2012 IEEE 33rd Real-Time Systems Symposium

$pCruise$: Reducing Cruising Miles for Taxicab Networks

Desheng Zhang and Tian He
Department of Computer Science and Engineering, University of Minnesota, Twin Cities, USA

***Abstract*—In taxicab industry, a long standing challenge is how to reduce taxicab's mileage spent without a fare, *i.e.*, cruising mile. The current solution for this challenge usually requires the participation of the passengers. To solve the issue without the passengers involved, in this paper, we propose a cruising system, $pCruise$, for taxicab drivers to maximize theirs profits by finding the optimal route to pick up a passenger, thus reducing the cruising mile. In $pCruise$, base on collected GPS records about other near taxicabs, a taxicab characterizes its cruising process with a *cruising graph*. When a taxicab becomes vacant and tries to find a passenger, *cruising graph* will provide the shortest cruising route with at least one expected available passengers for this taxicab. With the shortest cruising routes, taxicabs will significantly reduce theirs cruising miles. We evaluate $pCruise$ based on a 7 days 10 GB real world GPS dataset from a city with more than $15,000$ taxicabs. The evaluation results show that $pCruise$ can assist taxicab drivers to reduce cruising miles by 41% on average.**

I. Introduction

Nowadays, among all different transportation, taxicabs play an particularly prominent role in big metropolitan residents' daily commute. According to a recent survey in New York City [1], over 100 taxicab companies operate more than $13,000$ taxicabs in New York City. There is a fairly stable taxicab ridership of $660,000$ passengers per day, and taxicabs in New York City transport more than 25% of all paying passengers, accounting for 45% of all fares paid.

Taxicab service availability is one of most important service quality in taxicab industry. In a New York City taxicab survey, the top ranking unsatisfactory reason about taxicabs is that taxicabs are not available when needed [2], but nearly 40% of total taxicab mileage, a total of 314 million miles per year, is spent to cruise for passengers. Thus, it is important to address this discord between the high cruising miles for drivers and the low ride availability for passengers. This issue can be tackled from two different perspectives. From the passengers' perspective, the straightforward solution is to spend the minimal amount of time or distance to find an available taxicab on the street; from the drivers' perspective, the obvious solution is to find a passenger with the minimal amount of cruising miles (*i.e.*, miles spent without a fare). In this paper, we focus on the perspective of taxicab drivers to maximize their profits by reducing cruising miles.

In large metropolitan areas of United State and China, *e.g.*, New York City, Beijing, Shanghai, and Shenzhen, taxicabs are equipped with GPS and communication devices, and they upload their vehicle statuses back to taxicab companies' dispatching centers periodically. Thus, these dispatching centers can schedule taxicabs to the optimal routes to pick up passengers, which will reduce the cruising miles. However, the current solution in dispatching centers requires the participation of passengers. Typically, the existing dispatching centers function under the scenario where a passenger contacts a dispatching center, and then the dispatching center assigns a task about this passenger to a nearby vacant taxicab accordingly. But most passengers will hail a taxicab along the street directly, rather than booking a taxicab from a dispatching center. Therefore, we face a challenge that how to schedule taxicabs to find passengers with the minimal length routes to reduce their cruising miles, yet without the participation of passengers.

In this paper, we propose a cruising system, $pCruise$, *i.e.*, cruising with purposes, which schedules the taxicabs based on pick-up events from GPS records about their nearby taxicabs in taxicab networks. The key insight about $pCruise$ is that it utilizes only several key GPS records to model a cruising process about a taxicab, and then provides a scheduling strategy to find a passenger with the minimal cruising miles. Specifically, our key contributions are as follows:

- We introduce a mathematical concept, *cruising graph*, for $pCruise$, where vertices represent intersections and edges represent road segments connecting intersections. We characterize a cruising graph by weights on its edges, which are represented by the expected number of arrival passengers during a taxicab cruises on road segments.
- According to characterization on edges, $pCruise$ utilizes a cruising graph by an efficient scheduling. During the scheduling, $pCruise$ will select the cruising route for a taxicab with at least one arrival passenger yet having the minimal length, thus reducing the cruising miles for the taxicab driver.
- More importantly, we evaluate $pCruise$ with a real world 10 GB dataset, collected from taxicabs in Shenzhen, China. The dataset consists of 7 days of GPS traces from more than $15,000$ taxicabs. Based on this dataset, we conduct both a large scale trace-driven simulations and a small scale testbed experiment. The evaluation results show that with the assistance of $pCruise$, a taxicab driver can reduce the cruising miles by 41% on average.

The rest of the paper is organized as follows. In Section II, we present our design goal. In Section III, we propose our $pCruise$ design. In Section IV, we evaluate $pCruise$ through simulation and testbed study. In Section V, we present related work. In Section VI, we conclude the paper.

1052-8725/12 $26.00 © 2012 IEEE
DOI 10.1109/RTSS.2012.61

II. Design Goal

Our work is mainly motivated by the observation that the taxicab drivers suffer from long cruising miles. On the other hand, passengers cannot find a taxicab when they need one. We bridge this gap by proposing a cruising system which can guide vacant taxicabs to find a passenger with the minimal cruising miles, thus maximizing the profits of taxicab drivers. To illustrate the current cruising miles in a taxicab network and our design goal, we plot the average percentage of miles without passengers among total miles, *i.e.*, cruising miles, by comparing Ground Truth GPS traces and the traces obtained by a trace-driven simulation where 10% of total taxicabs using $pCruise$. The evaluation setup is given in Section IV.

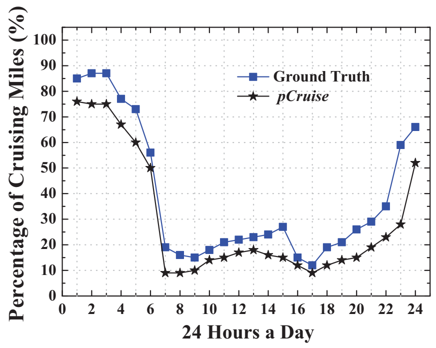

Fig. 1. Illustration of $pCruise$ Design Goal.

Figure 1 plots the percentages of cruising miles among total miles on every 1 hour time window from $0:00$ to $23:59$ of a day, comparing the ground truth of a real world dataset and a trace-driven simulation under $pCruise$. From this figure, we observe that during the rush hours of a day, *e.g.*, from $06:00$ to $10:00$ or from $16:00$ to $20:00$, both of two schemes have percentages of cruising miles below 20%; during the non-rush hours of a day, *e.g.*, from $00:00$ to $06:00$, percentages of cruising miles are higher than 60%. But $pCruise$ has a better performance than Ground Truth during the rush or non-rush hours by 41% and 21% on average, respectively. This figure illustrates the design goal of $pCruise$, which is to reduce the cruising miles for taxicab drivers.

