Online Cruising Mile Reduction in Large-Scale Taxicab Networks

Desheng Zhang, *Student Member, IEEE*, Tian He, *Senior Member, IEEE*, Shan Lin, *Member, IEEE*, Sirajum Munir, *Student Member, IEEE*, and John A. Stankovic, *Fellow, IEEE*

Abstract—In the taxicab industry, a long-standing challenge is how to reduce taxicabs' miles spent without fares, i.e., cruising miles. The current solutions for this challenge usually depend on passengers to actively provide their locations in advance for pickups. To address this challenge without the burden on passengers, in this paper, we propose a cruising system, $pCruise$, for taxicab drivers to find efficient routes to pick up passengers to reduce cruising miles. According to the real-time pick-up events from nearby taxicabs, $pCruise$ characterizes a cruising process with a *cruising graph*, and assigns weights on edges of the cruising graph to indicate the utility of cruising corresponding road segments. Our weighting process considers the number of nearby passengers and taxicabs together in real-time, aiming at two scenarios where taxicabs are explicitly or implicitly coordinated with each other. Based on a weighted cruising graph, when a taxicab becomes vacant, $pCruise$ provides a distributed online scheduling strategy to obtain and update an efficient cruising route with the minimum length and at least one arriving passenger. We evaluate $pCruise$ based on a real-world GPS dataset from a Chinese city Shenzhen with $14{,}000$ taxicabs. The evaluation results show that $pCruise$ assists taxicab drivers to reduce cruising miles by 42 percent on average.

Index Terms—Taxicab network, dispatching, cruising mile reduction

1 Introduction

Nowadays, among all transportation modes, taxicabs play a particularly prominent role in residents' daily commutes in many metropolitan areas [1]. Based on a recent survey in New York City [2], over 100 taxicab companies operate more than $13{,}000$ taxicabs, with a stable ridership of $660{,}000$ per day, and transport more than 25 percent of all passengers, accounting for 45 percent of all paid transit fares. Unfortunately, to fulfill such delivery requests, these taxicabs travel a total of 800 million miles per year [1]. Nearly 40 percent of this mileage (i.e., 314 million miles per year, consuming 14 million gallons of gas) is spent to cruise for passengers, therefore leading to severely harmful tailpipe emissions and energy consumption. On the other hand, the top comment from passengers about taxicabs is "I cannot get one when needed" [2]. Additionally, in carbon emission trading under the Kyoto Protocol [3], governments will provide economic incentives for achieving reductions in the carbon emissions. Therefore, to achieve less carbon emissions, better satisfactory services and more economic incentives, it is imperative to find a practical initiative to reduce cruising miles for taxicabs to quickly find passengers.

As a promising initiative, in large metropolitan areas, e.g., New York City, Singapore, Beijing, and Shenzhen [4], [5], [6], taxicabs are equipped with GPS and communication devices, and taxicabs' status (locations, speeds, with passengers or not, etc.) is uploaded to taxicab companies' dispatching centers periodically. Based on the status, these dispatching centers schedule taxicabs to efficient routes to pick up nearby known passengers, thus reducing cruising miles. However, these solutions usually depend on passengers to actively provide their pickup locations in advance. Typically, existing dispatching centers function under the scenario where a passenger contacts a dispatching center first, and then a dispatching center assigns a task about this passenger to a nearby vacant taxicab. But in developing counties or dense urban areas, most passengers hail taxicabs along streets directly, rather than booking taxicabs from dispatching centers in advance. Therefore, we face a challenge that how to dispatch taxicabs to quickly find passengers to reduce their cruising miles, yet without pickup locations provided by passengers.

In this paper, we propose a cruising system, $pCruise$, i.e., cruising with purposes. The key novelty about $pCruise$ is that it utilizes only several key GPS records of nearby taxicabs to model a cruising process about a particular taxicab, and then provides a distributed online scheduling strategy with the shortest cruising route for this taxicab to find a passenger. Specifically, our key contributions are as follows:

- To the best of our knowledge, we conduct the first comprehensive study of how to reduce cruising miles of taxicabs with a fully distributed online

- *D. Zhang and T. He are with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455. E-mail: {zhang, tianhe}@cs.umn.edu.*
- *S. Lin is with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794. E-mail: Shan.Lin.3@stonybrook.edu.*
- *S. Munir and J.A. Stankovic are with the Department of Computer Science, University of Virginia, Charlottesville, VA 22903. E-mail: {munir, stankovic}@cs.virginia.edu.*

Manuscript received 25 Mar. 2014; revised 13 Aug. 2014; accepted 21 Sept. 2014. Date of publication 19 Oct. 2014; date of current version 7 Oct. 2015. Recommended for acceptance by X. Cheng.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2014.2364024

Taxicab Network Summary	
Collection Period	6 Months
Collection Date	01/01/12-06/30/12
Numbe of Taxis	14,453
Number of Passengers	98,472,628
Total Live Mile	371,269,642
Total Cruising Mile	238,788,860
Average Live Mile	3.77
Average Cruising Mile	2.43

Fig. 1. Statistics.

method. We design a system, called $pCruise$, to let taxicab drivers find passengers with a cruising route based on pick-up events of nearby taxicabs, yet without locations provided by passengers.

