*Information* **2015**, *6*, 134-151; doi:10.3390/info6020134

OPEN ACCESS

*information*

**ISSN 2078-2489**

www.mdpi.com/journal/information

*Article*

# Analysis and Visualization for Hot Spot Based Route Recommendation Using Short-Dated Taxi GPS Traces

**Ying Shen [1,2], Ligang Zhao [1,2,†] and Jing Fan [1,2,†,*]**

[1] School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310014, China; E-Mails: shenying@zjut.edu.cn (Y.S.); strongzhaoligang@126.com (L.Z.)

[2] Key Laboratory of Visual Media Intelligent Process Technology of Zhejiang Province, Hangzhou 310023, China

[†] These authors contributed equally to this work.

[*] Author to whom correspondence should be addressed; E-Mail: fanjing@zjut.edu.cn; Tel.: +86-571-8529-0116; Fax: +86-571-8529-0668.

Academic Editor: Min Yao

*Received: 10 December 2014 / Accepted: 14 April 2015 / Published: 21 April 2015*

**Abstract:** Taxi GPS traces, which contain a great deal of valuable information as regards to human mobility and city traffic, can be extracted to improve the quality of our lives. Since the method of visualized analysis is believed to be an effective way to present information vividly, we develop our analysis and visualization method based on a city's short-dated taxi GPS traces, which can provide recommendation to help cruising taxi drivers to find potential passengers with optimal routes. With our approach, hot spots for loading and unloading passenger(s) are extracted using an improved DBSCAN algorithm after data preprocessing including cleaning and filtering. Then, this paper describes the start-end point-based similar trajectory method to get coarse-level trajectories clusters, together with the density-based ε distance trajectory clustering algorithm to identify recommended potential routes. A weighted tree is defined including such factors as driving time, velocity, distance and endpoint attractiveness for optimal route evaluation from vacant to occupied hot spots. An example is presented to show the effectiveness of our visualization method.

**Keywords:** taxi GPS traces; hot spots; trajectory cluster; recommended route

## 1. Introduction

Urbanization promotes not only economic development but also people's living style; meanwhile, it is associated with problems such as traffic jams. In recent years, a large amount of track data, such as taxi trajectory data, can be obtained with the development of mobile devices. Taxi trajectory data is used to improve taxi cruising planning [1] and analyze the city area that people are interested in [2]. In this paper, we use taxi trajectory data to obtain the hot traveling spots of urban residents, and try to help taxi drivers to find out the optimal path to the nearby hotspots where they can pick up a passenger. This paper adopts an improved clustering algorithm GADBSCAN to generate the hot spots subsequent to preprocessing of the taxi trajectory data. Then, we find out the potential trajectories between hotspots using start-end point based similar trajectory method and the density based epsilon distance clustering algorithm. Finally, in order to obtain the optimal route, we define a weighted tree to evaluate the potential trajectory with such factors as driving time, velocity distance and attractiveness of hot spot for loading passengers. Figure 1 shows the workflow framework of this method.

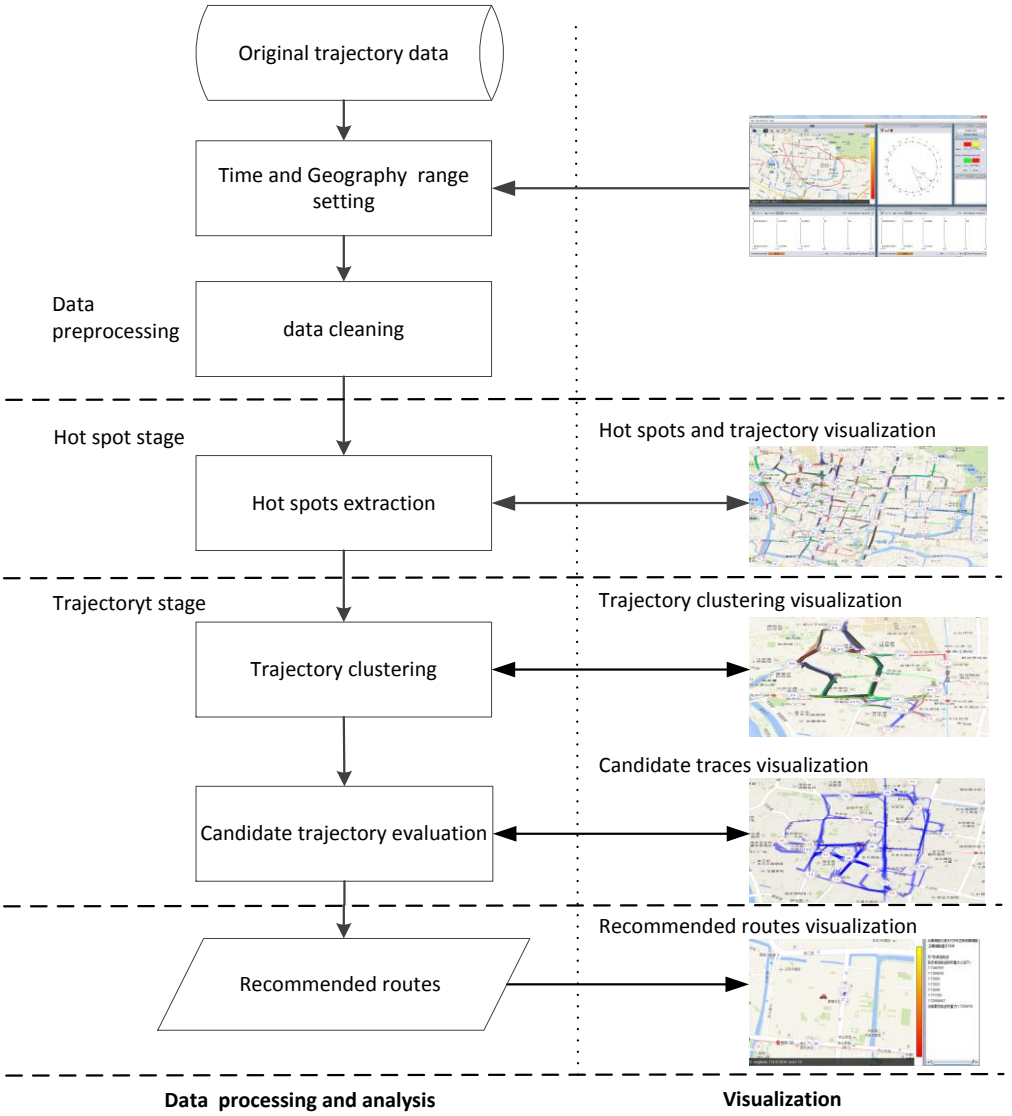



**Figure 1.** The workflow framework.

As regards to the rest of this paper, Section 2 introduces the related work; Section 3 describes the preprocessing of trajectory data; Section 4 presents the hotspots' extraction method; Section 5 demonstrates the trajectory clustering algorithm, while Section 6 shows the approach to evaluating the potential recommended routes for drivers, and presents recommended results using an example with visualization. In the end, Section 7 presents conclusions and prospects.