Given the existence of a plethora of dispatching and data mining schemes that extract the knowledge from the entire GPS dataset, we decide to design $pCruise$ as a lightweight yet effective system which can be implemented either in frontend, *i.e.*, taxicabs, or in backend, *i.e.*, dispatching centers. In addition, we design an efficient scheduling for $pCruise$ based on the GPS records of their own nearby taxicabs, instead of the entire GPS dataset. Since every taxicab has different nearby taxicabs and starting time, $pCruise$ introduces unpredictability and randomness in the system to compensate for more taxicabs head to the same area with the same cruising route. $pCruise$ provides a unified solution for a highly diverse, heterogeneous route selection scheme that may be deployed at individual mobile devices from various applications.

III. *pCruise* Design

In this section, we introduce the detailed design for *pCruise*. We first present the main idea of *pCruise*, which is to model a taxicab's cruising process to reduce cruising miles. Then, we demonstrate how to model a cruising process. Further, we explain how to characterize a cruising process with GPS records collected from nearby taxicabs. Finally, we show how to utilize a characterized cruising process to select the optimal cruising routes for taxicab drivers to reduce the cruising miles.

A. Main Idea

There are two basic situations for a taxicab, *i.e.*, driven with a fare or driven without a fare. Total miles spent with a fare are called live miles, while total miles spent without a fare are called ***cruising miles***, which consists of multiple ***cruising routes***. In *pCruise*, the optimal strategy to maximize drivers' profits is to minimize every cruising route and then to minimize the total cruising miles, while finding at least one expected passenger on every minimal cruising route. Note that to design *pCruise* with a more generic philosophy, we describe *pCruise* as a distributed solution, even though it can be easily customized to a centralized solution.

The key idea of *pCruise* is simple. During both live and cruising miles, every taxicab broadcasts its GPS records to other taxicabs periodically, while collecting GPS records from nearby taxicabs. Whereas during cruising miles, *pCruise* provides a model for a cruising process based on collected GPS records. This model can be used to find the optimal cruising route with the minimal length, while satisfying that at least one expected passenger will arrive during the cruising process.

Several questions rise up with respect to this idea.

1) How to model a cruising route: When a vacant taxicab is cruising on the street, a taxicab driver can make decisions about cruising routes anytime, but can only take actions at an intersection on the street. Therefore, an intersection is a basic yet crucial unit for the optimal cruising route. To capture fundamental characteristics about a cruising process on every intersection, we propose a mathematical concept *Cruising Graph*. Subsection *B* will elaborate this concept.

2) How to characterize the optimality of a cruising route: To characterize the optimality of a route on a given cruising graph, we characterize a cruising graph by assigning weights on its edges, representing corresponding road segments of a cruising route. The weights represent the lower bound of expected number of available passengers that will arrive on this road segment, during time period that a taxicab is cruising on it. This expected number of available passengers can be obtained by two metrics we proposed. Subsection *C* introduce the details of other two parameters.

3) How to utilize the characterized optimality to find the optimal cruising route: Based on the characterization of a cruising process, we propose an efficient scheduling scheme *pCruise* to obtain the optimal cruising route with the minimal cruising length, while satisfying that at least one passenger will arrive on this route during the cruising. Subsection *D* will present this scheduling scheme.

B. Creation of Cruising Graph

In this subsection, we demonstrate how to model a taxicab cruising process. Note that to show the key principle of *pCruise*, we assume a road map is given, and directly present a model based on it. But in the appendix, we provide a simple yet effective method to model a cruising process without a corresponding road map.

We model a taxicab cruising process with a mathematical concept, *i.e.*, ***Cruising Graph***. A cruising graph is a simple graph where vertices represent intersections and edges represent road segments between intersections. The key step to create a cruising graph is to identify intersections and road segments connecting them, which can be performed by several schemes [3] [4]. To fucus on a system level, we now assume that intersections are given in a road map, but in the appendix we propose a simple method to identify intersections. For every intersection, we create a corresponding vertex; for every two adjacent vertices, we create the two directed edges between them from one to another and *vice versa*. The rationale behind creating two directed edges, instead of one undirected edge, between two adjacent vertices is that the optimality from one intersection I_1 to another intersection I_2 could be total different to that from I_2 to I_1. In addition, in some one-way situations, it does not even exist. Figure 2 shows a cruising graph created according to a given road map.

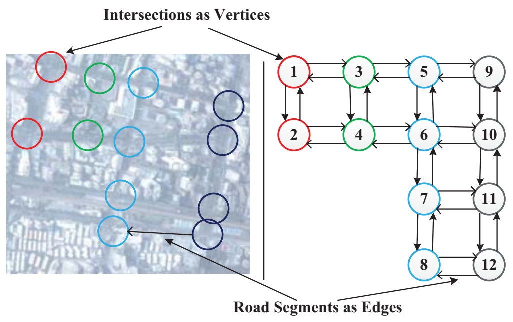

Fig. 2. Cruising Graph

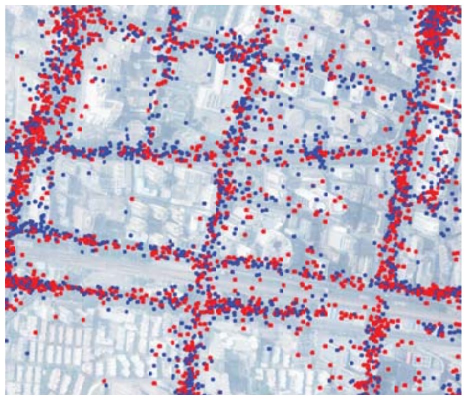

Fig. 3. Pick-up or Drop-off Spots. The most of the pick-up or drop-off spots is on main streets, only a small subset of them is off the street. In addition, pick-up or drop-off spots are roughly uniformly distributed on main streets.

C. Characterization of Cruising Graph

In this subsection, given a cruising graph obtained in last subsection, we characterize it by assigning weights on its edges, indicating the attractiveness of cruising on corresponding road segments. The key feature of our characterization is that we do not rely on the entire GPS trajectories, only a small portion of records, whereas most of current schemes functions with the entire GPS trajectories. Therefore, before presenting the characterization, we first introduce how to utilize GPS records about taxicab networks in Subsection *1)*.

With collected GPS records, we characterize an edge, *i.e.*, road segment $[s_{begin}, s_{end}]$ in a time duration $[t_{begin}, t_{end}]$ with a novel metric: ***Available Passenger Ratio*** ρ. It indicates that given an arbitrary location $s_x \in [s_{begin}, s_{end}]$ and an arbitrary moment $t_x \in [t_{begin}, t_{end}]$, the probability that there is an *available*, *i.e.*, unserved, passenger in two dimensional temporal-spacial spot $p_x = [t_x, s_x]$. We propose the available passenger ratio ρ in Subsection *2)*.

The available passenger ratio ρ alone is not enough to characterize a road segment, and the number of total passengers (including both available and unavailable) arriving at the road segment is also very important. This is because in a road segment with a high available passenger ratio ρ yet a low number of total arrival passengers, the number of available arrival passengers will still be small. But with current GPS records, even with the entire trajectories of all taxicabs, it is impractical to obtain such accurate number of passengers arriving at certain road segments. Nevertheless, with current GPS records, we can obtain with another metric on a road segment: ***Passenger Arrival Rate*** λ, which indicates the lower bound of the total passenger arrival rate, given a road segment $[s_{begin}, s_{end}]$ and a time duration $[t_{begin}, t_{end}]$. We present the passenger arrival rate λ in Subsection *3)*.

