6-2013

HUNTS: A Trajectory Recommendation System for Effective and Efficient Hunting of Taxi Passengers

Ye DING
Hong Kong University of Science and Technology

Siyuan LIU
Carnegie Mellon University

Jiansu PU
Hong Kong University of Science and Technology

Lionel NI
Hong Kong University of Science and Technology

Citation

DING, Ye; LIU, Siyuan; PU, Jiansu; and NI, Lionel. HUNTS: A Trajectory Recommendation System for Effective and Efficient Hunting of Taxi Passengers. (2013). *MDM '13 Proceedings of the 2013 IEEE 14th International Conference on Mobile Data Management*. 1, 107-116.
Available at: https://ink.library.smu.edu.sg/sis_research/3472

HUNTS: A Trajectory Recommendation System for Effective and Efficient Hunting of Taxi Passengers

Ye Ding[1], Siyuan Liu[3], Jiansu Pu[1], Lionel M. Ni[1,2]
[1] Department of Computer Science and Engineering; [2] Guangzhou HKUST Fok Ying Tung Research Institute
Hong Kong University of Science and Technology
[3] Carnegie Mellon University
[1,2] {valency, jspu, ni}@cse.ust.hk [3] syliu@andrew.cmu.edu

***Abstract*—Nowadays, there are many taxis traversing around the city searching for available passengers, but their hunts of passengers are not always efficient. To the dynamics of traffic and biased passenger distributions, current offline recommendations based on place of interests may not work well. In this paper, we define a new problem, *global-optimal trajectory retrieving (GOTR)*, as finding a connected trajectory of high profit and high probability to pick up a passenger within a given time period in real-time. To tackle this challenging problem, we present a system, called *HUNTS*, based on the knowledge from both historical and online GPS data and business data. To achieve above objectives, first, we propose a dynamic scoring system to evaluate each road segment in different time periods by considering both picking-up rate and profit factors. Second, we introduce a novel method, called *trajectory sewing*, based on a heuristic method and the Skyline technique, to produce an approximate optimal trajectory in real-time. Our method produces a connected trajectory rather than several place of interests to avoid frequent next-hop queries. Third, to avoid congestion and other real-time traffic situations, we update the score of each road segment constantly via an online handler. Finally, we validate our system using a large-scale data of around 15,000 taxis in a large city in China, and compare the results with regular taxis' hunts and the state-of-the-art.**

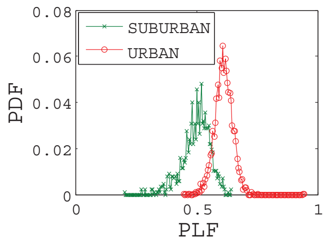


Fig. 1. The PLF of taxis, which is only 50% for suburban taxis, and around 60% for urban taxis.

Fig. 2. Passengers try to hire taxis in the midtown of Changsha, China [2]. In many cities of developing countries, there are often few taxi waiting lanes near POIs, and people have to hire taxis in the middle of the street.

I. Introduction

Taxi service is a major public transportation service in large cities nowadays, and there are often a huge number of unoccupied taxis traveling around the city. However, it is still difficult to hire taxis in crowded areas in the meantime [1]. Such problem is much more severe in large modern cities (e.g., New York, Beijing, and London). For example, Figure 1 shows the probability distribution of the *passenger load factor (PLF)* of around 4,000 taxis generated from our data of a large city in China in one month. PLF is the quotient of dividing the driving distance when a taxi is occupied by a passenger, by the total distance that the taxi has traveled. It shows the occupy rate of a taxi in distance domain. According to Figure 1, we can find that PLF is only 50% for suburban taxis, and around 60% for urban taxis. Due to the low occupy rate, a passenger-hunting recommendation system is an urgent demand of taxi drivers, as well as the city authorities, to improve the taxi utilization and reduce the energy cost.

To make a passenger-hunting recommendation, we can make use of the geographical data collected from the GPS equipments of taxis, and the business data collected from the taximeters. By analyzing the above data, traditional methods can identify the *place of interests (POIs)* [3], which are the locations with high probabilities to pick up passengers, and then recommend the best one to the taxi driver. However, a POI recommendation has several drawbacks.

First, if taxis are not allowed to wait in POIs and pick up passengers, POIs will not be suitable as recommendations. This is because if the taxi driver drives to the POI but cannot pick up a passenger, the taxi driver should then request another POI recommendation. In this situation, the efficiency of exhaustive querying is unacceptable. As a matter of fact, taxis are indeed forbidden to stop in the middle of most roads for the purpose of picking up passengers in many large cities. Moreover, in many developing countries, there are often few taxi waiting lanes near POIs, and people have to hire taxis in the middle of the street, as shown in Figure 2.

Second, the hunting trajectory of consecutive POIs may not be global-optimal. A taxi may pick up a passenger after driving through many POIs. Although each POI is the best recommendation of each step, but the resulting trajectory of the greedy POI search may not be the best selection. A

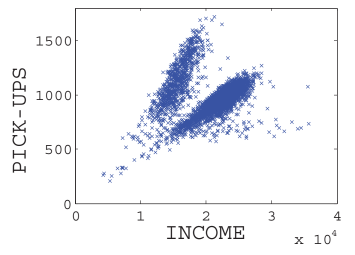


Fig. 3. The relationship between the income of taxis and the passengers that taxis have picked up in one month.

recommendation should be global-optimal comparing with all the possible hunting trajectories of the taxi, but not local-optimal on each step.

Third, the objective of a taxi is to earn money, but seeking for more passengers may not result in earning more money. In Figure 3, we can find that there are two different patterns of earning money: 1) picking up more passengers, and 2) picking up less passengers but with higher income. The reason of such phenomenon is that, a taxi may have to wait for a very long time to pick up a passenger in the terminal or the airport, but once the taxi picks up a passenger, the income could be quite high because these passengers often travel to some farther destinations. In an opposite situation, the picking-up rate may be very high in shopping malls, but passengers after shopping may travel to a near destination because they just cannot carry their stuffs. Hence, a recommendation should achieve not only higher picking-up rate, but also higher profit [4].

To overcome the drawbacks of the POI recommendation, in this paper, first, we construct a connected hunting trajectory rather than several POIs as a recommendation, and the taxi driver can drive through the trajectory until a passenger is picked up. Second, the hunting trajectory is approximate global-optimal, which can achieve more profit than a series of POI recommendations. Third, we consider both the picking-up rate and the average income of the recommendation, to make the hunting effective and efficient. Moreover, due to the constant changing of real-time traffic and guest distribution, our system can efficiently handle online data as input, and produce recommendations in real-time.

