendations, the total waiting time is just 10 minutes more than the *best strategy* in average, and the walking distance is only about 100 meters farther than the *nearest strategy*. This indicates that our recommendations balance the time and distance, and the two aspects are both close to the best situations.

However, there are still some limitations of our study, which would be the focuses of our future work:

- The rate function of piecewise linearity in NHPP is too simple for practical situations. We will try to use a continuous function to estimate the *leaving frequency of vacant taxis* λ which could change smoothly at any moment. This would make our model more flexible.
- The method we used to estimate the parameters such as $\lambda^{r,T}$ and $p_{imm}^{r,T}$ is just to make averaging on the historical data, and regard them as constants during different days. However, these parameters would also be changing slowly as time goes by. We consider weighing them differently based on period from that time to now, and then our method could adapt to the changes in the overall trend.
- The estimation and recommendation are not accurate enough. We will attempt to use or combine some other methods such as machine learning to improve the results, and we also plan to do some comparisons with different methods.
- We have not yet conducted in-the-field experiments to validate our approach. This type of experiments may be hard to do comprehensively. However, with the application *TaxiWaiter* we developed, we could receive feedbacks from users, which would give us a chance to validate and refine our approach.

10. ACKNOWLEDGMENTS

The research was supported by the National High Technology Research and Development Program of China (863 Program) (No. 2012AA011005), the fund of the State Key Laboratory of Software Development Environment (SKLSDE-2011ZX-02), and the Innovation Foundation of BUAA for PhD Graduates(YWF-12-RBYJ-036).

11. REFERENCES

[1] C. De Fabritiis, R. Ragona, and G. Valenti. Traffic estimation and prediction based on real time floating car data. In *Intelligent Transportation Systems*, pages 197–203, 2008.

[2] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani. An energy-efficient mobile recommender system. In *Proc. of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 899–908, 2010.

[3] S. Lee, J. Wilson, and M. Crawford. Modeling and simulation of a nonhomogeneous poisson process

having cyclic behavior. *Communications in Statistics-Simulation and Computation*, 20(2-3):777–809, 1991.

[4] L. Leemis. Estimating and simulating nonhomogeneous poisson processes. 2003.

[5] B. Li, D. Zhang, L. Sun, C. Chen, S. Li, G. Qi, and Q. Yang. Hunting or waiting? discovering passenger-finding strategies from a large-scale real-world taxi dataset. In *Pervasive Computing and Communications Workshops*, pages 63–68, 2011.

[6] X. Liang, X. Zheng, W. Lv, T. Zhu, and K. Xu. The scaling of human mobility by taxis is exponential. *Physica A: Statistical Mechanics and its Applications*, 2011.

[7] S. Liu, Y. Liu, L. Ni, J. Fan, and M. Li. Towards mobility-based clustering. In *Proc. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 919–928, 2010.

[8] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. In *Proc. of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 352–361, 2009.

[9] S. Phithakkitnukoon, M. Veloso, C. Bento, A. Biderman, and C. Ratti. Taxi-aware map: Identifying and predicting vacant taxis in the city. *Ambient Intelligence*, pages 86–95, 2010.

[10] M. Quddus, W. Ochieng, and R. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312–328, 2007.

[11] S. Ross. *Simulation*. Academic Press, 2006.

[12] S. Ross. *A first course in probability*. Prentice Hall, 2010.

[13] H. Yang and T. Yang. Equilibrium properties of taxi markets with search frictions. *Transportation Research Part B: Methodological*, 2011.

[14] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *Proc. of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 99–108, 2010.

[15] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G. Sun. An interactive-voting based map matching algorithm. In *Mobile Data Management (MDM)*, pages 43–52, 2010.

[16] J. Yuan, Y. Zheng, L. Zhang, and X. Xie. T-finder: A recommender system for finding passengers and vacant taxis. Submitted to TKDE, under second round review, 2013.

[17] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun. Where to find my next passenger? In *Proc. of the 13th ACM International Conference on Ubiquitous Computing*, 2011.

[18] D. Zhang, N. Li, Z. Zhou, C. Chen, L. Sun, and S. Li. ibat: detecting anomalous taxi trajectories from gps traces. In *Proc. of the 13th international conference on Ubiquitous computing*, pages 99–108, 2011.

Smarter Outlier Detection and Deeper Understanding of Large-Scale Taxi Trip Records: A Case Study of NYC

Jianting Zhang
Department of Computer Science
The City College of the City University of New York
New York, NY, 10031
jzhang@cs.ccny.cuny.edu

ABSTRACT

Outlier detection in large-scale taxi trip records has imposed significant technical challenges due to huge data volumes and complex semantics. In this paper, we report our preliminary work on detecting outliers from 166 millions taxi trips in the New York City (NYC) in 2009 through efficient spatial analysis and network analysis using a NAVTEQ street network with half a million edges. As a byproduct of large-scale shortest path computation in outlier detection, betweenness centralities of street network edges are computed and mapped. The techniques can be used to help better understand the connection strengths among different parts of NYC using the large-scale taxi trip records.

Categories and Subject Descriptors

H.2.8 [Database Management]: Database applications -Spatial databases and GIS

Keywords

Outlier Detection, Shortest Path, High-Performance, Taxi Trip

1. INTRODUCTION

Probe vehicle data, such as taxi trip records with GPS recorded pickup-up and drop-off locations/times and other related trip data, are valuable in understanding urban traffic and travel patterns. However, such data are often prone to various human, device and information system induced errors. Detecting outliers and subsequently cleaning up the data is a prerequisite for reliable analysis. Existing outlier detection and data clean approaches often use simple rules such as trip locations should be within city boundaries and geometric distances should be within certain thresholds [1]. When intermediate GPS readings are available, it is possible to examine the GPS traces and understand the identified outliers. However, in many cases, due to privacy concerns or data volume concerns or some other policy/management related reasons, the complete GPS traces are not available. Compared with using geometrical distances, shortest path distances can be more accurate and reliable in detecting outliers. However, shortest path computation in large street networks using traditional algorithms can incur unacceptable runtimes and make the computation impractical.

In this study, by integrating an open source package called MoNav [2] that implements an efficient Contraction Hierarchy (CH) based shortest path computation algorithm [3] and a spatial join algorithm to snap pickup and drop-off locations, we are able to compute the shortest paths of the 166 million taxi trip records in NYC in 2009 in less than two hours on a commodity desktop computer. The computed shortest paths not only facilitate outlier detection in a smarter way, but also allow generate edge betweenness centrality maps to understand the connection strengths among different areas of NYC from the taxi trip records. To the best of our knowledge, the approach has not been exploited previously in urban computing research. We believe the new approach can be used to understand the interactions between people and places deeper.

The rest of this paper is arranged as the following. Section 2 briefly introduces related works on outlier detections of taxi trip records, efficient shortest computation on road networks and betweenness centrality computation and applications. Section 3 presents the proposed method, including technique details on snapping taxi pickup and drop-off locations to street network and deriving betweenness centrality maps. Section 4 provides experiment results and performance evaluations. Finally Section 5 is the conclusions and future work directions.

2. BACKGROUND AND RELATED WORKS

The incompleteness, uncertainties and errors associated with taxi trip records have been well recognized in urban computing case studies. A commonly used practice to detect outliers and clean up data is to discard trips that are either too long or too short. For example, the research reported in [1] discarded trips with distances that are greater than 30 km (from one side of the city to the other) or shorter than 200 meters (which are considered unrealistic). A more general approach is to compute the distribution of a measurement (location, distance, duration et al) and then consider those that fall within the unusual ranges as outliers [4]. More domain-specific outlier detection approaches include marking taxi pickup and drop-off locations that are outside certain administrative regions or within regions of certain land use types (e.g. river/lake) as outliers through geospatial analysis such as point-in-polygon testing or nearest neighbor computing.

There are quite a few pioneering works in outlier detections from GPS traces. Some of them snap GPS trajectories to street networks and integrate street network topology into outlier detections [5] while others discretize GPS traces into grid cells and then use grid cells as the basic units for outlier detection [6, 7]. Unfortunately, these techniques can not be applied to our application as complete GPS traces are not available in our dataset and we are limited to use the two ends (Origin/Destination) of GPS trajectories.

In the context of processing GPS data, quite a few techniques have been developed to tackle uncertainty issues related to low GPS sampling rates by using road network geometry and topology [8, 9]. These techniques can be used to detect outliers as well by considering the non-matched GPS locations as outliers. In our study, in addition to applying the threshold-based, distribution-based and spatial analysis based outlier detection techniques, we propose to compute shortest paths and compare the recorded distances with computed shortest distances as a way to detect outliers. This approach can be considered as a network analysis based outlier detection technique. As detailed in the experimental section, in addition to be able to detect device-induced outliers, the approach is able to detect human induced outliers.

Shortest path computation is fundamental in both computer science research and virtually all application areas. A variety of shortest path algorithms, including the classic Dijkstra's algorithm and the A* algorithms, are available [10]. Several recent shortest path algorithms that are designed specifically for road networks have achieved signficant higher efficiencies than the generic ones. Among them, the Contraction Hierarchies (CH) algorithm developed by the Algorithm Engineering group at the KIT of Germany [3] has gained considerable popularities in practical applications. Two open source packages based on the CH algorithm, namely MoNav [3] and OSRM [11], are currently available. We found that the CH implementation in MoNav is easy to understand and easy to integrate in our application. As such, we have extracted the CH implementation module (termed as MoNav-CH hereafter) and used it in our study. As shown in the experiment section, we are able to compute over 25 million unique shortest paths in about 5,952 seconds using a single CPU core. The performance is more than three orders better than a commercial GIS software that we have tested. We also would like to note that a recent Graphics Processing Unit (GPU) based implementation of the CH algorithm called GPHAST [12] has achieved a throughput of less than 2 milliseconds in computing a single-source shortest path tree in an European road network with more than 18 million nodes and 42 million edges. The performance is about three orders faster than the Dijkstra's algorithm (without including path output overheads). We expect to gain higher performance in shortest path computation after incorporating the GPHAST algorithm. This has been planned for our U^{2}SOD-DB system [13] to provide a comprehensive and integrated data management and data mining platform for large-scale Origin-Destination (OD) data in ubiquitous urban sensing applications, including OD-based taxi trip records.

