T-Finder: A Recommender System for Finding Passengers and Vacant Taxis

Nicholas Jing Yuan, Yu Zheng, Liuhang Zhang, Xing Xie

Abstract—This paper presents a recommender system for both taxi drivers and people expecting to take a taxi, using the knowledge of 1) passengers' mobility patterns and 2) taxi drivers' picking-up/dropping-off behaviors learned from the GPS trajectories of taxicabs. First, this recommender system provides taxi drivers with some locations and the routes to these locations, towards which they are more likely to pick up passengers quickly (during the routes or in these locations) and maximize the profit of the next trip. Second, it recommends people with some locations (within a walking distance) where they can easily find vacant taxis. In our method, we learn the above-mentioned knowledge (represented by probabilities) from GPS trajectories of taxis. Then, we feed the knowledge into a probabilistic model which estimates the profit of the candidate locations for a particular driver based on where and when the driver requests the recommendation. We build our system using historical trajectories generated by over 12,000 taxis during 110 days and validate the system with extensive evaluations including in-the-field user studies.

Index Terms—Location-based services, urban computing, recommender systems, trajectories, taxicabs, parking place detection

1 Introduction

Have you ever suffered from waiting a long time for a taxicab? Actually, taxi drivers are also upset when cruising on road surfaces for finding passengers. The vacant taxis cruising on roads do not only waste gas and time of a taxi driver but also generate additional traffic in a city. Thus, how to improve the utilization of these taxis and reduce the energy consumption effectively poses an urgent challenge.

Recently, in many big cities, like New York, Beijing, and Singapore, taxicabs are equipped with GPS sensors for dispatching and safety. Typically, these taxis will report on their present locations to a data center in a certain frequency, e.g., 2 minutes [16]. Besides a geo-position and time stamp, the occupancy information of a taxi is also recorded (using some weight sensor or by connecting a taxi meter with the embedded GPS device). Therefore, a large number of GPS trajectories with occupancy information are being generated everyday. Intuitively, these taxi trajectories contain two aspects of knowledge. One is passengers' mobility, i.e., where and when passengers get on and off a taxi. The other are taxis' pick-up/drop-off behaviors. For example, where high-profit taxi drivers usually go and how they can find passengers quickly.

With these two aspects of knowledge, we present a recommender system for both taxi drivers and passengers using a huge number of historical GPS trajectories of taxis. Specifically, on the one hand, given the geo-position and time of a taxicab looking for passengers, we suggest the taxi driver with a location, towards which he/she is most likely to pick up a passenger as soon as possible and maximize the profit of the next trip. This recommendation helps reduce the cruising (without a fare) time of a taxi thus saves energy consumption and eases the exhaust pollution as well as helps the drivers to make more profit. On the other hand, we provide people expecting to take a taxi with the locations (within a walking distance) where they are most likely to find a vacant taxicab. Using our recommender system, a taxi will find passengers more quickly and people will take a taxi more easily thereby reducing the supply/demand disequilibrium problem to some extent.

The contributions of this work are summarized as follows:

- we propose an approach to detect parking places based on a large number of GPS trajectories generated by taxis, where the parking places stand for the locations where taxi drivers usually wait for passengers with their taxis parked. We devise a probabilistic model to formulate the time-dependent taxi behaviors (picking-up/dropping-off/cruising/parking) and enable a city-wide recommendation system for both taxi drivers and passengers.
- We improve the taxi recommender by considering the time-varying queue length at the parking places; We enhance the passenger recommender by estimating the waiting time on a specified nearby road segment in addition to calculating the probability of finding a vacant taxi. Besides, we develop a bisected clustering algorithm for categorizing the road segments in order to obtain a statistical reasonable result based on sparse data.

- *Nicholas Jing Yuan, Yu Zheng and Xing Xie are with Microsoft Research Asia, Beijing, China, Email:{nichy,yuzheng,xing.xie}@microsoft.com*
- *Liuhang Zhang is with Institute Of Computing Technology, Chinese Academy Of Sciences, Email: zhanglhjostey@gmail.com*

This article is an expanded version of [18], which appeared in Proceedings of the 13th ACM international conference on Ubiquitous computing (UbiComp 2011).

- For both of the taxi recommender and the passenger recommender, we build our model incorporating day of the week and historical weather conditions to tackle the varying pick-up/drop-off behaviors. We perform extensive evaluations including in-the-field user studies to validate our system.

2 Overview

2.1 Preliminary

Definition 1 (Road Segment): A *road segment* r is a directed edge that is associated with a direction symbol $r.dir$ (one-way or bidirectional), two terminal points $r.s$ and $r.e$, road level $r.level$ (e.g., level-0 roads are mainly high-ways), as well as the travel time $r.t$.

Definition 2 (Route): A *route R* is a sequence of connected road segments, i.e., R: $r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_n$, where $r_{k+1}.s = r_k.e$, $(1 \leq k < n)$. The start point and end point of a route can be represented as $R.s = r_1.s$ and $R.e = r_n.e$.

Definition 3 (State): We consider three states for a working taxi: *occupied* ($\mathcal{O}$), *cruising* ($\mathcal{C}$) and *parked* ($\mathcal{P}$), detailed in Table 1. The taxi is *non-occupied* for both the cruising and parked states.

TABLE 1
The states of a taxi

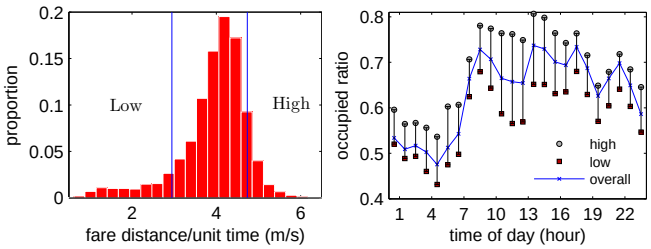
State	Taxi Status
Occupied ($\mathcal{O}$)	A taxi is occupied by a passenger.
Cruising ($\mathcal{C}$)	A taxi is traveling without a passenger.
Parked ($\mathcal{P}$)	A taxi is waiting for a passenger.

Note that the "parked" state proposed in this paper is the status that taxi drivers wait somewhere for business, i.e., stay and/or queue for a while with the intention to get a passenger on-board. This status is frequently found at airports, hotels, shopping centers, etc. We call the places where the taxis are often parked as *parking places*. The parking place here does not merely imply a parking lot for private vehicles (which is the typical definition for the "parking place").

Definition 4 (Trajectory and *Trip):* A taxi *trajectory* is a sequence of GPS points logged for a working taxi, where each point p has the following fields: time stamp $p.t$, latitude $p.lat$, longitude $p.lon$, located road segment (provided by map matching [17]) $p.r$, state $p.s$ (The raw GPS trajectory only indicates whether a point is occupied or non-occupied). A taxi *trip* is a sub-trajectory which has a single state, either cruising (need to be inferred) or occupied. Note that a taxi could generate multiple trips between two parking places.

2.2 Motivation

Different from other public transports like buses or subways, which follow the fixed routes everyday, taxi drivers plan their own routes once they drop off a passenger. This is the main reason that different drivers get discrepant incomes. Figure 1 reveals some statistics w.r.t. 12,000 taxicabs during 110 days. As shown in Figure 1(a), the profit of a taxi driver can be measured by the fare (occupied) distance per unit working time, based on which, we divide the taxi drivers into 3 groups, the top 10% are regarded as high-profit drivers, the bottom



(a) Distribution of profit (b) Occupied ratio during a day

Fig. 1. Motivation based on statistical results

10% are considered as the low-profit drivers and the rests are medium-profit drivers.

There is no doubt that at peak hours, taxicabs more easily find passengers. i.e., the taxis are often in short supply. However, at off-peak hours, the gap between the high-profit drivers and the low-profit drivers becomes obvious. Figure 1(b) further shows the time-variant occupied ratio (the quotient between the occupied distance and the whole distance) pertaining to the high/low-profit taxi drivers as well as the overall occupied ratio changing during a day. It's clear that from 10am to 3pm, the gap between the high-profit drivers and low-profit drivers is more significant. The critical factor determining the profit of a taxi driver depends on two folds. One is that the driver should know the places where he/she can pick up passengers quickly given a particular time of day. The other is the length of the typical trips that originate from a pick-up place. As we know, transportation terminals, shopping centers and hotels all generate demand for taxi service. A professional taxi driver usually knows when certain planes and trains arrive, when the movie is over at a local theater and even what time shifts change at certain businesses.

Typically, for experienced local drivers, instead of random cruising, they usually have a place to go with the intention to pick up new passengers after dropping off a passenger. Figure 2 presents an informative density scatter of the cruising distance per unit time w.r.t. the profit (measured by fare distance per unit time) for the time interval 10am to 3pm, where the color indicates the density of a point. The Pearson correlation coefficient of these two variables is only 0.0874 according to the plotted data. This figure shows us that cruising more does not mean earning more. Instead, waiting at some right places may bring more chance to pick up a passenger. As shown in the figure, quite a few drivers cruise more than the majority (the points on the upper left corner of the hot kernel), however, their profit is lower. The right bottom parts (of the hot kernel) are the drivers who earn more but cruise less than the majority.