The major contributions of this paper lie in the following aspects:

- First, we create a system, called *HUNTS*, which produces hunting trajectory recommendations for taxi drivers to earn more money.
- Second, to make the recommendation, we define a new problem, called *global-optimal trajectory retrieving*, as finding a connected trajectory with high profit within a given time period in real-time. To solve the problem, we propose a novel method, called *trajectory sewing*, based on a heuristic method and the Skyline technique, to utilize each road segment and provide a recommendation.
- Third, we employ large scale real life taxi data to evaluate and compare different methods, and our experiments show that our recommendations work much better than traditional next-hop POI recommendations, and regular taxis' hunts, in terms of effectiveness and efficiency. Moreover, we have discovered some interesting findings during our study.

TABLE I
SPECIFICATIONS OF BUSINESS DATA

Data Type	**Description**
Taxi ID	Taxi registration plate number.
Begin / End Time	Begin / end timestamp of the deal.
Distance	Total distance of the deal.
Time Cost	Total time of the deal.
Price	Taxi fare of the deal.
Free Distance	The distance between two deals while the taxi is unoccupied.
Free Time	The amount of time between two deals while the taxi is unoccupied.
Taxi Company	The company id of the taxi. One company can only have one color of taxis.
Taxi Color	The color of the taxi, green or red.

TABLE II
SPECIFICATIONS OF TRACE DATA

Data Type	**Description**
Taxi ID	Taxi registration plate number.
Timestamp	Timestamp of the sample point.
Latitude / Longitude	GPS location of the sample point.
Speed	Current speed of the taxi.
Angle	Current driving direction of the taxi.
Occupied Status	Indicator of whether the taxi is occupied by a passenger.

In the following sections, we will first define the problem in Section II, and then show our system consisted of a dynamic scoring system, a recommender, and an online data handler in Section III. In Section IV, we will evaluate the accuracy and performance of our system. In Section V, we will discuss some interesting findings during our study. At last in Section VI and VII, related works and conclusions will be provided.

II. Data Description and Problem Definition

A. Data Description

Our data were collected in a large city of China in September, 2009, and it contains two datasets: one is the taxis' traces collected from GPS equipments, and the other one is the business information describing the transactions for each deal of these taxis. The format of our data is shown in Table I and Table II. Our trace data contains the information of around 15,000 taxis, and the sampling rate is about 20 seconds. Our business data contains the information of 4,197 taxis, and there are 44 deals per taxi per day on average.

B. Terminologies

Road segment: A road segment τ_i is the road between two crossroads. Some roads, like a highway, may have two road segments between two crossroads, since they have different directions. A road network $R = \{\tau_i\}_{i=1}^n$ is consisted of road segments.

Deal: A deal is the procedure from picking up a passenger to dropping off the passenger.

Taxi fare: Taxi fare is the money paid by the passenger to the taxi in the end of a deal. In this paper, taxi fare is the only income of a taxi.

Taxi status: The status of a taxi can be either unoccupied or occupied. A taxi is occupied if the taxi is handling a deal, otherwise, it is unoccupied.

Picking-up rate: Picking-up rate is the probability of picking up a passenger in a road segment.

Average income: Average income is the average taxi fare of deals which start at a road segment within a time period.

Score: For a road network $R = \{\tau_i\}_{i=1}^{n}$, where τ_i is a road segment, we define δ_i^t as the score of τ_i on time t, which is the measurement of the gain of the road segment τ_i. δ_i^t may vary upon time. As mentioned previously, the design of δ_i^t should consider both the average income and the picking-up rate of τ_i. In this paper, δ_i^t is calculated via a dynamic scoring system as shown in Figure 5, and the details will be shown in section III.

C. Problem Definition

The problem, global-optimal trajectory retrieving, is to maximize the score of a trajectory consisted of connected road segments. The formal definition of the problem is shown in Definition 1.

Definition 1. ***Global-Optimal Trajectory Retrieving (GOTR):*** *Given a start point s and a start time t_s, find a trajectory $T = \{\tau_i\}_{i=1}^{m} \subseteq R$ of maximum $\sum_{i=1}^{m} \delta_i^t$ within a time period $\hat{t}$.*

The problem definition is based on a well-designed δ_i^t as we mentioned previously. Assume we have an appropriate design of δ_i^t, the problem can be modeled to a longest path problem with time constraint. Adopting from the definition of the time-dependent shortest path problem [5], we define a time-dependent longest path problem as shown in Definition 2.

Definition 2. ***Time-Dependent Longest Path (TDLP):*** *Given an undirected FIFO graph $G = (V, E, \Delta, T)$, let $e_{i,j} \in E$ be the edge incident to v_i and v_j, and $\delta_{i,j}(t) \in \Delta$ be the weight of $e_{i,j}$, where t is a time variable in a time domain, and $t_{i,j} \in T$ be the time delay of $e_{i,j}$. Find a path $P = (s, v_1, v_2, \ldots, v_n)$ starts at s on t_s of maximum weight $\sum \delta_{i,j}$, while $\sum t_{i,j} \leq \hat{t}$.*

Different from other path-finding problems, in this paper, we do not restrict the resulting path to be a simple path. This is because if there exists a cycle with high profit, the driver can drive through the cycle repeatedly. Since there is a time bound $\hat{t}$, we can always find a feasible solution to the problem. Moreover, similar as the time-dependent shortest path problem, we consider the graph is a FIFO (First-In-First-Out) graph. The definition of FIFO graph is shown in Definition 3.

Definition 3. ***FIFO Graph:*** *Let $d(i, j, t)$ be a delay function of edge $e_{i,j}$. For any $t, t' \geq 0, t \leq t'$, we have $t + d(i, j, t) \leq t' + d(i, j, t')$.*

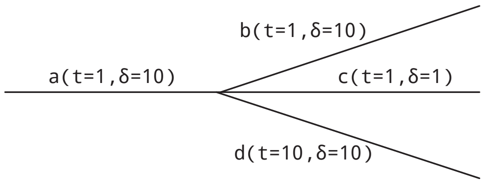


Fig. 4. An example of the global-optimal trajectory retrieving problem.