Node or edge based betweenness centrality has been widely used to study the relative importance of nodes and edges in a network [14]. The betweenness centrality of a node or an edge can be simply defined as the number of shortest paths that pass through the node or edge. As a by-product of computing shortest paths of all taxi trips, the betweenness centrality for all nodes and edges in a road network can be easily computed through simple aggregations on the edges or the nodes. Different from social network applications of betweenness centrality, in this study, we spatially map the numbers of computed shortest paths on all edges for deeper understanding of the connection strengths among different parts of the NYC in a visual manner. While our primary focus on this study has been on computing efficiency and data management sides, we note that betweenness centrality has been recently applied to study traffic flows [15, 16]. Compared with plotting aggregated pickup and drop-off locations separately [1] or examining an OD matrix [17], our approach provides details at the street segment level and relates origin and destination locations simultaneous. The standard map interfaces allows zooming, panning and querying the distributions of the betweenness centrality dynamically which are familiar to many users who use Internet mapping services (such as Google Map) on a daily basis.

3 METHOD AND DISCUSSIONS

Given a taxi trip dataset (T) with the following attributes: pickup location, pickup time, drop-off location, drop-off time and recorded distance, and, a street network (S) with N nodes and M edges, our method for outlier detection has three steps. The first step snaps both the pickup and drop-off locations to their nearest street segments. If a taxi trip has either a pickup location or drop-off location that can not be snapped to a street segment with a reasonable distance threshold (D_0), it is considered as a type I outlier. The node in the snapped segment that is closer to the pickup/drop-off locations is then assigned to the trip record as a pickup or drop-off node. The second step computes the unique combinations of the pickup and drop-off nodes of all trips. As demonstrated in the experiment section, computing shortest path of unique node combinations instead of individual taxi trips has reduced shortest path computation workload to nearly 1/7. The third step actually computes the shortest paths using the MoNav-CH module for each unique pickup-up and drop-off node pair. While the shortest paths are being computed, the node and edge centralities are updated for each of the nodes and edges along the computed shortest path. The computed shortest path distances are then compared with the recorded distances. If the computed distances are greater than a threshold D_1 and are W times longer than the recorded distances, then the records are marked as type II outliers.

We next briefly discuss some of the implementation details on snapping pickup and drop-off locations. It is obvious that the criteria on snapping point locations to street network segments can be viewed as a combination of window query and a nearest neighbor query which are supported in many Geographical Information Systems (GIS) and Spatial Databases (SDB). However, our experiments have shown that existing GIS and SDB software tools are not designed to handle location data at the scales of hundreds of million points and above. The performance is unacceptable using either ESRI ArcGIS or open source PostgreSQL. While we are aware of proper partitions and configurations can potentially improve performance but we are not sure whether the achievable performance can meet the level that is required by our application. As such, we leverage the P2N-D spatial join module in our U^{2}SOD-DB system for this purpose. While we refer to [13] for more details on this, we would like to note that by adopting a column-oriented, in-memory system with massively data parallel GPU hardware accelerations, the performance of the first step of the proposed method has been significantly improved.

We also note that using shortest path as a proxy of actual trip paths is approximate in nature, especially when computing shortest paths purely based on distances. As many previous studies have shown that, experienced taxi drivers usually take quite a few factors into consideration in choosing travel paths and it is not surprising that some researchers suggest use experiences of taxi drivers for online navigation purposes [18, 19]. Nevertheless, we believe that when the recorded trip distances deviate significantly from the computed shortest distances by setting D_1 and W properly, it can be an effective filtering mechanism to identify true outliers. In addition, the computed betweenness centrality measurements are aggregated values where the inaccuracies and uncertainties have a good chance to be canceled by each other and can be used as a visualization tool to understand the overall travel and traffic patterns in metropolitan areas such as NYC.

4 EXPERIMETNS AND RESULTS

4.1 Data and Experimental Setting

We use approximately 166 million taxi trips in NYC in 2009 in this study. While there are more than 20 attributes from the original database dump, we use only the five attributes as listed in previous section and we refer to [13] for more details regarding to the data management aspects of both taxi trip data and infrastructure data in U^{2}SOD-DB. For illustration purposes, we have plotted a distribution map of aggregated numbers of taxi pickup locations using a spatial resolution of 16 feet (~5 meters) and a grid dimension of 8192*8192. As shown in the right part of Fig. 1, clearly, the majority of taxi pickup locations are in the midtown and downtown Manhattan area. When the taxi pickup locations in the area are aggregated at 0.5 feet resolution, the effect of road network becomes apparent as shown in the left part of Fig. 1. The similar trends have been observed for the drop-off locations.

The street network in the NYC area provided by NAVTEQ through its University Program not only includes the five boroughs in NYC but also includes three neighboring counties, i.e., Westchester, Nassau and Suffolk. The expanded spatial coverage allows examine taxi trips with either pickup locations or drop-off locations outside the NYC boundary. Significant preprocessing work has been carried to bridge between the map data format used by NAVTEQ and the graph data format used by MoNav. While we could have extracted both nodes and edges in the ***streets*** shapefile according to NAVSTREETS data format documentation, we have decided to incorporate intermediate nodes that are separately stored in ***zlevels*** for the purpose of snapping taxi locations to street segments more accurately. The resulting network has 434,521 nodes and 501,756 nodes. As the network is very sparse, the CH-based algorithm is especially suitable as many natural shortcuts exist.

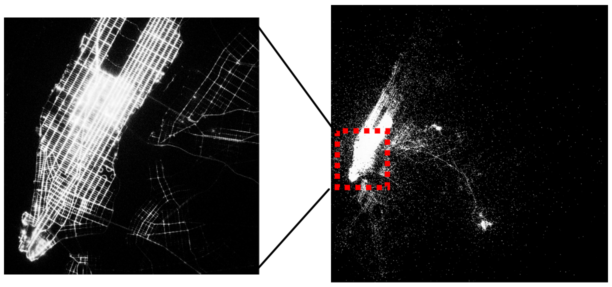


Fig. 1 Aggregated Pickup Locations in NYC in 2009 with Details in Manhattan

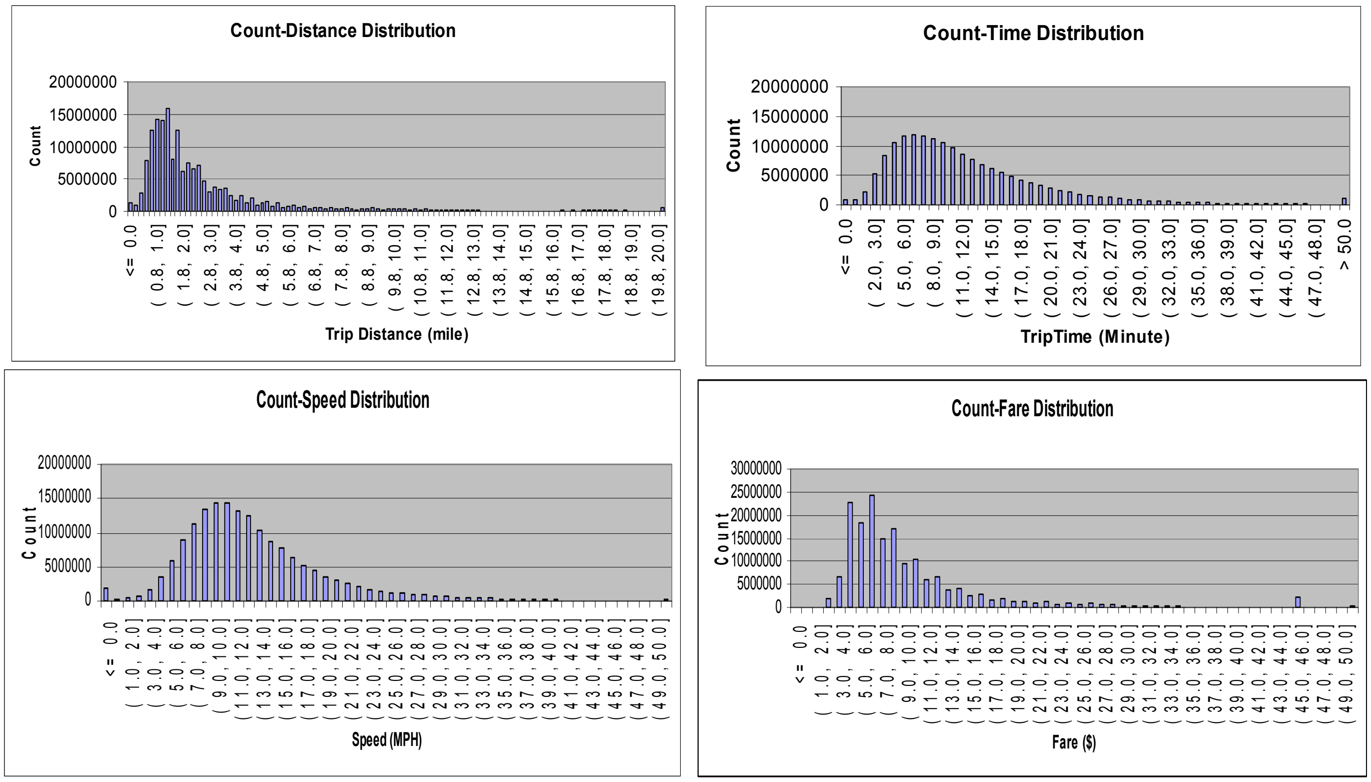


Fig. 2 Histograms of Distributions of Trip Distance, Time, Speed and Fare

4.2 Distributions of Trip Distances, Time, Speed and Fare

In addition to the spatially aggregated pickup and drop-off counts at different grid scales, we have also computed the histograms of travel distances, travel times, travel speeds and travel fares from the 2009 trip records and they are shown in Fig. 2. Note that the first bins in the histograms represent invalid values and the last bin represents excessive values that are larger than the pre-defined thresholds and their counts can be larger than the neighboring bin counts. From the figure we can see that there are quite some short trips (travel distance <1.0 miles and/or travel time <5 minutes) in NYC which may warrants further studies. While the distributions of both travel distances and travel times are heavily tailed, the speed distribution exhibit a nice normal distribution centered at 10 MPH (Miles per Hour) which may reflect heavy traffic congestion in NYC. The start/end overheads of short trips may also contribute to the low average speed. Despite waiting times are counted towards fares, the distribution of fares closely resembles the distribution of trip distances which may suggest that trip distance is still the dominate factor of fares. It might be interesting to separate midtown and downtown Manhattan area with the rest of the NYC and look into the differences of their travel speed distributions. The same idea can be applied to the time dimension to compare the differences of distributions during peak/off-peak time slots, weekdays/weekends, normal/special events days and their combinations.