2.3 Framework

The framework of the system is illustrated in Figure 3. We develop an approach to detect the parking places (Section 3) from GPS trajectories and segment the GPS trajectories according to Definition 4, then map-match the GPS trajectories to road networks using the IVMM algorithm [17], which

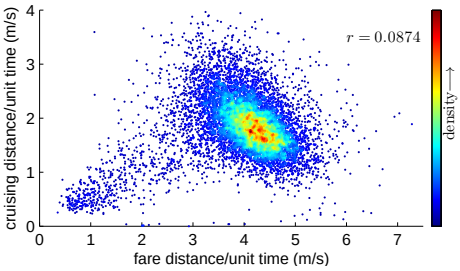


Fig. 2. Density scatter of cruising distance/unit time w.r.t. profit

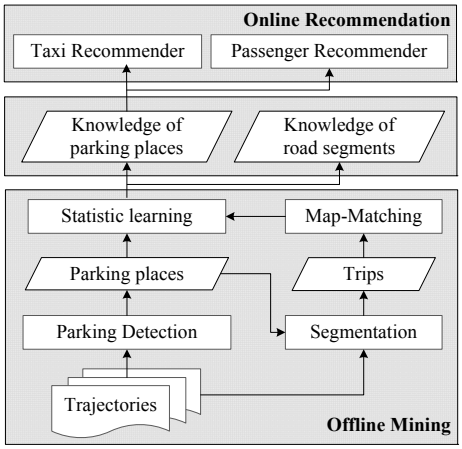


Fig. 3. System Overview

outperforms other approaches for low-sampling-rate GPS trajectories. Later, we utilize the detected parking places and the mapped trajectories to learn the time-dependent statistical results based on a probabilistic model (Section 4). To tackle the data sparseness problem, we devise a road segment clustering method and perform statistical learning on each road segment cluster instead of a single road segment (Section 5). The above processes are performed offline and will be repeated only when the trajectory data is updated (e.g., once a month). Based on this model, we perform recommendations to taxi drivers and passengers, given their locations and current times (Section 6).

3 Parking Places Detection

This section details the process for detecting parked status from a *non-occupied trip* and accordingly finding out the parking places in the urban area of a city based on a collection of taxi trajectories.

3.1 Candidates Detection

Figure 4 demonstrates the parking candidate detection approach, given a non-occupied trip $p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_7$. We first keep checking the distance (Great-circle distance) between the current point and the latter point until the distance is smaller than a threshold. As depicted in Figure 4 B), since $dist(p_1, p_2)$ exceeds the distance threshold δ, we move next, fixing p_2 as the "pivot" point and find that $dist(p_2, p_3) < \delta, dist(p_2, p_4) < \delta$ while $dist(p_2, p_5) > \delta$

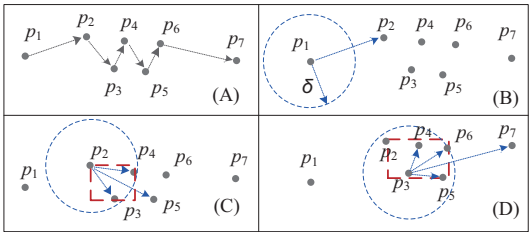


Fig. 4. Detection of candidate parking places

(Figure 4 C). If the time interval between $p_2.t$ and $p_4.t$ is larger than the time threshold τ, the three points that form a small cluster represent a possible parking candidate. Next, we fix p_3 as the pivot point and keep on the procedure to check latter points. Finally, as shown in Figure 4 D), we detect $(p_2, p_3, p_4, p_5, p_6)$ as a parking candidate because we cannot expand this group any further, i.e., all the points in this group have a distance farther than δ to p_7. The pseudocode is provided in Algorithm 1.

3.2 Filtering

Essentially, the candidate detection algorithm finds out the locations where the GPS points of a taxi are densely clustered, with spatial and temporal constraints. However, a parking candidate could sometimes be generated by taxis stuck in a traffic jam, or waiting for signals at a traffic light, instead of a real parking. To reduce such false selections, we design a supervised model for picking out the true parked status from the candidate set, using the following features:

- *Spatial-Temporal features* including 1) Minimum Bounding Ratio (MBR). As shown in Figure 5(A),B)), MBR is the area ratio between the bounding box of the road segment (MBRr) and the bounding box of the GPS points (MBRc) in the candidate set. 2) AverageDistance. The average distance d_c between points in the candidate set and their nearest road segments, as

Algorithm 1: ParkingCandidateDetection

Input: A road network G, a trajectory J, distance threshold δ, time threshold τ
Output: A set of parking candidates $\mathbb{P} = \{P\}$

```
i ← 0, M ← ||J||, P ← ∅, ℙ ← ∅;
while i < (M − 1) do
    j ← i + 1; flag ←false;
    while j < M do
        dist ←Distance (p_i, p_j);
        if dist < δ then j ← j + 1;flag =true;
        else break;
    if p_{j−1}.t − p_i.t > τ and flag =true then
        foreach point p ∈ J[i, j) and p ∉ P do
            P.Add(p);/* build a candidate */
        if i = j − 1 then
            ℙ.Add(MB (P)); P ← ∅;
            /* add the minimum bounding box of P
               into ℙ */
        i ← i + 1;
    return ℙ
```

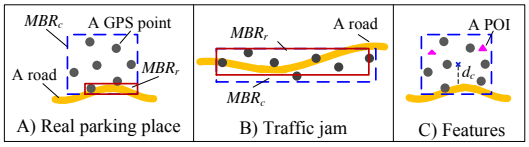


Fig. 5. Distinguishing parking places from traffic jams

shown in Figure 5 C). 3) CenterDistance. The distance between center point in MBRc of the candidate set and the road segments. 4) Duration. The parking duration of a candidate. 5) LastSpeed. The speed of the last point leaving a parking candidate (we found a high LastSpeed is a strong indicator that a taxi changes its status from parked to occupied).

- *POI feature*. As we know, a parking place is highly relevant with the points of interests (POI) around it, e.g., subway exits, theaters, shopping malls within 50 meters, shown in Figure 5 C). We employ the *term frequency-inverse document frequency* (tf-idf)[14] to measure the importance of a POI to a parking place. Specifically, for a given parking place, we formulate a POI vector, $\langle v_1, v_2, \ldots, v_k \rangle$ where v_i is the tf-idf value of the i-th POI category, given by:

$$v_i = \frac{n_i}{N} \times \log \frac{\|\mathbb{P}\|}{\|\{P \in \mathbb{P} | \text{the } i\text{-th POI category} \in P\}\|},$$

where n_i is the number of POIs belonging to the i-th category and N is the number of POIs lying around the parking candidate. The idf item is calculated using the logarithm of the number of parking candidates divided by the number of parking candidates which have the i-th POI category.

- *Collaborative feature*. For a real parking place, other drivers should also park historically at that place. Otherwise, it's not a common parking place. So we use the number of parking candidates within 50 meters in the past 7 days of a candidate as a collaborative feature to enhance the classifier.

We use a human-labeled dataset to learn the threshold and train a bagging [3] classifier model to guarantee the high precision and recall of the detected parking candidate. Then we utilize the model to inference whether a candidate is a really parking or a traffic jam.

3.3 Parking Place Clustering

The parked status is detected for each trajectory separately. However, the parking place detected from a single trajectory is only a portion of a real parking place. Thus different parking places may be actually the same one. We use a density based clustering method OPTICS [2] to discover what are essentially the same parking places. The reason for using this method is that it outperforms other methods when the clustered region may have an arbitrary shape and the points inside a region may be arbitrarily distributed.

4 Model Description

Both the taxi recommender and the passenger recommender are multiple-criteria recommendation systems. In particular, given the current location L_c and time T_0 of a taxi driver or a passenger, the taxi recommender provides the driver with top-k parking places and routes to these parking places while the passenger recommender suggests a set of road segments within a walking distance, both according to a specified recommendation strategy, which is either identified by the user's preference or automatically set to a default value. We detail the computation of each criterion in this section and then describe the strategies later in Section 6.

4.1 Taxi Recommender

The taxi recommender aims to *provide the taxi drivers with the best parking places and the routes to these parking places*. But how to define a "good" parking place? Different drivers may have different definitions. However, it's intuitively obvious that a good parking place should bring a *high probability* (during the routes or at the parking place) to get a passenger, a *short waiting time*, a *short queue length* at the parking place and a *long distance* of the next trip.

Definition 5 (Action Λ_{RP}): Assume P is a certain parking place and $R : r_1 \to r_2 \to \ldots \to r_n$ is a route to P. We say the driver *takes action* Λ_{RP} if he/she drives along R until finding a new passenger and queues at P for at most $t_{\max}$ time if he/she fails to pick up a passenger along R.

Actually, the recommendation adopts the risk-cost-benefit analysis used by many decision-making system. Specifically, the criteria used for the taxi recommender are as follows:

1) [Risk] How likely will the driver pick up a passenger if he/she takes the action Λ_{RP}?
2) [Cost1] If the driver takes action Λ_{RP} and succeeds in finding a new passenger, what is the expected duration from T_0 to the beginning of the next trip?
3) [Cost2] If the driver takes action Λ_{RP} and arrives at a parking place, what is the expected queue length at that parking place?
4) [Benefit] If the driver takes action Λ_{RP} and succeeds in finding a new passenger, how long is the expected distance/travel time of the next trip ?

We answer the above questions in the following subsections.