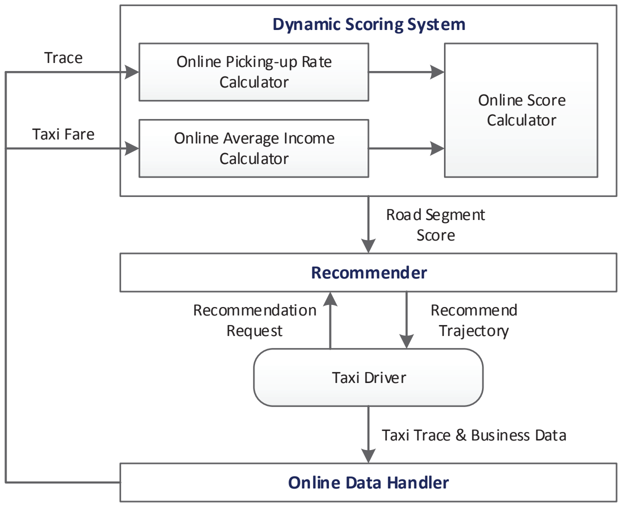


Fig. 5. System structure of HUNTS.

As shown in Definition 2, we consider the delay function $d(i, j, t) = t_{i,j}$ is irrelevant to t, hence it is a FIFO graph. Since the decision problem of the longest path problem is NP-complete, longest path problem is NP-hard. Hence the time-dependent longest path problem, as well as the global-optimal trajectory retrieving problem, are also NP-hard.

To clearly illustrate the global-optimal trajectory retrieving problem, please refer to the example of Figure 4. In Figure 4, let $\hat{t} = 2$ and $R = \{a, b, c, d\}$, we aims to find the trajectory $\{a, b\}$ rather than $\{a, c\}$ since the later one has a lower score $\sum \delta$, and also not $\{a, d\}$ since it exceeds the time limit $\hat{t}$.

III. HUNTS: The Hunting System

A. Overview

In this paper, our solution to the problem is based on the system structure shown in Figure 5. The following subsections will introduce the structure step by step in details, which include the dynamic scoring system, the recommender, and the online data handler.

B. Dynamic Scoring System

In this section, we will discuss the scoring system that evaluates the score δ_i^t for each road segment τ_i. Since the calculation for each road segment is the same, we will use δ^t instead of δ_i^t in this section for convenience. In this paper, we define δ^t as:

$$\delta^t = \frac{p^t w^t}{d} \tag{1}$$

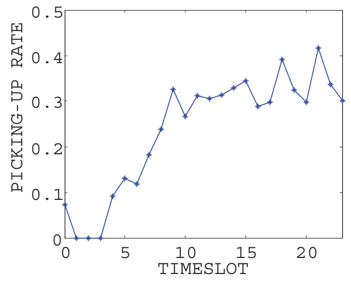


Fig. 6. Picking-up rate of one road segment in one day. The size of the timeslot is one hour, where 0 stands for 00:00 - 00:59, 1 stands for 01:00 - 01:59, etc.

where p^t is the picking-up rate at time t, w^t is the average income, and d is the length of the road segment. Since p^t and w^t are statistical, superscript t here is actually a small time period.

Our design of score is based on the following observations: 1) the higher p^t and w^t is, we more tend to choose the road segment since it is more possible to pick up a passenger with higher income in the road segment; and 2) we more tend to choose short road segments rather longer ones, since δ^t may vary in different parts of a very long road segment, and it is only higher because of some parts of the road segment. In conclusion, we design the score of a road segment in a time period by considering higher p^t and w^t, and lower d. In the following subsections, we will discuss how to generate these three values.

1) Picking-up Rate: To generate the picking-up rate, we divide the number of taxis which picked up passengers in a road segment, by the total number of taxis which passed by the road segment while unoccupied. Since picking-up rate may vary upon time, e.g., the picking-up rate could be much higher on 19:00 when people are getting off work than 03:00 when people barely appear on street, we use a timeslot t to do the division for each road segment as follows:

$$p^t = \frac{|status(0 \to 1)|_t}{|status(0)|_t} \tag{2}$$

where $|status(0 \to 1)|$ is the number of taxis which passed by the road segment and changed their status from unoccupied to occupied, and $|status(0)|$ is the number of unoccupied taxis which passed by the road segment. Here we define status 0 as unoccupied, and status 1 as occupied. Figure 6 shows the picking-up rate of one day generated using Formula 2.

In Figure 6, we can find that in the midnight (01:00-03:00), the picking-up rate is almost 0 since it may not be possible to pick up a passenger while people are sleeping. The picking-up rate remains high during the night (21:00-24:00), and it may because although the number of passing-by taxis is small, these taxis often successfully pick up passengers.

Before calculating the picking-up rate, we should do a map matching to map the sample points to road segments. In this paper, we use a greedy map matching algorithm [6][7] as shown in Algorithm 1.

Algorithm 1 Greedy Map Matching Algorithm

```
1: Separate the map into a set of subspaces $S = \{s_i\}_{i=1}^n$ of equal size, where each $s_i$ contains at least one road segment;
2: Find the subspace $s_i$ where the sample point $p \in s_i$;
3: $s'_i \leftarrow s_i \cup$ neighbors of $s_i$;
4: $d_{min} \leftarrow \infty$;
5: $r \leftarrow \emptyset$;
6: for each road segment $\tau_j \in s'_i$ do
7:     Find the minimum perpendicular distance $d$ from $p$ to $\tau_j$;
8:     if $d < d_{min}$ then
9:         $d_{min} \leftarrow d$;
10:        $r \leftarrow \tau_j$;
11:    end if
12: end for
13: Resulting $r$ is the matching of $p$.
```

Algorithm 2 Algorithm of Minimum Perpendicular Distance

```
1: $d_{min} \leftarrow \infty$;
2: for each component segments $\gamma_i \in$ road segment $\tau_j$ do
3:     $d \leftarrow$ perpendicular distance of $p$ to $\gamma_i$;
4:     if $d < d_{min}$ then
5:         $d_{min} \leftarrow d$;
6:     end if
7: end for
8: Resulting $d_{min}$ is the minimum perpendicular distance from $p$ to $\tau_j$.
```

Algorithm 1 maps a sample point to the nearest road segment of minimum perpendicular distance. In this paper, since a road segment is actually a polyline, we define the minimum perpendicular distance as the minimum distance between a point p and all the component segments of a road segment. The algorithm of finding minimum perpendicular distance is shown in Algorithm 2. Algorithm 1 has a time complexity of $O(n^2)$, because it enumerates all the road segments and calculates the perpendicular distance of each road segment.