- In $pCruise$, we create a mathematical model, called *cruising graph*, to capture key ingredients of a taxicab's cruising process. Vertices of the cruising graph represent intersections and its edges represent road segments connecting intersections.
- We characterize a cruising graph by weights on its edges, which are represented by the expected number of *remaining passengers* (obtained by the number of *potential passengers* minus the number of *competing taxicabs*) during a taxicab cruising road segments. We employ the pickup events from nearby taxicabs to obtain the number of potential passengers; we present two methods to calculate the number of competing taxicabs under explicit and implicit coordinations among taxicabs. The formulation of weights in a cruising graph is based on only nearby taxicabs' GPS records uploaded to dispatching centers, without additional marginal costs.
- According to the weights on the cruising graph, we propose a fully distributed online scheduling for $pCruise$. It selects a cruising route for a specific taxicab by considering both the weight (i.e., expected number of remaining passengers) on this route and the length of this route together. Furthermore, $pCruise$ supports an online scheduling where a cruising route for a particular taxicab is updated at every intersection according to newly received GPS records from nearby taxicabs.
- We evaluate $pCruise$ with a real-world half-year dataset, collected from taxicabs in Shenzhen. The dataset consists of 6 months of GPS records from $14{,}000$ taxicabs. The evaluation results show that with the assistance of $pCruise$, a taxicab reduces cruising miles by 42 percent on average.

The rest of the paper is organized as follows. Section 2 proposes the motivation based on our empirical results. Section 3 presents the $pCruise$ design. Section 4 validates $pCruise$ with a large-scale dataset. Section 5 presents the related work. Section 6 gives the conclusion.

2 Motivation

In this section, we employ the empirical data to show our motivation. In the taxicab business, the total travel miles of a taxicab consist of live miles (driven with paying passengers) and cruising miles (driven without paying passengers). The key inefficiency of taxicab networks is the long cruising miles for taxicabs to find passengers, which leads to a high gas consumption without any passengers. Based on a GPS dataset of $14{,}000$ taxicabs in Shenzhen (with the highest people density of $17{,}510$ per KM^2 in China), we present the evidence about the inefficiencies of current taxicab networks.

Fig. 1 summarizes statistics about taxicab networks studied. In Fig. 1, we observe that the average cruising mile is 2.43 to find a passenger with an average live mile of 3.77 per taxicab per trip. Fig. 2 gives the average cruising mileage of all taxicabs in the network on an hourly basis. We observe that the cruising miles are high in the morning, and typically peak at 7 AM around 11 miles per hour.

From the above two figures, we argue that such high cruising miles contribute to the key inefficiency of current taxicab networks, since long cruising miles or a high percentage of cruising indicate inefficient consumption of gas. On the other hand, passengers would select another transportation mode if they cannot find taxicabs in time. Thus, the key motivation of this paper is to reduce cruising miles, thus addressing a discord where taxicab drivers suffer from long cruising miles, and passengers cannot find a taxicab when they need one. As follows, we bridge the above gap by proposing $pCruise$ which guides vacant taxicabs to find a passenger with the minimum cruising miles.

3 $pCruise$ Design

In this section, we present the detailed design for $pCruise$. We introduce the current taxicab infrastructure in Section 3.1. Built upon this infrastructure, we present the main idea of $pCruise$ in Section 3.2. We demonstrate how to model a cruising process in Section 3.3. By this modeling, we explain how to characterize a cruising process in Section 3.4. Finally, we show how to utilize a characterized cruising process to select cruising routes for taxicab drivers to reduce cruising miles in Section 3.5.

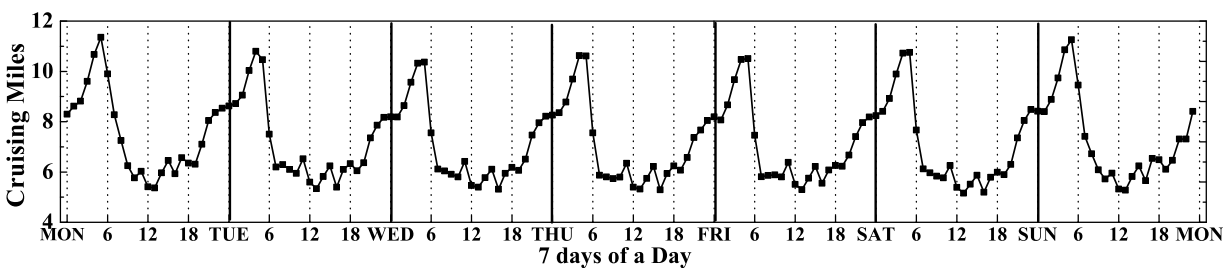


Fig. 2. Weekly cruising miles.