## 2. Related Work

Taxi GPS data contains a bundle of knowledge and it has been widely applied in various areas. Researchers have been concerned with understanding the mobility patterns of a city's population as well as traffic flow movement and the corresponding benefits for drivers [3]. For example, there are papers [4] focusing on discovering the functionality of different regions in a city. Li *et al.* [5] find the on-off hotspots and build a model of loading probability forecast to help taxi drivers as regard to their routing strategy. Liu *et al.* study the relationship between taxi drivers' revenue and their routing strategies [6]. Zhang [7] proposes effective service strategy with GPS data, while Wang [8] predicts the driving time of different paths based on road network data and track data in real-time. Different from the above studies, which are based on historical data and offline test, Moreira-Matias *et al.* [9] present a short-term prediction model through online test-bed.

Hot spots related research is popular in taxi GPS data mining. Yue [10] focuses on discovering hotspot distribution features. Pan *et al.* generate the hotspots of a city with an iterative DBSCAN algorithm [11] for the feature classification of urban land. Chen *et al.* introduce the application of hot spots in night bus route planning [12].

Trajectory clustering is essential for GPS data processing [13,14]. Yuan proposes an index tree based trajectory clustering method [15] with factors of directional angle, speed, start and end points. Yu *et al.* introduce an efficient and continuous density-based [16] trajectory clustering algorithm. Besides these segment partition methods, GenLIP [17], route similarity [18] and density-based k nearest neighbor trajectory [19] are all related trajectory clustering algorithms.

Our work is similar to [1,6,20], but with a different goal. Our goal is to help vacant taxi drivers find the best route to pick up the next passenger with short-dated taxi dataset and no supporting road network.

## 3. Data Description and Preprocessing

### *3.1. Data Description*

The taxi GPS data used in this paper is downloaded from the website of Datatang [21]. It is generated by about 7600 taxis in the city of Nanjing from September 1–2, 2010. Its sampling time interval ranges from 30–40 s. Data Dimensions of the taxi GPS data are described as follows:

**VehicleSimID**: a unique taxi ID;
**GPSLongtitude**: longitude of current sampling point;
**GPSLatitude**: latitude of current sampling point;
**GPSSpeed**: the current speed of sampling point, in km/h;
**GPSDirection**: current driving direction of sampling point—from 0°–360° clockwise (north direction is 0°);

**PassengerState**: current state of sampling point; "0" indicates "no passenger", and "1" indicates "more than 1 passenger (1 included)";
**CreateData**: current sampling time;
**ReadFlag**: current status of the taxi GPS device, "0" indicates "normal", and "1 indicates "abnormal".

### *3.2. Data Preprocessing*

Due to abnormalities such as GPS device failure, data should be cleaned at first. In order to find out the effective taxi trajectories, data is organized in ascending order by "VehicleSimID" and "CreateData" fields. Then, we extract a taxi trajectory according to the following steps:

**Step 1 [Raw Trajectory Extraction]**: Based on the field of "PassengerState", each unique vehicle's sample points can be assembled as a sequence like (1). We define "passenger on event" as a shift from 0–1, and "passenger off event" as a shift from 1–0, which are also called "occupied event" and "vacant event". An "occupied trajectory" is defined as a point sequence beginning with occupied event and ending with vacant event, while the "vacant trajectory" begins with vacant event and ends with occupied event. The trajectory is represented by *TR*, and the *i*th sampling point is represented by $Tr_i$.

$$..0\ \underbrace{1111111}_{\text{occupied } trajectory}\ 01...1\overbrace{0000000001}^{vacant\ trajectory}1... \quad (1)$$

**Step 2 [Abnormal Filtering]**: During this stage, abnormal events mainly including flipping, parking and over-short trajectory are filtered from raw trajectory. Figure 2 shows two examples of these situations that need to be filtered.

Flip trajectory filtering: A flip detection function is defined as:

$$H(Tr_i) = \begin{cases} true & v_{i,i-1} \le V_T \wedge v_{i,i+1} \le V_T \\ false & v_{i,i-1} > V_T \vee v_{i,i+1} > V_T \end{cases} \quad (2)$$

In Equation (2), $V_T$ is a speed threshold value. And $v_{i,i+1}$ is calculated as the Manhattan distance divided by time interval between point *i* and point (*i+1*). Given a trajectory *TR*, for each sampling points $Tr_i$, if $H(Tr_i)$ equals false, then $Tr_i$ is called a "flip point". All the flip points are filtered from raw trajectory.

- Parking sub trajectory filtering: Given a trajectory *TR*, if it meets $TR = (Tr_1\ Tr_2 ...\ Tr_n) | \exists\ i, j, 1 < i < j \le n, d(Tr_i, Tr_j) \le D_T \wedge \Delta T_{i,j} > T$, then *TR* is called "a parking trajectory", here $d(Tr_i, Tr_j)$ is a Manhattan distance between point $Tr_i$ and $Tr_j$, $\Delta T_{i,j}$ is the driving time from $Tr_i$ to $Tr_j$. We set $D_T$ to 50 m, *T* to 5 min.
- Short trajectory filtering: Delete all the trajectories with less than five sampling points.


(**a**) (**b**)

**Figure 2.** (**a**) flip trajectory example; (**b**) parking trajectory example.

## 4. Hot Spot Extraction

The DBSCAN algorithm [22] can discover density-based clusters with an arbitrary shape in a noise spatial database, which takes a cluster as the max density-reachable point set. DBSCAN with proper parameters has greater advantage in dealing with outliers and noises than pure partition-based clustering or hierarchical clustering [23]. However, it works poorly in areas with sparse vacancy or occupied events. The taxi GPS data set is typically a noisy space data with different sizes and shapes [24], which is particularly suited for DBSCAN. Since DBSCAN is sensitive with parameters [11], this paper improves it by the idea of grid partition [2] and introduces average density threshold to adaptively select parameters for clusters in each grid. The algorithm to generate hotspots is called GADBSCAN.

**Definition 1**: A passenger-loading (unloading) point set $\boldsymbol{S_{Tr}}$ is a set containing all the points meeting the vacant (occupied) event in trajectory.

**Definition 2**: For a loading(unloading) point $Tr_i$, its **Δ-neighborhood** $\boldsymbol{N_\Delta(Tr_i)}$ is defined as a point set satisfying the situation that the distance between each point in the set t and $Tr_i$ is no more than Δ.