To increase performance, we split the map into grid subspaces before finding the minimum perpendicular distance. For each sample point, only the road segments lie in the same subspace of the sample point are compared. Nevertheless, a border problem, as shown in Figure 7, may occur if we only consider the subspace that the sample point lies in. In Figure 7, although p is close to B, but since $p \in A$, p will be mapped to A. In this paper, we also consider the eight neighbors of the subspace to avoid the problem, as shown in line 3 of Algorithm 1.

Figure 8 shows the traffic of urban area generated from our trace data using Algorithm 1. The average traffic in the off-work time is 14.85 cars per road segment, while it is 7.11 cars per road segment in the midnight. In Figure 8, the gradient color from red to green shows the traffic situation from busy to free, where busy indicates more cars in the road segment.

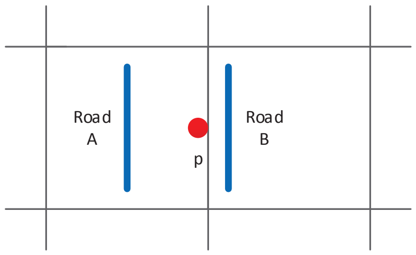
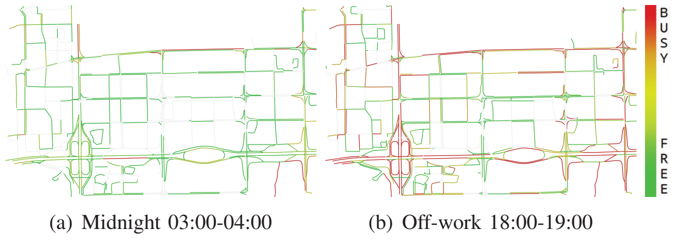


Fig. 7. The border problem of map matching. p is close to B, but since $p \in A$, p will be mapped to A.



(a) Midnight 03:00-04:00 (b) Off-work 18:00-19:00

Fig. 8. Traffic of urban area in one day. Off-work time is obviously busier than midnight.

It is obvious that off-work time is busier than midnight, but not the same road segments. It explains the importance of the time constraint.

2) Average Income: In this section, we will discuss how to generate the average income for each road segment. As mentioned previously, our observation shows that always heading to the road segment with highest picking-up rate may not be the best choice, since passengers in the terminal or the airport may travel much farther than those in normal streets or business areas. Hence, we should also consider the average income of a road segment as one part of the score.

Similar as finding the picking-up rate, we use the timeslot to calculate the average income of each road segment. In this paper, we use the start time of a deal as the indicator of which timeslot the deal belongs to. For example, if a deal starts at 23:50 on September 1st and ends at 00:10 on September 2nd, we consider the deal belongs to September 1st, timeslot 23:00-23:59 if the timeslot size is one hour.

In this paper, we only consider the average income of taxis which picked up passengers in the road segment. The calculation of the average income is shown in Formula 3.

$$w^t = \frac{\sum fare(0 \rightarrow 1)_t}{|status(0 \rightarrow 1)|_t} \tag{3}$$

where $|status(0 \rightarrow 1)|$ is the number of taxis which passed by the road segment and changed their status from unoccupied to occupied, and $fare(0 \rightarrow 1)$ is the taxi fare of those taxis. Figure 9 shows the average income of one road segment in one day. In figure 9, we can find that the average income in the midnight is zero since there are barely people hiring taxis in the midnight, and the average income during daytime is about 25 dollars, which is reasonable according to our experience.

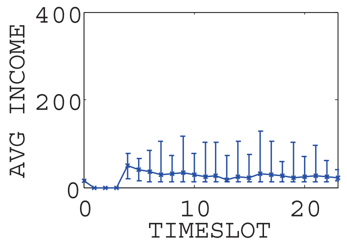
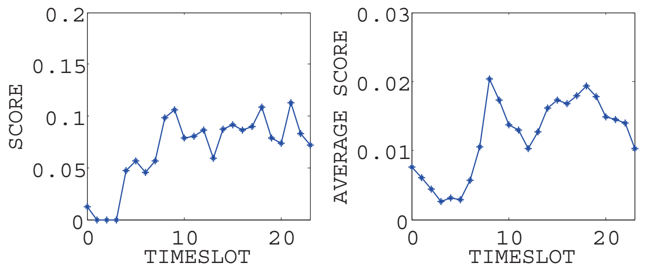
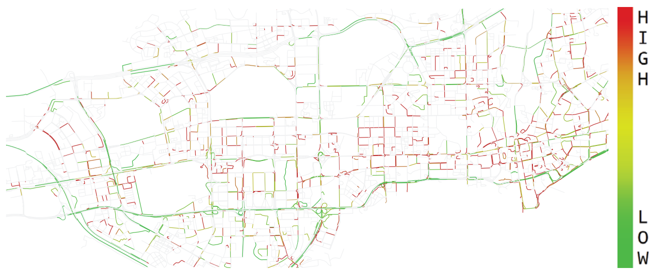


Fig. 9. Average income of one road segment in one day.



(a) Score of one road segment in one day. (b) Average score of all road segments in one day.



(c) Average score of urban area on 18:00-19:00 of one day. The color of red to green shows the score of higher to lower, and grey stands for no score, i.e, no passengers are picked up by any taxis on the road segment.

Fig. 10. Score of one road segment and the average score of all road segments in one day.

3) Scoring: Once we have the picking-up rate and the average income, we can calculate the score δ^t based on Formula 1. Figure 10 shows the score of one road segment, as well as the average score of all road segments in one day. In Figure 10, we can find that the score is higher on average during commuting (around 08:00 and 18:00), and lower during midnight.

C. Recommender

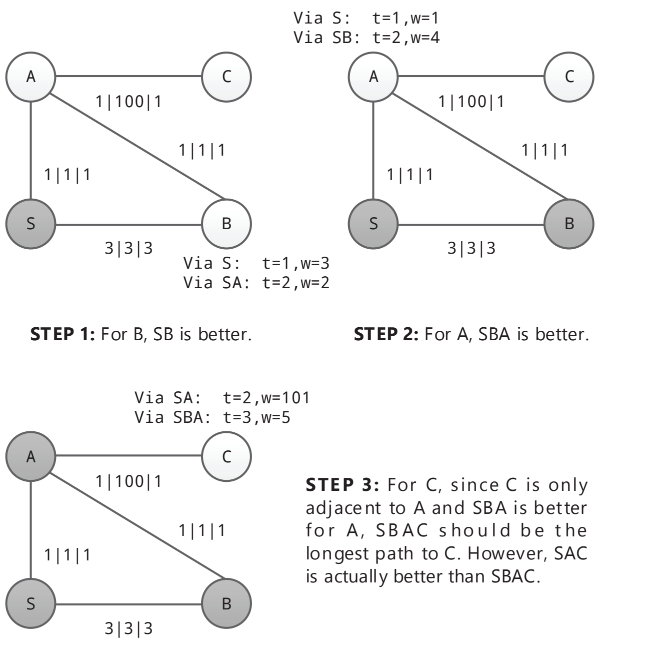
Once the score of all the road segments are obtained, we can recommend a trajectory to a taxi given its current location and time. In this section, we will introduce our recommender which produces a connected trajectory based on the score calculated previously.