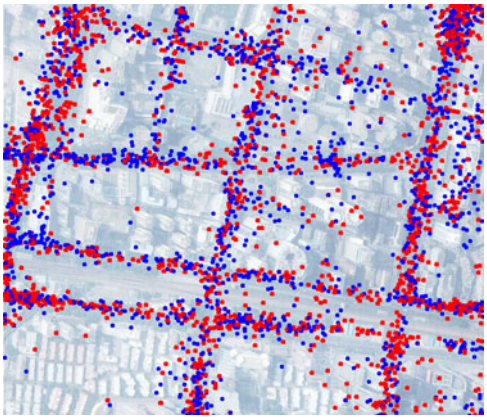


Fig. 3. Pick-up (red) or drop-off (blue) spots.

3.1 Current Infrastructure

In current taxicab networks of large cities such as New York City, Beijing, and Shenzhen, taxicabs are equipped with GPS and communication devices, in addition to the traditional fare meters. With the GPS devices, taxicabs log their physical status, such as current locations, speeds, directions, etc.; with the fare meters, taxicabs log their logical status, such as with passengers or not. Thus, with communication devices, both physical and logical status of taxicabs is uploaded periodically to either a government agency server that stores the status of all taxicabs, or a taxicab company server that stores the status of the taxicabs belonging to this company. The physical and logical status of a taxicab is uploaded in terms of GPS records, which consist of the following parameters: (i) Plate Number; (ii) Date and Time; (iii) GPS Coordinates; and (iv) Availability Bit: whether or not a passenger is in this taxicab when the record is being uploaded. Based on this infrastructure, we introduce semantics mined from it, and how to configure the networking of taxicabs as follows.

3.1.1 Semantics Mined from Current Infrastructure

In the above infrastructure, the only information taxicabs periodically uploaded is their GPS records. By observing the consecutive GPS records from the same taxicab, we data mine passenger distributions from them. Instead of considering all consecutive GPS records from a taxicab, we shall focus on the GPS records with a change on the availability bit compared to the previous GPS records for the same taxicab. 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 at the location indicated by the corresponding GPS coordinates. Therefore, we name this physical location s_i and the 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. Fig. 3 gives an example of these spots within three blocks. In this paper, taxicabs on roads are seen as probing mobile sensors to detect passenger distributions, which significantly assists vacant taxicab drivers to find better routes to pick up new passengers.

3.1.2 Networking under Current Infrastructure

Given the current infrastructure and the mined semantics, a straightforward dispatching is to establish a centralized dispatching center to dispatch taxicabs based on mined passenger distributions to reduce cruising miles for all taxicabs. But this centralized solution is usually infeasible in the real-world, since taxicabs in the same city may belong to more than 100 taxicab companies [1]. So, sharing GPS records or dispatching taxicabs across different companies is logistically difficult to be performed in practice.

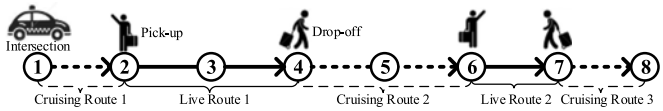


Fig. 4. Taxicab operating scenario.

In this paper, utilizing the broadcast nature of wireless uploading, we argue that by overhearing GPS records from other taxicabs, a vacant taxicab shall find better routes to reduce cruising miles. A system built upon this approach has the following potential advantages: (i) this approach is based on the current taxicab network infrastructure where it is mandatory for taxicabs to upload their records, so this approach does not involve additional marginal costs; (ii) this approach provides distributed coordinations for taxicab drivers to break the barriers between different companies and operators.

The rationale between our distributed networking (i.e., GPS broadcasting and overhearing) is that for a taxicab, not all other taxicabs' GPS records are useful for reducing cruising miles. For example, GPS records from another taxicab several miles away may not be helpful for a taxicab to find a nearby passenger; similarly, if a taxicab tries to find a passenger now, it may also not be helpful to analyze GPS records received at several hours ago. Therefore, according to the current infrastructure, we envision that GPS records uploaded by a taxicab are overheard by other taxicabs located within a certain range, indicated as a *Broadcasting Range*. A taxicab stores and analyzes the GPS records from another nearby taxicab, as long as it is within nearby taxicabs' broadcasting ranges. In addition to the broadcasting range, a broadcasting speed, i.e., how often a taxicab uploads its GPS records, is also important, since a faster uploading speed enables nearby taxicabs to more quickly receive GPS records. We test effects of the GPS broadcasting range and speed on the system performance in Section 4.

3.2 Main Idea

Based on the semantics and the network setting in the last section, we first introduce some concepts in the taxicab business, before presenting the main idea. Fig. 4 gives an example of a taxicab operating scenario where its total travel miles are divided into *cruising routes* (driving without paying passengers) and *live routes* (driving with paying passengers) by pick-up or drop-off locations.