1) GPS Records: In $pCruise$, every taxicab will broadcast its GPS records to other nearby taxicabs periodically. A GPS record consists of following parameters: (1) Plate Number; (2) Date and Time; (3) GPS Coordinates; and (4) Availability Bit: whether or not a passenger is in this taxicab when the record is broadcasted. Thus, we can map GPS coordinates received by a taxicab into a temporal-spacial coordinate system where all GPS records are represented by points.

Instead of considering all GPS records, in $pCruise$, we shall focus on the records with a change on Availability Bit compared to previous records. For example, if an Availability Bit turns to 1 from 0 in two consecutive records of a Taxicab i, then it indicates that this taxicab just *picks up* a passenger in a location indicated by the GPS coordinates. Therefore, we name this physical location s_i and corresponding moment t_i as a *pick-up spot* $p_i = (t_i, s_i)$. Similarly, we name a physical location as a *drop-off spot*, if an Availability Bit turns to 0 from 1. Figure 3 gives an example of these spots on a road map, which implies that pick-up or drop-off spots can be treated as representative samples for all GPS records to reduce the total number of processed GPS records for characterization of a taxicab cruising process on a cruising graph.

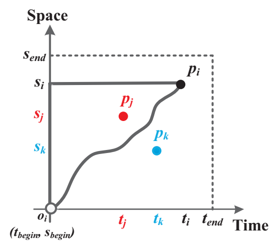

Fig. 4. ρ' in One Taxi Scenario

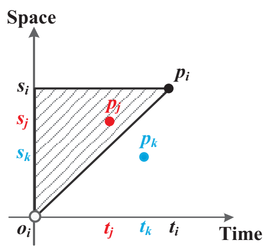

Fig. 5. ρ in One Taxi Scenario

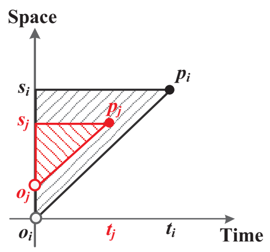

Fig. 6. ρ in Two Taxis Scenario

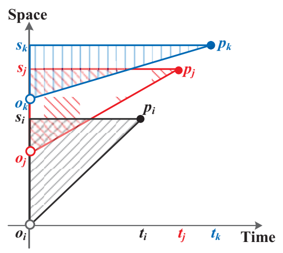

Fig. 7. ρ in Three Taxis Scenario

2) Available Passenger Ratio ρ: In general, a road segment with more available passengers will lead to a high possibility of pick-up, thus minimizing cruising miles for a taxicab driver. In *pCruise*, a taxicab driver shall favor a road segment with a higher available passenger ratio ρ, which can be obtained by collected pick-up spots in GPS records, via observing how long nearby vacant taxicabs picked up passengers on this road segment before. The sooner a nearby taxicab picked up a passenger on this road segment, the higher available passenger ratio ρ on this road segment. In the following, we show how to quantify ρ based on pick-up spots.

For the simplicity, when computing a weight on a road segment, we transform a two-dimensional GPS coordinate (x, y) into a one dimensional variable $s \in [s_{begin}, s_{end}]$ on this road segment. Based on a pick-up spot p_i of Taxicab i, we can map p_i in a temporal-spacial cartesian coordinate system in terms of a pick-up location s_i on a road segment and a pick-up moment t_i, as in Figure 4. The origin of coordinates o_i is (t_{begin}, s_{begin}) where t_{begin} is the moment that Taxicab i enters this road segment; s_{begin} is the beginning location of this road segment.

If *pCruise* is under the assumption that all the trajectories are given, in this two dimensional temporal-spacial system, the triangle-like shape $\triangle' o_i s_i p_i$ indicates a temporal-spacial area without any available passenger. This is because if there is an available passenger within $\triangle' o_i s_i p_i$, *e.g.*, a spot $p_j = (t_j, s_j)$ in Figure 4, then the pick-up location of Taxicab i should be s_j, instead of s_i. But there could be an available passenger outside $\triangle' o_i s_i p_i$, *e.g.*, a spot $p_k = (t_k, s_k)$ in Figure 4, since at moment t_k, Taxicab i has already passed location s_k, and cannot verify whether or not there is a passenger waiting at location s_k from moment t_k, by picking the passenger up.

Therefore, in this 1 taxicab scenario, for a road segment r connecting an intersection m and an intersection n, we can obtain the available passenger ratio $\rho_r'^{(1)}$ at a time period of $[t_{begin}, t_{end}]$ as follows.

$$\rho_r'^{(1)} = 1 - \frac{|\triangle' o_i s_i p_i|}{|t_{end} - t_{begin}| \times |s_{end} - s_{begin}|},$$

where $|\triangle' o_i s_i p_i|$ is the area of $\triangle' o_i s_i p_i$. The physical meaning of $|t_{end} - t_{begin}| \times |s_{end} - s_{begin}|$ is the total 2 dimension in terms of time and space, while the physical meaning of $|\triangle' o_i s_i p_i|$ is the time and space that confirmed by Taxicab i that there is no passengers. Note that in Figure 4, any available passenger can exist within square $|t_{end} - t_{begin}| \times |s_{end} - s_{begin}|$ but outside $\triangle' o_i s_i p_i$. This is the reason why ρ is the probability that there exists an available passenger in arbitrary location $s_x \in [s_{begin}, s_{end}]$ at arbitrary moment $t_x \in [t_{begin}, t_{end}]$.

In the above analysis, we utilize the trajectory of Taxicab i to obtain $\triangle' o_i s_i p_i$. However, since *pCruise* is designed to employ only a few pick-up spots instead of all the trajectories, we approximate the triangle-like shape $\triangle' o_i s_i p_i$ with the triangle $\triangle o_i s_i p_i$ in Figure 5. The rationale behind this approximation is that in a road segment we can assume that a taxicab is with relatively even speed. We will verify the effect of this approximation on Evaluation section. After the approximation, we have a new available passenger ratio $\rho_r^{(1)}$ as in Figure 5, which is given by

$$\rho_r^{(1)} = 1 - \frac{|\triangle o_i s_i p_i|}{|t_{end} - t_{begin}| \times |s_{end} - s_{begin}|},$$

where $|\triangle o_i s_i p_i|$ is the area of $\triangle o_i s_i p_i$.

Figure 6 and 7 consider multiple taxicabs scenarios where the triangles of every taxicab overlap with each other. The union area of all triangles indicates an area without any available passengers in terms of space and time. For example, in Figure 6, when Taxicab i enters the road segment from o_i, Taxicab j has already been on this road segment at o_j. At moment t_j, Taxicab j first picks up a passenger at location s_j. After that at moment t_i, Taxicab i then picks up a passenger at location s_i. Since $t_j < t_i$ and $s_j < s_i$, $\triangle o_j s_j p_j$ is inside of $\triangle o_i s_i p_i$. Note that even though there is a passenger at point p_j that is inside $\triangle o_i s_i p_i$, but this passenger is not *available* for Taxicab i, since she or he is picked up by Taxicab j.