4.1.1 The Probability of Picking up the Next Passenger

If the driver takes action Λ_{RP}, he may pick-up a passenger on R; or pick-up a passenger at P; or fail to pick-up a passenger after waiting at P for $t_{\max}$ time. Let S be the event that the driver *succeeds* in picking up the next passenger if he/she takes the action Λ_{RP} and $\overline{S}$ be the opposite situation (fails to get the next passenger). Then we have

$$S = \bigcup_{i=1}^{n+1} S_i, \tag{1}$$

where S_i for $i = 1, 2, \ldots, n$ is the event that the driver picks up a passenger at road segment r_i, and S_{n+1} is the event that the driver picks up a passenger at the parking place. Note that both S and S_i are with respect to the current time T_0. Let $t_i = \sum_{j=1}^{i} r_j.t$, i.e., the travel time from the start point to r_i. Denote the probability that a cruising taxi picks up a passenger at road segment r_i and at time $T_0 + t_i$ by

$$p_i = \Pr(\mathcal{C} \rightsquigarrow \mathcal{O} | r_i, T_0 + t_i). \tag{2}$$

Let

$$p_* = \Pr(\mathcal{P} \xrightarrow{(0, t_{\max}]} \mathcal{O} | T_0 + t_n) \tag{3}$$

be the probability that a taxi succeeds in picking up a passenger at parking place P and waiting time $T_P \in (0, t_{\max}]$ if the driver reaches P at $T_0 + t_n$. Then

$$\Pr(S_i) = \begin{cases} p_1, & i = 1 \\ p_i \prod_{j=1}^{i-1} (1 - p_j), & i = 2, 3, \ldots, n, \\ p_* \prod_{j=1}^{n} (1 - p_j), & i = n + 1. \end{cases} \tag{4}$$

Now the answer of question 1 is clear:

$$\begin{aligned} \Pr(S) =& 1 - \Pr(\overline{\bigcup_{i=1}^{n+1} S_i}) \\ =& 1 - (1 - p_*) \prod_{j=1}^{n} (1 - p_j). \end{aligned} \tag{5}$$

The factor $\prod_{j=1}^{n}(1 - p_j)$ in Equation 5 is the probability that the driver fails to find a passenger along R. We denote this event by $\overline{S_R}$. Note the route from the current position of the driver to P is not unique. Should we suggest the driver with the route that has the minimum $\Pr(\overline{S_R})$? It's obviously absurd since the driver can traverse all the road network in that case. In practice, we can provide the fastest route or a route with the minimum $\Pr(\overline{S_R})$ conditioned by that the distance does not exceed a threshold. This can be implemented by a simple generalization of the constrained shortest path problem [19].

In practice, we assume the probability is stable during time interval $[t - \triangle t, t + \triangle t]$, where $\triangle t$ is a fixed threshold. This is reasonable, since the probability changes gradually instead of sharply. For computing the time-dependent probability, a common way is to partition a day into fixed slots (e.g., one hour a slot), and calculate the result for each slot beforehand. Different from this method, we develop a *partition-and-group* approach so as to compute this probability "just-in-time" and enable real-time recommendation. More concretely, we partition one day into K small time units, the length of each unit is τ (where $\triangle t$ can be divisible by τ, e.g., $\tau = 5$ minutes, $\triangle t = 15$ minutes). Thus, the k_{th} interval is

$$\Phi_k = [(k - 1)\tau, k\tau], \ k = 1, 2, \ldots, K. \tag{6}$$

Then we count the instances pertaining to each Φ_k (for needed probabilities) offline. In the online phase, when the time t of a taxi driver/passenger is input, we retrieve the corresponding intervals (i.e., a set of Φ_k) which belong to $[\lfloor t/\tau \rfloor \tau - \triangle t, \lfloor t/\tau \rfloor \tau + \triangle t]$, then compute the corresponding probability using the statistical results obtained from all the retrieved intervals. The intuition of this partition-and-group approach is much like the Riemann Integral. The advantage of the above "just-in-time" way compared to the fixed slot method is that we can avoid the discontinuity when crossing the boundary of a interval (e.g., the probability at 10:59 am may entirely different with the one at 11:00am if 11:00am is the boundary of a fixed time slot) as well as make the most use of the sparse data in each small interval.

Let $\#_k(\mathcal{C} \rightsquigarrow \mathcal{O}; r)$ be the number of trips that the taxis transfer from the cruising state to occupied state, i.e., pick up a passenger during Φ_k when cruising on road segments r, then p_j in Equation 5 can be computed by:

$$\Pr(\mathcal{C} \rightsquigarrow \mathcal{O} | r, t) = \frac{\sum_{k=\lfloor (t-\triangle t)/\tau \rfloor}^{\lfloor (t+\triangle t)/\tau \rfloor} \#_k(\mathcal{C} \rightsquigarrow \mathcal{O}; r)}{\sum_{k=\lfloor (t-\triangle t)/\tau \rfloor}^{\lfloor (t+\triangle t)/\tau \rfloor} (\#_k(\mathcal{C}; r))}. \tag{7}$$

For each cluster of the parking places, we have a set of trajectories which are at the parked status with varied arriving time and leaving time. Let $\#_k(t_a, t_b, \mathcal{P} \rightsquigarrow \mathcal{O}; P)$ be the number of trips that originate from parking place P when the taxis arrive at P during Φ_k and finally become occupied with the waiting time $T_P \in (t_a, t_b]$. Let $\#_k(\mathcal{P} \rightsquigarrow \mathcal{O}; P)$ be the number of trips that originate from P after the taxis arrive at P during Φ_k and become occupied when leaving P, versus $\#_k(\mathcal{P} \rightsquigarrow \mathcal{C}; P)$ denotes the taxis which are still non-occupied (cruising) when leaving P. Then the probability that the waiting time $T_P \in (t_a, t_b]$ when reaching P at t can be calculated by

$$\begin{aligned} &\Pr(\mathcal{P} \xrightarrow{(t_a, t_b]} \mathcal{O} | T_P > 0, t) = \\ &\frac{\sum_{k=\lfloor (t-\triangle t)/\tau \rfloor}^{\lfloor (t+\triangle t)/\tau \rfloor} \#_k(t_a, t_b, \mathcal{P} \rightsquigarrow \mathcal{O}; P)}{\sum_{k=\lfloor (t-\triangle t)/\tau \rfloor}^{\lfloor (t+\triangle t)/\tau \rfloor} (\#_k(\mathcal{P} \rightsquigarrow \mathcal{O}; P) + \#_k(\mathcal{P} \rightsquigarrow \mathcal{C}; P))}, \end{aligned} \tag{8}$$

which can be used to compute the p_* in Equation 5.

4.1.2 Duration Before the Next Trip T

Let random variable T be the duration from current time T_0 to the beginning of the next trip, given that the taxi driver takes the action Λ_{RP}. Then T is a summation of two random variables: the cruising time along R, denoted by T_R and the waiting time at P, termed as T_P, i.e.,

$$T = T_R + T_P. \tag{9}$$

Note that T_R and T_P are not independent. Actually,

$$\begin{cases} T_P = 0, & \text{if } T_R \leq t_n, \\ T_R = t_n, & \text{if } T_P > 0. \end{cases} \tag{10}$$

TABLE 2
Major Notations in This Paper

T_0	Current time	P	Parking place
R	The route to P	r_i	The i-th road segment of R
p_i	Equation 2	p_*^j	Equation 15
p_*	Equation 3	q_i^j	Equation 20
S	The event that the driver succeeds in picking up the next passenger		
t_i	Travel time from the current position to r_i		
T	Duration before the taxi picks up the next passenger		
T_R	Duration of driving along the route R		
T_P	Duration of waiting at parking place P		
D_N	Distance of the next trip		
T_N	Duration of the next trip		
$L(t)$	Queue length at a parking place at time t		
$T_w(t)$	Waiting time of a passenger arriving at time t		
$\mu(t)$	Arriving rate of vacant taxi on a road segment at time t		

Based on Bayes rules, the probability mass function is given by

$$\begin{aligned}&\Pr(T_R = t_i|S)\\ =&\Pr(T_R = t_i, S)/\Pr(S)\\ =&\begin{cases}\Pr(S_i)/\Pr(S), & i = 1, 2, \ldots, n-1,\\ \dfrac{\Pr(S_n)+\Pr(S_{n+1})}{\Pr(S)}, & i = n,\end{cases}\end{aligned} \tag{11}$$

thus the conditional expectation of T_R is

$$\begin{aligned}\mathbf{E}[T_R|S] =&\sum_{i=1}^{n} t_i \Pr(T_R = t_i|S)\\ =&\frac{1}{\Pr(S)}\left(\sum_{i=1}^{n} t_i \Pr(S_i) + t_n \Pr(S_{n+1})\right).\end{aligned} \tag{12}$$

Let W be the event that the driver waits at P, we have

$$\Pr(W) = \prod_{j=1}^{n}(1 - p_j). \tag{13}$$

To learn the distribution, we break the interval $(0, t_{\max}]$ into m buckets (in our system, we set $m = 200$ and $t_{\max} = 100$ min. We later discuss the parameter selection in Section 8.2). Specifically, let

$$\begin{cases} t_0 = 0,\\ \triangle t^* = t_{\max}/2m,\\ t_j^* = (2j-1)\triangle t^*,\ j = 1, 2, \ldots, m,\end{cases} \tag{14}$$

where t_j^* is the average waiting time for the j-th bucket. Denote the probability that the taxi succeeds in picking up a passenger and the waiting time T_P belongs to the j-th bucket by

$$p_*^j = \Pr(\mathcal{P} \xrightarrow{(t_j^*-\triangle t^*, t_j^*+\triangle t^*]} \mathcal{O}|T_P > 0, T_0 + t_n). \tag{15}$$

Actually, recall Equation 3, we have $p_* = \sum_{j=1}^{m} p_*^j$.

The conditional probability

$$\Pr(T_P = t_j^*|S) = \begin{cases}\dfrac{(1-\Pr(W))}{\Pr(S)}, & j = 0,\\ \dfrac{\Pr(W)p_*^j}{\Pr(S)}, & 1 \le j \le m.\end{cases} \tag{16}$$

Therefore, the conditional expectation of T_P is

$$\mathbf{E}[T_P|S] = \frac{\Pr(W)}{\Pr(S)}\sum_{j=1}^{m} p_*^j t_j^*. \tag{17}$$

Then, the conditional expectation of T is

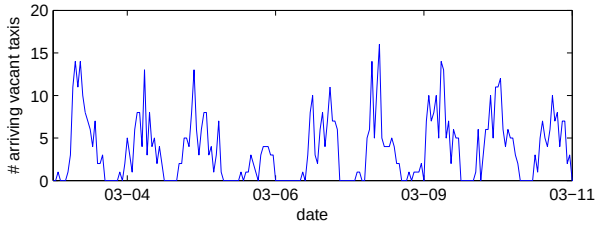
$$\boxed{\begin{aligned}&\mathbf{E}[T|S]\\ =&\mathbf{E}[T_R|S] + \mathbf{E}[T_P|S]\\ =&\frac{\sum_{i=1}^{n} t_i \Pr(S_i) + t_n \Pr(S_{n+1}) + \Pr(W)\sum_{j=1}^{m} p_*^j t_j^*}{\Pr(S)},\end{aligned}} \tag{18}$$

where $\Pr(S_i)$ is given by Equation 4, Equation 7 and p_*^j is computed using Equation 8.