**Definition 3**: For a passenger loading (unloading) guest point set $S_{Tr}$, the average density threshold, which is denoted as ***avgMinPts***, is calculated by summing element number in the Δ-neighborhood of each point, and dividing with count of $S_{Tr}$, as Equation (3).

$$avgMinPts(S_{Tr}) = \frac{\sum_{i=1}^{n} |N_\Delta(Tr_i)|}{|S_{Tr}|} \quad Tr_i \in S_{Tr} \tag{3}$$

**Definition 4**: A loading (unloading) hot spot set $\boldsymbol{HS_{Tr}}$ is a non-empty subset of loading (unloading) point $S_{Tr}$, and each point pair is density-reachable or density-connect with respect to $N_\Delta(Tr_i)$ and *avgMinPts*, which is output by typical DBSCAN algorithm.

With the above definitions, a Grid Adaptive DBSCAN (GADBSCAN) algorithm is designed to make DBSCAN fit for Taxi trajectory. Its critical parameters are radius *d* and density threshold *MinPts*, and *d* is a distance used to delimiting the neighborhood while *MinPts* is a density measure that indicates the amount of points needed in a neighborhood of the point in order to assign the point and its neighbors to a cluster. The algorithm is described as follows:

Algorithm: GADBSCAN
Input: area A, $S_{Tr}$, the minimum area radius *d*, density threshold *MinPts*
Output: Hot Spot Set $HS_{Tr}$
1: Subdivides A into small grids gs as each one is 1 km × 1 km, set $HS_{Tr}$ for each grid empty
2: Subdivides $S_{Tr}$ by allocated each point in its specific grid, get $\{S_{Tr,\,gs[i][j]} | each\ point\ in\ S_{Tr,\,gs[i][j]}$ located in grid gs[*i*][*j*]}
3: FOR each subdivides $S_{Tr,\,gs[i][j]}$ of $S_{Tr}$
4: calculate $averMinPts_{gs}$
5: IF ($averMinPts_{gs} \leq MinPts$)
6: $HS_{Tr,\,gs[i][j]}$. add (DBSCAN(*gs*, *d*, $averMinPts_{gs}$))
7: ELSE $HS_{Tr,\,gs[i][j]}$ add (DBSCAN(*gs*, *d*, *MinPts*))
8: Joins density-connected clusters in adjacent grids to assemble all the subdivision to get $HS_T$

Figure 3 shows the clustering results of the above method. Circles with red numbers in them represent loading-hot spot while circles with blue numbers in them are unloading-hot spot. By the aid of visualization, hot spot distribution can be observed and explored with different parameters such as Δ and density threshold *MinPts*.

At the end of this stage, loading (unloading) hot spot distribution is presented in visualization especially with map and timeline tools. The behavior patterns of people travelling in rush hours, day and night, work day and weekend will also be reflected in the hot spot distribution.

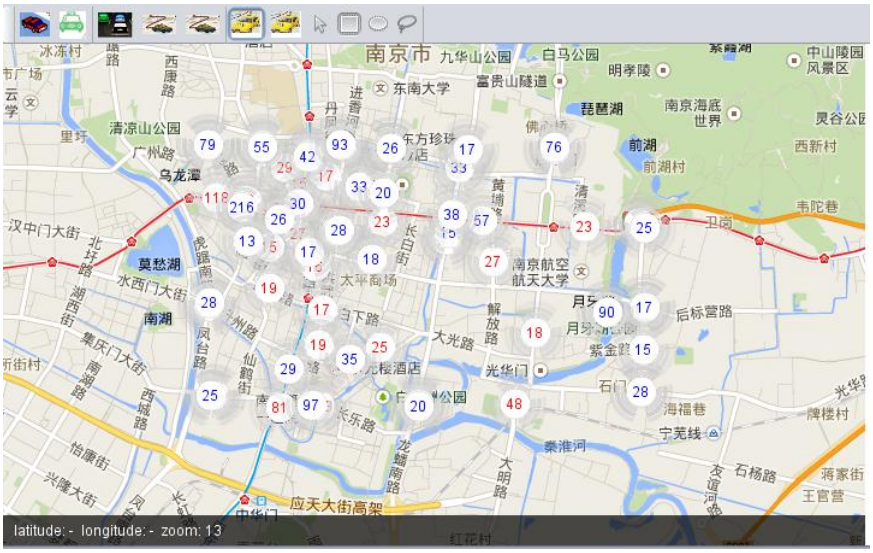



**Figure 3.** Hot spot clustering results.

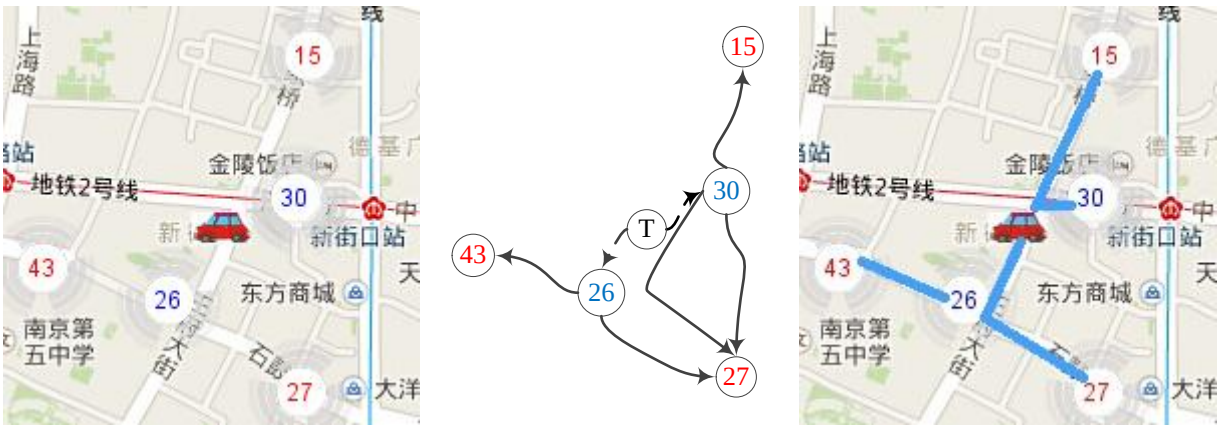
## 5. Trajectory Clustering

Taxi drivers are always eager to find the next passenger(s) instantly after their current passenger(s) is(are) unloaded. Since the loading (unloading) hot spot means there are more passengers loading (unloading) in that spot, if vacant taxi drivers go to the nearest loading hot spot at the very beginning, they intuitively have a higher chance of getting passengers.

The model for taxi in arbitrary position to loading hot spots is similar to traditional driving route recommendation models. This is usually complex in computing, and needs the support of a road network and extensive historical data. On the other side, short-day taxi data is too sparse to support the calculation from an arbitrary position to loading hot spots. When a vacant event happens, a taxi has greater possibility to stop near an unloading hot spot, so establishing a model from unloading hot spot to loading hot spot nearby may simplify the process of making a route recommendation for cruising drivers.