Therefore, for 2 taxicabs scenario in Figure 6,

$$\rho_r^{(2)} = 1 - \frac{|\triangle o_i s_i p_i| \cup |\triangle o_j s_j p_j|}{|t_{end} - t_{begin}| \times |s_{end} - s_{begin}|}.$$

Similarly, for 3 taxicabs scenario in Figure 7,

$$\rho_r^{(3)} = 1 - \frac{|\triangle o_i s_i p_i| \cup |\triangle o_j s_j p_j| \cup |\triangle o_k s_k p_k|}{|t_{end} - t_{begin}| \times |s_{end} - s_{begin}|}.$$

To generalize the above results, for an edge connecting vertex m and vertex n, associating the road segment r connecting intersection m and intersection n, we have

$$\rho_r = 1 - \frac{\cup_{\forall i \in I} |\triangle o_i s_i p_i|}{|t_{end} - t_{begin}| \times |s_{end} - s_{begin}|},$$

where I is a set of all pick-up spots.

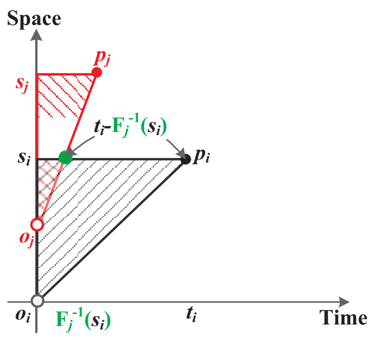

Fig. 8. λ under Scenario 1

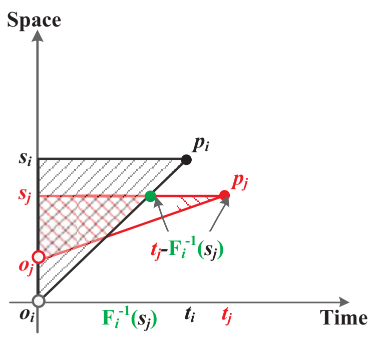

Fig. 9. λ under Scenario 2

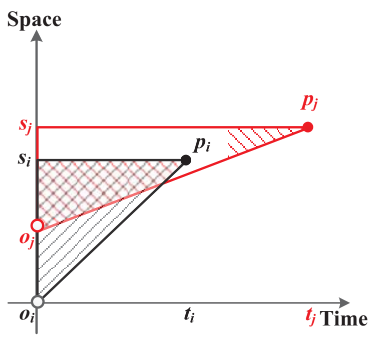

Fig. 10. λ under Scenario 3

3) Passenger Arrival Rate λ: In this subsection, we introduce λ, which indicates the lower bound of passenger arrival rate for a road segment $[s_{begin}, s_{end}]$ in time duration $[t_{begin}, t_{end}]$. With λ, we can have the lower bound of total arrival passenger number at a time duration.

To obtain an accurate passenger arrival rate, we need a time period and the number of passengers arriving during this time period. But based on the collected GPS records alone, it is impractical to obtain such accurate information. Alternatively, we may obtain the lower bound of passenger arrival rate λ^{s_i} at a pick-up location s_i during a time period.

For example, in Figure 8, at moment t_{begin}, Taxicab j is ahead of Taxicab i, but the pick-up location s_i of Taxicab i is ahead of the pick-up location s_j of Taxicab j. This indicates when Taxicab j arrives in location s_i at moment $F_j^{-1}(s_i)$, there is no available passenger, where F_j^{-1} is the inverse function of line segment $\overline{o_j p_j}$. This is because if these is an available passenger, then the pick-up spot p_j of Taxicab j should be $(F_j^{-1}(s_i), s_i)$, instead of (t_j, s_j). But since the pick-up spot of Taxicab i is $p_i = (t_i, s_i)$, it indicates that when Taxicab i arrives at location s_i, there is a passenger in location s_i at moment t_i. This implies that *at least* one passenger arriving at location s_i in time duration $[F_j^{-1}(s_i), t_i]$. Note that there may be more than one passenger at location s_i arriving at duration $[F_j^{-1}(s_i), t_i]$, but since a taxicab can pick up only one passenger at a time, we only can confirm that *at least* one passenger arrived. This is the reason why λ is the lower bound of passenger arrival rate.

Therefore, the lower bound of passenger arrival rate at a pick up location s_i is given by

$$\lambda^{s_i} = \frac{1}{t_i - F_j^{-1}(s_i)}.$$

The similar situation is in Figure 9 where at least one available passenger arrived at the location s_j in time duration $[F_i^{-1}(s_j), t_j]$. But in Figure 10, we observe that a situation where λ cannot be computed, because no pick-up happens when a later taxicab passes a former taxicab's cruising route.

Finally, the lower bound of passenger arrival rate at road segment r during $[t_{begin}, t_{end}]$ is given as follows.

$$\lambda_r = \Sigma_{i \in I} \lambda^{s_i},$$

where I is a set of all pick-up spots.

D. Utilization of Cruising Graph

In this subsection, based on the creation and characterization of a cruising graph, we present how to utilize a characterized cruising graph to select the optimal cruising route with minimal cruising miles, while having a reasonable number of passengers. This is because that a taxicab driver shall select a cruising route with at least one passenger arrival, since a taxicab can only take one passenger at a time. Note that in our context a passenger means a ridership which may contain more than one person.

Given a cruising graph, we can have different routes for a taxicab driver by different traversal strategies. Since every route consists of several road segments, we characterize a route based on its road segments, which are evaluated by the two metrics proposed, *i.e.*, Available Passenger Ratio ρ and Passenger Arrival Rate λ. During the time period that a taxicab is cruising on a selected route R, several passengers may arrive at every road segment on this route R. Since there are other competing taxicabs, among total arrival passengers, only some of them are *available* for pick-ups of this taxicab. Therefore, for a road segment r in a route R, given its crossposting passenger arrival rate λ_r and available passenger ratio ρ_r, we can obtain Available Passenger Arrival Rate, *i.e.*, $\rho_r \times \lambda_r$. In addition, for a given road segment r, the time spent on it, τ_r, can be obtained by a road segment length set S based on GPS records. Therefore, we can obtain the expected number of arrival available passengers, κ_r, on the road segment r by

$$\kappa_r = \rho_r \times \lambda_r \times \tau_r.$$

Since a cruising route R consists of multiple route segments, we can obtain the expected number of arrival available passengers κ_R on a route R by the following.

$$\kappa_R = \sum_{r \in R} \kappa_r = \sum_{r \in R} \rho_r \times \lambda_r \times \tau_r.$$

Based on the above formula of κ_R, we can obtain the expected number of arrival available passengers in every intersection of a route. Since only one available passenger will be enough for a taxicab, we compute all the routes with the expected number of arrival available passengers larger than 1. Among these candidate routes, we select the one with minimal length as the optimal route to reduce the cruising miles.