In Fig. 4, a taxicab starts its first cruising route 1 at intersection 1 to find a passenger. At intersection 2, by picking up a passenger, the taxicab ends cruising route 1 and starts its live route 1. At intersection 4, by dropping off the passenger, the taxicab ends its live route 1 and starts its cruising route 2, and so on and so forth. The process during which a driver cruises streets to find a passenger is called a *cruising process*, which starts at the last drop-off location and ends at

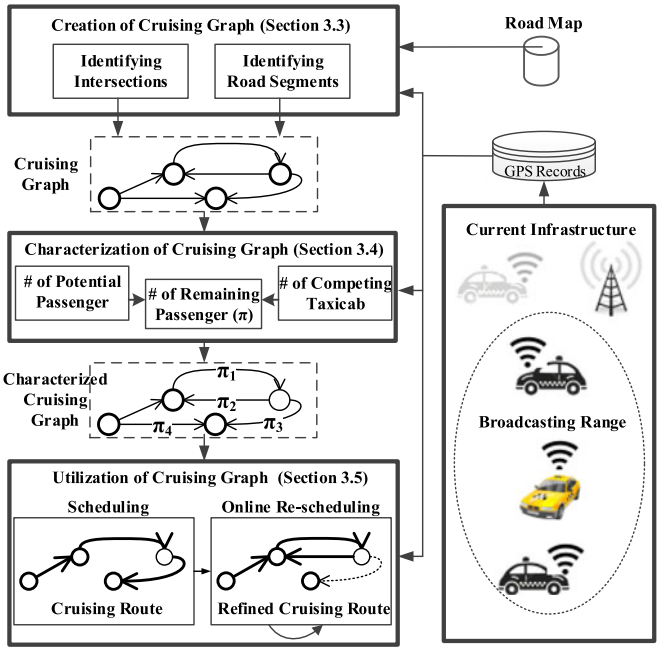


Fig. 5. *pCruise* architecture.

the next pick-up location. The total miles of all cruising routes or live routes are called *cruising miles* or *live miles*, respectively. For example, supposing that Fig. 4 gives the entire operating scenario of this taxicab in a day, the total cruising miles are the sum of lengths of the three cruising routes in Fig. 4.

In a live route, a taxicab is occupied by a passenger who pays a fare according to the length of a live route; whereas in a cruising route, a taxicab is vacant without paying passengers yet consuming gas. Therefore, a profitable strategy for a cruising process is *to minimize the length of every cruising route, thus minimizing the total cruising miles, while finding at least one expected passenger on every cruising route.* Based on this cruising strategy, the key idea of *pCruise* is as follows.

During both cruising and live routes, as introduced in Section 3.1, a taxicab (i) broadcasts its GPS records as it did in the existing infrastructure, (ii) overhears and stores the GPS records broadcasted by the taxicabs located within its broadcasting range, and (iii) deletes the GPS records overheard from the taxicabs moving out of its broadcasting range.

In addition to the above operations, when a taxicab becomes vacant and tries to find a new passenger (beginning of a cruising route), according to GPS records overheard from taxicabs within its broadcasting range, it (i) *models* its cruising process at road segment levels, (ii) *characterizes* road segments in the model, (iii) *selects* a cruising route based on characterized road segments, and *refines* the selected cruising route based on newly received GPS records.

Three questions rise up with respects to this idea. Based on the overall architecture of *pCruise* given in Fig. 5, we answer those questions as follows:

- *How to model a cruising process.* When a vacant taxicab is cruising road segments, a driver only takes actions at an intersection. Thus, intersections and road segments are crucial units for our design. To capture fundamental characteristics about a cruising process, we propose a mathematical concept *Cruising Graph*. As in the first box of Fig. 5, a cruising graph is created based on a road map, as introduced in Section 3.3.
- *How to characterize a road segment in a cruising process.* Given a cruising graph, to show the utility of cruising a road segment, we characterize a cruising graph by assigning weights on its edges, corresponding to road segments. As in the second box of Fig. 5, weights are indicated by the expected numbers of *remaining passengers* π, which are obtained by the number of *potential passengers* minus the number of *competing taxicabs*, as introduced in Section 3.4.
- *How to select and refine a cruising route by utilizing characterized road segments.* Given a characterized cruising graph, we propose an efficient online scheduling to obtain the shortest cruising route at which at least one passenger arrives. As in the last box of Fig. 5, given a characterized cruising graph, a cruising route is scheduled at first, and then based on newly received GPS records, this cruising route is refined repeatedly via an online re-scheduling during the process, which is introduced in Section 3.5.

Note that to design *pCruise* with a generic philosophy, we describe *pCruise* as a distributed solution, although it can be easily customized to a centralized solution. Thus, *pCruise* serves as a lightweight yet effective system which can be implemented in frontend (i.e., taxicabs) or in backend (i.e., dispatching centers). Further, to be compatible with the current infrastructures, *pCruise* is designed as a byproduct without marginal costs, so there is no *ad hoc* direct communications among taxicabs, and the only communications are uploading of GPS records.

Most importantly, given the existence of a plethora of dispatching and recommendation systems, *pCruise* serves as a middleware under any existing dispatching system to enhance its performance. This is because the creation, the characterization, and the utilization about a cruising graph are fully independent from each other, and can be used to supplement other theoretical models. As a result, *pCruise* provides a unified solution for a highly diverse, heterogeneous route selection that may be deployed among mobile units for various applications.