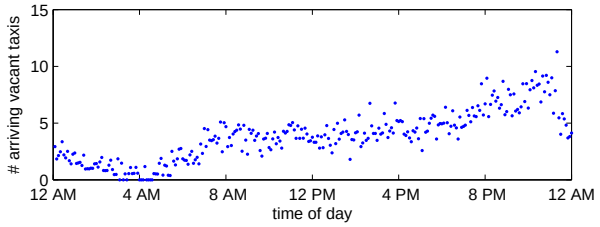
4.1.3 Queue Length

In some parking places, taxis often queue to wait for passengers, e.g., at the airports and railway stations. We learn the average queue length at a given time from the historical data. Figure 6(a) presents the number of arriving vacant taxis at a parking place along the time line. It's clear that the data reveals a certain cyclical characteristics. Since the data in a single day is not enough to get a statistical reasonable result, we first partition the data according to the day of week, and weather condition (normal weather and severe weather), then aggregate the corresponding data into one day (as shown in Figure 6(b)) and estimate the queue length.

We note that given the arriving time t at a parking place P, the number of parked taxis at P is only related with the



(a) Taxi arriving sequence at a parking place



(b) Average number of arriving vacant taxis per day

Fig. 6. Data aggregation for queue length estimation

taxis which arrive at P after $t - t_{\max}$ (recall Definition 5, the taxis arriving before $t - t_{\max}$ will leave P before t, thus do not affect the number of parked taxis at time t). Note that since the data is derived from multiple days, we regard the taxis arriving in different days as different taxis even if they have an identical taxi id. We denote the set of the taxis arriving at P during the interval Φ_k by $\mathbb{T}_k^{in}(P)$ and let $\mathbb{T}_k^{out}(P)$ be the set of taxis that leave P during the time interval Φ_k. We assume that in each interval Φ_k (we set the length of Φ_k as 5 minutes in our system), the taxi driver can only join and leave the queue once. We retrieve the intervals $\Phi_{k_1}, \Phi_{k_1+1}, \ldots, \Phi_{k_2}$, where $k_1 = \lfloor (t - t_{\max})/\tau \rfloor$ and $k_2 = \lfloor t/\tau \rfloor$.

We devise an iterative method to compute the average queue length at t, denoted by $\overline{L}(t)$. At the end of interval Φ_{k_1}, the set of parked taxis arriving after Φ_{k_1} (as stated above, we are only concerned about the taxis arriving after $t - t_{\max}$) is the set difference of $\mathbb{T}_{k_1}^{in}(P)$ and $\mathbb{T}_{k_1}^{out}(P)$. At the end of the second interval Φ_{k_1+1}, the parked taxis are increased by the taxis arriving at P in Φ_{k_1+1} and decreased by the taxis that leave P in Φ_{k_1+1}. As shown in Equation 19, we can compute the set of taxis queueing at the end of Φ_{k_2} and get the queue length $\overline{L}(t)$.

$$\begin{cases} \mathbb{T}'_{k_1}(P) = \mathbb{T}_{k_1}^{in}(P) \setminus \mathbb{T}_{k_1}^{out}(P), \\ \mathbb{T}_k(P)' = \left(\mathbb{T}'_{k-1} \bigcup \mathbb{T}_k^{in}(P)\right) \setminus \mathbb{T}_k^{out}(P), \quad k_1 + 1 \le k \le k_2, \\ \overline{L}(t) = \|\mathbb{T}'_{k_2}\|. \end{cases} \tag{19}$$

4.1.4 Distance/Travel Time of the Next Trip D_N, T_N

Let random variable D_N be the distance of the next trip if the driver takes the action Λ_{RP} conditioned by that S happens. Let q_i^j be the probability that the distance of the next trip satisfies $d_{j-1} < D_N \le d_j$, when S_i happens (note the time at that moment is $T_0 + t_i$), i.e., $\forall i = 1, 2, \ldots, n+1$,

$$q_i^j = \Pr(d_j - \triangle d < D_N \le d_j + \triangle d | S_i, T_0 + t_i). \tag{20}$$

Here,

$$\begin{cases} d_0 = 0, \\ \triangle d = d_{\max}/2s, \\ d_j = (2j-1)\triangle d, \quad j = 1, 2, \ldots, s, \end{cases} \tag{21}$$

where $d_{\max}$ is the maximum distance of the next trip (in our system, we set $s = 80$ and $d_{\max} = 80$ km). Then, the conditional probability distribution is given by:

$$\Pr(D_N = d_j | S) = \sum_{i=1}^{n+1} \Pr(S_i) q_i^j / \Pr(S), \tag{22}$$

for $j = 1, 2, \ldots, s$. Thus, the conditional expected distance of the next trip is

$$\boxed{\begin{aligned} \mathbf{E}[D_N|S] &= \frac{1}{\Pr(S)} \sum_{j=1}^{s} \left(d_j \sum_{i=1}^{n+1} \Pr(S_i) q_i^j \right) \\ &= \frac{1}{\Pr(S)} \sum_{i=1}^{n+1} \Pr(S_i) \left(\sum_{j=1}^{s} d_j q_i^j \right). \end{aligned}} \tag{23}$$

Similarly, we apply the partition-and-group method to calculate q_i^j, given by,

$$\Pr(d_a < D_N \le d_b | r, t) = \frac{\sum_{k=\lfloor (t-\triangle t)/\tau \rfloor}^{\lfloor (t+\triangle t)/\tau \rfloor} \#_k(d_a, d_b; r)}{\sum_{k=\lfloor (t-\triangle t)/\tau \rfloor}^{\lfloor (t+\triangle t)/\tau \rfloor} \#_k(0, d_{\max}; r)},$$

where $d_{\max}$ is the maximum distance of a trip and $\Pr(t_a < T_N \le t_b \mid r, t)$ is similarly computed.

Note that the conditional expected travel time of the next trip $\mathbf{E}[T_N|S]$ is computed in exactly the same way as $\mathbf{E}[D_N|S]$, thus we omit the detail.

4.2 Passenger Recommender

Different from the taxis, the passengers do not want to walk too long for hailing a taxi. If a passenger is close to at least one parking place, we provide him with the nearby parking places with the maximum expected queue length calculated with the method described in 4.1.3. Otherwise, we suggest the passenger with the nearby road segments (in a walking distance) considering two criteria 1) the possibility to find a vacant taxi and 2) the average waiting time.

4.2.1 Probability of Finding a Vacant Taxi

Let $\Pr(\mathcal{C}; r|t)$ be the probability that there is a vacant taxi on road segment r at time t. Given the passenger's current position, we suggest him with the road segments, which have the highest probability of finding a cruising taxi among a reachable region Ω of the passenger, i.e.,

$$r = \operatorname*{argmax}_{r \in \Omega} \Pr(\mathcal{C}; r|t) \tag{24}$$

$$= \operatorname*{argmax}_{r \in \Omega} \Pr(r|t) \Pr(\mathcal{C}|r, t). \tag{25}$$

Let $\{r_i\}_{i=1}^{R}$ be the set of all the road segments in the road network, then the first factor on the right side of Equation 25 can be calculated by:

$$\Pr(r|t) = \frac{\sum_{k=\lfloor (t-\triangle t)/\tau \rfloor}^{\lfloor (t+\triangle t)/\tau \rfloor} \left(\#_k(\mathcal{C}; r) + \#_k(\mathcal{O}; r) \right)}{\sum_{i=1}^{R} \sum_{k=\lfloor (t-\triangle t)/\tau \rfloor}^{\lfloor (t+\triangle t)/\tau \rfloor} \left(\#_k(\mathcal{C}; r_i) + \#_k(\mathcal{O}; r_i) \right)} \tag{26}$$

where $\#_k(\mathcal{C}; r)$ and $\#_k(\mathcal{O}; r)$ denote the number of cruising and occupied taxis on road segment r within the interval Φ_k respectively. Similarly, the second factor on the right side of Equation 25 is given by:

$$\Pr(\mathcal{C}; r|t) = \frac{\sum_{k=\lfloor (t-\triangle t)/\tau \rfloor}^{\lfloor (t+\triangle t)/\tau \rfloor} \#_k(\mathcal{C}; r)}{\sum_{k=\lfloor (t-\triangle t)/\tau \rfloor}^{\lfloor (t+\triangle t)/\tau \rfloor} \left(\#_k(\mathcal{C}; r) + \#_k(\mathcal{O}; r) \right)}. \tag{27}$$

Combining Equation 25, 26 and 27 (note that we only consider the road segments in Ω), we have

$$r = \underset{r \in \Omega}{\operatorname{argmax}} \frac{\sum_{k=\lfloor (t-\triangle t)/\tau \rfloor}^{\lfloor t+\triangle t/\tau \rfloor} \#_k(\mathcal{C}; r)}{\sum_{r_i \in \Omega} \sum_{k=\lfloor (t-\triangle t)/\tau \rfloor}^{\lfloor (t+\triangle t)/\tau \rfloor} \left(\#_k(\mathcal{C}; r_i) + \#_k(\mathcal{O}; r_i) \right)}. \tag{28}$$

4.2.2 Average Waiting Time

For a passenger, the waiting time is a crucial concern in addition to the probability. The arrival of the vacant taxis on a given road segment can be modeled using a non-homogeneous Poisson process [13] (which is a Poisson process with a time-dependent arriving rate function) with arriving rate $\mu(t)$. Let $T_w(t)$ be the waiting time when a passenger arrive at time t on a given road segment. Note that the expected waiting time should be less than the average interval of two consecutive vacant taxis, i.e.,

$$\mathbf{E}[T_w(t)] \leq \frac{2\triangle t}{\int_{t-\triangle t}^{t+\triangle t} \mu(x)dx} = \hat{T}_w(t). \tag{29}$$

Hence, we use $\hat{T}_w(t)$ as an upper bound to approximate the expected waiting time.