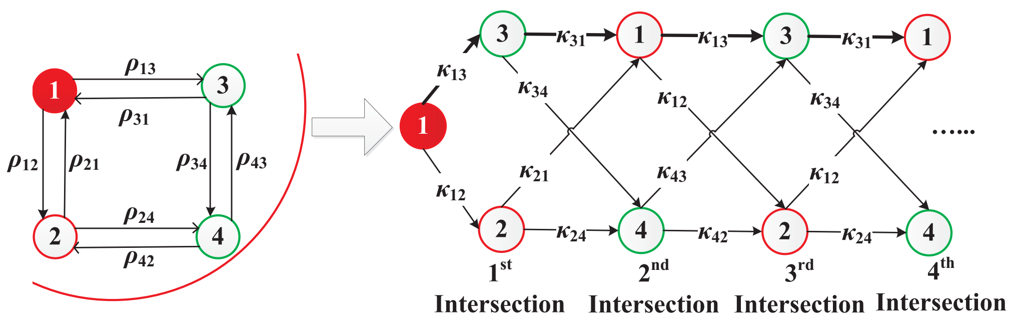

Fig. 11. Utilization of Cruising Graph. Given a cruising graph in the left, we give an example how $pCruise$ utilizes it to obtain the optimal cruising route for a Taxicab i starting at intersection 1. During an instance of a depth-first traversal, $pCruise$ will compute the expected number of arrival available passengers κ_R by the unit of road segments between intersections. For example, in the first depth-first traversal (bold) with intermediate vertices 1, 3, 1, 3, 1, 3,..., the first vertex traversed is vertex 3 which is corresponding to the intersection 3 adjacent to the intersection 1. So the expected number of arrival available passengers, κ_R^1, on this traversal at 1^{st} intersection is $\kappa_R^1 = \kappa_{13} = \rho_{13} \times \lambda_{13} \times \tau_{13}$. κ_R^2 on this traversal at 2^{nd} intersection is $\kappa_R^2 = \kappa_{13} + \kappa_{31}$. Similarly, $\kappa_R^3 = \kappa_{13} + \kappa_{31} + \kappa_{13}$. In ith traversed vertex, $pCruise$ compares κ_R^i with 1. If κ_R^i is larger than 1, then $pCruise$ ends this traversal, and considers it as a candidate for the optimal cruising route.

Given a cruising graph G and a road segment length set S, the following algorithm describes the scheduling scheme of $pCruise$, which will provide the optimal cruising route R with minimal cruising miles.

Algorithm 1 *pCruise* Scheduling

```
Require: (1) Cruising Graph $G$; (2) Road Segment Length Set $S$;
Ensure: Optimal Cruising Route $R$;
1: Based on $G$, perform a depth-first traversal; and obtain a route $R'$ whose expected number of arrival available passengers $\kappa_R$ is larger than 1;
2: Compute the cruising miles of $R'$ based on road segment length set $S$, and compare it with the cruising miles of current optimal cruising route $R$;
3: Select the cruising route from $R'$ or $R$ with shorter cruising miles as the new optimal cruising route $R$;
4: Continue to perform depth-first traversal until all cruising routes with expected number of arrival available passengers larger than 1 are obtained;
```

In the above scheduling, $pCruise$ obtains all qualifying cruising routes one by one, and compares them with the current optimal cruising route R. If there is a cruising route R' shorter than the current optimal cruising route R, then R' becomes the new current optimal cruising route R. $pCruise$ scheduling is over when all qualifying cruising routes are considered, and the running time of this algorithm is bounded by the terminal condition that the expected number of arrival available passengers is larger than 1.

In Figure 11, we give an example of the utilization of cruising graph in $pCruise$ scheduling. Based on the cruising graph in the left figure, $pCruise$ can compute the all qualifying cruising routes that satisfy the terminal condition in the right of Figure. To show the principle of our $pCruise$ scheduling, we assume that a taxicab can only receive or broadcast the GPS records from or to other taxicabs within one block.

IV. Trace-driven Evaluation

In this section, to examine the effectiveness of *pCruise*, we perform a large-scale simulation and a small-scale testbed in subsection A and B, respectively.

To evaluate *pCruise* in a real-world scenario, we collected a real world 10 GB dataset about 7 days of GPS traces of 15405 taxicabs belonging to different taxicab companies in Shenzhen, a Chinese city with 10 million population. This dataset is from an government agency for urban transportation pattern search. The dataset is obtained by letting every taxicab upload 4 records with a speed $1/s$ to redundantly report its GPS trace records to a base station every 30 seconds on average. A record mainly contains following information: **(1) Plate Number:** as a primary key for every taxicab; **(2) Date and Time:** date and time of every record uploaded; **(3) Availability:** whether or not a passenger is on this taxicab when the record is uploaded; **(4) GPS Coordinates:** GPS coordinates when the record is uploaded. Based on the dataset of above GPS trace records, we can obtain location and time distributions of pick-up events, which are used to evaluate the performance of *pCruise*.

To show the effectiveness of *pCruise*, we compare the performance of *pCruise* with **Ground Truth**, which is the original GPS traces from the dataset without any modifications. In addition, since *pCruise* only employs the pick-up spots, instead of the entire trajectories, we also compare *pCruise* with an **Oracle** scheme, which can store and process all the collected trajectories of nearby taxicabs, and utilizes the triangle-like shape $\triangle'$ to compute available passenger ratio ρ' as in Section III.*C.2)*. In contrast, *pCruise* utilizes the triangle shape $\triangle$ to obtain ρ based on only pick-up spots.

The performance metric is the percentage of cruising miles in total driving miles. We investigate this metric on every 1 hour time window of a day. In addition, we investigate the sensitivities of *pCruise*'s performances on two key parameters of taxicab networks, *i.e.*, broadcasting speed as well as broadcasting range.

