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, Tsinhua 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.

UrbComp'12, August 12, 2012, Beijing, China.

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.

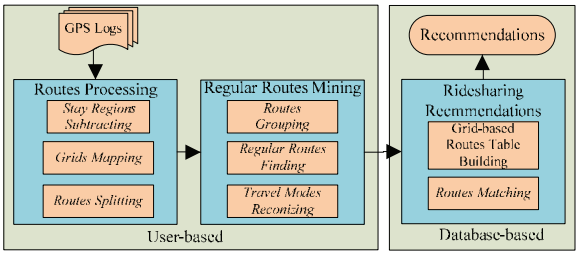


Figure 1. Architecture

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

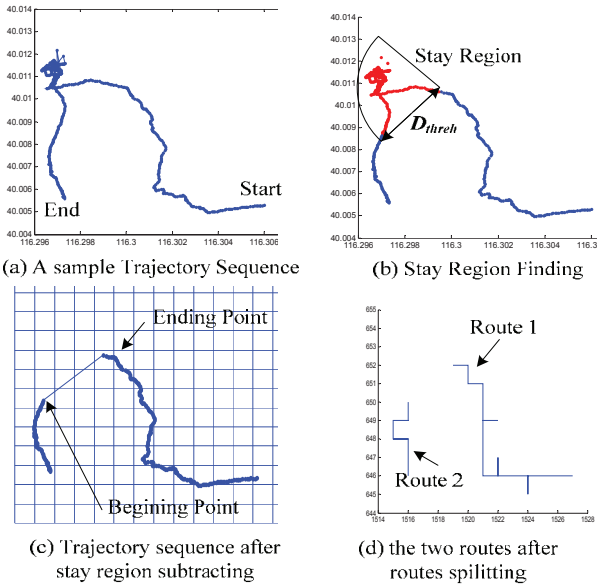
4.1 Routes Processing

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.

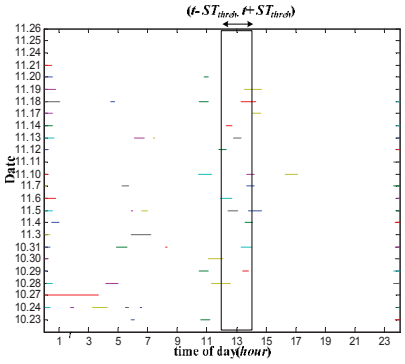
4.2 Regular Routes Mining

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* $tR\{R_m,...R_n\}$, where ($R_i.st$, $R.et$)$\cap$ (t- $SimT_{threh}$, t+ $SimT_{threh}$) $\neq\Phi$, $R_i \in tR$. 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:

$$DE.v = \frac{Distance(TG_m, TG_n)}{TG_n.t_l - TG_m.t_l} \quad (1)$$

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 $DE.sup=\{R_1,R_2,\ldots R_{DE.num}\}$. 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*.

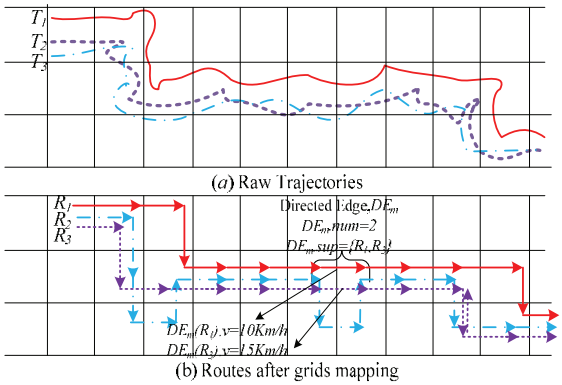


Figure 4. Definitions in a route

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

$$DE.rc = \sum_{i=1}^{DE.num} (f_c(DE.\sup_i) > fc_{threh}) \quad (2)$$

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*)