To learn the arriving rate $\mu(t)$ from the data, we incorporate the maximum likelihood method proposed in [11], which regards the arriving rate function as a piece-wise linear function. Here, we also employ the partition-and-group method (proposed in Section 4.1.1). Specifically, we count the number of vacant taxis arriving on a given road segment r for each time interval Φ_k. Later, we aggregate the data into one single day (for weekdays and weekends, we perform the aggregation separately since they may have different patterns). Given the arriving time at a road segment t, we retrieve the subintervals Φ_k within the time slot $[t-\triangle t, t+\triangle t]$, say $\Phi_1, \Phi_2, \ldots, \Phi_m$. Denote the realization of the average count vector (number of counts per day) $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_m)$ for the m subintervals by $\mathbf{n} = (n_1, n_2, \ldots, n_m)$, where Y_i is a random variable produced by a non-homogeneous Poisson process with mean value

$$\mu_k = a + bx_k, \quad 1 \leq k \leq m, \tag{30}$$

where $x_k = 2(k-1/2)\triangle t/m$. As stated above, we assume the arriving rate function is linear in the short time slot $[t-\triangle t, t+\triangle t]$, written as

$$\mu(t) = a + bt. \tag{31}$$

Then the goal is to find the estimator $\hat{a}$ and $\hat{b}$ to maximize the following log-likelihood

$$\begin{aligned} \ln \Pr(\mathbf{Y}; a, b) &= -\sum_{k=1}^{m} \mu_k + \sum_{k=1}^{m} Y_k \ln \mu_k - \sum_{k=1}^{m} \ln(Y_k!) \\ &= -a\triangle t - \frac{b\triangle t^2}{2} \\ &\quad + \sum_{k=1}^{m} Y_k \ln((a+bx_k)(\triangle t/m)) + C, \end{aligned}$$

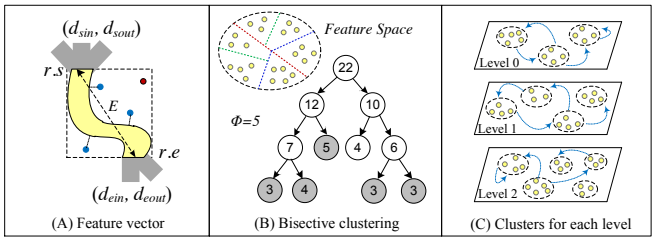


Fig. 7. Road segment clustering methodology

where C is a constant. This maximization problem can be efficiently solved using Newton's iterative method[11].

5 Road Segment Clustering

On some road segments, the data is too sparse to perform the statistical learning proposed in Section 4. Instead of computing the probability on each road segment, we first conduct a road segment clustering to integrate the road segments with similar features so as to tackle the data sparseness problem and accelerate the online computing.

We identify the following features (as input for road segment clustering), which are derived from the underlying road structure and POIs.

- L: The actual length of a road segment.
- L/E: The ratio between L and the Euclidean length (between the terminal points) E of a road segment. In general, the larger the value is, the more tortuous the road segment is.
- *dir*: The direction of a road segment (one-way/two-way).
- *Lanes*: The number of lanes in a given road segment.
- *degree*: Given a road segment r, we define the in-degree of the start node (d_{sin}) as the number of directed edges terminating at node s , while the out-degree of the start node (d_{out}) as the number of directed edges originating at node s. Similarly, we define the in-degree and out-degree for the terminal node e as d_{ein} and d_{eout}, as shown in Figure 7 A.
- *POI*: The POI features used are similar with the ones used in parking place detection.

Later, we devise a *Bisected Clustering* algorithm, which respectively group the road segments of the same level into a certain number of clusters according to the above-mentioned features. This algorithm is based on K-means (with $K = 2$) clustering and is performed recursively.

In this algorithm, we first partition all the road segments into 2 respective clusters $\{left, right\}$ by using a 2-means clustering algorithm. Now for each cluster $C \in \{left, right\}$, keep on sub-dividing the cluster C via 2-means unless and until the total of samples (GPS points) inside $C \leq \phi$. In this example, if we set $\phi = 5$, the five clusters with gray background cannot be further divided. So, in total, six clusters will be generated as a result of the clustering. Note that this clustering is only performed on the road segments with the same level, e.g., $r.level = 2$, as illustrated in Figure 7 B.

The reasons to use the Bisected clustering (with ϕ constraint) as opposed to simple K-means lie in two aspects: First, we need to avoid clusters with few GPS points, which would then result in biased and incorrect computation of prior probabilities. Second, we do not know how to define the number of clusters K in advance.

As a result, we obtain a collection of clusters, each of which contains a set of road segments with similar features. In the off-line learning stage, we learn the needed probabilities of our model introduced in Section 4 with respect to each cluster of road segments instead of a single road segment so as to gain a statistical reasonable result based on the sparse data. For example, let $\#_k(\mathcal{C}\rightsquigarrow\mathcal{O};\tilde{r})$ be the number of trips that the taxis transfer from the cruising state to the occupied state, i.e., pick up a passenger during Φ_k when cruising on all road segments within cluster $\tilde{r}$, then the probability of Equation 6 is actually calculated by:

$$\Pr(\mathcal{C}\rightsquigarrow\mathcal{O}|r,t)=\frac{\sum_{k=\lfloor t/\tau\rfloor}^{\lfloor (t+\triangle t)/\tau\rfloor}\#_k(\mathcal{C}\rightsquigarrow\mathcal{O};\tilde{r})}{\sum_{k=\lfloor t/\tau\rfloor}^{\lfloor (t+\triangle t)/\tau\rfloor}(\#_k(\mathcal{C};\tilde{r}))}. \quad (32)$$

Other probabilities related to the road segments, such as Equation 26 and Equation 27, are similarly computed. Additionally, in the later on-line phase described in Section 6, the computation cost is also saved thanks to the road segment clustering, since the number of candidate road segments to be computed is significantly reduced to a comparatively smaller cluster space.

6 Online Recommendation

In this stage, given the location and time of a taxi driver/passenger, we provide recommendations based on the proposed model and the derived statistical knowledge.

For the **taxi recommender**, we first perform a range query according to the location of the taxi, and then retrieve a set of potential parking places. For each parking place P, we generate the route R with the minimum $\Pr(\overline{S_R})$ constrained by a distance threshold (1.5× distance of the shortest path) using a dynamic programming recursion [19] in parallel. Then we compute the probability $\Pr(S)$ and the conditional expectations: $\mathbf{E}[T|S]$, $\mathbf{E}[D_N|S]$, $\mathbf{E}[T_N|S]$ with respect to the current time of the driver T_0. Later, we rank the candidate parking places with (but not limited to) the following strategies (S1–S4) and accordingly recommend top-k parking places to the driver in real-time. The thresholds P_θ, D_θ and F_θ can either be learned from the training data or be set by the user.

S1. (Most profitable) $Topk_{\max}\{\mathbf{E}[D_N|S]/\mathbf{E}[T+T_N|S] : \Pr(S) > P_\theta\}$. The candidate parking places of this strategy are restricted to the ones which have a $\Pr(S)$ larger than a threshold P_θ, among which, we provide the taxi driver with the top-k profitable parking places, i.e., the driver can earn the most money per unit time by traveling to these k parking places.

S2. (Fastest) $Topk_{\min}\{\mathbf{E}[T|S] : \Pr(S) > P_\theta, D_N > D_\theta\}$. This strategy retrieves k parking places which have the minimum expected duration before picking up a new passenger and have at least P_θ possibility to pick up a passenger as well as D_θ distance of the next trip.

S3. (Highest probability) $Topk_{\max}\{\Pr(S) : \mathbf{E}[D_N|S]/\mathbf{E}[T+T_N|S] > F_\theta\}$. This strategy retrieves the parking places, towards which the drivers are most likely to pick up a passenger and has a guaranteed profit (at least F_θ).

S4. (Shortest queue) $Topk_{\min}\{E(L(t)) : \mathbf{E}[D_N|S]/\mathbf{E}[T+T_N|S] > F_\theta, \Pr(S) > P_\theta\}$. This strategy provides the parking places with the minimum expected queue length, a guaranteed profit (at least F_θ) and a guaranteed possibility of finding a passenger (at least P_θ).

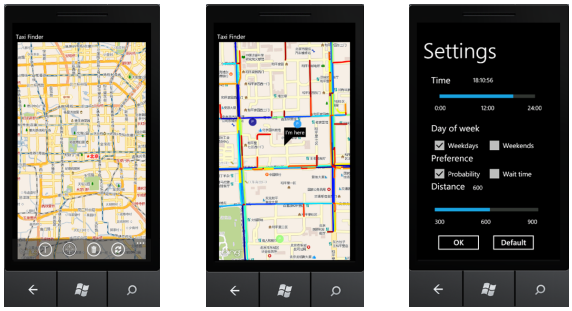


Fig. 8. WP7 App for the passenger recommender

For the **passenger recommender**, we also perform a range query so as to obtain a region, which is within a walking distance of the passenger. As shown in Figure 8, the recommender provides the user with nearby parking places as well as the road segments with the colors indicating the possibility to find a vacant taxi. The user can choose to rank top-k road segments based on either the **probability** of finding a vacant taxi (denoted by $Rank_p$) or the **average waiting time** (denoted by $Rank_t$). By default, the system returns the top-k road segments using $Rank_p$ and provides the estimated waiting time of the suggested k road segments so as to be more user friendly. The results presented to the users are also time-dependent.