In this paper, we build such a hot spot-based model by extracting and clustering trajectories through loading and unloading hot spots in a specific area and period. The hot spot based clustered trajectories reflect situations on the main roads in the city. Learning from the short-dated taxi data with referenced time, velocity, distance and up spot attractiveness, we turn to evaluate potential routes from taxi position to the loading hot spot nearby. Although such a simplified model may have lower accuracy than that of the traditional route recommendation models, it works easily and effectively for short route recommendation and does not need road network information but only short-term taxi data.

Figure 4 explains our model with an instance of a vacant taxi hunting for new passengers. Figure 4a describes a scene that a taxi becomes vacant at the moment. There are some hot spots near the current taxi position. Then, a hot spot based model can be constructed in Figure 4b. The map is abstracted as a unidirectional graph with unloading (blue) hot spot node to adjacent loading (red) hot spot node. The solid curves represent clustered trajectory which can be calculated; "T" represents current taxi position and the dashed curves represent the simple distance estimation in this model. Then, the problem turns to retrieving and finding the best adjacent routes. It suggests that the taxi is driven to a nearby unloading (blue) hot spot (usually no more than 1000 m), then to the next loading (red) hot spot, which gives the driver greater probability of finding new passengers. The route can be divided into two parts: the first part is the taxi to the blue point, which is marked with dotted curves such as "T" to "26", and the second part is the trajectories from blue point to red spot marked with solid curves such as "26" to "43". Trajectory clustering algorithm will help to find optimal routes such as the ones in Figure 4c. For example, the taxi can go to "43" through "26", or go to "27" through "26". In our models, the taxi can be guided to the hotter red spots with factors such as distance, driving time, and road congestion considered simultaneously.




(**a**) (**b**) (**c**)

**Figure 4.** (**a**) Taxi hunting for loading hot spot; (**b**) Hot spot based model; (**c**) Trajectory clustering.

The routing process can be divided into two stages. At first, sub trajectories starting from unloading hot spots to loading hot spots are extracted with the start-end point based similar trajectory clustering algorithm. Then, the clustering method of density based epsilon distance trajectory is used to find out potential routes.

### *5.1. Start-End Point Based Similar Trajectory Clustering*

Since our simplified model only focuses on the trajectories from unloading hot spots to loading hot spots, all the trajectories will be retrieved to extract sub trajectories from unloading hot spot to its nearby loading hot spots based on the output of hot spots set in Section 4. Then, we can focus on coarse level trajectory clustering.

Based on the idea of paper [13], we propose a start-end point based similar trajectory method with which we can get a series of trajectory clusters that have the nearby start and end points. One pair of trajectories is compared each time with this method. Figure 5 shows factors in the calculation of distance between two trajectories.

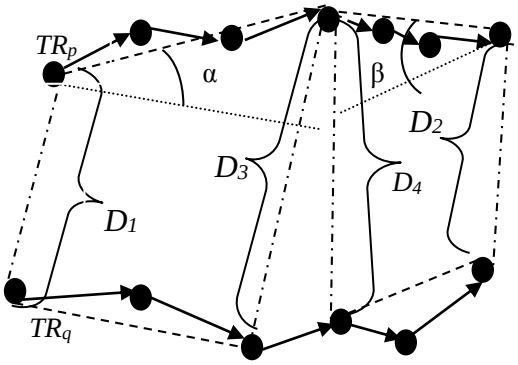



**Figure 5.** Distance metric for two trajectories with similar start-end point.

In Figure 5, $TR_p$ and $TR_q$ represent two trajectories, $D_1$, $D_2$, $D_3$, $D_4$ represent Euclidean distance and α, β represent angle. $D_1$ is the distance between the two start points of trajectory $TR_p$ *and* $TR_q$, and $D_2$ is the distance between the two end points of the trajectories, while $D_3$ and $D_4$ are the two midpoints distance of the trajectories marked as d ($TR_{pc}$, $TR_{qc}$). It should be especially stressed that if $TR_p$ and $TR_q$ contains an odd number of sampling points, there is only one middle distance $D_3$, which is d ($TR_{pc}$, $TR_{qc}$). Otherwise, there are two middle distances ($D_3$ and $D_4$). d ($TR_{pc}$, $TR_{qc}$) can be calculated with Equation (4).

$$d(TR_{pc}, TR_{qc}) = \begin{cases} D_3 & \mathrm{TR_p\ and\ TR_q\ both\ have\ odd\ number\ of\ points} \\ (\mathrm{D_3 + D_4})/2 & \mathrm{TR_p\ or\ TR_q\ has\ even\ number\ of\ points} \end{cases} \quad (4)$$

Angle α is the angle between two start-midpoint dashed line segments of $TR_p$ and $TR_q$, which reflects the degree of divergence between the two trajectories, while angle β is the angle between two midpoint-end dashed line segments, which reflects the degree of convergence.

We marked "GPSDirection" angle of start, middle and end sampling points on $TR_p$ and $TR_q$ as $\theta^p{}_s$, $\theta^p{}_c$, $\theta^p{}_e$, $\theta^q{}_s$, $\theta^q{}_c$, $\theta^q{}_e$, then α and β can be calculated with Equations (5) and (6); here, angle of middle sampling point should be handled as distance.

$$\alpha = \left| \left( \theta^{q}{}_{c} - \theta^{q}{}_{s} \right) - \left( \theta^{p}{}_{c} - \theta^{p}{}_{s} \right) \right| \tag{5}$$

$$\beta = \left| \left( \theta^{q}{}_{e} - \theta^{q}{}_{c} \right) - \left( \theta^{p}{}_{e} - \theta^{p}{}_{c} \right) \right| \tag{6}$$

In our method, distance between two trajectories is defined with Equation (7).

$$d(TR_p, TR_q) = \omega_1 \times D_1 + \omega_2 \times D_2 + \omega_3 d(TR_{pc}, TR_{qc}) \tag{7}$$

$d\ (TR_{pc}, TR_{qc})$ represents the mid-point distance between $TR_p$, and $TR_q$, can be calculated as above. $\omega_1$ and $\omega_2$ are similar as those in paper [16], calculated with Equations (8) and (9); $\omega_3$ is set at 1/3.

$$\omega_1 = \begin{cases} \frac{1}{3}[1 - \frac{1}{2}\sin\alpha - \frac{1}{2}\sin(|\alpha - \beta|)] & \alpha \in \left[0, \frac{\pi}{2}\right) \\ \frac{1}{3}(1 + \frac{1}{2}\sin\alpha + \frac{1}{2}\sin(|\alpha - \beta|)) & \alpha \in \left[\frac{\pi}{2}, \pi\right] \end{cases} \tag{8}$$

$$\omega_2 = \begin{cases} \frac{1}{3}[2 + \frac{1}{2}\sin\alpha + \frac{1}{2}\sin(|\alpha - \beta|)] & \alpha \in \left[0, \frac{\pi}{2}\right) \\ \frac{1}{3}[2 - \frac{1}{2}\sin\alpha - \frac{1}{2}\sin(|\alpha - \beta|)] & \alpha \in \left[\frac{\pi}{2}, \pi\right] \end{cases} \tag{9}$$

When $d\ (TR_p, TR_q)$ is less than the threshold $t$ (here $t$ is 0.8 km), the two trajectories can be merged into one category.