4.3 Results on taxi trip outlier detection

Using the method discussed in Section 3, we have set D_0=200 feet, D_1=3 miles and W=2 for our experiments in outlier detections. Among the 166 million taxi trip records, approximately 2.5 millions (1.5%) whose pickup or drop-off locations can not be snapped to a street segment of the NAVTEQ street map dataset and are identified as Type I outliers. While the majority of these outliers could be induced by GPS device errors, some of them may be associated with incompleteness of street networks, e.g., picking up and dropping off at private land parcels, and thus requires further investigations. Fig. 3 plots all the street segments symbolized by the numbers of computed shortest paths that pass through them (c.f. Section 2 and Section 4.4) overlaid with the map of the NYC Community District data [20]. We can clearly see that quite some taxi trips cross the boundary of NYC while the majority of the trips are in the midtown and downtown Manhattan area (also see Fig. 1 and Fig. 5).

Among the 166,384,464 taxi trips whose pickup and drop-off locations are successfully snapped to NAVTEQ street network segments, about 18,000 are identified as type II outliers. For illustration purposes, the top 9 outliers in January 2009 are listed in Fig. 4 ordered by the decreasing ratio of calculated distance (calc_dist) over recorded distance (record_dist). The computed shortest paths clearly suggest that the recorded distances are incorrect. An explanation is that the metering devices, which may be separate from GPS devices, were reset during the trips for various reasons. As the number of Type II outliers is fairly small, our outlier detection technique has made it possible to examine these trip records individually in a GIS (as exemplified in Fig 4). We have tried to decrease D_1 to identify more outliers. However, for short trips, it is likely that taxi drivers may not follow shortest paths for various reasons and thus the proposed approach may increase the chances of false positives on outlier detection. We are trying to incorporate complete GPS traces and learn the differences between real trip paths and computed shortest paths.

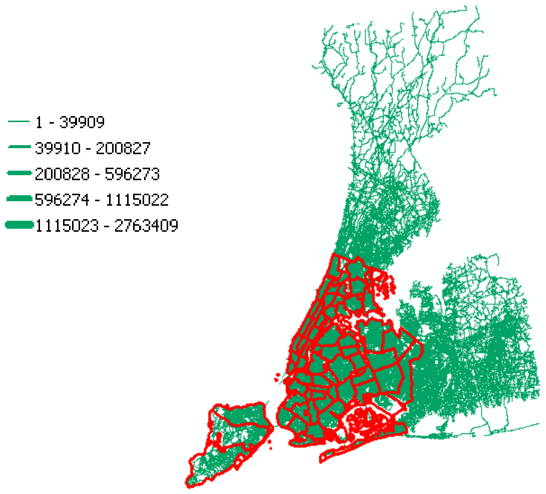
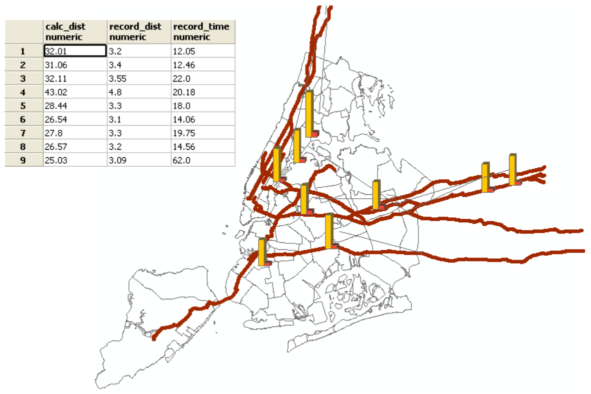


Fig. 3 Mapping of Computed Shortest Paths Overlaid with NYC Community Districts Map



	calc_dist numeric	record_dist numeric	record_time numeric
1	32.01	3.2	12.05
2	31.06	3.4	12.46
3	32.11	3.55	22.0
4	43.02	4.8	20.18
5	28.44	3.3	18.0
6	26.54	3.1	14.06
7	27.8	3.3	19.75
8	26.57	3.2	14.56
9	25.03	3.09	62.0

Fig. 4 Examples of Detected Type II Outliers

4.4 Results on Betweenness Centrality

The 166 million taxi trips have resulted in approximately 25 million unique combinations of NAVTEQ street network nodes and it took 5,952 seconds to compute the shortest paths, including aggregating the hourly and overall centrality measurements for both nodes and edges. The result is encouraging in the sense that MoNav-CH has achieved a throughput of roughly 0.2 second per 1000 pair shortest path computation on a reasonably large street network with more than half a million edges using a single CPU core. While computing overhead has been considered as a major hurdle for urban computing in understanding human mobility [21], our results have demonstrated the feasibility of large-scale computation through efficient algorithm engineering.

The aggregated edge centrality map in NYC area using the 2009 data is shown in Fig. 5. We can see that there are strong connections between the midtown and downtown Manhattan area with the two major airports (LGA at top-right and JFK at bottom-right) in NYC. The bridges, tunnels and highways are extensively used for the connections. We can also see that while the connections among different parts of Manhattan are even stronger, a few street segments, such as Broadway and the 5th/6th/7th avenues, have much higher centralities (and hence are more important) according to the taxi trips as shown in the left part of Fig. 5.

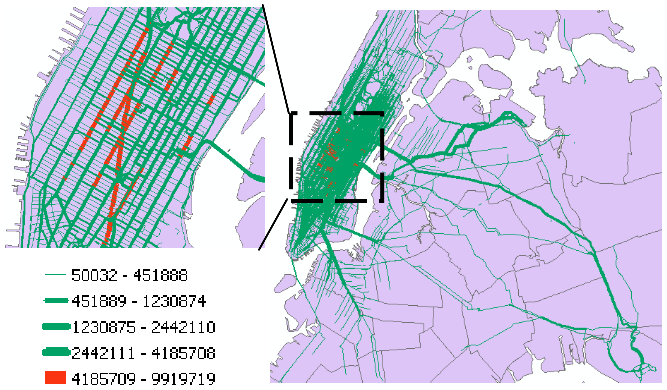


Fig. 5 Mapping Betweenness Centralities (All hours)

To further compare the dynamics of the street segment centralities across times, we have generated hourly centrality maps for the majority of the NYC area. The bi-hourly maps in the midtown and downtown Manhattan area are show in Fig. 6 where the widths of the street segments are symbolized using three levels of shortest path counts as indicated by the legend located at the bottom of the figure. From Fig. 6 we can see that many of the street segments in the area have more than 10^5 trips in 2009 (i.e., ~250 trips per day in an hour) for all hours. There are very few changes across hours from 08H to 22H except a slight recession at 16H (mid-afternoon). However, we do see the connection strengths decrease significantly during 00H-08H compared to the rest of the day. The connection strengths reach a minimum point around 04H where only Broadway and a few other streets still maintain the highest level of connection strength. While NYC has a well-known nickname, i.e., "The City That Never Sleeps" [22], the taxi trip records reveal that it does take a long nap in the early morning and a short nap in the afternoon.

CONCLUSION AND FUTURE WORKS

Large-scale taxi trip records are error-prone due to a combination of device, human and information system induced errors. In the cases that accesses to complete taxi trip trajectories are limited due to either policy, privacy or technical constraints, outlier detections based on pickup/drop-off locations and times and trip distance/duration information are especially challenging. In this study, by leveraging the U^{2}SOD-DB system that we have developed to facilitate efficiently managing large-scale OD data and the MoNav open source package for efficient shortest path computation, we are able to detect outliers that can not be snapped to street segments and/or have significant differences between computed shortest distances and recorded trip distances. The derived edge betweenness centralities can be used to understand connection strengths among different parts of metropolitan areas such as NYC.

For future work, first, we plan to develop a more comprehensive framework for outlier detections and data cleaning by incorporating pickup and drop-off times, trip duration and fare information. Second, we would like to generate dynamics of betweenness maps at different traffic conditions, e.g., peak/non-peak, morning/afternoon and weekdays/weekends. Finally, it is desirable to further improve the efficiencies of shortest path computation by exploiting parallel computing on distributed systems, multi-core machines and GPU devices.

REFERENCES

1. Veloso, M., Phithakkitnukoon, S and Bento, C. (2011). Urban mobility study using taxi traces. Proceedings of the international workshop on Trajectory Data Mining and Analysis (TDMA)
2. http://code.google.com/p/monav/
3. Geisberger, R., Sanders, P., Schultes, D., Delling, D. and McGeoch, C. (2008). Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. Proceedings of Experimental Algorithms.
4. Wang, Y., Zhu, Y., He, Z., Yue, Y. and Li, Q. (2011). Challenges and Opportunities in Exploiting Large-Scale GPS Probe Data. Technical report. HP. Laboratories.
5. Liu, W., Zheng, Y., Chawla, S., Yuan, J. and Xing, X. (2011). Discovering spatio-temporal causal interactions in traffic data streams. Proceedings of the ACM SIGKDD Conference.
6. Ge, Y., Xiong, H., Liu, C. and Zhou, Z.-H. (2011). A Taxi Driving Fraud Detection System. Proceedings of IEEE ICDM.
7. Zhang, D., Li, N., Zhou, Z.-H., Chen, C., Sun, L. and Li, S. (2011). iBAT: detecting anomalous taxi trajectories from GPS traces. Proceedings ACM UbiComp
8. Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W. and Huang, Y. (2009). Map-matching for low-sampling-rate GPS trajectories. Proceedings of ACM-GIS.
9. Zheng, K., Zheng, Y., Xie, X. and Zhou, X. (2012). Reducing Uncertainty of Low-Sampling-Rate Trajectories. Proceedings of IEEE ICDE.
10. http://en.wikipedia.org/wiki/Shortest_path_problem
11. http://project-osrm.org/
12. Delling, D., Goldberg, A. V., Nowatzyk, A. and Werneck, R. F. (2011). PHAST: Hardware-Accelerated Shortest Path Trees. Proceedings of IEEE IPDPS.
13. Zhang, J., Gong, H., Kamga, C. and Gruenwald, L. (2012). U^{2}SOD-DB: A Database System to Manage Large-Scale Ubiquitous Urban Sensing Origin-Destination Data. Technical report online at: http://134.74.112.65/pub/u2soddb_tr.doc.
14. Brandes, U. 2008. On variants of shortest-path betweenness centrality and their generic computation. Social Networks 30, 136--145.
15. Kazerani, A. and Winter, S. (2009). Can betweenness centrality explain traffic flow. Proceedings of the 12th AGILE International Conference on GIS.
16. Leung, I., Chan, X. Y., S.-Y, Hui, P. and Lio, P. (2011). Intra-City Urban Network and Traffic Flow Analysis from GPS Mobility Trace. http://arxiv.org/abs/1105.5839.
17. Qi, G., X. Li, Li, S., Pan, G., Wang, Z. and Zhang, D. (2011). Measuring social functions of city regions from large-scale taxi behaviors. Proceedings of IEEE PERCOM Workshops.

18. Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G. and Huang, Y. (2010). T-drive: driving directions based on taxi trajectories. Proceedings of ACM-GIS.
19. Li, Q., Zeng, Z., Z., Zhang, T., Li, J. and Wu, Z. (2011). Path-finding through flexible hierarchical road networks: An experiential approach using taxi trajectory data. International Journal of Applied Earth Observation and Geoinformation 13(1): 110-119.
20. http://www.nyc.gov/html/dcp/html/bytes/applbyte.shtml
21. Jiang, B., Yin, J. and Zhao, S. (2009). Characterizing the human mobility pattern in a large street network. Physical Review E 80(021136).
22. Wikipedia. http://en.wikipedia.org/wiki/The_City_That_Never_Sleeps

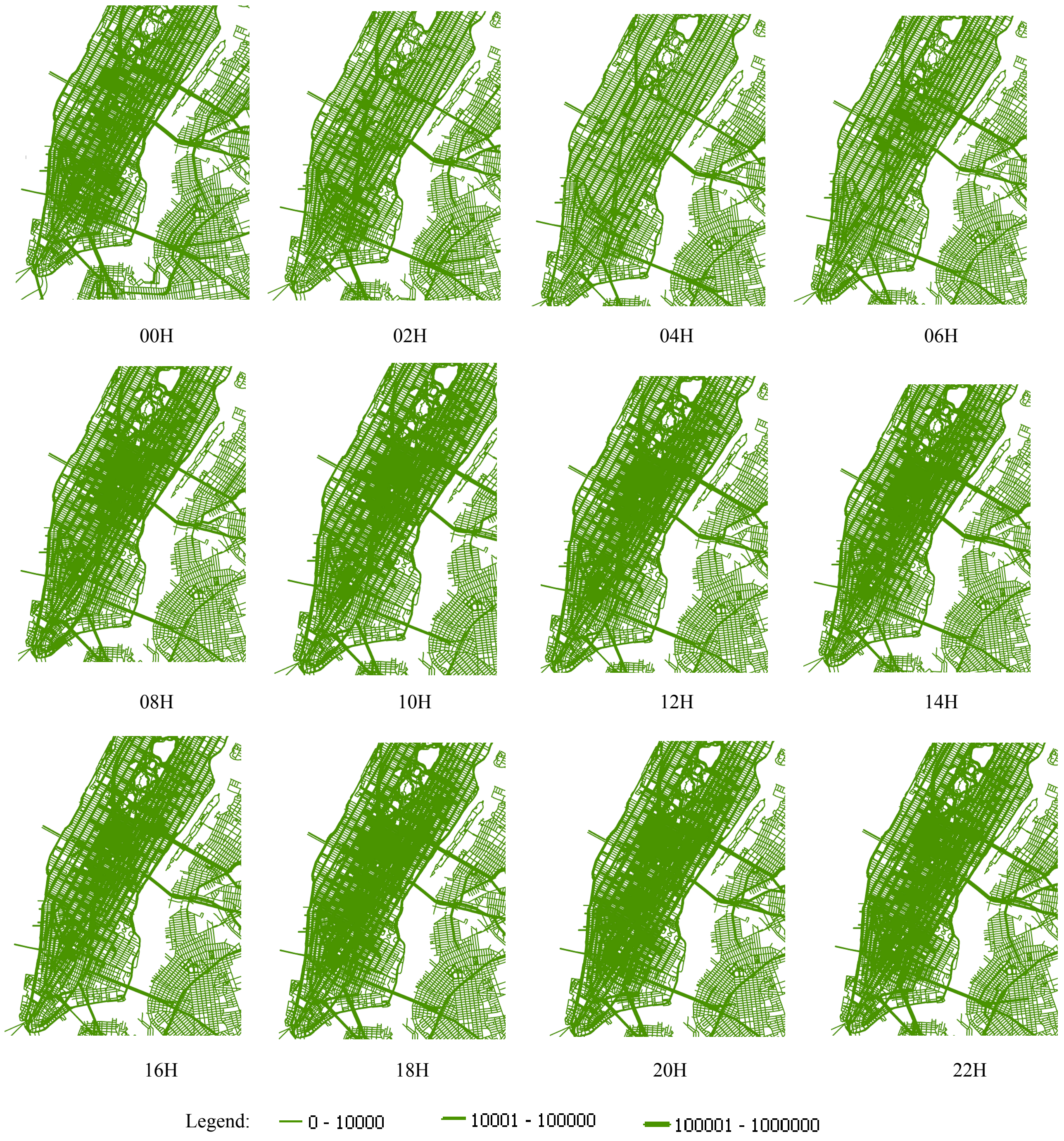


Fig. 6 Mapping Bi-Hourly Edge Betweenness Centrality in the Midtown and Downtown Manhattan Area

U^2SOD-DB: A Database System to Manage Large-Scale Ubiquitous Urban Sensing Origin-Destination Data

Jianting Zhang
Department of Computer Science
City College, the City University of New York
New York, NY, 10031
jzhang@cs.ccny.cuny.edu

Hongmian Gong
Department of Geography
Hunter College, the City University of New York
New York, NY, 10065
gong@hunter.cuny.edu

Camille Kamga
Department of Civil Engineering
City College, the City University of New York
New York, NY, 10031
ckamga@utrc2.org

Le Gruenwald
School of Computer Science
University of Oklahoma
Norman, OK 73071
ggruenwald@ou.edu

ABSTRACT

Volumes of urban sensing data captured by consumer electronic devices are growing exponentially and current disk-resident database systems are becoming increasingly incapable of handling such large-scale data efficiently. In this paper, we report our design and implementation of U^2SOD-DB, a column-oriented, Graphics Processing Unit (GPU)-accelerated, in-memory data management system targeted at large-scale ubiquitous urban sensing origin-destination data. Experiment results show that U^2SOD-DB is capable of handling hundreds of millions of taxi-trip records with GPS recorded pickup and drop-off locations and times efficiently. Spatial and temporal aggregations on 150 million pickup locations and times in middle-town and downtown Manhattan areas in the New York City (NYC) can be completed in a fraction of a second. This is 10-30X faster than a serial CPU implementation due to GPU accelerations. Spatially joining the 150 million taxi pickup locations with 43 thousand polygons in identifying trip purposes has reduced the runtime from 30.5 hours to around 1,000 seconds and achieved a two orders (100X) speedup using a hybrid CPU-GPU approach.

Categories and Subject Descriptors

H.2.8 [Database Management]: Database applications – Data mining, Spatial databases and GIS

General Terms

Management, Design

Keywords

Urban Sensing, Origin-Destination, High-Performance, GPGPU, Spatial Aggregation, Temporal Aggregation, Spatial Join

1. INTRODUCTION

With the increasing availability of imaging, locating and other types of sensing technologies on portable wireless devices, huge amounts of pervasive urban sensing data are being captured at ever growing rates. For example, the approximately 13,000 taxicabs in the New York City (NYC) equipped with GPS devices generate more than half a million taxi trip records per day. Cell phone call logs represent a category of data at an even larger scale [1, 2]. Also as visitors travel around the world more frequently, location-dependent social networks such as FourSquares [3], and, location-enhanced social media such as text posted to wiki sites [4], images and videos posted to Flickers and YouTube [5], can also potentially generate large-amounts of spatial-temporal data. All the three types of data have a few features in common: (1) they are produced and collected by end users using commodity sensing devices and are rich in data volumes in urban areas; and (2) they are a special type of spatial-temporal data with an origin location and a destination location in geo-referenced space domain and a starting time and an ending time in the time domain. However, the intermediate locations between origins and destinations are either unavailable, inaccessible or unimportant. Compared with traditional geographical data collected by government agencies for urban planning and city administration purposes, these data can be more effective to help people understand the real dynamic of urban areas with respect to spatial/temporal resolutions and representativeness. We term such data as Ubiquitous Urban Sensing Origin-Destination data, or U^2SOD data, for notation convenience. Despite the close relationships between U^2SOD data and the Spatial Databases (SDB) [14] and Moving Object Databases (MOD) [6, 7], our experiences have shown that traditional disk-resident and tuple/row oriented spatial databases and moving object databases are ineffective in processing large-scale U^2SOD data for practical applications.

Towards this end, we have designed a new data management system, termed U^2SOD-DB, to address the technical challenges that we have encountered when processing about 300 million taxi trip records during an approximate two years period in NYC. U^2SOD-DB is an in-memory database system and adopts a time-segmented column-oriented data layout strategy. A set of high-performance spatial and temporal indexing and query processing techniques has been developed to efficiently aggregate and join U^2SOD data and their auxiliary geographical data (e.g., urban infrastructure) to understand urban dynamics. Parallel accelerations using General Purpose computing on Graphics Processing Units (GPGPU) technologies [8] are incorporated where appropriate. The rest of the paper is arranged as the follows. Section 2 introduces background and related work. Section 3 presents the system architecture and implementation details of the key components. Section 4 reports the performance evaluations. Finally, Section 5 concludes the paper and outlines the future work.

2. BACKGROUND, MOTIVATIONS AND RELATED WORK

The increasingly available location data generated by consumer wireless portable devices, such as GPS, GPS enhanced cameras and GPS/WiFi/Cellular enhanced mobile phones, have significantly changed the ways of collecting, analyzing, disseminating and utilizing urban sensing data. Traditionally city government agencies are responsible for collecting various types of geographical data for city management purposes, such as urban planning and traffic control. The data collection is usually done through sampling, typically at coarse-resolutions, and questionnaire-based investigations which often incur long turn-around periods. In contrast, as consumer mobile devices become ubiquitous, similar data obtained from GPS-traces and mobile phone call logs, has much finer resolutions. With the help of privacy and security related technologies, the aggregated records from such ubiquitous urban sensing data can be enormously helpful in understanding and addressing a variety of urban related issues. Research groups from both academia (e.g., MIT Sensible City Lab) and industries (e.g., IBM Smart Plant Initiative and Microsoft Research Asia) have developed quite a few techniques to utilize such data and understand the interactions among people and their location/mobility at both the city level (e.g., Beijing [9], Boston [10] and Rome [1]), social group level (e.g., friends [11] and taxi-passenger pairs [12]) and the individuals level [13]. However, most of the existing studies focus on the data mining aspects of such ubiquitous urban sensing data through case studies while largely leaving the data management aspects untouched. Lacking proper data management techniques can result in signficant technical hurdles to make full use of such data for societal impacts.