Recall Definition 1, the recommender aims to find a trajectory $T = \{\tau_i\}_{i=1}^m \subseteq R$ with maximum $\sum_{i=1}^m \delta_i^t$ within a time period $\hat{t}$ from the start point s with time t_s. As mentioned previously in Definition 2, we can model the global-optimal trajectory retrieving problem to a time-dependent longest path problem. Although the time-dependent longest path problem seems like a longest path problem, or a time-dependent shortest path problem, the solutions to these path-finding problems cannot be directly adopted to solve the time-dependent longest path problem. This is because of three major limitations:

1) A traditional longest path problem often considers weight of the graph is static. But in a dynamic weighted graph, since weight varies upon time, we cannot compare two weights in different time, hence traditional algorithms may not work. For example, in Figure 11, assume we start at S and try to find the longest path in the graph, where the time delay of each edge is one, and the weight of each edge is assigned as shown in the figure. In the first step, since SB with weight 3 is better than SAB with weight 2, we consider SB is the longest path to B. Similarly in the second step, SBA with weight 4 is better than SA with weight 1 for A. Now consider the idea of dynamic programming in a static weighted graph, in the third step, since C is only adjacent to A, the longest path to C should be equal to the longest path to A plus the weight of AC, which is 5 via $SBAC$. However, since the weight of AC varies upon time, we may miss the longer path SAC with weight 101. The dynamic weighting problem is the main difference between the longest / shortest path problem and time-dependent longest / shortest path problem. Similar as those shortest path algorithms based on static graphs cannot be used to solve the time-dependent shortest path problem, traditional algorithms of longest path problem cannot be directly adopted to solve the time-dependent longest path problem, too.
2) A traditional path-finding problem often assume there is only one weight (often the physical distance) associated with each edge in the graph, but in this paper, there are two weights should be considered: score and time delay. Since we have to compromise between these two weights, it makes the problem very difficult, and traditional algorithms often do not consider such situation.
3) As mentioned in Definition 2, different from traditional path-finding problems, in a real-life situation, we do not restrict the resulting path to be a simple path, i.e., cycles are allowed. Hence traditional algorithms cannot be adopted.

To solve the global-optimal trajectory retrieving problem, a straight-forward method is to exhaustively enumerate all the feasible trajectories and find the one with the highest score. Algorithm 3 shows the exhaustive algorithm.

1) Exhaustive Algorithm: In algorithm 3, we construct a list $\{T_i\}$ of trajectories, and enumerate all the possible trajectories start at s within the time period $\hat{t}$. Finally it returns the trajectory with highest score $T_i(\delta)$. As mentioned in Definition 1 and Definition 2, each road segment τ_i is associated with its score δ_i^t and time delay t_i. Similarly, for each T_i, $T_i(\delta)$ is the total score of the trajectory, and $T_i(t)$ is the total time delay.



Via S: t=1,w=1
Via SB: t=2,w=4
A
C
S
B
1|100|1
1|1|1
1|1|1
3|3|3
Via S: t=1,w=3
Via SA: t=2,w=2
STEP 1: For B, SB is better.
STEP 2: For A, SBA is better.
Via SA: t=2,w=101
Via SBA: t=3,w=5
STEP 3: For C, since C is only adjacent to A and SBA is better for A, SBAC should be the longest path to C. However, SAC is actually better than SBAC.

Fig. 11. Dynamic weighting problem.

Algorithm 3 Exhaustive Algorithm
```
1: $T_0 \leftarrow \{s\}$;
2: while there exists an open $T_i$ where $T_i(t) < \hat{t}$ do
3:   for each subsequent road segment $\tau_j$ of $\tau_m \in T_i$ do
4:     Create $T' \leftarrow T_i \cup \tau_j$;
5:     if $T'(t) > \hat{t}$ then
6:       Drop $T'$;
7:     end if
8:   end for
9:   if all $T'$ are dropped then
10:    Mark $T_i$ as closed;
11:  else
12:    Remove $T_i$;
13:  end if
14: end while
15: Resulting $T_i$ with $max(T_i(\delta))$ is the optimal selection.
```

Each T_i is consisted of several road segments $\{\tau_1, \tau_2, \ldots, \tau_m\}$ sorted by arriving time, and they are connected one by one.

In this paper, if two road segments share a same crossroad, i.e., two edges are incident to a same vertex in the graph, we say the two road segments are neighbors. Since each edge is incident to two vertices, we can partition the neighbors of a road segment into two sets N_L and N_R. For a road segment $\tau_j \in T_i$, assume $\tau_{j-1} \in N_L$ and $\tau_{j+1} \in N_R$, we say the road segments in N_R are the subsequent road segments of τ_j with respect to τ_{j-1}. The reason of introducing the subsequent road segments is that, one cannot drive back to the previous road segment, or it will violate the FIFO restriction. The details are shown in Figure 12.

During the enumeration process, Algorithm 3 behaves like breadth-first search, and it stops when there are no more

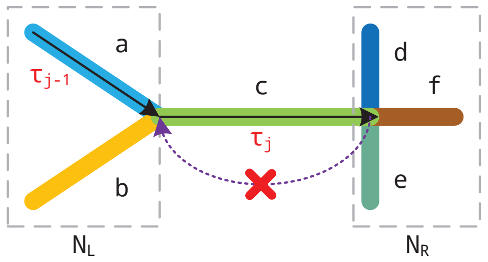


Fig. 12. An example of the subsequent road segments. Let $\tau_j = c$ and $\tau_{j-1} = a$, then $N_L = \{a, b\}$, $N_R = \{d, e, f\}$. Clearly the taxi cannot drive back to N_L, or the time delay of c will double. Hence the recommender can only recommend the road segments in N_R, which are the subsequent road segments of τ_j with respect to τ_{j-1}.