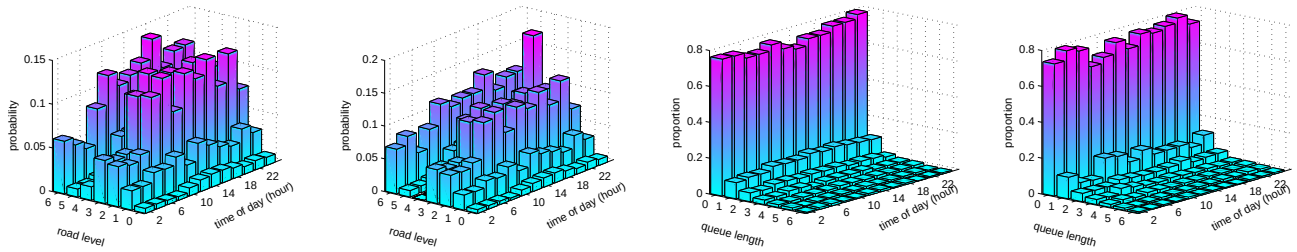
7 Validation

7.1 Settings

7.1.1 Dataset

Road networks: We evaluate our method using the road network of Beijing, which contains 106,579 road nodes and 141,380 road segments.

Trajectories: The dataset contains the GPS trajectory recorded by over 12,000 taxis in a period of 110 days in the year of 2010. The total distance of the data set is more than 200 million kilometers and the number of points reaches to 577 million. After trip segmentation, there are in total 20 million trips, among which 46% are occupied trips and 54% are non-occupied trips. Since in Beijing there are about 67,000 taxis, we enlarge the results with a scale factor 5.83(=67000/12000)



(a) $\Pr(\mathcal{C} \rightsquigarrow \mathcal{O})$, normal weather (b) $\Pr(\mathcal{C} \rightsquigarrow \mathcal{O})$, severe weather (c) Queue length, weekdays (d) Queue length, weekends

Fig. 9. Statistics results on road segments and parking places

when calculating the queue length and arriving rate (based on the additivity of Poisson distribution). A sample data is available at [1].

7.2 Evaluation on Statistical Learning

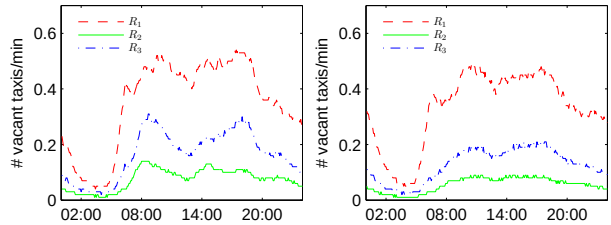
We present some of the statistical results associated with parking places and road segments as shown in Figure 9. For example, Figure 9(a) depicts the average probability (for each level of road segments) that a taxi transfers from the cruising status to the occupied status changing over time on weekdays when the weather condition is normal. Since the level-0 roads and level-1 roads are mainly high-ways or main roads, the probability is reasonably lower compared with level-2/3. Figure 9(b) reveals the corresponding results under severe weather condition. It is clear that during inclement weather, there are more passengers picked up on low-level road segments, especially in the evening (see Figure 9(b)). Figure 9(c) and Figure 9(d) depict the distribution of queue length of all the parking places changing over time. We note that on weekends, during midnight (10pm–2am) the queue length is comparatively longer than on weekdays. The reason may be that people go out more often on the evenings of weekends.

Based on the method proposed in Section 4.2.2 and Section 5, we calculate the average arriving rate of vacant taxis on each road segment cluster. For example, Figure 10 plots the results of three road segments pertaining to different clusters on weekdays and weekends changing over time. Obviously, during the period around 4am, the arriving rate is the lowest for all of the road segments. In addition, we note that on weekdays the arriving rate around morning rush hour (8am) is significantly higher (with a gap of 0.1) than on weekends, since the taxi drivers can take this chance to earn money from the passengers who go to work or go to school.

7.3 Evaluation on Road Segment Clustering

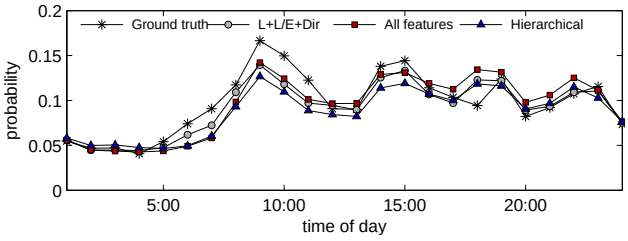
The road segment clustering process aims to group the road segments which have similar statistical results based on a set of features we choose. Here, we evaluate the performance of the road segment clustering by verifying that the road segments with similar feature values are close in terms of statistical results.

Specifically, we select some road segments having enough number of GPS points, and directly calculate the statistical results (like $\Pr(\mathcal{C} \rightsquigarrow \mathcal{O}|r,t)$) according to the method proposed

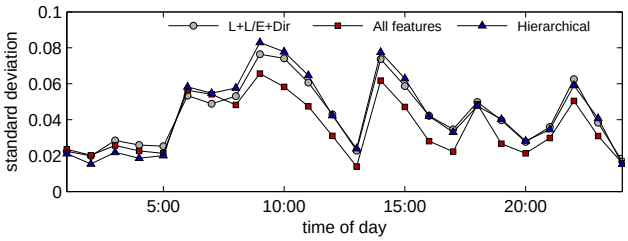


(a) weekdays (b) weekends

Fig. 10. Vacant taxi arriving rate on different road segment clusters changing over time



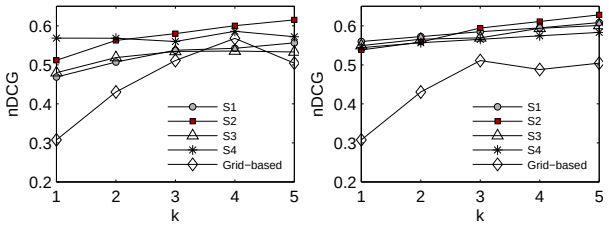
(a) Overall $\Pr(\mathcal{C} \rightsquigarrow \mathcal{O}|r,t)$ under normal weather condition, weekdays



(b) Standard deviation of $\Pr(\mathcal{C} \rightsquigarrow \mathcal{O}|r,t)$ compared with the ground truth

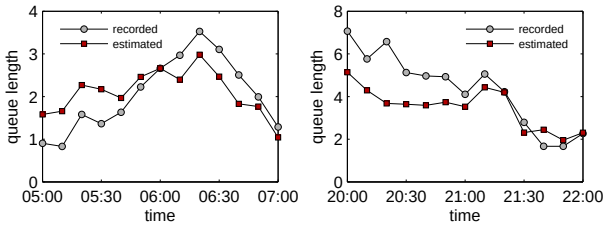
Fig. 11. Evaluation results on road segment clustering

in Section 4 based on their own data. The statistical results of such road segments are considered as the ground truth (since the data is enough for an accurate estimation) to evaluate the clustering results. Later, we randomly remove the samples (the samples are used in the POI feature) in each of these road segments and then perform our clustering method. Next, we evaluate the closeness of the statistical results on the road segments within each cluster and study the effects of the features we described in Section 5. In addition, we compare our method with the hierarchical clustering method proposed in [10]. For example, Figure 11(a) presents the overall probabilities of $\Pr(\mathcal{C} \rightsquigarrow \mathcal{O}|r,t)$ on all the clusters, changing over time. As is shown, our method using all the features outperforms both the baseline method and our



(a) weekdays (b) weekends

Fig. 12. NDCG of different recommendation methods



(a) weekdays (b) weekends

Fig. 13. Recorded queue length compared with estimated queue length changing over time

method considering only partial features. Meanwhile, Figure 11(b) plots the corresponding standard deviation for the above methods at each time stamp. It's clear that our method using all the features has the lowest standard deviation (with the ground truth) among all the compared methods.

7.4 Evaluation on Taxi Recommender

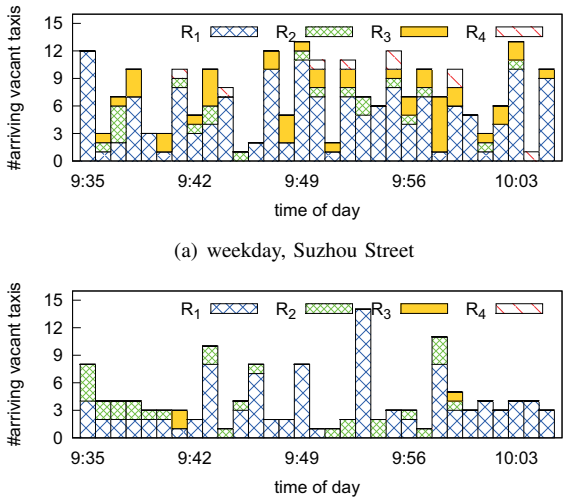
In our previous paper [18], we evaluated the parking place detection method and the taxi recommender (using strategies S1–S3). The result showed that our method can precisely detect the parking places with a precision over 90% and effectively provide taxi drivers with high-profit parking places, e.g., our method outperforms the baseline methods (e.g., kNN-based method) with an 4.5% improvement in precision and an 8% improvement in recall, based on the data of high-profit taxi drivers. Here, we further conduct experiments to evaluate the effectiveness of the taxi recommender.