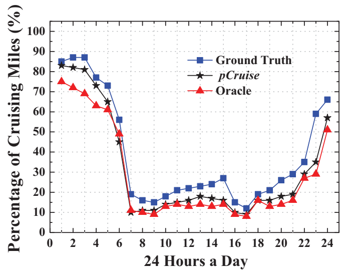

Fig. 12. Cruising Miles VS. Time of One Taxicab

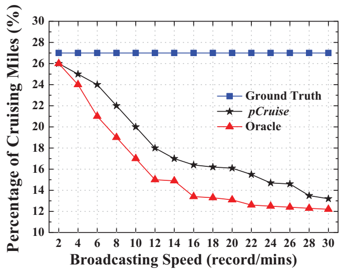

Fig. 13. Cruising Miles VS. Speed of One Taxicab

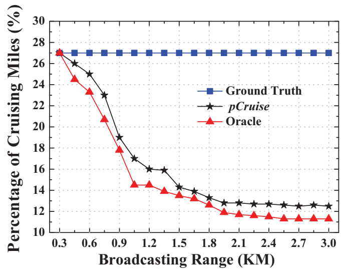

Fig. 14. Cruising Miles VS. Range of One Taxicab

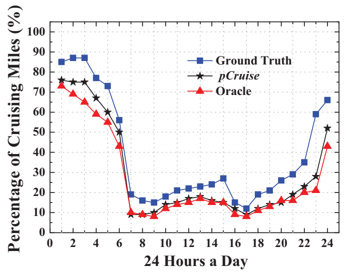

Fig. 15. Cruising Miles VS. Time of 10% Taxicab

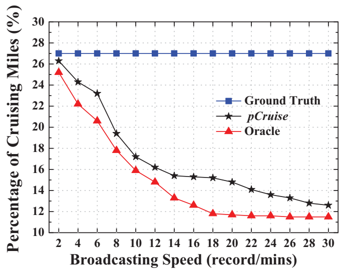

Fig. 16. Cruising Miles VS. Speed of 10% Taxicab

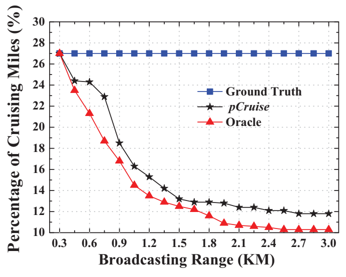

Fig. 17. Cruising Miles VS. Range of 10% Taxicab

A. Trace-driven Simulation Evaluation

In this simulation, we report the results of only one taxicab or 10% of total taxicabs using *pCruise* or Oracle.

1) Only one taxicab with pCruise or Oracle: Figure 12 plots the percentage of cruising miles in every 1 hour time window of a day. We observe that in the rush hours of a day, *e.g.*, $06:00-10:00$, the percentages of cruising miles for all three schemes are below 20%. In contrast, in non-rush hours of a day, *e.g.*, $00:00-6:00$, the percentages of cruising miles for all three schemes are over 60%. But during every 1 hour time window of a day, we observe that both *pCruise* and Oracle outperform Ground Truth with an average gain of 32% and 37%, which verifies the effectiveness of taking nearby taxicab's activities into consideration. In addition, Oracle outperforms *pCruise* by 11% on average in non-rush hours of a day, and by 3% on average in rush hours, indicating that even though in rush hours Oracle considers the entire trajectories, the effect is limited. It implies during rush hours, pick-up spots alone used by *pCruise* can effectively reduce the cruising miles, and considering the entire trajectories in rush hours is not as effective as in non-rush hours.

Figure 13 plots the effect of different broadcasting speeds on the percentage of cruising miles. In Figure 13, we observe that with the increase of the speed, the percentages of cruising miles in *pCruise* and Oracle decrease significantly as much as 36%, when the speed is larger than 6 records per mins, and the decreasing slows down when the speed is larger than 12 records per mins. This is because when the speed first becomes larger, *pCruise* and Oracle will have more frequently updated GPS records to obtain the optimal route, but when the speed is larger than 12 records per mins, the records received by taxicabs are becoming redundant, and cannot be efficiently used for route selections. We also observe that Oracle outperforms *pCruise* slightly when the speed is small, *e.g.*, 6 records per mins. The gain is becoming bigger when the speed is between 6 to 22 records per mins.

Figure 14 plots the effect of different broadcasting ranges on the percentage of cruising miles. We observe that with the increase of ranges from 0.3KM to 3.0KM, the percentage of cruising miles in *pCruise* and Oracle continuously decrease. When the range is larger than 0.6KM, the decrease becomes more significant, but when the range is larger than 1.5KM, it becomes more stable. This is because when the ranges become larger, *pCruise* and Oracle will select a route based on more nearby taxicabs' information. But when the range is larger than 1.5KM, the GPS records about the taxicabs that are far away cannot receive any related information about nearby road segments. Thus, the decrease of cruising miles becomes less obvious. Again, Oracle always has a better performance than *pCruise* by average 8, which thanks to the utilization of entire trajectories in Oracle.

2) 10% of total taxicabs with pCruise or Oracle: Figure 15 plots the performance of 10% of total taxicabs in terms of the percentage of cruising miles. We observe that in the rush hours of a day, *e.g.*, $06:00-10:00$ or $16:00-18:00$, the percentages of cruising miles for all three schemes are below 20%. In contrast, in non-rush hours of a day, *e.g.*, $00:00-6:00$ or after $22:00$, the percentages of cruising miles for all three schemes are significantly higher than these of rush hours. In addition, both *pCruise* and Oracle outperform Ground Truth with an average gain of 36% and 41%, which implies that when more taxicabs take other pick-up spots into consideration, the whole system has a better performance. Furthermore, Oracle outperforms *pCruise* by 8%. This performance gain decreases compared to one taxicab using *pCruise* scenario, which implies that *pCruise* performs better when more taxicabs employ *pCruise*.

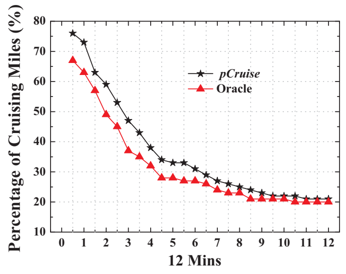

Fig. 18. Cruising Miles VS. Time in Testbed

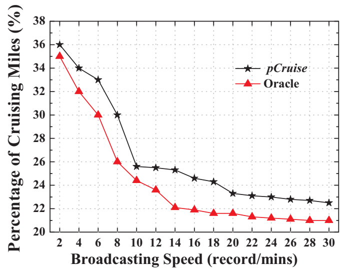

Fig. 19. Cruising Miles VS. Speed in Testbed

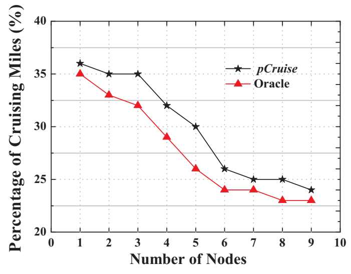

Fig. 20. Cruising Miles VS. Density in Testbed

Figure 16 and 17 plot the effects of different broadcasting speeds and ranges on the percentage of cruising miles, respectively. In both Figure 16 and 17, we observe that with the increases of broadcasting speeds and ranges, the percentage of cruising miles in Ground Truth keeps the same, while those in *pCruise* and Oracle decrease significantly when the broadcasting speed and range is larger than 6 record/mins and 0.75KM, respectively. Compared to results in Figure 13 and 14, the results in Figure 16 and 17 have the similar tendency yet with better performances. But the values from which the percentages of cruising miles decrease significantly are smaller. One explanation for this is that when more taxicabs take other pick-up spots into consideration, the routes used by taxicabs with *pCruise* or Oracle will distribute more uniformly on the entire road network, which reduces the cruising miles.

B. Trace-driven Testbed Evaluation

To evaluate *pCruise* in a real world setting, we conduct a testbed experiment with the real world taxicab dataset.