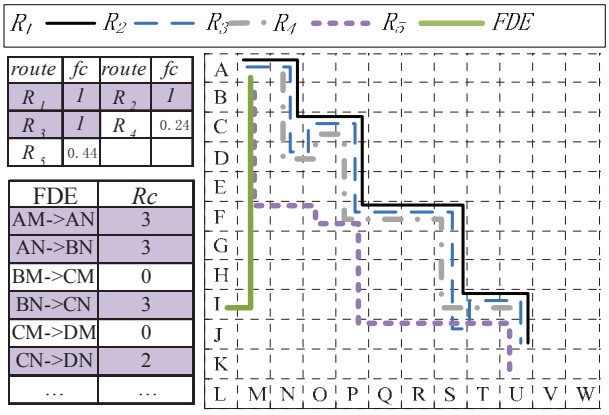
Then a *FDE* has $DE.rc > F_{threh}$ is extracted as an *RFDE,* and an *RR* is the collection of *RFDEs*, which is denoted as $RR\{DE_{m,\ldots,}DE_n\}$, where $DE_i \in RFDEs$, $i = m,\ldots n$. If we set F_{threh} to 2, the *RRs* in Figure 5(a) is $RR\{AM \rightarrow AN$, $BN \rightarrow CN$, $CN \rightarrow DN,\ldots$, $IU \rightarrow JU\}$ 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 Recogning

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:

$$DE.fr = \{DE.sup_n\} \quad if\ (fc(DE.sup_n) > fc_{thre})\ \ n=1,...DE.num$$

Then the stop probability is defined as

$$SP(DE) = \frac{\sum_{i=1}^{DE.rc} ((DE(DE.fr_i).v < V_{thre})}{DE.rc} \tag{5}$$

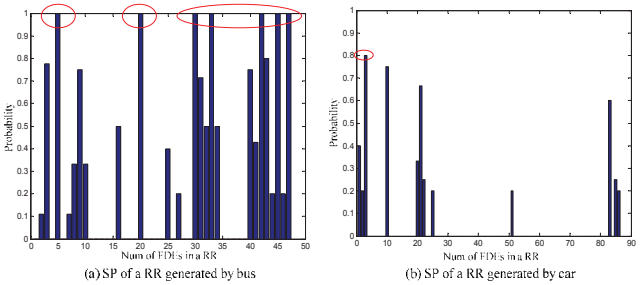
Then an *RFDE* with *SP* lower than P_{stop} is a stop region.

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 $g_i(RR_{m,...}RR_n)$. 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:

$$CR(RR_n) = \frac{Distance(RR_n)}{Distance(RR_i)} \quad (7)$$

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.

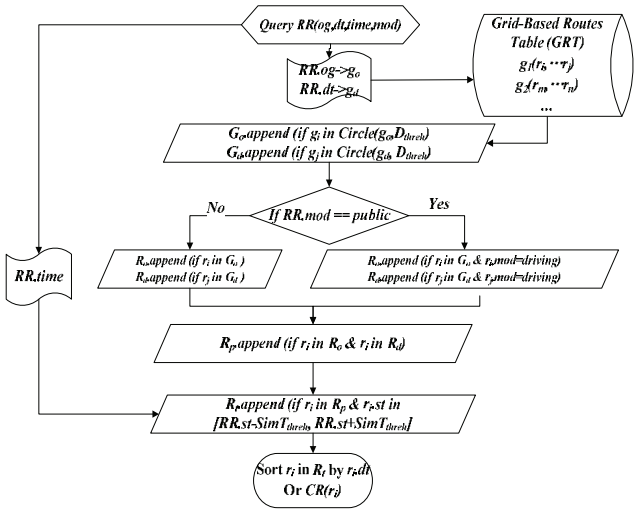


Figure7. Flowchart of finding similar routes

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.

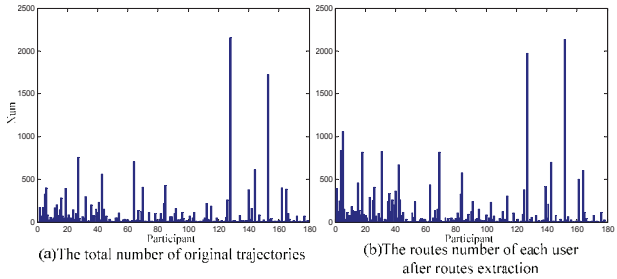


Figure8. The number changing after routes extraction

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.

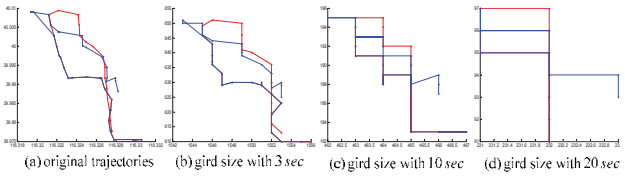


Figure9. Comparison with different size of grids

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).