3.3 Creation of Cruising Graph

In this section, we model a taxicab cruising process with a mathematical concept, i.e., *Cruising Graph* where for every intersection in the real-world, we create a corresponding vertex; for every two adjacent intersections, we create two directed edges between them. Thus, the key step to create a cruising graph is to identify intersections and road segments connecting adjacent intersections. Much work has been proposed to use GPS traces to infer a road map [7], [8]. Therefore, to focus on the system level, we assume that a road map is given. Thus, the creation of cruising graph is straightforward, since intersections and segments can be easily identified by a road map. Note that though all segments are bidirectional, one way situation is considered by assigning weight 0 to corresponding edges. Fig. 6 shows an example of a cruising graph created according to a road map.

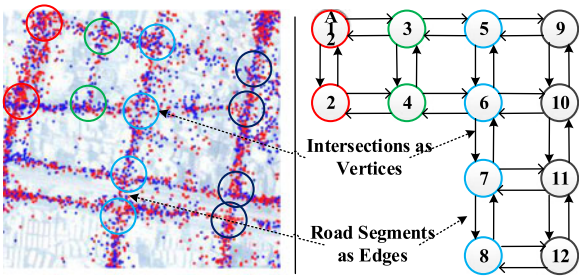


Fig. 6. Cruising graph.

3.4 Characterization of Cruising Graph

For a cruising graph in the last section, we characterize its edges (i.e., real-world road segments) with weights, i.e., numbers of *remaining passengers*. For the same edge, its weight varies for different taxicabs and time durations, which is based on the real-time pickup events from nearby taxicabs.

As in Fig. 7, given the current time t_0, for a specific taxicab T_1, we present how to obtain the number of remaining passengers π_r^t on a specific segment $r = [s_{begin}, s_{end}]$ during a future period $t = [t_{begin}, t_{end}]$. In the evaluation and real-word setting, t_{begin} and t_{end} of a travel period on a segment are decided by both the current location of a target taxicab and the historical travel time for this segment.

For a better presentation, we omit the temporal superscript. First, assuming we are given a *passenger arrival rate* λ_r and a *travel time* τ_r for r during t, we have the number of *arriving passengers* as $\lambda_r \times \tau_r$. Second, we consider two kinds of vacant taxicabs that will be on r during t: (i) the vacant taxicabs (called original taxicabs) that are on r at t_0; (ii) the vacant taxicabs (called competing taxicabs) that are not on r at t_0. Third, for all these arriving passengers on r during t, we exclude the impacts of the original taxicabs by a *potential passenger ratio* ρ_r, and thus we have the number of *potential passengers* as $\kappa_r = \lambda_r \times \tau_r \times \rho_r$; further, we exclude the impacts of the competing taxicabs by numbers of *competing taxicabs* ω_r, and thus we have the number of *remaining passengers* as $\pi_r = \kappa_r - \omega_r$, which is the key formula for our weighting. The details of κ_r, ω_r and π_r are given in Section 3.4.1, 3.4.2, and 3.4.3.

3.4.1 Number of Potential Passengers κ_r

Among all arriving passengers $\lambda_r \times \tau_r$ on r during t, we obtain the number of potential passengers κ_r with a potential passenger ratio ρ_r, which gives the probability that there is a *potential* passenger (i.e., not served by the original

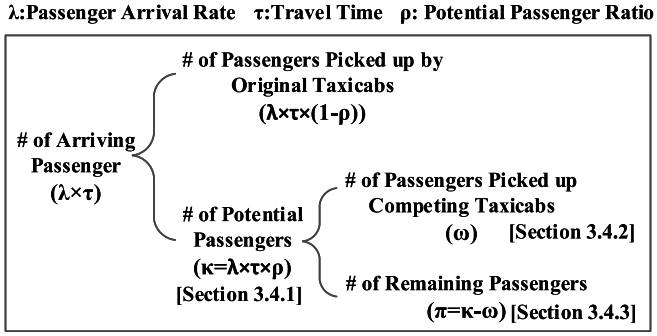


Fig. 7. Overview of characterization.

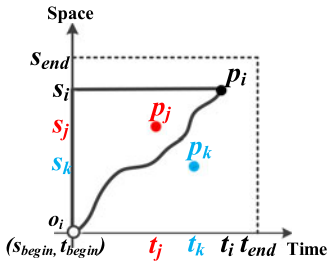


Fig. 8. ρ' about 1 taxi.

taxicabs already on r) in two dimensional temporal-spatial spot $p_x = [t_x, s_x]$ where $t_x \in [t_{begin}, t_{end}]$ and $s_x \in [s_{begin}, s_{end}]$, as follows

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

The rationale behind this equation is that for r, among all $\lambda_r \times \tau_r$ arriving passengers, some of them are picked up by vacant taxicabs already on r (i.e., original taxicabs) at t_0, and only the rest (with a number of $\rho_r \times \lambda_r \times \tau_r$) is treated as the potential passengers for the taxicabs (i.e., competing taxicabs) that will arrive on r during t. Since a travel time τ_r can be easily estimated by the historical dataset, we present how to obtain ρ_r and λ_r as follows:

Potential passenger ratio ρ. In general, a road segment with a high potential passenger ratio leads to a high possibility of pick-ups, thus minimizing cruising miles for a taxicab driver. Therefore, a taxicab shall select a segment with a higher potential passenger ratio ρ, which is obtained by collected pick-up events in GPS records, via observing how long nearby vacant taxicabs picked up passengers on this segment before. The sooner the nearby taxicabs pick up passengers on this segment, the higher the potential passenger ratio ρ on this segment. Note that we utilize the collected pick-up events in a previous period to infer ρ for a future period. The assumption is that the potential passenger ratio does not change dramatically in a short time period during which a taxicab passed a segment.