1) Testbed Setup: We implement *pCruise* and Oracle in fully connected networks with 10 TelosB sensor devices on the TinyOS/Mote platform. We simulate two street blocks for a mobile toy car attached with a TelosB node as in Figure 21, which provides the toy car 6 intersections and 7 road segments to make different routes. The other 9 nodes serve as nearby taxicabs to broadcast normalized GPS records about this two street blocks based on the real-world dataset. The normalization of GPS records is to map GPS records of two real-world street blocks into these two simulated blocks in terms of time and space. We utilize pick-up spots of same time period but in the previous day in the dataset to simulate passenger distributions for the pick-up events of the toy car. The toy car is controlled by the attached TelosB node and imaginarily picks up a passenger at a spot on these two street blocks, only if there is a similar pick-up event in the previous day of GPS records. We also use the normalized trip length of similar pick-up events in the previous day to decide trip length for a simulated pick-up event. All the experiments are conducted 10 times and average results are reported.

2) Testbed Evaluation: Since we cannot obtain Ground Truth of taxicab traces constrained in real world two street blocks, we only compare *pCruise* and Oracle in different time windows and broadcasting speeds. Because we conduct the testbed experiment in a fully connected network, we use the different numbers of broadcasting nodes to change the density of networks, instead of broadcasting ranges.

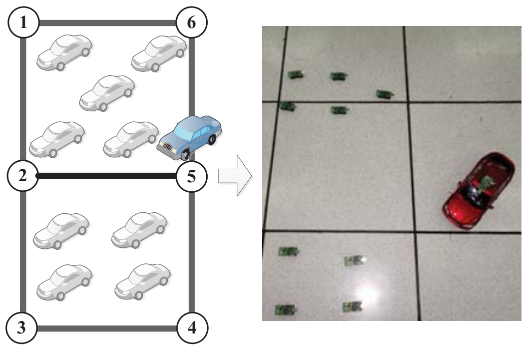

Fig. 21. Testbed Setup

Figure 18 plots the percentage of cruising miles in every 30 seconds time window of total 12 mins, which is mapped by 2 rush hours in real world dataset. We observe that as cumulative time increases, the percentages of cruising miles for both of schemes decrease, and at the end of the experiment the curves of both schemes overlap with each other. This is because after cumulatively collecting GPS records from nearby taxicabs simulated by sensor nodes, both of schemes can select more effective cruising routes. In addition, the performance gain between Oracle and *pCruise* decreases from 12% to 4%, when cumulative time increases from 1 to 8 mins. This is because even though Oracle has more information than *pCruise* to select the route, the number of different routes obtained by two street blocks is limited .

Figure 19 and 20 plot the effects of different broadcasting speed and the number of broadcasting nodes on the percentage of cruising miles, respectively. In both Figure 19 and 20, we observe that with the increases of the speed and the number of nodes, the percentages of cruising miles in *pCruise* and Oracle decrease significantly when the speed and the number of broadcasting nodes is larger than 6 record/mins and 4, respectively. When the speed is 10 record/mins and the number of broadcasting nodes is 6, the percentage of cruising miles achieves its maximal decreasing ratio, respectively. Compared to results in simulation, the testbed results has a similar tendency. Nevertheless the values from which the percentages of cruising miles decrease significantly are different due to the configuration of testbed. The testbed results verify that selecting an appropriate broadcasting speed and the density of networks will reduce the cruising miles significantly.

V. Related Work

There are two types of systems related to our work, *i.e.*, Dispatching Systems and Recommendation Systems.

A. Dispatching Systems

Taxicab dispatching systems are proposed along with the development of intelligent transportation systems and the popularization of GPS sensors [5] [6]. Currently, in most of dispatching systems, a dispatching center assigns a pick-up task to taxicab drivers according to the nearest neighbor principle in terms of distance or time. Phithakkitnukoon *et al*. [7] employ the naive Bayesian classifier with an error-based learning approach, which can obtain the number of vacant taxicabs at a given time and location to enhance the dispatching system. Yang *et al*. [8] propose a mode for urban taxicab services, which indicates the vacant and occupied taxicab movements as well as the relationship between passengers and taxicab waiting time. Yamamoto *et al*. [9] present an adaptive routing scheme and a clustering scheme to enhance dispatching system via assigning vacant taxicabs to the locations with a high expected potential passengers number adaptively. Chang *et al*. [10] propose a model that can predict taxicab demand distribution based on weather condition, time and locations. Gonzalez *et al*. [11] compute the fastest route by taking into account the speed and driving patterns of taxicabs, which are obtained from historical GPS trajectories. Ziebart *et al*. [12] utilize GPS trajectories obtained from 25 taxicabs, instead of providing the fastest route for drivers, aiming to predict the destination of drivers.

Different from the above centralized dispatching systems, our cruising system $pCruise$ provides suggestions to taxicab drivers with an efficient scheduling, and provides the optimal cruising route at a road segment level. Most importantly, for a dispatching system, the passengers need to provide the demand for taxicab companies by booking a taxicab via telephone or Internet in advance, and the reservation is usually not free of charge. In contrast, most passengers hail a taxicab along the street directly, rather than booking a taxicab from a taxicab company. In addition, our approach can be used as a middleware under an existing dispatching system to enhance its performance.

B. Recommendation System

Recommendation system are proposed to provide useful business intelligence for drivers, passengers, or taxicab companies to maximize their profits [13] [14]. Based on GPS data from a metropolitan area, Zheng *et al.* [15] [16] [17] present several novel methods to model the transportation in the area. Ge *et al*. [18] present a model to recommend a taxicab driver with a sequence of pick-up points so as to maximize the profit, via a centralized solution. According to a mobile sequential recommendation problem, the author formulates the target problem. Li *et al*. [19] study how to find passengers via several strategies for taxicab drivers in Hangzhou. They select several features to classify the passenger-finding schemes in terms of performances. Recently, Powell *et al*. [20] propose an approach to suggest profitable grid-based locations for taxicab drivers by constructing a profitability map where according to the potential profit calculated by the historical data, the nearby regions of the driver are scored serving as a metric for a taxicab driver decision making process.

Our cruising system $pCruise$ is different from the above schemes in the following two key aspects: (1) The most of above schemes utilize the entire GPS trajectories of taxicabs to analyze or model the behaviors of taxicabs, but $pCruise$ only employs an "on/off" information, *i.e.*, Pick-up or Drop-off spots, for the analysis. Therefore, we significantly reduce the size of data needed to be proceeded, thus making $pCruise$ suitable for the implementation on low-cost devices, *e.g.*, smart phones. (2) Existing systems suffer from the fact that multiple taxicabs may have the same recommended route, which compromises the performance of the system. In contrast, $pCruise$ introduces unpredictability and randomness in the system by letting every taxicab to create its own cruising graph, which solves the problem of multiple taxicabs selecting the same optimal route.

VI. Conclusion

In this paper, we introduce *pCruise*, a cruising system to reduce cruising miles for taxicab networks. Our work is motivated by the fact that current schemes to reduce taxicabs' cruising miles require the participation of the passengers. To reduce cruising miles without the passengers involved, we create a cruising graph for every taxicab and characterize it with weights on its edges based on GPS records about nearby taxicabs. According to the cruising graph, we propose an efficient scheduling scheme to provide the shortest routes with at least one passenger. Based on a 10 GB real world dataset, we evaluate *pCruise* in both simulation and testbed. The results show that *pCruise* is able to effectively reduce 41% of total cruising miles on average compared to the Ground Truth. In this work, we mainly focus on the efficiency and leave the fairness among the taxicabs in the network as a in the future.