Although Geographical Information Systems (GIS) and Spatial Databases (SDB) [14] are the commonly used tools to handle geo-referenced spatial data, we argue that existing GIS and SDB technologies and available tools are inefficient and/or insufficient in managing large-scale, location dependent and time-varying ubiquitous urban sensing data. First, when looking back through the history, GIS and SDB technologies are mostly driven by the needs of managing cadastral types of geographical data more than five decades ago which are quite different from the ubiquitous urban sensing data we have today. Second, although GPS trajectories can be naturally modeled as moving object data and existing moving object databases technologies for indexing and query processing (e.g., [6, 7, 26]) can be exploited, many other types of ubiquitous urban sensing data can not be efficiently managed by existing databases. Third, existing commercial and open source GIS and SDB are slow in adopting hardware acceleration techniques. Existing disk-resident GIS and SDB software tools are mostly designed for online transactions rather than analytics and they can be orders faster if they are tailored for read-only data [15, 16]. Additional signficant speedup is possible if modern hardware acceleration techniques are exploited, e.g., flash memory for off-chip data access, CPU caches and GPU shared memory for on-chip data access, and, multi-core CPUs and many-core GPUs for parallel computing [17]. Our own experiences in processing 300 millions taxi trip records in NYC have shown that the performance of existing GIS and SDB software tools is unacceptable when processing urban sensing data at this scale – a simple spatial join may take dozens of days even on a current high-end server. This may also explain why most of existing studies that utilize ubiquitous urban sensing data do not adopt a database approach which is arguably more convenient from a user perspective.

Our research on developing U^{2}SOD-DB is motivated by the following two aspects. First, we observe that Origin-Destination (OD) data is much simpler than the data types that are modeled by both classic GIS, SDB and MOD systems. An OD record basically has a pair of spatially and temporally referenced points besides additional associated attributes. An OD record can be modeled as a fixed sized relational tuple although the spatial and temporal components need to be handled specially for efficiency purposes. In contrast, both classic SDB and MOD have variable sized records to handle polygon/polyline/trajectories. Many bounding box based indexing techniques that are effective for polygons/polylines/trajectories become ineffective for OD data as the data volumes of bounding boxes can be as large as the data volumes of the OD data itself. Second, recent research works on in-memory based columnar store of relational data have shown that various in-database data compression and relational operations on multicore processors using parallel threads can achieve significant speedups [18, 19]. As many spatial and temporal operations on urban sensing data are both computational and I/O intensive, it is desirable to exploit the parallel processing powers of commodity hardware, including both multi-core CPUs and many-core GPUs.

It is easy to observe that GPS-recorded OD data is a subset of GPS trajectory data with only the first and the last GPS reading in a trip. Obviously adequately sampled GPS traces provide more spatiotemporal information than an OD pair. A GPS trajectory can actually be considered as the concatenation of consecutive OD pairs. However, we argue that OD data may have richer semantics than GPS traces as the origins and destinations of trips are explicitly marked to delineate the starting position/time and the ending position/time. Additional trip-specific attributes might have been attached to OD pairs when they are collected. In contrast, GPS traces are often needed to be segmented or clustered to identify trips based on various heuristics [20] or processed manually using a travel survey approach [21]. Furthermore, the data volumes of OD pairs are usually much smaller than those of GPS traces which make the OD data more suitable for large-scale studies. Although the ideal situation is to integrate the trip data with GPS traces in many applications, we argue that techniques that are designed for management and analysis of OD data can NOT be subsumed by those that are designed for GPS data. For cell phone call log data and location dependent social network data where the exact geometric traces are either not available or not important, techniques that are designed for OD data can be more suitable for such applications.

To the best of our knowledge, the work reported in this paper is the first in designing and implementing a high-performance data management system for large-scale ubiquitous urban sensing data. While it is difficult to develop a full-fledged system to handle all types of U^{2}SOD data, we have taken an incremental approach to developing modules for such purpose. Our current research and development are driven by the practical needs of processing large-scale taxi-trip records in NYC. We expect the accumulated modules will eventually lead to an integrated database system to lower the barrier of managing and data mining of large-scale U^{2}SOD data, so that domain users can focus on urban research issues rather than computing techniques. This is arguably a significant contribution from the Urban Computing research community.

3 SYSTEM ARCHITECTURE AND IMPLEMENTATION DETAILS

Our design of U^{2}SOD-DB has three layers. The bottom tier is closely related to physical data layout and we have adopted a time-segmented, column-oriented data layout approach. The raw data are first transformed into binary representations and attributes are clustered into groups based on application semantics. The data corresponding to the attribute groups at a certain time granularity are stored as a single database file with the relevant metadata registered with the database system. We assume one or more database files can be streamed into CPU main memory as a whole to maximize disk I/O utilization. Multicore CPU processors can access the data files in parallel once they are loaded into the CPU main memory. They can also be transferred to GPU global memories should the system determine that GPU parallel processing is more advantageous. The middle tier is designed for efficient data accesses and aggregations, including compression, aggregation and indexing. While we skip the implementation details of most of the modules in this layer due to space limit, we would like to note that a few in-memory based indexing techniques for both categorical (e.g., trip mode and trip purpose), 1D numeric (e.g., time/distance/fare/tip) and 2D numeric data (mostly x and y coordinates of pickup/drop-off locations) are being explored. The third tier handles efficient joins between OD data and urban infrastructure data, such as road networks and administrative regions. The overall architecture is shown in Fig. 1. We next provide implementation details of several key components in the design.

3.1 TIME-SEGMENTED COLUMN-ORIENTED DATA LAYOUT

As shown in Fig. 2, we group the attributes that are associated with trip records into several groups and data of the attributes in the same group are stored in a single database file by following the column-oriented layout design principle - attributes in the same group are likely to be used together and thus it is beneficial to load them into main memory as a whole to reduce I/O overheads [16]. Given a fixed amount of main memory, as the attribute field lengths of individual groups are much smaller than the length of all attributes, more records that are related to analysis can be loaded into main memory for fast data accesses. In general, the column-oriented data layout design improves traditional tuple based physical storage in relational databases by avoiding reading unneeded attribute values into main memory buffers and subsequently increasing the number of tuples that can be read into the buffers in a single I/O request. We note that while traditional relational databases use page as the basic unit for I/Os (e.g., 8 kilobytes), modern hard drives often have large hardware caches (e.g., 32 megabytes). As such, reading large chunks of tuples can be advantageous for read-only data to fully utilize hardware capacity and improve overall disk I/O performance which is arguably the most severe bottleneck in processing large-scale data. We also note that combining several attribute groups into one and extracting attributes from multiple groups to form materialized views can be beneficial for certain tasks. For example, in Fig. 2, attribute group 5 (start_x, start_y, end_x and end_y) can be considered as a materialized view of the attribute group 2 (start_lon,start_lat, end_lon and end_lat) by applying a local map projection to the lat/lon pairs. Since the projected data are frequently used in calculating geometric and shortest path distances and map projections are fairly expensive, materializing attribute group 5 can significantly improve system performance. Another example to demonstrate the utilization of materialized views on the physically grouped attributes is verifying the recorded trip times (indicated by trip_time in group 6) with computed trip times (by subtracting pickup time from drop-off times in group 3) by materializing the respective attributes in the two groups. We further note that among the attributes in the original dataset shown in Fig. 2, some of them can be derived from others. For example, both start/end zip codes (group 9) and addresses of pickup and drop-off locations (group 8) can be derived from start/end latitudes and longitudes (group 2) through reverse geocoding or other related techniques.

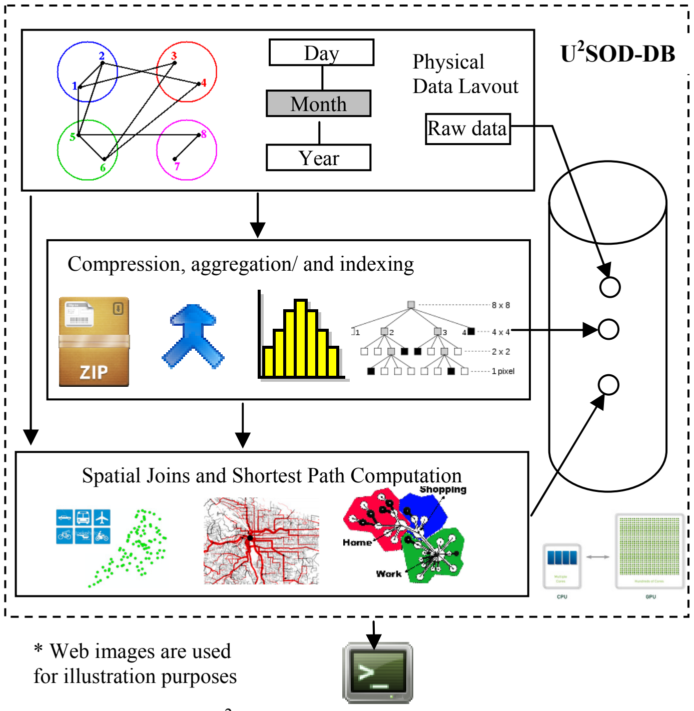

U^{2}SOD-DB
Day
Month
Year
Physical Data Layout
Raw data
Compression, aggregation/ and indexing
ZIP
8 x 8
4 x 4
2 x 2
1 pixel
Spatial Joins and Shortest Path Computation
Shopping
Home
Work
CPU
GPU
* Web images are used for illustration purposes

Fig. 1 U^{2}SOD-DB System Architecture

After determining data layout by grouping attributes (vertical partitioning), the next issue is to determine the appropriate number of records in database files so that these files can be efficiently streamed among hard drives, CPU main memory and GPU device memory. The sizes of the database files should be larger than those of hard disk caches but small enough to accommodate multiple database files simultaneously in CPU/GPU memories so that typical operations can be completed in the designated memory buffers. At the same time, the numbers of records should correspond to certain time granularities as much as possible. In our design, we use month as the basic unit (temporal granularity) to segment taxi trip records (horizontal partition). Given that there are about half million taxi trip records per day in NYC, assuming that an attribute group has a record length of 16 bytes (e.g., four attributes with each represented by a 4-bytes integer), the database file would be 16*0.5*30=240 megabytes. Since the hard drive cache and the main memory capacity in our experiment system are 16 megabytes and 16 gigabytes, respectively, the database file size seems to be appropriate although other sizes might be suitable as well. We plan to develop a data layout advisor when U^{2}SOD-DB is applied to other types of OD data or applied in a different region by incorporating user-defined parameters.