We compare our recommender with the grid-based method, which partitions the map into grids and recommends the taxi driver with a nearby high-profit grid (for comparison, the grid size is maintained in the same scale with a parking place), based on the historical statistics. In this experiment, we utilize the high-profit drivers' data as ground truth. The query points are generated by two categories: 1) the drop-off points and 2) the points which are 500m, 1000m and 1500m before the next parking place along a cruising trip. For each method, we retrieve the recommended top-k locations as well as the routes to these locations. We measure the performance of different recommendations using NDCG (normalized discounted cumulative gain) and employ the scoring metrics proposed in [18] to calculate the NDCG@k for each method. Figure 12 presents the results for both the weekdays and weekends. As is shown, our methods outperforms the competing method, where the improvements in terms of nDCG are more significant on weekends.

In addition, we perform in-the-field studies to explore the accuracy of the proposed queue length estimation (w.r.t. strategy S4). Specifically, we invite 4 users to stand by 7 different parking places and record the number of parked taxis every 5 minutes during a period of 2 hours. This evaluation is repeated in a period of 3 weeks and is conducted on both the weekdays and weekends. Later, we compare the evaluation results with the recommended results to validate the effectiveness of our system.

Figure 13 presents the overall results (10 days for weekdays and 5 days for weekends) of the recorded queue length compared with the estimated queue length changing over time on weekdays and weekends. The results show that the difference between the estimated and recorded queue length is quite small (RMSE=0.5576) on weekdays. Besides, the trends changing over time are accurately modeled for both the weekdays and weekends.

7.5 Evaluation on Passenger Recommender



(a) weekday, Suzhou Street

(b) weekend, Wudaokou

Fig. 14. Number of vacant taxis passing through the road segments close to two areas, counted per minute

We perform the in-the-field study for the passenger recommender in 5 areas of Beijing: Zhichun Road, Suzhou Street, Wudaokou, Zhongguancun (shortened as ZGC) East Road and Zhongguancun Main Street. For each area, the users are involved to record the number of vacant taxis passing by 4 nearby road segments every minute during a period of 30 minutes. For each road segment close to an area, we ask one user to record the number of vacant taxis passing through the road segment every minute. This evaluation is also repeated in a period of 3 weeks and is conducted on both the weekdays and weekends. For example, Figure 14 presents the results for one test in Suzhou Street on a weekday and another in Wudaokou on a weekend, where the road segments R_1, R_2, R_3, R_4 are ranked by $Rank_p$, i.e., the possibility

TABLE 3
Overall evaluation results of passenger recommender on weekdays

area	Zhichun Rd.				Suzhou St.				Wudaokou				ZGC East Rd.				ZGC Main St.			
time	8:40-9:10				9:35-10:05				15:55-16:20				8:45-9:15				14:00-14:30			
road	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
$\#$	20.8	3.3	0.8	0.8	128.3	35.0	16.7	7.5	59.2	38.3	3.3	1.7	45.8	15.8	1.7	1.2	48.3	20.0	13.3	9.2
$Rank$	1	2	3	3	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
$Rank_p^d$	2	1	4	3	1	3	2	4	1	2	3	4	1	2	4	3	2	1	3	4
$Rank_t^d$	2	1	3	4	1	2	4	3	1	2	3	4	2	1	3	4	1	2	4	3
$Rank_p^w$	1	3	2	4	1	3	2	4	1	2	4	3	1	4	3	2	1	3	2	4
$Rank_t^w$	1	2	4	3	2	1	3	4	1	2	3	4	1	2	4	3	1	2	4	3
$Rank_p^{d,w}$	1	2	3	4	1	3	2	4	1	2	3	4	1	2	3	4	1	2	3	4
$Rank_t^{d,w}$	1	2	4	3	1	2	3	4	1	2	3	4	1	2	3	4	1	2	4	3

TABLE 4
Overall evaluation results of passenger recommender on weekends

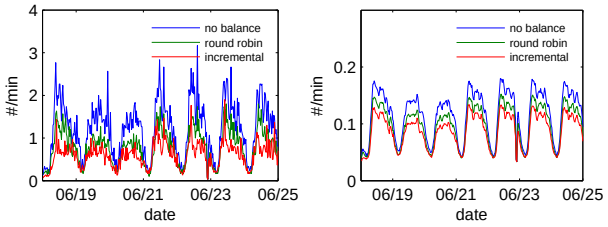
area	Zhichun Rd.				Suzhou St.				Wudaokou				ZGC East Rd.				ZGC Main St.			
time	10:30-11:00				13:10-13:40				9:40-10:10				14:15-14:45				15:35-16:05			
road	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
$\#$	35.0	18.3	8.3	0.8	22.5	10.0	7.5	7.5	82.5	28.33	2.5	0.7	59.1	48.3	10.8	4.2	10.0	8.3	6.7	3.3
$Rank$	1	2	3	4	1	2	3	3	1	2	3	4	1	2	3	4	1	2	3	4
$Rank_p^{d,w}$	1	2	3	4	1	2	3	4	1	2	3	4	1	2	4	3	1	2	3	4
$Rank_t^{d,w}$	1	3	2	4	1	2	4	3	1	2	3	4	1	2	3	4	1	2	3	4

to find a vacant taxi (see Section 6). It's clear that the recommended road segment R_1 has the most vacant taxis passing by in the period of 30 minutes. Table 3 gives the overall results of the in-the-field evaluation on weekdays. In the first column, $Rank_p^{d,w}$ stands for ranking according to probability of finding a vacant taxi considering *day of the week* (d) and *weather conditions* (w); $Rank_p^d$/$Rank_p w$ only considers day of the week/weather conditions; $Rank_p$ considers neither of them. The notations for $Rank_t$ are similar. We compare the above ranking with the real ranking (denoted by $Rank$) according to the average number of vacant taxis encountered in 30 minutes (denoted by $\#$). It's obvious that the methods considering both the day of the week and weather conditions are better than the competing methods. We note that in all of the tested areas, the rank-1 road segments of $Rank_p^{d,w}$ and $Rank_t^{d,w}$ match the real ranking perfectly. For rank-2∼rank-4 road segments, in some places $Rank_p^{d,w}$ is better than $Rank_t^{d,w}$ (e.g., ZGC Main St.) and vise versa. Tabel 4 shows the corresponding results on weekends (due to the space limitation, here we only present the results of our combined approaches, i.e., $Rank_p^{d,w}$ and $Rank_t^{d,w}$).

8 Discussion

8.1 Load Balance

The load balance problem is an open challenge for many recommendation systems and is also widely studied in many other fields such as distributed networks[7] and web services[4]. According to the places where the balancing approaches are used, the balancing strategies can be categorized into two groups: client-side balancing and server-side balancing. We note that our recommender is location-variant and time-dependent, i.e., vary for the queries in different places and



(a) maximum loads (b) average loads

Fig. 15. The performance of different balancing strategies

different time periods. Hence, the load balance problem is weakened in the "client"-side to a certain extent. Meanwhile, since our recommender is based on a probabilistic model learned from the historical data, the "high-loaded" parking places/road segments where many drivers/passengers often choose are already considered in the model (e.g., the queueing model of taxis in Section 4.1.3). In addition, the taxi recommender aims to provide recommendations to taxi drivers and let them make their own decisions. However, this recommender can also be combined with the central taxi dispatching system (which takes the responsibility for controlling the loads of taxis in different places) so as to complement each other.

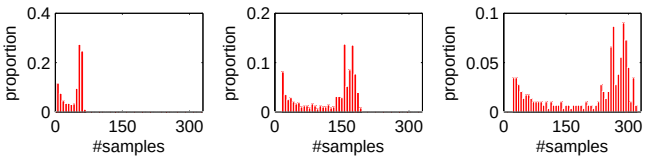
In practice, in order to further lower down the risk of overloads, we can adopt the *round-robin* or *incremental-load* balancing strategies in the "server"-side. For each time unit τ (see Equation 6), the round-robin strategy (as utilized in [6]) maintains the number of queries on each road segment (only the ones on which the queries are already sent), chooses one from the top-k lists calculated by the system in a circular manner, e.g., for the first query, return the top-1 parking place, and for the second query, return the top-2 parking place..., and return the top-1 again for the $k+1_{th}$ query.

Instead, the incremental-load strategy returns the least-loaded parking place to answer the current query for the queries sent from the same road segment and at the same time unit. We investigate the maximum loads (i.e., the maximum number of times that a certain parking place is recommended to taxi drivers) per minute and average loads with different balancing strategies in Figure 15(a) and Figure 15(b) respectively. Here, we simulate the queries with all the drop-off points (in total 1,151,703 points) during a week (from 2010 Jun. 18 to 2010 Jun. 24), where the average query interval is 0.53 seconds. As a result, both of the proposed balancing strategies alleviate the loads significantly, while the incremental-load method has a better performance. Note that the chain reactions, such as the traffic dynamics induced by mobilizing the taxis using the above strategies, are not taken into account, however, this is beyond the focus of this paper.