Figure 6a shows 45 trajectories of a taxi, with occupied ones shown in red and vacant ones shown in blue. Figure 6b shows the results of the above start-end point based similar trajectory method with a distance threshold of 0.8 km, which outputs four sub trajectory sets with different colors. We can see that the green cluster has a similar endpoint but two different routes. That is the reason why we should continue to work on a finer trajectory clustering.


(**a**) (**b**)

**Figure 6.** (**a**) Forty-five original trajectories; (**b**) four sub trajectory sets after clustering ($t$ = 0.8 km).

### *5.2. Epsilon Distance Trajectory Clustering*

The distance calculation of two trajectories in our paper is based on the idea of [18], which is done through repeated scanning of two trajectories to find the closest pair of positions, then by computing the mean distance of the corresponding positions and penalty distance. We introduce current driving angle

to computing in order to shorten the scan range of trajectories in similar directions, and to eliminate the restriction that trajectories should have the same number of sampling points in a trajectory cluster.

Given the two trajectories *TR* and *TR'*, if their angle difference is within the threshold, then we go on to traverse all the points in them. The algorithm of improved distance calculation is described below:

Function: Distance between two trajectories
Input: trajectory *TR*, *TR'*, distance threshold *dThred*, angle threshold θ
Output: The $D_{pq}$ distance between *TR* and *TR*'

1: $D_{pq}$, Codistance, Conum = 0; $i = j = 1$
2: Δα←|*TR*. start angle-*TR*'. start angle|
3: IF Δα > θ RETURN 2 × *dThred*
4: ELSE
5: WHILE ($i < TR.length \wedge j < TR'.length$)
6: d←d($TR_i$, $TR'_j$)
7: WHILE ($i + 1 < TR.length \wedge d(TR_{i+1},TR'_j) < d$)
8: $D_{pq}$ += $d(TR_{i+1},TR_i)$
9: *i*++
10: d←d($TR_i$,$TR'_j$)
11: WHILE ($j + 1$ < TR'.*length* ∧ d($TR_i$,$TR'_{j+1}$) < *d*)
12: $D_{pq}$ += d($TR'_j$,$TR'_{j+1}$)
13: *j*++
14: *d*←*d*($TR_i$,$TR'_j$)
15: Codistance += *d*; // find the nearby point
16: Conum++;
17: IF $D_{pq}$/Conum > *dThred*
18: RETURN $D_{pq}$ + *dThred*
19: $D_{pq}$ −= Codistance; Codistance /= Conum
20: *i*++;*j*++
21: WHILE ($i \leq TR.length$)
22: $D_{pq}$ += $d(TR_{i+1},TR_i)$
23: *i*++;
24: WHILE ($j \leq TR'.length$)
25: $D_{pq}$ += d($TR'_j$,$TR'_{j+1}$)
26: j++
27: RETURN $D_{pq}$

**Definition 5:** Given a trajectory set $S_{TR}$ and $d_\varepsilon(TR_0)$, the neighbor set of the epsilon distance of $TR_0$ is denoted as $N_{d\varepsilon}$ ($TR_0$) which meets:

$$N_{d\varepsilon}(TR_0)=\{TR|TR \in S_{TR}\text{-}\{TR_0\} \wedge d(TR,TR_0) \leq \varepsilon)\} \quad (10)$$

**Definition 6:** [core trajectory]: Given a trajectory set $S_{TR}$, trajectory density threshold k and the distance threshold ε, if there is $TR_0 \in S_{TR}$ and satisfies $|N_{d\varepsilon}(TR_0)| \geq k$, then $TR_0$ is named a **core trajectory**.

Distance in Definitions 5 and 6 is the distance between two trajectories calculated by the distance function above. The core trajectory and its neighbor set of the epsilon distance are in a trajectory cluster. Figure 7 illustrates the two definitions above. If $k = 2$, then the red trajectory $TR_0$ can be called a core trajectory since $|N_{d\varepsilon}(TR_0)| \geq 2$.

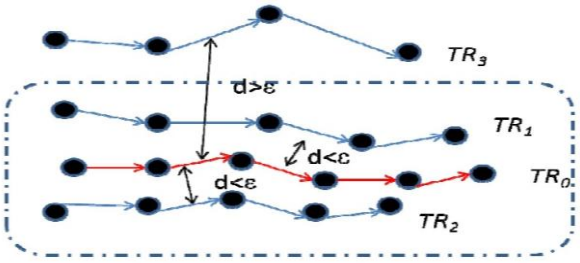



**Figure 7.** The illustration of trajectory cluster.

By using density based ε distance trajectory clustering algorithm, we can find trajectory cluster like {$TR_0$, $TR_1$, $TR_2$} in Figure 7. The algorithm traverses each trajectory $TR_0$ in set $S_{TR}$, constructs its ε neighborhood set $N_{d\varepsilon}$ ($TR_0$), and determines whether $TR_0$ is a core trajectory. If $TR_0$ is a core trajectory, then delete all the trajectories of $N_{d\varepsilon}$ ($TR_0$) from $S_{TR}$, and put them in a trajectory cluster.

Algorithm: Density based ε distance clustering
Input: Trajectory set $S_{TR}$, density threshold $k$, distance threshold $\varepsilon$
Output: Trajectory Clustering Set List *scluster*
1: *scluster* = NULL; item = 1;
2: WHILE ($S_{TR}$ != Φ)
3: FOR each $TR_0$ ϵ $S_{TR}$
4: $N_{d\varepsilon}(TR_0)$.APPEND($TR_0$)
5: FOR each $TR$' in $S_{TR}$-{$TR_0$}
6: IF $distance(TR_0,TR') \leq \varepsilon$
7: $N_{d\varepsilon}(TR_0)$.APPEND ($TR$')
8: IF $|N_{d\varepsilon}(TR_0)| \geq k$ THEN
9: $S_{TR} = S_{TR}$-$N_{d\varepsilon}(TR_0)$
10: $scluster_{[item]}$. APPEND ($N_{d\varepsilon}(TR_0)$)
11: item++
12: ELSE $S_{TR} = S_{TR}$-{$TR_0$}
13: RETURN *scluster*

The result of our density ε based distance clustering algorithm is related to the trajectory order since it uses $TR_0$ as default cluster center. However, if we set a proper value for distance threshold ε within road width, then the cluster result is acceptable.