For simplicity, when computing a weight on a road segment, we transform a two-dimensional GPS coordinate (x, y) into an 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 map p_i in a temporal-spatial 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 Fig. 8. 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.

In this two dimensional temporal-spatial system, the triangle-like shape $\triangle' o_i s_i p_i$ indicates a temporal-spatial area without any potential passenger. This is because if there were a potential passenger within $\triangle' o_i s_i p_i$, e.g., a spot $p_j = (t_j, s_j)$ in Fig. 8, then the pick-up location of Taxicab i should be s_j, instead of s_i. But there could be a potential passenger outside $\triangle' o_i s_i p_i$, e.g., a spot $p_k = (t_k, s_k)$ in Fig. 8, 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.

Therefore, in this one taxicab scenario, for a road segment r connecting an intersection s_{begin} and an intersection

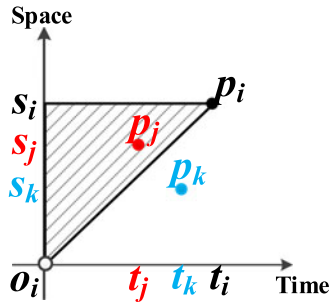


Fig. 9. ρ about 1 taxi.

s_{end}, we obtain the potential 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 two dimensions in terms of time and space, while the physical meaning of $|\triangle' o_i s_i p_i|$ is the time and space area confirmed by Taxicab i that there is no potential passengers.

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 Fig. 9. The rationale behind this approximation is that in a road segment we assume that a taxicab is with relatively uniform speed. We verify the effect of this approximation on the evaluation section. After the approximation, a new potential passenger ratio $\rho_r^{(1)}$ for the road segment r as in Fig. 9 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 $\triangle o_i s_i p_i$.

Figs. 10 and 11 consider multiple taxicab scenarios where the triangles of taxicabs overlap with each other. The union area of all triangles indicates an area without any potential passengers in terms of space and time. For example, in Fig. 10, 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 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

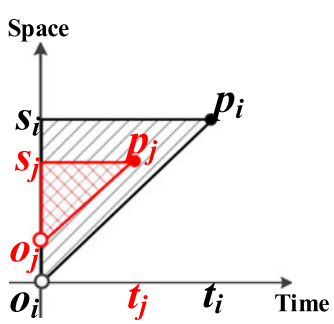


Fig. 10. ρ about 2 taxis.

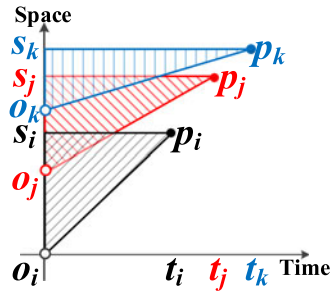


Fig. 11. ρ about 3 taxis.

$\triangle o_i s_i p_i$. 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 a *potential* passenger for Taxicab i, since she or he is picked up by Taxicab j.

Similarly, for the three taxicabs scenario in Fig. 11,

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

To generalize the above results, for an edge associating the road segment r,

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

where I is a set of all corresponding pick-up spots. In the real-world setting, t_{begin} and t_{end} are decided by both the current location of the target taxicab and the historical travel time from s_{begin} to s_{end}.

Passenger arrival rate λ. To obtain an accurate passenger arrival rate, we need a time period and the number of arriving passengers during this period. But based on GPS records alone, it is impractical to obtain such accurate information. Thus, we introduce λ, which indicates the lower bound of the passenger arrival rate for segment $r = [s_{begin}, s_{end}]$ in duration $t = [t_{begin}, t_{end}]$. With λ, we have the lower bound of the total arriving passenger number at a time duration. As follows, we first introduce the lower bound of passenger arrival rate λ^{s_i} at a pick-up location s_i on r during t, which is obtained by the time difference of two vacant taxicabs passing the same location.

For example, in Fig. 12, at moment O_i, 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 passenger (F_j^{-1} is the inverse function

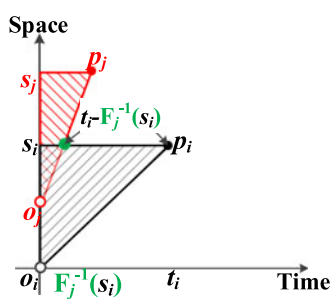


Fig. 12. λ in scenario 1.