8.2 Parameter Selection

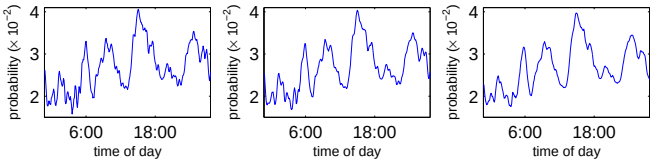
In Section 4.1.1, we propose the partition-and-group framework for calculating the required probabilities of the model, which is then utilized many times, such as in Equation 7,8,26 and 27. In our model, the time unit τ is the smallest unit (the *partition* part in the partition-and-group framework) for counting the number of samples based on the historical data. Thus a smaller τ means a higher resolution segmentation in the time line, whereas the number of samples in a single time unit is reduced simultaneously. Hence, our selection rule is to obtain a τ as small as possible, while providing sufficient data for a statistical reasonable estimation. For example, Figure 16 presents the distribution (over different time units during a day) of average number of samples on each *road segment cluster* (keep in mind that the statistical learning are performed with respect to each road segment cluster instead of individual road segments, as described in Section 5). As is shown, when $\tau = 1$ min or 3 min, the number of samples in most time units is less than 150, however, when τ increases to 5 min, the number of samples in most time units is close to 300. Thus in our system, we set $\tau = 5$ min. We use similar method to determine the value of $\triangle t^*$ in Equation 14 and $\triangle d$ in Equation 21.

Figure 17 gives another example showing the selection of parameter $\triangle t$, which is the smallest time interval (the *group* part in the partition-and-group framework) for answering a time-dependent query. Here, given the query time t, the time interval with length $\triangle t$ is utilized as a sliding window to aggregate statistical results in the time units around t, learned in the off-line phase, so as to smooth the values changing over time (which is similar to the "moving averaging smoothing" method). As is shown, when $\triangle t = 30$ min, we obtain a trade-off between the smoothness and the sensitivity (of the time-dependent probabilities).



(a) $\tau = 1$ min (b) $\tau = 3$ min (c) $\tau = 5$ min

Fig. 16. Distribution of average #samples w.r.t. τ



(a) $\triangle t = 15$ min (b) $\triangle t = 20$ min (c) $\triangle t = 30$ min

Fig. 17. $\Pr(\mathcal{C} \rightsquigarrow \mathcal{O})$ at different time of day w.r.t. $\triangle t$

9 Related Work

9.1 Dispatching Systems

Taxi dispatching systems are attracting growing attention from researchers with the development of intelligent transportation systems and the popularization of GPS sensors [8]. Most existing dispatching systems assign a task to taxi drivers based on nearest neighbor principle in terms of distance or time. Phithakkitnukoon et al. [12] use the naive Bayesian classifier with developed error-based learning approach to infer the number of vacant taxis at a given time and location which can be used to enhance the dispatching system. Yamamoto et al. [15] propose a fuzzy clustering and adaptive routing approach to improve dispatching system by assigning vacant taxis adaptively to the locations with high expectation of potential customers.

Different from the centralized dispatching, our recommendation system provides suggestions to taxi drivers/passengers, allowing them to make their own decisions. Typically, for a dispatching system, the customers need to book a taxi by telephone/internet in advance, and it is usually not free of charge. Most passengers hail a taxi along the road or stand where the taxis are available instead of booking a taxi. Besides, our method aims to maximize the profit for a particular taxi driver instead of balancing the income of all the taxi drivers, which is usually a goal of a dispatching system. In addition, our approach can be combined with a dispatching system so as to complement each other.

9.2 Location Recommendation For Taxi Drivers

A number of recent works provide recommendations for taxi drivers. Ge et al. [6] present a novel model to recommend a taxi driver with a sequence of pick-up points so as to maximize a taxi driver's profit. This work formulates the target problem by a mobile sequential recommendation (MSR) problem. Combined with a taxi driving fraud detection method and some business insights such as tip distributions, Ge et al. introduce a taxi business intelligence system in [5]. Li et al. [9] study the passenger-finding strategies (hunting/waiting) of taxi drivers in Hangzhou. In this work, L1-Norm SVM is used to select features for classifying the passenger-finding strategies in terms of performance.

Our approach is different from the above methods in the following aspects: 1) We provide recommendations to both

taxi drivers and passengers, which mobilizes them and reduces the disequilibrium of the demand and supply. 2) Instead of a grid/cell-based partition of the map, our recommendation is provided on road-segment level, which enables more accurate and meaningful understanding of the taxi drivers' behaviors as well as a more practical recommendation for both the taxi drivers and the passengers. 3) We focus on the off-peak hours to help the driver make the first step decision whenever and wherever they want to decide a destination to go. In practice, the "first step" recommendation would be more effective since usually the drivers are not willing to remember a sequence of places. 4) We develop an algorithm to distinguish the parked status from traffic jams and propose a solution to detect the parking places in an urban area. 5) We target the challenges when building the system based on sparse data and facilitate the on-line recommendation with a partition-and-group framework.

10 Conclusion

To save the time for finding a taxicab and reduce unnecessary traffic flows as well as energy consumptions caused by cruising taxicabs, we proposed a taxi-passenger recommender system based on the pick-up behaviors of high-profit taxi drivers and the mobility patterns of passengers learned from a large number of taxi trajectories. We built the recommender system with a dataset generated by 12,000 taxicabs in a period of 110 days, and evaluated the system by extensive experiments including a series of in-the-field studies. As a result, the taxi recommender accurately predicts the time-varying queue length at parking places and effectively provides the high-profit parking places; the passenger recommender successfully suggests the road segments where users can easily find vacant taxis, e.g., the top-1 road segment recommended by our system considering day of the week and weather conditions matches the ground truth for all of the tested areas. In the future, we plan to deploy our recommender in the real world so as to further validate and improve the effectiveness and robustness of this system.

References

[1] http://research.microsoft.com/apps/pubs/?id=152883.

[2] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *SIGMOD 1999*, pages 49–60. ACM Press, 1999.

[3] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[4] V. Cardellini, M. Colajanni, and P. Yu. Dynamic load balancing on web-server systems. *Internet Computing, IEEE*, 3(3):28–39, 1999.

[5] Y. Ge, C. Liu, H. Xiong, and J. Chen. A taxi business intelligence system. In *Proc. KDD 2011*, pages 735–738. ACM.

[6] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani. An energy-efficient mobile recommender system. In *Proc. KDD 2010*, pages 899–908.

[7] D. Grosu and A. Chronopoulos. Algorithmic mechanism design for load balancing in distributed systems. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(1):77–84, 2004.

[8] D. Lee, H. Wang, R. Cheu, and S. Teo. Taxi dispatch system based on current demands and real-time traffic conditions. *Transportation Research Record: Journal of the Transportation Research Board*, 1882(-1):193–200, 2004.

[9] 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 (PERCOM Workshops), 2011 IEEE International Conference on*, pages 63 –68, march 2011.

[10] Q. Li, Z. Zeng, B. Yang, and T. Zhang. Hierarchical route planning based on taxi gps-trajectories. In *Geoinformatics, 2009 17th International Conference on*, pages 1–5. Ieee, 2009.

[11] W. Massey, G. Parker, and W. Whitt. Estimating the parameters of a nonhomogeneous poisson process with linear rate. *Telecommunication Systems*, 5(2):361–388, 1996.

[12] S. Phithakkitnukoon, M. Veloso, C. Bento, A. Biderman, and C. Ratti. Taxi-aware map: Identifying and predicting vacant taxis in the city. In *Proc. AMI 2010*, page 86.

[13] S. Ross. *Stochastic processes*. Wiley, New York, 1996.

[14] H. Wu, R. Luk, K. Wong, and K. Kwok. Interpreting TF-IDF term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):1–37, 2008.

[15] K. Yamamoto, K. Uesugi, and T. Watanabe. Adaptive routing of cruising taxis by mutual exchange of pathways. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 559–566. Springer, 2010.

[16] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proc. KDD 2011*, pages 316–324. ACM.

[17] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G. Sun. An interactive-voting based map matching algorithm. In *Proc. MDM 2010*, pages 43–52.

[18] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun. Where to find my next passenger? In *Proceedings of the 13th international conference on Ubiquitous computing*. ACM, 2011.

[19] M. Ziegelmann. *Constrained Shortest Paths and Related Problems*. PhD thesis, Universität des Saarlandes, 2001.

Nicholas Jing Yuan is an associate researcher in Microsoft Research Asia. He acquired his Ph.D in Computer Science in 2012 and his B.S in Mathematics in 2007, both from University of Science and Technology of China. During the past few years, he has published a dozen papers in top conferences/journals such as KDD, TKDE, Ubicomp, ICDE. He was awarded the Microsoft Fellowship in 2011. Currently, his research interests include spatial-temporal data mining, urban computing and computational social science.

Yu Zheng is a researcher from Microsoft Research Asia and a senior member of both IEEE and ACM. His research interests include location-based services, spatio-temporal data mining, and mobile social applications. He has published over 70 referred papers at international conferences and journals, such as SIGMOD, SIGKDD, AAAI, WWW, Ubicomp, ICDE, and TKDE, and received 3 best paper awards and 1 best paper nominee. He has served the editorial board of 4 international journals and over 30 international conferences as a chair or committee member. He received his Ph.D. degree in communication & information system from Southwest Jiaotong University.

Liuhang Zhang is a master in Institute Of Computing Technology(ICT), Chinese Academy Of Sciences, under the supervision of Prof. Lixin Zhang, who is a Vice General Engineer at ICT, Chinese Academy of Sciences. He worked in Microsoft Research Asia as a full time research intern from November 2010 to June 2011, mentored by Dr. Yu Zheng and Dr. Xing Xie. His main research interests include mobile location search, spatial-temporal data mining, distributed computing and data center computing systems.

Xing Xie is a lead researcher in the Web Search and Mining Group of Microsoft Research Asia, and a guest Ph.D. advisor for the University of Science and Technology of China. He received his B.S. and Ph.D. degrees in Computer Science from the University of Science and Technology of China in 1996 and 2001, respectively. He joined Microsoft Research Asia in July 2001, working on spatial data mining, location based services, and mobile and pervasive computing. He has served on the organizing and program committees of many international conferences such as WWW, UbiComp, GIS, CIKM, and KDD. During the past years, he has published over 90 referred journal and conference papers. He is also a senior member of both ACM and the IEEE.