Algorithm 4 Greedy Algorithm (GA)

```
1: $T \leftarrow \{s\}$;
2: while $T(t) < \hat{t}$ do
3:     Find the subsequent road segment $\tau_j$ of $\tau_m \in T$ with the
       highest score among all the subsequent road segments
       of $\tau_m \in T$;
4:     Append $\tau_j$ to $T$;
5: end while
6: Resulting $T$ is the approximate selection.
```

possible trajectories. Assume the lowest time delay of all the road segments is t, Algorithm 3 takes at most $\hat{t}/t$ loops. Assume the largest number of subsequent road segments of all road segments is r, for each loop, Algorithm 3 creates at most r new trajectories. Hence, the time complexity of Algorithm 3 is at most $O(1 + r + r^2 + \cdots + r^{\hat{t}/t}) \le O(r^{\hat{t}/t+1})$. Since all possible trajectories are enumerated, Algorithm 3 is an exact algorithm.

2) Greedy Algorithm: As mentioned previously, a traditional recommender often recommends the road segment with the highest probability to pick up a passenger to the taxi. After driving to the road segment, if the taxi cannot pick up a passenger, a new recommendation will be generated. Such next-hop recommendation repeats until the taxi picks up a passenger. If we consider there is a time limit $\hat{t}$ of the recommendation, the consecutive next-hop recommendation is actually a greedy algorithm as shown in Algorithm 4.

Algorithm 4 is an approximation algorithm, as it tries to find the best subsequent road segment at each step, but these road segments may not lead to an optimal result. The accuracy limitation of Algorithm 4 is the main reason why we introduce the following two new algorithms.

3) Heuristic Algorithm: Inspired by the A^* algorithm, in this paper, we introduce a new heuristic algorithm as shown in Algorithm 5, which is faster than Algorithm 3, and more accurate than Algorithm 4.

In Algorithm 5, we use a heuristic estimation to evaluate the optimality of a trajectory:

$$h(T) = T(\delta) + (\hat{t} - T(t)) * \rho \tag{4}$$

where ρ is the highest time-efficiency value of the road

Algorithm 5 Heuristic Algorithm (k-HA)

```
1: Create empty stack $I$;
2: $T \leftarrow \{s\}$;
3: Push $T$ into $I$;
4: $T^* \leftarrow null$;
5: while $I$ is not empty do
6:     Pop a trajectory from $I$ to $T$;
7:     for each subsequent road segment $\tau_j$ of $\tau_m \in T$ do
8:         Create $T' \leftarrow T \cup \tau_j$;
9:         if $T'(t) \le \hat{t}$ and $T'(\delta) + (\hat{t} - T'(t)) * \rho \ge T^*(\delta)$
           then
10:            Push $T'$ into $I$;
11:            if $T'(\delta) > T^*(\delta)$ then
12:                $T^* \leftarrow T'$;
13:            end if
14:        end if
15:    end for
16:    Reduce the size of $I$ to $k$ by removing lower-scored
       trajectories in $I$;
17: end while
18: Resulting $T^*$ is the approximate selection.
```

segments within a feasible region, which is the largest sphere that a taxi can possibly reach within $\hat{t}$. The time-efficiency of a road segment is the quotient of dividing its score by its time delay.

Different from Algorithm 3, during each iteration of Algorithm 5, we check whether the new trajectory is possible to be optimal by estimating the highest score it can reach. If the highest score it can reach is even lower than the score of the best trajectory so far, it will be pruned. Furthermore, in the end of each iteration, we reduce the size of the stack to a user-defined higher bound k, by removing lower-scored trajectories in the stack. The higher k is, the more accurate trajectory we can find. If k is set to 1, Algorithm 5 will find the same trajectory as using Algorithm 4. Hence the accuracy of Algorithm 5 is higher than Algorithm 4, and its time complexity is lower than Algorithm 3.

Similar to other heuristic algorithms, Algorithm 5 is step-to-step approaching to the optimal solution. Hence we can extract a part of the resulting trajectory to the taxi while running the algorithm. This is another benefit of Algorithm 5.

4) Trajectory Sewing Algorithm: To improve the accuracy of Algorithm 5, we introduce another novel method, called trajectory sewing, by using the Skyline operation [8] as shown in Algorithm 6.

In this paper, we define the domination of the Skyline operation in two dimensions: time delay $T(t)$ and score $T(\delta)$. If T_1 has a lower time delay and a higher score than T_2, we say T_1 dominates T_2. For all $T \in I$, if there exists another $T' \in I - T$ that dominates T, T will be pruned. Since the Skyline operation considers two dimensions rather than one dimension in Algorithm 5, it is more accurate than Algorithm 5.

In conclusion, benefits from our scoring scheme, our rec-

Algorithm 6 Trajectory Sewing Algorithm (TS)

```
1: ...
2: while $I$ is not empty do
3:     ...
4:     Remove trajectories which can be dominated by others in $I$;
5: end while
6: ...
```

ommender 1) considers the time influence of picking up passengers; 2) considers both picking-up rate and average income of a road segment; 3) produces a connected trajectory rather than several POIs; and 4) finds an approximate global-optimal trajectory.

D. Online Data Handler

As mentioned previously, since trace data and business data are always streaming to the system, we should update the score of road segments consecutively. In this paper, the score of a road segment should always be the average of history. However, since traffic and urban plan may change in real life, there is no need to calculate the average of all the historical values of score. Hence, we use the simple moving average to calculate the average of score:

$$SMA(n+1) = SMA(n) - \frac{v_{n+1-m}}{m} + \frac{v_{n+1}}{m} \tag{5}$$

where $SMA(n)$ is the simple moving average of n numbers, v_n is the value the n-th number, and m is the size of a stack M. When we are calculating the simple moving average of $n+1$ numbers, we pop one number from the stack M, and insert the (n+1)-th number into the bottom of M, and then get the average value of the stack via Formula 5.

One critical problem of calculating the average is that, trace data and business data may not come simultaneously. Trace data is always available from GPS, whereas business data is only available when the taxi drops off a passenger and reports the taxi fare of the deal. Hence in this paper, we update the picking-up rate and the average income separately, and recalculate the score after the updating.

$$\delta^t = \frac{SMA(p^t)SMA(w^t)}{d} \tag{6}$$

Since the simple moving average requires a stack size m, δ^t is always using the recent m records. This implies only m days of data are used to train the score of road segments if δ^t is updated daily. Please note that the average here could be day average, or week average, i.e., one can assign seven score values to a road segment and each score value represents one day of a week. The reason of using week average is that the picking-up rate and the average income may be different on weekdays and weekends. In this paper, we use day average in the experiments.