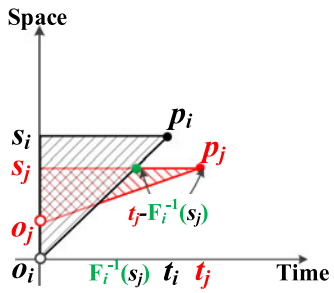


Fig. 13. λ in scenario 2.

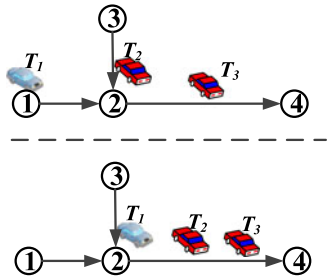


Fig. 15. Competing taxis.

of line segment $\overline{o_j p_j}$). This is because if there were a 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 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 arrives 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 can only 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 the 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 Fig. 13 where at least one passenger arrived at the location s_j in time duration $[F_i^{-1}(s_j), t_j]$. But in Fig. 14, we observe a situation where λ cannot be obtained, because no pick-up happens when a later taxicab passes a former taxicab's cruising route.

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

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

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

3.4.2 Number of Competing Taxicabs ω_r

For a specific taxicab T_1, there are two types of vacant taxicabs competing with T_1 on a road segment r when T_1 enters r: (i) vacant taxicabs (called original taxicabs) are on r now, and will still be on r when T_1 enters r. (ii) vacant taxicabs (called competing taxicabs) are not on r now, but will be on r when T_1 enters r. Fig. 15 gives examples about them. Supposing that at the current time (the top subfigure), taxicab T_1, T_2 and T_3 is on road segment $r_{1\to2}$, $r_{3\to2}$, and $r_{2\to4}$, respectively. At the time when T_1 enters $r_{2\to4}$ (the bottom subfigure), T_3 will be still on $r_{2\to4}$, and T_2 has already entered $r_{2\to4}$. Thus, T_3 is the original taxicab on $r_{2\to4}$ for T_1, while T_2 is the competing taxicab on $r_{2\to4}$ for T_1. In the previous section, we have excluded the effects of the original taxicabs with the potential passenger ratio ρ_r. Thus, in this section we exclude the effects of the competing taxicabs, which is shown by the number of competing taxicabs (indicated as ω_r) on a road segment r.

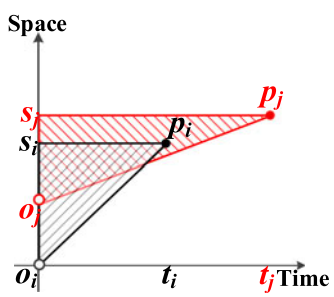


Fig. 14. λ in scenario 3.

Note that in the current infrastructure introduced in Section 3.1, vacant taxicabs do not upload future cruising routes to base stations (only uploading real-time GPS locations). Thus, by overhearing, a taxicab cannot directly obtain the number of competing taxicabs ω_r on a road segment r. To solve this issue, we propose two methods to obtain ω_r, according to *explicit coordinations* (cooperative) or *implicit coordinations* (non-cooperative) among taxicabs as follows.

Under the explicit coordination, vacant taxicabs are considered to be cooperative to each other, and explicitly share their cruising routes with each other in addition to GPS records. Thus, taxicabs coordinate with each other by lowering weights of the edges with many vacant taxicabs. This cooperative scenario is applicable to the situation where all taxicabs belong to the same taxicab company. Admittedly, a taxicab network may achieve a better performance under this explicit coordination, but it involves extra communications in addition to the normal GPS record uploading, and also may not be logistically possible for the situation where taxicabs belong to many different competing taxicab companies. In contrast, under the implicit coordination, vacant taxicabs are considered to be non-cooperative to each other, and do not share their cruising routes with each other. Thus, a taxicab has to infer future cruising routes about nearby taxicabs based on their GPS records. We introduce these two kinds of coordinations as follows.

Explicit coordinations among taxicabs. For the explicit coordination, a vacant taxicab shares its cruising routes to nearby taxicabs by additional communications. For a vacant taxicab, its cruising route starts at its current location and includes several road segments connecting corresponding intersections. Thus, to share its cruising routes, a vacant taxicab simply broadcasts a sequence of intersections in its cruising route to nearby taxicabs within its broadcasting range. To focus on the cruising system design, we configure this cruising route broadcasting together with the GPS record uploading.

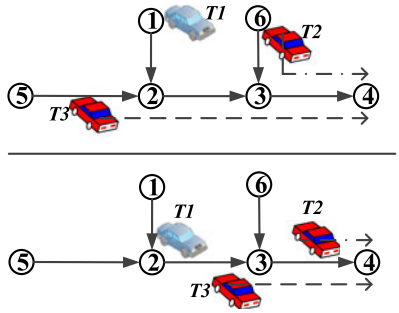


Fig. 16. Explicit coordinations.