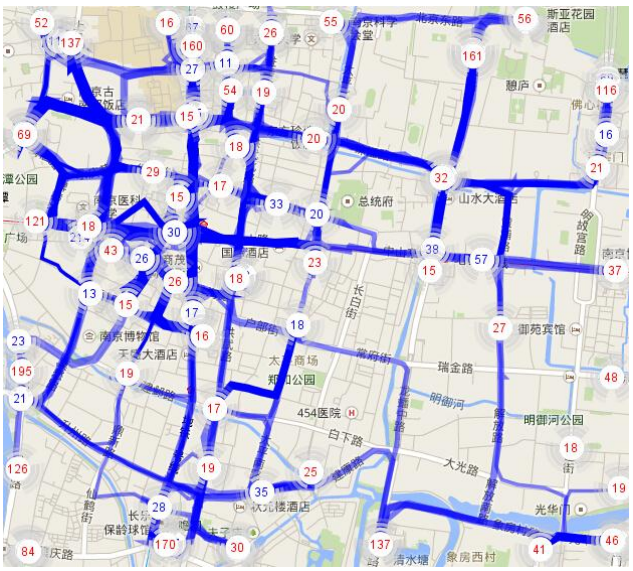



**Figure 8.** Clustered trajectories based on hot spots.

After the two-stage trajectory clustering, we are ready to pick optimal routes for cruising taxi drivers. An example of clustering result is presented in Figure 8. The figure includes 50 loading hot spots, 35

unloading hot spots and 311 potential trajectory clusters, which are extracted from 19861 original trajectories limited in an area of (118.76375,32.02073) to (118.81593,32.05973) during 8:00–9:00 in the morning of September 1, 2010. With wideness of trajectories indicating traffic flow, and darkness of colors indicating driving time cost, it is clear that the west part of the area has much heavier traffic and is busier than the east part in the rush hour.

## 6. Route Recommendation

With the clustered trajectories based on hot spots, a three-layered weighted tree is used to evaluate potential routes (see Figure 9); it is a sub tree with root "T" from the graph similar as Figure 4b. Current cruising taxi position T is represented as the root in the weighted tree. All the unloading hot spots near T are represented as the nodes in the second layer. All the loading hot spots near the nodes in the second layer are represented as the nodes in the third layer, which may have duplicated copies since there may be several paths between a pair of hot spots. Since a route can be represented as a weighted path from T to leaves, we can evaluate each potential route through its corresponding factors.

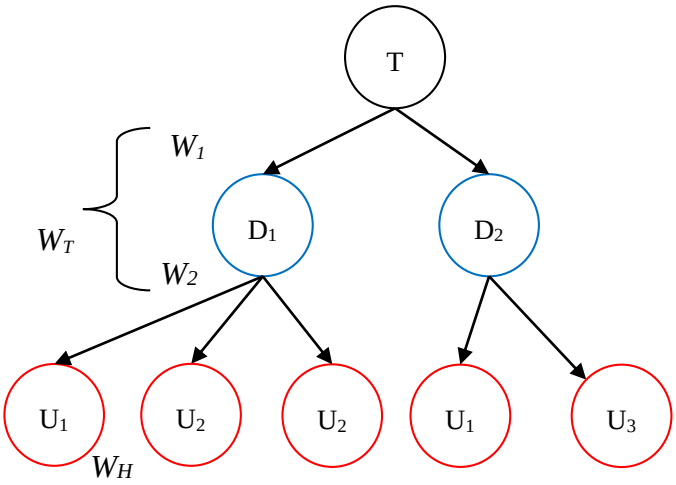



**Figure 9.** A weighted tree for route recommendation.

Here, score of each route is called $W_F$, which contains the path cost factor $W_T$ and the hot-spot attraction factor $W_H$, as shown in Equation (11).

$$W_F = W_T + W_H \tag{11}$$

$W_H$ can be calculated with Equation (12). The parameter $k$ in Equation (12) with empirical range [10,15] is used to adjust $W_H$, and we set $k$ = 11. Here, $H_u$ equals the number of loading-passenger(s) event happening at a loading hot spot, which is the end point of potential route. And $H_{max}$ and $H_{min}$ indicate, respectively, the maximum and minimum number of loading-passenger event happening at a loading hot spot as of current area and time.

$$\mathrm{W_H} = k\frac{H_\mathrm{u} - H_{min}}{H_{max} - H_\mathrm{min}} \tag{12}$$

$W_T$ contains two parts: $W_1$, the value from the root node to the node in the second layer, representing the score of path from the current taxi position to the unloading-hot spot nearby; $W_2$, value from the node in the second layer to its sub nodes in the third layer, representing the score of path from the current unloading hot spot to the loading hotspot nearby. $W_T$ can be described with Equation (13). Here, we set $\omega_1$ to 1/3 and $\omega_2$ to 2/3, since $W_1$ contains one factor, while $W_2$ consists of two factors.

$$W_T = \omega_1 W_1 + \omega_2 W_2 \tag{13}$$

For $W_1$, we focus on the distance factor between the unloading-hotspot and the position of the taxi. In Equation (14), $d_0$ indicates the Euclidean distance between the current position T and the unloading hotspot in this path. $d_{max}$ and $d_{min}$ indicate the Euclidean distance between T and the farthest and closest down-hot spot, respectively.

$$W_1 = (1 - \frac{d_0 - d_{\min}}{d_{\max} - d_{\min}}) \times 100 \tag{14}$$

For $W_2$, we take into consideration the distance factor $A$ and the traffic condition factor $B$ between the current unloading hot spot and the potential loading hot spot in Equation (15).

$$W_2 = \frac{1}{2}(A + B) \times 100 \tag{15}$$

Here, $A$ can also be calculated with Equation (14) in the same way $W_1$ is calculated; all the distances in $A$ are calculated by average time multiplied by average speed for each trajectory cluster; while $B$ is more complicated in demonstrating traffic status. As we know, the trajectory with shorter driving time and higher average speed is more attractive to a taxi driver. We set B as Equation (16). Here, $t_{avg0}$ is calculated with Equation (17), which shows the average driving time of the current trajectory's cluster $Sc$. The parameters $t_{avgmax}$ and $t_{avgmin}$ indicate the maximal and minimal time value of $Sc$, respectively. Similarly, $v_{avg0}$ is the average speed of $Sc$, calculated wit Equation (18). The parameters $v_{avgmax}$ and $v_{avgmin}$ indicate the maximal and minimal value of $Sc$, respectively.

$$B = \frac{1}{2}(1 - \frac{t_{avg0} - t_{avgmin}}{t_{avgmax} - t_{avgmin}} + \frac{v_{avg0} - v_{avgmin}}{v_{avgmax} - v_{avgmin}}) \tag{16}$$

$$t_{avg0} = \frac{\sum_{i=1}^{|S_c|} t_i}{|S_c|} \tag{17}$$

$$v_{\mathrm{avg0}} = \frac{\sum_{i=1}^{|S_c|} \frac{\sum_{j=1,k=j+1}^{n-1} \frac{d_{jk}}{\Delta t_{jk}}}{n-1}}{|S_c|} \tag{18}$$

In Equation (18), $d_{jk}$ represents the Euclidean distance between sampling point $j$ and $k$ of any trajectory in $Sc$, and $\Delta t_{jk}$ represents the time interval between two sampling points, while $n$ represents the number of the trajectories at the sampling point.