With the online data handler, we do not have to calculate the score every time when recommending a trajectory, but just query the latest score from the database. Such online data handler dramatically increases the efficiency of the system.

IV. Evaluation

A. Experiment Data

In this paper, we train our system using the historical data of one week, which includes 15,231 taxis with 154 million records. We use the online data handler to import all the records, and use a timeslot of 10 minutes to calculate the score of road segments. We compare our recommendations with both ground truth data and traditional methods, and the comparison covers both accuracy and performance. Since the online data handler is used to import data, and it can be proved as accurate, there is no need to evaluate it. The comparison includes:

1) **GT**: The ground truth hunting trajectories of taxis in one day that start at t_s and end at $t_s + \hat{t}$.
2) **GA**: The recommendation generated via the traditional next-hop greedy algorithm (Algorithm 4).
3) **TS**: The recommendation generated via the trajectory sewing algorithm (Algorithm 6).
4) **10-HA**: The recommendation generated via the heuristic algorithm (Algorithm 5), where $k = 10$. Since the heuristic algorithm requires a very long time to execute, we only retrieve the result within three minutes.

The exhaustive algorithm is not included in the experiments because it requires too much time to execute, which is not acceptable. To compare the recommendations in different situations, we conduct 90 different types of queries based on different start time t_s, different start point s, and different time period $\hat{t}$. In details:

1) t_s: 09:00:00, 14:00:00, and 18:00:00.
2) s: 35 random road segments.
3) $\hat{t}$: 300 seconds, 600 seconds, and 1800 seconds.

In conclusion, we generate 315 recommendations per method and 36,534 ground truth trajectories. Since the accuracy of a recommendation is not related to both t_s, s, and $\hat{t}$, we treat all the recommendations equally.

B. Accuracy

In this paper, we use the unit potential income to evaluate the accuracy of trajectories. As mentioned previously, the picking-up rate reflects the probability of picking up a passenger, and the average income reflects the potential income if the taxi picks up a passenger. Hence, we can evaluate the effectiveness of a recommendation via its picking-up rate, and the efficiency via its average income. However, we cannot evaluate the two metrics in separate ways, since efficiency depends on effectiveness. For example, if the picking-up rate of a recommendation is very low, no matter how high the average income is, the recommendation still makes no sense. Hence in this paper, we evaluate the accuracy in terms of effectiveness and efficiency of each road segment via its *potential income* as:

$$f^t = p^t w^t \tag{7}$$

where f^t is the potential income of a road segment at time t, p^t is the picking-up rate, and w^t is the average income if the taxi

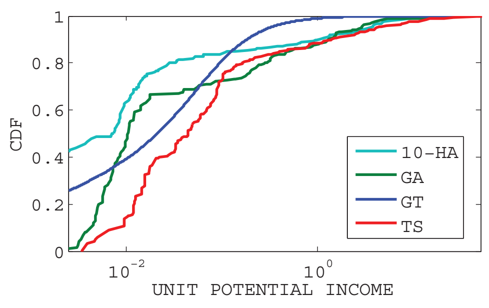
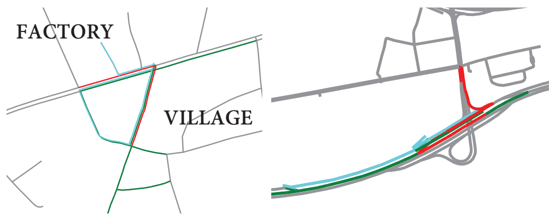


Fig. 13. CDF of the average potential income of GT, GA, TS, and 10-HA.



(a) Case 1 (b) Case 2

Fig. 14. Comparing recommendations via different methods, GA is colored green, 10-HA is colored cyan, and TS is colored red. The start point of each case is the road segment colored by all three colors.

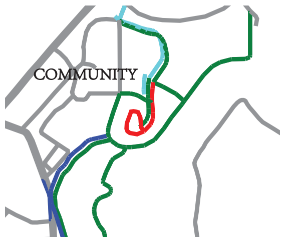


Fig. 15. Comparing recommendations with ground truth hunting trajectories, GA is colored green, 10-HA is colored cyan, TS is colored red, and GT is colored blue.

picks up a passenger. To fairly compare the potential income of trajectories, we use the value of potential income per unit distance as the *unit potential income* of the trajectory. In this paper, we use 100 meters as the unit length. The comparing of accuracy is shown in Figure 13.

In Figure 13, we can find that the accuracy of TS is always better than GA for around 70% of experiments. In addition, all the recommendations generated by TS are better than the ground truth hunting trajectories. It explains that a taxi driver may be experienced in finding shortest paths, but not in hunting for passengers.

To clearly show the recommendations, let us look at some examples in Figure 14. In case 1, the road segments recommended by TS are the entrance of some factories, as well as the restaurants in the other side of the main road. The area in the south of the main road is a countryside village (villages are often near factories in China because of low labor costs), and the recommendations here make no sense. In case 2, the road segments recommended by GA are through a highway with high time cost but no passengers. This is because traditional next-hop recommendations are lack of a global view of real-life situations. These cases explain that our trajectory sewing method is better than the traditional next-hop method. In Figure 15, the circle recommended by TS is a community entrance, and other road segments are the links between the community and the main road. In real life, passengers often wait in the entrance of the community to hire taxis. We can find that regular taxis do not aware this situation. This case explains that our recommendation generated by TS is indeed better than regular taxis' hunts.

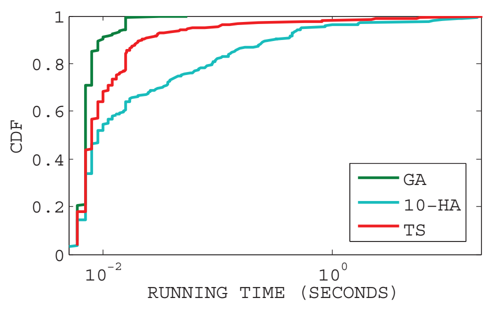


Fig. 16. CDF of execution time of 10-HA, TS, and GA.

In conclusion, our design of the score and correspondent algorithms, especially the trajectory sewing algorithm (Algorithm 6), achieves better accuracy comparing with traditional next-hop recommendations, as well as regular taxis' hunts.