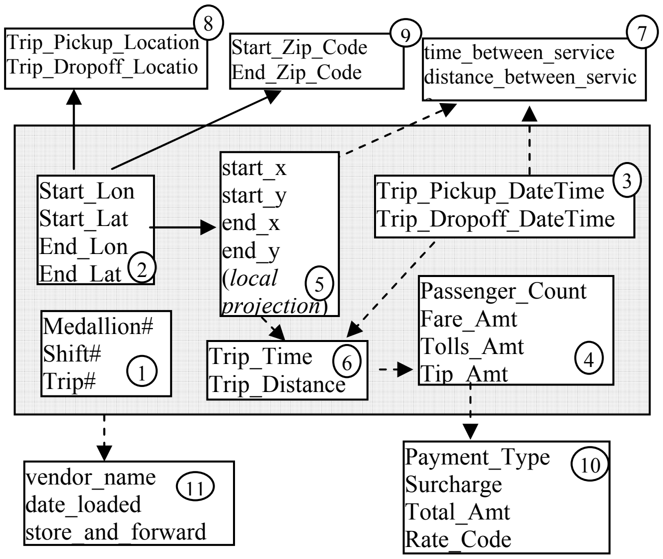


Fig. 2 Column-Oriented Physical Data Layout in U^{2}SOD-DB

3.2 Efficient Spatial -Temporal Aggregations

While many operations and analytical tasks can be performed on the U^{2}SOD data, spatial and temporal aggregations are among the fundamental ones. Fig. 3 illustrates the general framework and example spatial and temporal aggregations that are supported by our U^{2}SOD-DB design. Most of the temporal aggregations (e.g., daily and hourly) and some of the spatial aggregations (e.g., grid based) can use universal schemas, while more complex aggregations rely on the schemas provided by infrastructure data such as road network and administrative hierarchies. While we defer the discussions on associating OD data with the infrastructure data through spatial joins in the next subsection, we note that, as shown in Fig. 3, once the OD locations (i.e., pickup and drop locations in the taxi trip data) are associated with street segments or different types of zones, the aggregations can be reduced to simple grouping without involving expensive spatial and/or temporal operations any more. These relatively simple aggregations should be supported efficiently as much as possible to allow interactive visual explorations.

Towards this end, we have designed and implemented a GPU based spatial and temporal aggregation module in our U^{2}SOD-DB. As shown in the experiment section (Section 4), the GPU based aggregation module can significantly improve aggregation performance by 1-2 orders by making use of the massively data parallel computing power on GPUs [8] that are already widely available on commodity computers. The design uses four parallel primitives, namely *Transform*, *Sort*, *Reduce_By_Key* and *Scatter*. Take the grid-based spatial aggregation for an example, the *Transform* primitive converts x/y coordinate pairs into row-major ordered array indices and uses the indices as keys for the subsequent three primitives. The *Sort* primitive sorts the keys. As a single commodity GPU device can sort integers at the order of a few hundreds of millions to over a billion integers per second, this step can be efficiently processed for reasonably sized OD datasets in the orders of a few hundreds of millions on current commodity GPUs. The *Reduce_By_Key* primitive sums up the numbers of the same keys in the input vector and output the unique keys and their numbers of occurrences. In the grid-based spatial aggregation case, the numbers of pickup or drop-off locations within all the non-empty grid cells are computed. The last step essentially transforms the aggregation results from a sparse array representation into a dense array presentation should the dense array representation be more desirable. The step makes use of the *scatter* primitive and is optional. In the grid-based spatial aggregation case, a 2D grid is created and the numbers of non-empty grid cells are written to the corresponding positions by using the keys as the indices in the 2D grid (row-major ordered) while leaving the empty grid cells to have the initial zero value.

We note that the similar process can be applied to many aggregations in Fig. 3 as long as a key can be identified for aggregation. In addition to simply counting the numbers of records under a same key using the *Reduce_By_Key* primitive, many other types of operations can be implemented by using user defined binary functions, i.e., functors [22]. Given that GPUs are designed to support large-scale floating point computation, we believe it is promising to design more insightful but more computationally intensive measurements on GPUs during the aggregations. This is left for future work.

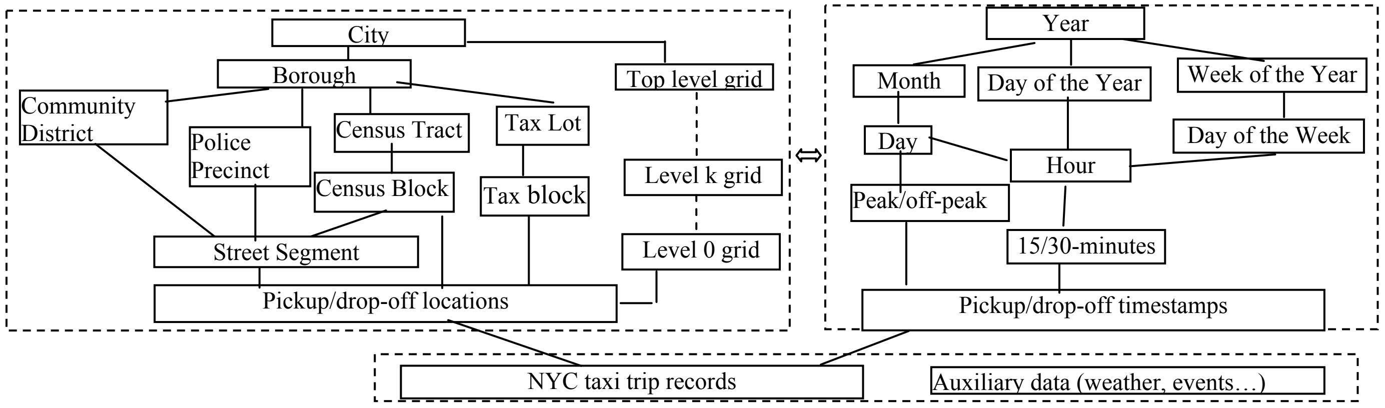


Fig. 3 Example Spatial and Temporal Aggregations in U^{2}SOD-DB

3.3 Spatial Join with Infrastructural Data

Very often OD data needs to be associated with infrastructure data, such as street networks, administrative regions and census zones. In NYC, the MapPluto tax lot dataset [23] has detailed land use information of each tax block. As tax blocks are typically at the finer spatial granularity than census blocks, they can be very useful in identifying trip purposes. In addition, in big cities such as NYC, usually there are large numbers of categorized Points of Interests (POI) sites that are collected by companies such as NAVTEQ for navigation purposes. While the infrastructure data are dynamic themselves, the change rates are relatively slow and can be considered static for reasonable long study periods. Associating OD data with these infrastructure data can be done offline to speed up online query processing in many cases. Based on the observation that inefficient disk I/Os dominate existing SDB and GIS systems for such spatial joins, we leverage the in-memory friendly physical data layout and our existing work on GPU-based indexing of large-scale point data [24] for this purpose. An open source spatial indexing package called libspatialindex [25] is integrated to develop a CPU-GPU hybrid approach to achieve the desired level of performance of spatial joins in U^{2}SOD-DB. Before introducing the design and implementation details, we would like to provide more details on the three types of spatial joins that U^{2}SOD-DB currently supports based on application needs, i.e., P2N-D, P2P-T and P2P-D.

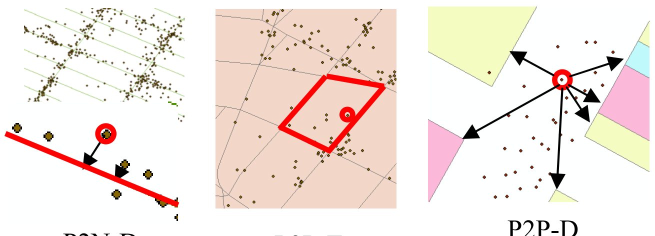


Fig. 4 Illustrations of Three Types of Spatial Joins between OD Data and Urban Infrastructure Data

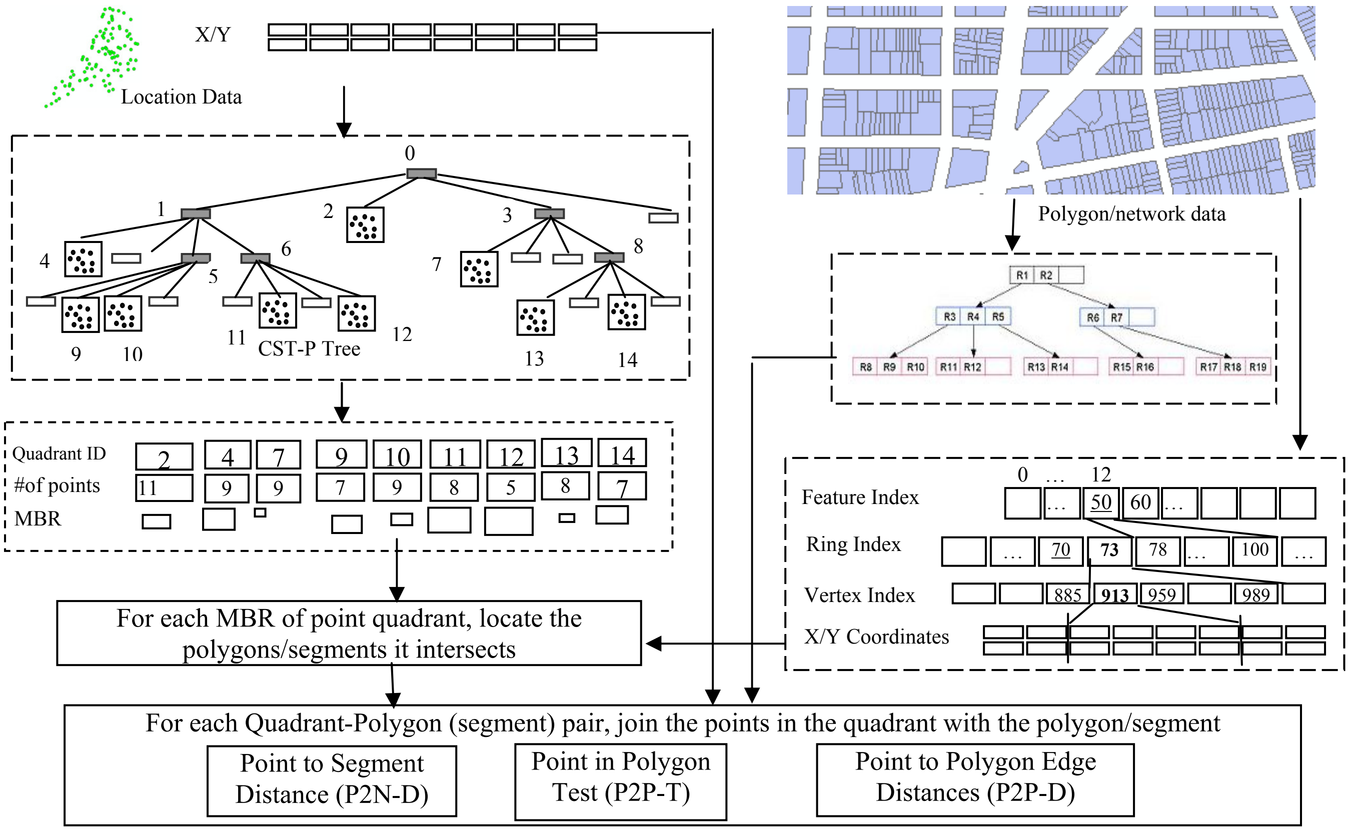

X/Y
Location Data
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
CST-P Tree
Polygon/network data
R1 R2
R3 R4 R5
R6 R7
R8 R9 R10 R11 R12 R13 R14 R15 R16 R17 R18 R19
Quadrant ID
#of points
MBR
2 4 7 9 10 11 12 13 14
11 9 9 7 9 8 5 8 7
Feature Index
0 … 12
… 50 60 …
Ring Index
… 70 73 78 … 100 …
Vertex Index
885 913 959 989
X/Y Coordinates
For each MBR of point quadrant, locate the polygons/segments it intersects
For each Quadrant-Polygon (segment) pair, join the points in the quadrant with the polygon/segment
Point to Segment Distance (P2N-D)
Point in Polygon Test (P2P-T)
Point to Polygon Edge Distances (P2P-D)