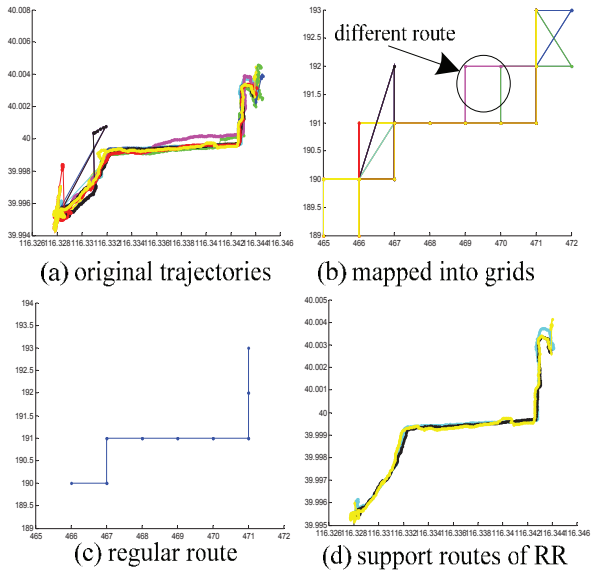


Figure 10. Example of mining *RR*s

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.

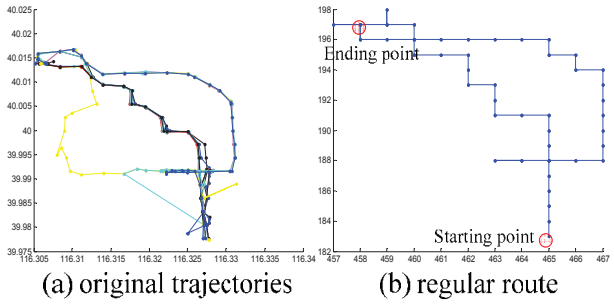


Figure11. Example 2 of mining *RR*s

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 $2Km$ 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*.

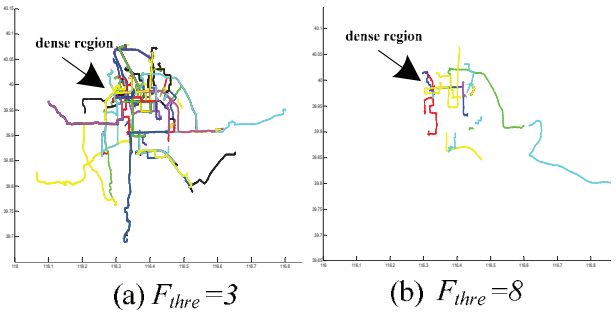


Figure12. Extracted regular routes

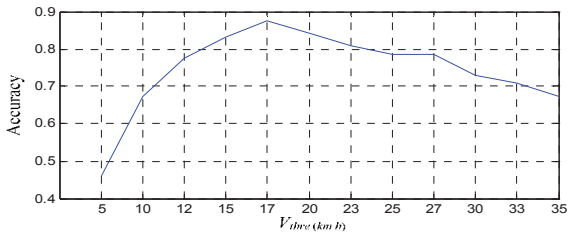


Figure13. Selecting threshold for *FSR*

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.

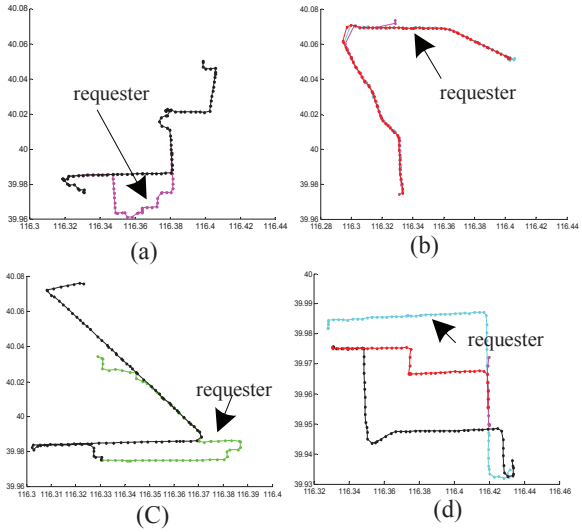


Figure14. Examples in similar routes matching

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