C. Performance

In our experiments, we use a computer with Intel Core 2 Duo CPU @ 2.53GHz and 4 GB physical memory to evaluate the performance of our system, and all the experiments are running within one thread. As mentioned previously, we conduct 90 experiments with different s, t_s, and $\hat{t}$ for the three algorithms. Since s is a random variable, we use the average execution time of experiments that with different s but same t_s and $\hat{t}$ as the execution time of the algorithm upon t_s and $\hat{t}$. Figure 16 shows the performance of the three algorithms.

Intuitively, we know the execution time of the three algorithm should be GA$\leq$TS$\leq$10-HA, and it is so according to Figure 16. Although TS requires more time than GA to execute, its execution time is still in seconds, hence we believe it is acceptable.

V. Discussions

We discover several interesting things when exploring our data. For example, there are two types of taxis in our data, one is red, which is allowed to drive through the whole city, and another one is green, which is allowed to drive in the suburb only. The driving patterns of these two types of taxis are quite different. Green taxis tend to pick up more passengers, while red ones tend to pick up few passengers but with longer distances. The reason of the choices could be the area limitations. Figure 17 shows the relationship between distance and pick-ups of green and red taxis in one month.

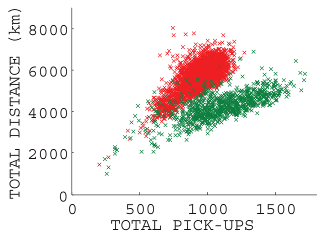


Fig. 17. Relationship between distance and pick-ups of green and red taxis in one month.

Another interesting discovery is that, the driving patterns of a taxi between before and after picking up a passenger are quite different. For example, the average speed after picking up is much higher than before. Moreover, the traces of taxis after picking up passengers have many commons, while they are almost totally different before. This also proves that taxi drivers are experienced drivers in path-finding, but not in passenger hunting.

VI. Related Work

Trajectory mining: Mining knowledge from trajectories has been studied for a long time. Some works studied the features of moving objects from their trajectories [9]; some works studied the techniques of trajectory analyzing, including calibration [10], classification [11], clustering [12][13], anomaly detection [14], etc; some works made further analysis based on these techniques, like in [15], they detected flawed road segments in the view of urban planning according to taxi trajectories. The knowledges discovered in this field are the basis of route planning.

Route planning: Route planning is an important application that leveraging the knowledge discovered from historical trajectories, which includes taxi hunting route recommendation [4], energy-efficient driving route recommendation [16], fast driving route recommendation [17] [3], personalized route planning [18], dynamic route planning [19] [20] [21], etc. These works are either focus on identifying POIs, or combining popular route fragments. In this paper, we focus on both identifying POIs and combining these POIs as a connected trajectory.

VII. Conclusion and Future Work

In this paper, we create a system, called HUNTS, to make hunting trajectory recommendations for taxi drivers to earn more money. To make recommendations, we study a new problem, called global-optimal trajectory retrieving, and to solve the problem, we propose a novel trajectory sewing method, based on the score which considers both picking-up rate and average income of each road segment. The recommendations are approximate global-optimal trajectories rather than several POIs. At last, we evaluate the accuracy and the performance of our system, by comparing with traditional methods and ground truth data.

Choosing routes is like gambling, if all the taxis in the same road segment choose the same route at the same time, the picking up rate could be lower because of too many competitors. In the future, we will study the game intelligence of taxi competitions, and which is also a challenging problem.

VIII. Acknowledgment

This paper was supported in part by Huawei Corp. under Contract YBCB2009041-27; Hong Kong, Macao and Taiwan Science & Technology Cooperation Program of China under Grant No. 2012DFH10010; Science and Technology Planning Project of Guangzhou, China, under Grant No. 2012Y2-00030; Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office; and a University Transportation Center grant (DTRT12-G-UTC11) from the US Department of Transportation.

References

[1] Taxi number stopped growing for 10 years in beijing. Beijing Daily. [Online]. Available: http://politics.people.com.cn/n/2012/0816/c1001-18753269.html

[2] Taxi number stopped growing for 8 years in changsha. Changsha Evening. [Online]. Available: http://www.voc.com.cn/Topic/article/201109/201109061730026480.html

[3] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *ACM SIGKDD 2011*.

[4] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun, "Where to find my next passenger?" in *UbiComp 2011*, 2011.

[5] A. Ziliaskopoulos and H. Mahmassani, *Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications*, 1993.

[6] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate gps trajectories," in *ACM SIGSPATIAL GIS 2009*, 2009.

[7] O. Pink and B. Hummel, "A statistical approach to map matching using road network geometry, topology and vehicular motion constraints," in *IEEE ITSC 2008*, 2008.

[8] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *IEEE ICDE 2001*, 2001.

[9] Q. He, K. Chang, and E.-P. Lim, "Analyzing feature trajectories for event detection," in *ACM SIGIR 2007*.

[10] S. Liu, C. Liu, Q. Luo, L. Ni, and R. Krishnan, "Calibrating large scale vehicle trajectory data," in *IEEE MDM 2012*, 2012.

[11] J. Lee, J. Han, X. Li, and H. Gonzalez, "Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering," in *VLDB 2008*, 2008.

[12] S. Liu, Y. Liu, L. M. Ni, J. Fan, and M. Li, "Towards mobility-based clustering," in *ACM SIGKDD 2010*.

[13] J. gil Lee and J. Han, "Trajectory clustering: A partition-and-group framework," in *ACM SIGMOD 2007*, 2007.

[14] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework," in *IEEE ICDE 2008*.

[15] Y. Zheng, Y. Liu, J. Yuan, and X. Xie, "Urban computing with taxicabs," in *UbiComp 2011*, 2011.

[16] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani, "An energy-efficient mobile recommender system," in *ACM SIGKDD 2010*.

[17] J. Yuan, Y. Zheng, C. Zhang, and W. Xie, "T-drive: Driving directions based on taxi trajectories," in *ACM SIGSPATIAL GIS 2010*.

[18] J. Letchner, J. Krumm, and E. Horvitz, "Trip router with individualized preferences (trip): Incorporating personalization into route planning," in *AAAI 2006*, 2006.

[19] H. Kanoh and K. Hara, "Hybrid genetic algorithm for dynamic multi-objective route planning with predicted traffic in a real-world road network," in *GECCO 2008*, 2008.

[20] N. Malviya, S. Madden, and A. Bhattacharya, "A continuous query system for dynamic route planning," in *IEEE ICDE 2011*, 2011.

[21] J. Xu, L. Guo, Z. Ding, X. Sun, and C. Liu, "Traffic aware route planning in dynamic road networks," in *DASFAA 2013*, 2012.