Figure 10 presents a case study of the potential route recommendation. Figure 10a shows 116 potential routes around a cruising taxi. All the potential routes are evaluated by Equation (9). The top seven routes are drawn in Figure 10b with their scores listed in Table 1. The first is the recommended route of our system.

Unloading hot spot "30" is the only competitive unloading hot spot at the current position of the taxi, because all routes with the seven highest scores go through it (Table 1).


(**a**) (**b**)

**Figure 10.** (**a**) 116 potential routes around a taxi; (**b**) the top seven routes.

**Table 1.** Scores for the top seven routes.

| Routes Between Hot Spots | 43 | 27–1 | 27–2 | 87 | 120 | 18 | 15(bottom) |
|---|---|---|---|---|---|---|---|
| **30** | 89.66 | 88.61 | 87.06 | 87.05 | 86.65 | 86.41 | 83.44 |

From Figure 10b, we can see that almost all the loading hot spots near "30" have been included in the top seven routes except for loading hot spot "15(loading)". The potential route from "30" to "15(loading)" has the obvious advantage with the shortest distance, and it only got a score of 66.64. Because of the one-way rule, the real route from "30" to "15(loading) has very long distance that goes through unloading hot spot "30", loading hot spot "18", and loading hot spot "29", which means our method can mostly extract the correct routes.

Figure 11 explores details of the routes with highest scores through visualization. All the routes are ordered descendingly by score, which means Figure 11a is the first recommended route.

In Figure 11, trajectories in darker color are of lower average velocity, and wider lines mean more trajectories in the cluster. As we can see, Figure 11b shows the shortest route; (a) and (c) also have advantage at a shorter distance. The score is not only related with the distance, but also with other factors like hot spot attractiveness, driving time and speed. Figure 11a is the winner because of its overall advantages in speed, distance and attractiveness compared with the other routes. The broad width of trajectory cluster means that the route in Figure 11a is the favorite route for taxi drivers.

Figure 11b,c represent two routes from "30" to "27"; (b) distinguishes itself as shorter distance and (c) has a little higher velocity. We evaluate (b) as being better than (c) and more taxi drivers agree with this because line (b) is wider than (c). Figure 11d,e are typical in attractiveness. Figure 11e is both the fastest in traffic speed and longest in distance among the top seven. Figure 11f,g lose their scores with less attractiveness and longer distance.

Above all, our evaluation provides insightful information for taxi drivers to find suitable routes for subsequent passengers.

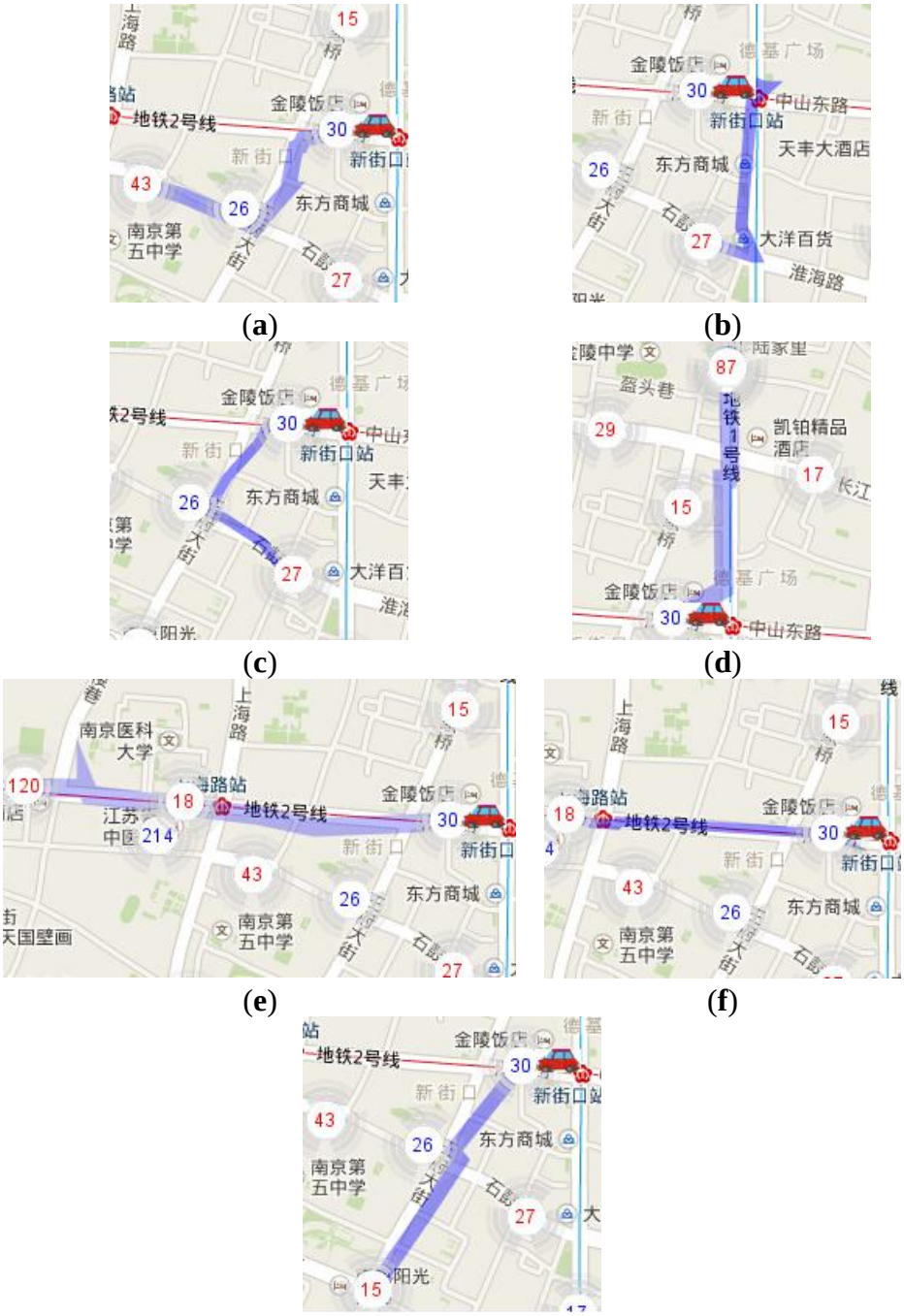



**Figure 11.** (**a**) "30"–"43"; (**b**) "30"–"27"-1; (**c**) "30"–"27"-2; (**d**) "30"–"87"; (**e**) "30"–"120"; (**f**) "30"–"18"; (**g**) "30"–"15(bottom)".