Fig. 5 Illustrations of Spatial Join Processing using a Hybrid Framework and Cache-Friendly In-Memory Data Representations

The P2N-D spatial join associates each point with its nearest street network segment. The P2P-T spatial join associates each point with a polygon that it falls within, i.e., through point-in-polygon test. The P2P-D spatial join associates a point with a set of polygons ordered by the smallest distance to the edges of such polygons. The three types of spatial joins are illustrated in Fig. 4. The purposes of P2N-D and P2P-T are straightforward and we next briefly explain the utilization of the P2P-D spatial join in U^{2}SOD-DB for processing taxi trip records. Conceptually, most of taxi pickup drop-off locations (or origins and destinations in general) should be close to street segments but outside of buildings and other types of properties that are not accessible to public vehicles. The distances between the locations and the boundaries of the properties can be important in helping identify trip purposes. However, the same O/D location can be associated with multiple polygonal property boundaries and there are some inherent uncertainties of the approach. Since we are interested in aggregated trip patterns rather than individual trips, the uncertainties are not a problem in computing aggregated trip patterns. Compared with P2N-D and P2P-T spatial joins, P2P-D can also be useful in handling positional errors that are inherent to GPS or GPS-equipped devices, especially in NYC where urban canyon effect can be significant [21]. We note that P2P-D is related to KNN queries in classic SDB when the distance between a point and a polygon is used to define nearest neighbors. However, in addition to providing parameter K, P2P-D spatial join also has an influence window and only the top-K polygons that intersect with the influence window are joined with a query point. As such, P2P-D spatial join can be considered as a combination of window query and KNN query for each query point. From a computing perspective, P2P-D is more complex than P2N-D and P2P-T. This is because a set of (identifier, distance) pairs, instead of a single identifier, needs to be returned from this type of spatial join for each location in an OD record.

The proposed solution is a hybrid CPU-GPU approach. First, a Constrained Spatial Partition Tree for Point Data (CSPT-P Tree) is constructed on GPU using the Thrust parallel library [22] which has achieved a 23X speedup than a serial CPU implementation as detailed in our technical report [24]. Second, an R-Tree is constructed using the libspatialindex package [25] on CPU on the infrastructure data (street network segments or zones). Third, the bounding boxes of the set of points that fall into the leaf nodes of the CSPT-P tree are queried against the R-Tree to associate the leaf nodes with polygons. Assuming the influence window size is (w, h) and the bounding box is (x1,y1, x2, y2), then if the query window (x1-w,y1-h,x2+w,y2+h) intersects with the Minimum Bounding Rectangle (MBR) of a polygon/segment indexed by the R-Tree, then the CSPT-P tree node and the polygon are paired and queued for further processing. This is essentially the filtering phase in classic spatial joins [14]. The final step is to actually join the points with the polygon/segment. As a polygon/segment is a collection of points, this step essentially requires pair-wise computation among the points in the two datasets before applying an aggregation function g(x) to each of the query points. For P2N-D and P2P-D, g(x) is the minimum function on distance. For P2P-T, the function g(x) can be a boolean function on the number of intersections if a ray-casting algorithm is applied. As there are multiple polygons/segments returned from a query on the R-Tree, a second aggregation function f(x) is needed for reduction. For P2N-D, f(x) is the minimum function on distance. For P2P-T, f(x) is a boolean testing function as a point can only fall within a single polygon. For P2P-D, f(x) is the top-K ranking function on distances.

Among the differences between our spatial join framework and the existing SDB implementations (e.g., PostgreSQL), the most signficant one is that our framework exploits the cache-conscious array representations extensively for both OD location data and the infrastructure data as well as their auxiliary indices whereas possible. While existing SDB implementations utilize sophisticated data structures to efficiently map between database files and main memory buffers to reduce I/Os and minimize computation when the available main memory buffer is small, this is unnecessary in our framework as we aim at making full use of large memory capacities available to modern computing systems. Our time-segmented column-oriented physical layout and the data compression modules ensure that the memory footprint within a batch computing does not exceed available memory capacity. Compared with using memory pointers which is required in many dynamic data structures, the array based representation not only is cache friendly but also reduces memory footprints significantly. The technique is especially useful on modern hardware as data accesses are becoming much more expensive than computing [17].

As detailed in the evaluation section (Section 3), our implementation is able to complete a P2P-D spatial join of 150 million taxi pickup locations with 43 thousands MapPluto tax lot polygons in about 500 seconds using an influence window size of (100,100) feet. Analyzing the runtimes of the P2P-D join reveals that building the CSPT-P tree for the location data took only about 6-7 seconds on GPUS while nearly an order more time was spent on querying the intersecting polygons of CSPT-P tree quadrants and two orders more time was spent on distance computation. This clearly indicates that a more efficient spatial indexing approach for infrastructure data is needed in an in-memory computing environment. We are considering replacing the libspatialindex with an in-house GPU-based R-Tree implementation or using a simpler multi-level grid file approach. It is also relatively straightforward to parallelize distance computation on GPUs by assigning a quadrant-polygon pair to a GPU computing block and launching multiple threads in a computing block proportional to the number of points in a quadrant of the location data. Our preliminary results have shown more than 150X speedup on distance computation using GPU acceleration and we expect 10-40 times speedups for a pure GPU implementation. This will bring a total of 3- 4 orders of speedup (~2000X) when compared with the baseline serial CPU implementations using the state-of-the-art open source spatial indexing/query packages.

4 CASE STUDIES AND PERFORMANCE EVALUATIONS

4.1 Data and Experimental Setting

In addition to the taxi trip data with 300 million records in about two years that have been discussed above, our case studies also use several infrastructure datasets, including NYC DCPLION street network data and census 2000/2010 data that are publically available on the NYC-DCP website [23], and the NYC MapPluto Tax Lot data [23] which requires a license. Both the taxi trip data and the infrastructure data require several preprocessing steps before they can be used in the U^{2}SOD-DB system. The details of preprocessing are omitted due to space limit. All experiments are performed on a Dell Precision T5400 workstation equipped with dual quadcore CPUs running at 2.26 GHZ with16 GB memory, a 500G hard drive and an Nvidia Quadra 6000 GPU device. The sustainable disk I/O speed is about 100 megabytes per second while the theoretical data transfer speed between the CPU and the GPU is 4 gigabytes per second through a PCI-E card.

The taxi trip data are partitioned both vertically and horizontally as described in Section 3.1. Among the 11 attribute groups, only groups 3, 4, 5, 6 are used in this study. While more sophisticated analysis is possible, for example, analyzing profitability at the shift level, in this paper, we focus on two aspects. The first one is the essential functionality that is important to taxi trip analysis. In this case, the importance of performance is secondary to functionality and U^{2}SOD-DB will be evaluated by its unique functionality. The second aspect is the performance of key operations, including both online aggregations and offline spatial joins. In this case, we will compare the performance of the U^{2}SOD-DB modules with the best-effort serial CPU implementations.

4.2 Performance Evaluations on parallel spatial and temporal aggregations

To evaluate the GPU-based spatial and temporal aggregation approach presented in Section 3.2, we feed the 177 million taxi trips in 2009 to the system with a spatial coverage of approximately 3*3 miles that covers the middle- and low-Manhattan at a 0.5 feet spatial resolution which results in over 150 million pickup locations in the area. While the GPU memory footprint of the algorithm depends only on the number of locations prior to generating a final grid with the value in each cell representing the number of pickup locations that fall within the cell, the GPU memory limit (6GB on Nvidia Quadro 6000 device) prevents us from experimenting CSPT-P tree levels 14 or above where the number of grid cells is 256 million (2^{28}) or higher. As such, we have performed grid-based spatial aggregations at the levels 8-13 (6 levels) using the 2D sparse matrix presentation with additional aggregations on levels 14-16 using a 1D vector representation (c.f. Section 3. 2). The resulting 2D grids at the levels 8 and 13 are shown in Fig. 6. The picture on the right clearly shows the effects of the street network on the taxi pickup locations.

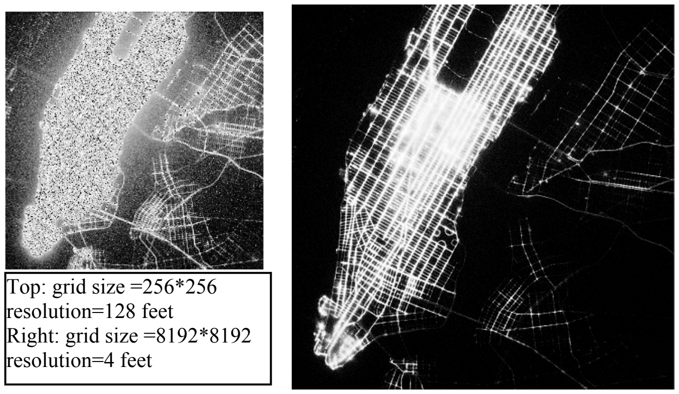


Fig. 7 Grid-Based Spatial Aggregations of Pickup Locations in 2009 with two Resolutions

Since the grid-based spatial aggregations can also be easily implemented on CPUs by sequentially looping through all locations and increasing the counters at the corresponding grid cells, our evaluations focus on the performance gains due to GPU accelerations in U^{2}SOD-DB. First, the total GPU runtimes vary from 240 ms at level 8 to 326 ms at level 13, a 38.5% increment. In contrast, the CPU times increase from 2497 ms to 17786 ms, a 712.3% increment. The results clearly demonstrate the scalability of our GPU-based implementation. While GPU memory capacity currently is a limiting factor, we can coordinate the radix sort algorithm on both CPUs and GPUs to allow larger numbers of data points to be sorted which is left for future work. Second, more importantly, when comparing GPU total runtimes with CPU runtimes, we see a gain of 10-30X speedup as the grid levels increase. We would also like to note that our GPU implementation uses the Thrust parallel primitives [22]. Although very often a good tradeoff between coding complexity and code efficiency, the primitives based implementations usually are not the most efficient implementations achievable. By re-implementing the module using a native programming language to consolidate the four steps (c.f. Section 3.2) and use GPU shared memory efficiently, it is quite possible to further reduce GPU runtimes at all levels for higher speedups. We note that the CPU implementation only involves the transform and scatter steps without needing the sort and reduce steps which is only possible for sequential algorithms. In addition, the CPU implementation also has been optimized with –O3 compilation flag and thus we do not expect further performance improvements. We have also experimented temporal aggregations at the hour and minute temporal granularities using the same set of data after spatial filtering. The runtimes for the GPU and CPU implementations are 197.63 ms and 1709.02 ms, respectively, at the minute granularity with a bin number of 24*60=1440, i.e., a speedup of 8.6X has been achieved. The runtimes are slightly lower at the hour granularity (bin number = 24) where the runtimes are 165.12ms for GPU and 1598.89 ms for CPU, respectively. Again, a 9.7X speedup has been achieved.