VII. Acknowledgements

This research was supported in part by the US National Science Foundation (NSF) grants CNS-0845994, CNS-0917097,CNS-1239226 and IBM OCR Fund.

References

[1] N. Y. C. Taxi and L. Commission, "Taxi of tomorrow survey results," 2011.

[2] S. Consulting, "The new york city taxicab fact book," *http://www.schallerconsult.com/taxi/taxifb.pdf*.

[3] Y.-Y. Chiang and C. A. Knoblock, "Automatic extraction of road intersection position, connectivity, and orientations from raster maps," in *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, ser. GIS '08, 2008.

[4] A. Fathi and J. Krumm, "Detecting road intersections from gps traces," in *Proceedings of the 6th international conference on Geographic information science*, ser. GIScience'10, 2010.

[5] D. Lee, H. Wang, R. Cheu, and S. Teo, "Taxi dispatch system based on current demands and real-time traffic conditions," in *Journal of the Transportation Research Board*, 2004.

[6] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li, "ibat: detecting anomalous taxi trajectories from gps traces," in *Proceedings of the 13th international conference on Ubiquitous computing*, ser. UbiComp '11.

[7] 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*, 2011.

[8] H. Yang, C. Fung, K. Wong, and S. Wong, "Nonlinear pricing of taxi services," in *Transportation Research Part A: Policy and Practic*, 2010.

[9] 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*, 2010.

[10] H.-w. Chang, Y. Tai, and J. Y. Hsu, "Context-aware taxi demand hotspots prediction," *Int. J. Bus. Intell. Data Min.*, vol. 5, no. 1, Dec. 2010.

[11] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag, "Adaptive fastest path computation on a road network: a traffic mining approach," in *Proceedings of the 33rd international conference on Very large data bases*, ser. VLDB '07, 2007.

[12] B. D. Ziebart, A. L. Maas, A. K. Dey, and J. A. Bagnell, "Navigate like a cabbie: probabilistic reasoning from observed context-aware behavior," in *Proceedings of the 10th international conference on Ubiquitous computing*, ser. UbiComp '08, 2008.

[13] R. Raymond, T. Sugiura, and K. Tsubouchi, "Location recommendation based on location history and spatio-temporal correlations for an on-demand bus system," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '11, 2011.

[14] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher, "Greengps: a participatory sensing fuel-efficient maps application," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, ser. MobiSys '10, 2010.

[15] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma, "Understanding transportation modes based on gps data for web applications," *ACM Trans. Web*, vol. 4, no. 1, Jan. 2010.

[16] Y. Zheng, L. Liu, L. Wang, and X. Xie, "Learning transportation mode from raw gps data for geographic applications on the web," in *Proceedings of the 17th international conference on World Wide Web*, ser. WWW '08, 2008.

[17] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on gps data," in *Proceedings of the 10th international conference on Ubiquitous computing*, ser. UbiComp '08, 2008.

[18] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani, "An energy-efficient mobile recommender system," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '10, 2010.

[19] 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 *PerCom Workshops*, 2011.

[20] J. Powell, Y. Huang, F. Bastani, and M. Ji, "Towards reducing taxicab cruising time using spatio-temporal profitability maps," in *Proceedings of the 12th International Symposium on Advances in Spatial and Temporal Databases*, 2011.

APPENDIX

In this appendix, we demonstrate how to create a cruising graph without the requirement of a background road map. In Figure 3, we observe that pick-up or drop-off spots can be treated as representative samples for all GPS records, and these spots are uniformly distributed in the main streets. It implies that these spots can serve as a virtual road map used to create a cruising graph. Among these spots, some of special spots form a *concave spot group*, which can identify a physical intersection on the underlying road map. We can identify these concave spot groups by the geometrical properties of spots in them, and thus we can identify intersections.

To obtain these concave spots, we uniformly select pairs of spots, and then construct the shortest paths among them. Based on the geometrical properties of concave spots, the most frequently chosen spots among all the shortest paths should be the concave spots, if we set the radius of a spot is slighter longer than the average distance between spots. For example, in Figure 22, given several pick-up or drop-off spots, we show how to identify a concave spot group and thus an intersection, by constructing the shortest paths among uniformly selected pairs of spots.

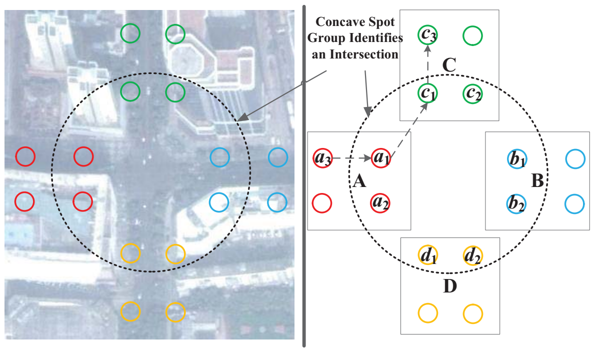

Fig. 22. Identify an Intersection

In Figure 22, the spots a_1, a_2, b_1, b_2, c_1, c_2, d_1 and d_2 will become the most frequent spots in all shortest paths, and thus form a concave spot group to identify this intersection. All the shortest paths from spots in Group A to spots in Group B have a_1 as an intermediate point, *e.g.*, the shortest path from a_3 to c_3 will go through a_1. The similar situations exist for other concave spots. Thus, these spots can identify this intersection.

Based on concave spot groups we identified, we can create a cruising graph to capture the natures of taxicab cruising processes. For every intersection identified by concave spot groups, we create a vertex. To examine the number of pick-up or drop-off spots between two vertices, we shall identify adjacent vertices. For every two adjacent vertices, we create the two directed edges between them from one to another and *verse visa*. Figure 23 shows an example of cruising graph created according to pick-up or drop-off spots.

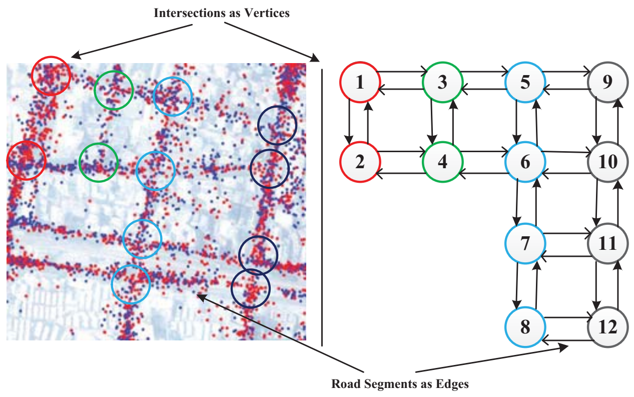

Fig. 23. Cruising Graph

In the above figure, we can see that for every intersection in left figure, there is a corresponding vertex in right figure; every road segment connecting two adjacent vertices has a corresponding edge in right figure.