## 7. Conclusions

This paper presents a recommended method for finding optimal routes for drivers of vacant taxis to pick up new passenger(s) using short-dated trajectory data. A workflow of taxi GPS data processing and analysis is designed with each stage supported by visualization. An improved DBSCAN algorithm is integrated in hot spot extraction. A similar start-end point-based trajectory method at the first stage and a density-based ε distance method at the refined stage are used in our trajectory clustering. Weighted tree based route evaluation is defined including the factors of distance, driving time, velocity and end point attractiveness. A case study is then done to verify the whole analytical process with the most suitable routes recommended in the end. More work needs be done to validate our method with a larger data set, and to find proper learning techniques in hotspot extraction, as well as to improve evaluation accuracy in recommending the optimal routes to pick up passenger(s) with the help of the temporal and special distribution pattern of hot spots.

## Acknowledgments

This paper is supported by Key Science and Technology Project of Zhejiang province (No. 2013C01112). The authors would like to thank Datatang for the data supplied. We also thank the anonymous reviewers for their valuable comments and suggestions to improve this work.

## Author Contributions

Shen Ying designed the research, edited and revised the manuscript. Zhao Ligang performed the research, analyzed the data and prepared the manuscript. Fan Jing integrated the entire study through designing, revising and approving the manuscript. All authors have read and approved the final manuscript.

## Conflicts of Interest

The authors declare no conflict of interest.

## References

1. Yuan, J.; Zheng, Y.; Zhang, C.; Xie, W.; Xie, X.; Sun, G.; Huang, Y. T-drive: Driving directions based on taxi trajectories. In Proceedings of the 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, San Jose, CA, USA, 2–5 November 2010; ACM: New York, NY, USA; pp. 99–108.
2. Li, X.; Pan, G.; Wu, Z.; Qi, G.; Li, S.; Zhang, D.; Zhang, W.; Wang, Z. Prediction of urban human mobility using large-scale taxi traces and its applications. *Front. Comput. Sci.* **2012**, *6*, 111–121.
3. Castro, P.; Zhang, D.; Chen, C.; Li, S.; Pan, G. From taxi GPS traces to social and community Dynamics: A Survey. *ACM Comput. Surv. (CSUR)* **2013**, *46*, doi:10.1145/2543581.2543584.
4. Yuan, J.; Zheng, Y.; Xie, X. Discovering regions of different functions in a city using human mobility and POIs. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing, China, 12–16 August 2012; ACM: New York, NY, USA, 2012; pp. 186–194.

5. Li, B.; Zhang, D.; Sun, L.; Chen, C.; Li, S.; Qi, G.; Yang, Q. Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset. In Proceedings of the 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), Seattle, WA, USA, 21–25 March 2011; IEEE: New York, NY, USA; 2011; pp. 63–68.
6. Liu, S.; Liu, Y.; Ni, L.M.; Fan, J.; Li, M. Towards mobility-based clustering. In Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 25–28 July 2010; ACM: New York, NY, USA, 2010; pp. 919–928.
7. Zhang, D.; Sun, L.; Li, B.; Chen, C.; Pan, G.; Li, S.; Wu, Z. Understanding Taxi Service Strategies from Taxi GPS Traces. *IEEE Transa. Intell. Transp. Syst.* **2014**, *16*, 123–135.
8. Wang, Y.; Zheng, Y.; Xue, Y. Travel Time Estimation of a Path using Sparse Trajectories. In Proceedings of the 20th SIGKDD Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014.
9. Moreira-Matias, L.; Gama, J.; Ferreira, M.; Mendes-Moreira, J.; Damas, L. Predicting taxi-passenger demand using streaming data. *IEEE Trans. Intell. Transp. Sys.* **2013**, *14*, 1393–1402.
10. Yue, Y.; Zhuang, Y.; Li, Q.; Mao, Q. Mining time-dependent attractive areas and movement patterns from taxi trajectory data. In Proceedings of the 2009 17th International Conference on Geoinformatics, Fairfax, VA, USA, 12–14 August 2009; IEEE: New York, NY, USA; pp. 1–6.
11. Pan, G.; Qi, G.; Wu, Z.; Zhang, D.; Li, S. Land-use classification using taxi GPS traces. *IEEE Trans. Intell. Transp. Syst.* **2013**, *14*, 113–123.
12. Chen, C.; Zhang, D.; Li, N.; Zhou, Z. B-Planner: Planning Bidirectional Night Bus Routes Using Large-scale Taxi GPS Traces. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 1451–1465.
13. Lee, J.G.; Han, J.; Whang, K.Y. Trajectory clustering: A partition-and-group framework. In Proceedings of the 2007 ACM SIGMOD International Conference on MANAGEMENT of Data, Beijing, China, 11–14 June 2007; ACM: New York, NY, USA, 2007; pp. 593–604.
14. Chen, Y.; Shen, H.; Tian, H. Clustering Subtrajectories of Moving Objects Based on a Distance Metric with Multi-dimensional Weights. In Proceedings of the 2014 Sixth International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), Beijing, China, 13–15 July 2014; IEEE: New York, NY, USA, 2014; pp. 203–208.
15. Yuan, G.; Xia, S.; Zhang, L.; Zhou, Y.; Ji, C. An efficient trajectory-clustering algorithm based on index tree. *Trans. Inst. Meas. Control* **2011**, doi:10.1177/0142331211423284.
16. Yu, Y.; Wang, Q.; Wang, X. Continuous clustering trajectory stream of moving objects. *Communications (China)* **2013**, *10*, 120–129.
17. Pelekis, N.; Andrienko, G.; Andrienko, N.; Kopanakis, I.; Marketos, G.; Theodoridis, Y. Visually exploring movement data via similarity-based analysis. *J. Intell. Inf. Syst.* **2012**, *38*, 343–391.
18. Andrienko, G.; Andrienko, N.; Wrobel, S. Visual analytics tools for analysis of movement data. *ACM SIGKDD Explor. Newsl.* **2007**, *9*, 38–46.
19. Akasapu, A.K.; Rao, P.S.; Sharma, L.K.; Satpathy, S.K. Density based k-nearest neighbors clustering algorithm for trajectory data. *Int. J. Adv. Sci. Technol.* **2011**, *31*, 47–58.
20. Yuan, J.; Zheng, Y.; Zhang, L.; Xie, X.; Sun, G. Where to find my next passenger. In Proceedings of the 13th International Conference on Ubiquitous Computing, Beijing, China, 17–21 September 2011; ACM: New York, NY, USA; pp. 109–118.

21. Datatang. Available online: http://www.datatang.com/data/44060 (accessed on 20 April 2015). (in Chinese)
22. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, OR, USA, 2–4 August 1996; Volume 96, pp. 226–231.
23. Chang, H.; Tai, Y.; Hsu, J. Context-aware taxi demand hotspots prediction. *Int. J. Bus. Intell. Data Min.* **2010**, *5*, 3–18.
24. Gui, Z.; Yu, H. Mining traffic hot spots from massive taxi trace. *J. Comput. Inf. Syst.* **2014**, *10*, 2751–2760.