While it seems that the runtimes for both spatial and temporal aggregations on CPUs are already acceptable in many offline applications, we argue that GPU accelerations can further make real time and interactive spatial and temporal aggregations possible by providing sub-second response times. As the experiments have shown, this currently is not possible on CPUs, especially in the case of high-resolution spatial aggregations. While using indexing and/or materialization can potentially also reduce the runtimes of CPU-based applications to sub-second level, our primitives based GPU implementation is preferable due to its simplicity. The implementation essentially requires only 4 lines of code with each line corresponding to a parallel primitive and a few lines to implement a functor (C++ functional object) to map between values and bin numbers (either a grid cell in spatial aggregations or a time index in temporal aggregations). When the aggregation rules get more complex, such as combinations of spatial and temporal queries, we expect our GPU based implementation will achieve higher speedups over CPU based ones. We leave the performance studies of complex spatial, temporal and spatiotemporal aggregations for future work.

4.3 Performance Evaluations on P2P-D Spatial Join

To test the efficiency of the P2P-D spatial join implementation, we have built a CSPT-P tree for the taxi trips whose pickup locations fall within the study area shown in Fig. 7. We limit the spatial extent of the study area mostly because our current CSPT-P tree construction algorithm can only calculate Morton codes from points with 32 bits (the extension is left for future work). We could have covered all NYC areas using a coarser spatial resolution. Since more than 85% of pickup locations of all the taxi trips in 2009 fall within the study area, we believe it is acceptable to limit our study area but use a higher spatial resolution (0.5 feet). For the CSPT-P tree construction, the only parameter to set is the maximum number of points that is allowed in the leaf nodes of the CSPT-P tree (N). Table 1 lists the end-to-end runtimes using three N values of1000, 2000 and 5000. We have used the default parameters for the R-Tree. The number of nearest neighbors is empirically set to 10. From the results listed in Table 1 we can see that our current implementation is able to achieve an end-to-end runtime in the order of 500-2000 seconds using a variety of parameter combinations for P2P-D spatial join between the 150,417,865 pickup locations and 43,252 polygons. Compared with pure CPU implementation which requires about 30.5 hours to complete the same P2P-D join, we have achieved more than two orders (100X) of speedup. Given the involved computational intensities in terms of numbers of locations (A), quadrant-polygon pairs (D), point to polygon edge distance computations (E) and the resulting records (F) listed in Table 1, it is clear that modern processors are quite capable of processing large-scale U^{2}SOD data and there are great potential for parallel processing for larger-scale data.

Table 1 Computation Intensities and Runtimes of P2P-D spatial join on 150 million locations and 43 thousands polygons

Max # of points in a CSP-Tree quadrant		1000	2000	5000
	A	15,0417,865		
	B	893,871	458,737	166,601
	C (sec.)	7.422	7.184	6.829
Query Window Size w=100 h=100	D (million)	7.036	3.430	1.370
	E (billion)	15.401	12.309	13.623
	F (billion)	1.025	0.997	1.022
	G (sec.)	115.377	53.501	20.253
	H (sec.)	595.061	462.516	519.809
	I (sec.)	36.723	34.813	46.260
	K (sec.)	754.583	558.014	593.151
Query Window Size w=200 h=200	D (million)	20.015	9.850	3.752
	E (million)	29.312	25.848	27.078
	F (billion)	1.377	1,290	1.313
	G (sec.)	171.317	80.353	30.508
	H (sec.)	1,197.532	1,090.533	1,118.527
	I (sec.)	269.790	210.558	227.604
	K (sec.)	1646.061	1388.628	1383.468

(A) - #of pickup locations, (B) - #of CSPT-P Tree quadrants, (C) - CSPT-P Tree construction time (sec.) (D) - #of quadrant-polygon pairs (E)- #of point to polygon edge distance computation (F) - #of resulting records (G) - index query time (sec.) (H) data query time (sec.) (I) - result gathering time (sec.) (K)- total time (sec.) = C+G+H+I

7 CONCLUSION AND FUTURE WORK

In this paper, we reported our design and implementation of U2SOD-DB, a column-oriented, GPU-accelerated, in-memory data management system targeted at large-scale ubiquitous urban sensing origin-destination data. Experiment results show that U^{2}SOD-DB is capable of handling hundreds of millions of taxi-trip records with GPS recorded pickup and drop-off locations and times efficiently.

The accomplished work so far is preliminary in nature when compared to our ultimate goals. First, while we target at the general U^{2}SOD data when we abstract data models, design operations and develop efficient implementations, they may be specific to taxi trip data and thus more generalizations are needed. Second, while a few GPGPU based algorithms have been proposed for efficient implementations of essential functionality of the system, there are quite a few that still rely on existing packages that are serial CPU code. More research efforts are needed to make the system achieve even better performance by fully exploiting GPGPU technologies. Finally, a SQL front end is needed to make the system a true database system. These are left for future work.

REFERENCES

1. Reades, J., Calabrese, F., et al. 2007. Cellular Census: Explorations in Urban Data Collection. Pervasive Computing, IEEE 6(3): 30-38.
2. Calabrese, F., Colonna, M. et al. 2010. Real-Time Urban Monitoring Using Cell Phones: A Case Study in Rome. IEEE Transactions on Intelligent Transportation Systems 12(1): 141-151.
3. Vasconcelos, M. A., Ricci, S., et al. 2012. Tips, Dones and Todos: Uncovering User Profiles in Foursquare. Proceedings of the fifth ACM International Conference on Web Search and Data Mining.
4. Calabrese, F., Kloeckl, K., et al. 2008. WikiCity: Real-time Location-sensitive Tools for the City. In Handbook of Research on Urban Informatics: The Practice and Promise of the Real-Time City (Foth, M. eds) 390-413. IGI Global.
5. Friedland, G., Choi, J., et al. 2011. Video2GPS: A Demo of Multimodal Location Estimation on Flickr Videos. Proceedings of the 19th ACM International Conference on Multimedia.
6. Düntgen, C., Behr, T., et al. 2009. BerlinMOD: A Benchmark for Moving Object Databases. The VLDB Journal 18(6): 1335-1368.
7. Sakr, M. and Güting, R. 2011. Spatiotemporal Pattern Queries. GeoInformatica 15(3): 497-540.
8. Kirk, D. B. and Hwu, W.-M. W. 2010. Programming Massively Parallel Processors: A Hands-on Approach Morgan Kaufmann.
9. Zheng, Y., Liu, Y., et al. 2011. Urban Computing with Taxicabs. Proceedings of ACM UbiComp.
10. Phithakkitnukoon, S., Horanont, T., et al. 2010. Activity-Aware Map: Identifying Human Daily Activity Pattern Using Mobile Phone Data. Proceedings of the First International Conference on Human Behavior Understanding.
11. Zheng, Y., Chen, Y., et al. 2009. GeoLife2.0: A Location-Based Social Networking Service. Proceedings of the 10th Tenth International Conference on Mobile Data Management.
12. Yuan, J., Zheng, Y., et al. 2011. Where to Find My Next Passenger? Proceedings of ACM UbiComp.
13. Xie, R., Ji, Y., et al. (2011). Mining Individual Mobility Patterns from Mobile Phone Data. Proceedings of the 2011 International Workshop on Trajectory Data Mining and Analysis.
14. Shekhar, S. and Chawla S. 2003. Spatial Databases: A Tour Prentice Hall.
15. Harizopoulos, S., Liang, V., et al. 2006. Performance Tradeoffs in Read-optimized Databases. Proceedings of VLDB.
16. Abadi, D. J., Madden, S. R., et al. 2008. Column-stores vs. row-stores: how different are they really? Proceedings of ACM SIGMOD.
17. Hennessy, J. L. and D. A. Patterson (2012). Computer Architecture: A Quantitative Approach. Morgan Kaufmann.
18. Holloway, A. L., Raman, V., et al. 2007. How to Barter Bits for Chronons: Compression and Bandwidth Tradeoffs for Database Scans. Proceedings of the ACM SIDMOD.
19. Bakkum, P. and Skadron K., 2010. Accelerating SQL Database Operations on a GPU with CUDA. Proceedings of GPGPUs.
20. Zheng, Y., Li, Q., et al. 2008. Understanding Mobility based on GPS Data. Proceedings of the ACM UbiComp.
21. Chen, C., Gong, H., et al. 2010. Evaluating the Feasibility of a Passive Travel Survey Collection in a Complex Urban Environment: Lessons learned from the New York City Case Study. Transportation Research Part A: Policy and Practice 44(10): 830-840.
22. Thrust Parallel library. http://code.google.com/p/thrust/.
23. http://www.nyc.gov/html/dcp/html/bytes/applbyte.shtml
24. Zhang, J. and Gruenwald, L. 2012. Spatial Indexing of Large-Scale Geo-Referenced Point Data on GPGPUs Using Parallel Primitives. Technical Report. Online available at http://www-cs.ccny.cuny.edu/~jzhang/papers/TR-CSTP.pdf
25. Libspatialindex. http://libspatialindex.github.com/
26. Wang, L., Zheng, Y., et al. 2009. A Flexible Spatio-Temporal Indexing Scheme for Large-Scale GPS Track Retrieval. Proceedings of MDM.

Correspondence Chair: Yu Zheng, Microsoft Research Asia, China
http://research.microsoft.com/en-us/people/yuzheng/

Conference ID - 2012-30939