With the above explicit cruising route broadcasting, a taxicab T_1 decides whether or not another taxicab T_2 is on a specific road segment r when T_1 enters r, based on the following information:

i) the current locations of T_1 and T_2;
ii) the cruising route of T_2;
iii) the entering intersection and the exiting intersection of road segment r;
iv) the time period $[t^{T_2}_{r_{\text{enter}}}, t^{T_2}_{r_{\text{exit}}}]$ from T_2 entering r to T_2 exiting r;
v) the time $t^{T_1}_{r_{\text{enter}}}$ when T_1 enters r.

If $t^{T_1}_{r_{\text{enter}}}$ is between $t^{T_2}_{r_{\text{enter}}}$ and $t^{T_2}_{r_{\text{exit}}}$, then T_2 is on r, when T_1 enters r. Similarly, we decide the same situation for other nearby taxicabs.

Finally, the number of competing taxicabs ω_r for a taxicab T_1 is given by the following.

$$\omega_r = \sum_{n \in N} \left\{ C^r_n = \begin{cases} 1; & t^n_{r_{\text{enter}}} < t^{T_1}_{r_{\text{enter}}} < t^n_{r_{\text{exit}}} \\ 0; & \text{otherwise} \end{cases} \right\}, \quad (1)$$

where N is the set of taxicabs that are within the broadcasting range of taxicab T_1.

For example, as in the top subfigure of Fig. 16, a vacant taxicab T_1 tries to calculate the expected number of competing taxicabs ω_r for road segment $r_{2\to3}$. Two other taxicabs T_2 and T_3 are within its broadcasting range, and have already selected their cruising routes shown by two dashed lines. Based on the explicit coordinations, T_2 and T_3 broadcast their cruising routes as $6 \to 3 \to 4$ and $2 \to 3 \to 4$ with their GPS records, respectively. Thus, T_1 has both the cruising routes and GPS records from T_2 and T_3. Then, T_1 decides that T_2 will not arrive at $r_{2\to3}$ based on the cruising route of T_2. Next, T_1 calculates (i) the time period that T_3 is on road segment $r_{2\to3}$, and (ii) the time that T_1 arrives at road segment $r_{2\to3}$. As shown by the bottom subfigure of Fig. 16, T_3 will still be on $r_{2\to3}$ when T_1 arrives at road segment $r_{2\to3}$. Finally, T_1 has $C^r_{T_2} = 0$ and $C^r_{T_3} = 1$, leading to $\omega_r = 1$ for road segment $r_{2\to3}$.

Note that during the cruising road segment $r_{2\to3}$, ω_r may change due to different situations, e.g., a new vacant nearby taxicab because of a new drop-off or the entrance of the broadcasting range.

Implicit coordinations among taxicabs. In the implicit coordination, we envision a different situation where all taxicabs are from different competing companies, and thus additional coordinating communications are not allowed. Thus, a taxicab cannot directly obtain nearby taxicabs' cruising routes. Therefore, we employ a method to infer cruising routes of nearby vacant taxicabs.

A naive method for a vacant taxicab T_1 to infer the cruising route of another nearby vacant taxicab T_2 is to assume that T_2 uniformly selects a direction at all intersections. Obviously, this naive method overlooks the fact that taxicab T_2 also overhears the GPS records from nearby taxicabs as taxicab T_1 does. If we envision that all vacant taxicabs in the network are cruising based on *pCruise*, then T_1 uses its cruising graph (Section 3.3), weights on the edges (Section 3.4), and a scheduling algorithm (Section 3.5) to calculate the cruising route of T_2 based on the current location of T_2. One key issue is that under the implicit coordination, T_1 cannot obtain the expected number of competing taxicabs ω_r for a road segment r, since ω_r is inherently depended on cruising routes. So T_1 cannot obtain the weights on the edges. To address this issue, we let the number of potential passengers (κ_r) equal to the number of remaining passengers (π_r) for any road segment r, assuming there are no competing taxicabs ($\omega_r = 0$). Thus, T_1 obtains T_2's cruising route.

After we obtain all cruising routes of nearby vacant taxicabs, we use the same method as in Equation (1) for the explicit coordination to obtain the number of competing taxicabs ω_r for taxicab T_1.

3.4.3 Number of Remaining Passengers π_r

So far, we have introduced how to obtain the number of potential passengers κ and the number of competing taxicabs ω in Sections 3.4.1 and 3.4.2. For a road segment r, considering the arriving competing taxicabs (with number of ω_r), only a subset of the total potential passengers (with number of κ_r) remains for a targeted taxicab. Thus, we define the weight on an edge r (corresponding to a real-world road segment r) as the number of remaining passengers π_r as follows:

$$\pi_r = \kappa_r - \omega_r = (\lambda_r \times \tau_r \times \rho_r) - \omega_r.$$

3.5 Utilization of Cruising Graph

The key objective of this section is based on a taxicab's weighted cruising graph to schedule a cruising route, which starts at its current location and includes several following road segments and corresponding intersections. When a taxicab becomes vacant, *pCruise* first performs an *initial scheduling* to obtain a cruising route. But during the taxicab cruises this route, *pCruise* updates the weights on its cruising graph based on the newly received GPS records. With the updated weights, *pCruise* performs an *online rescheduling* to obtain a new cruising route at every intersection, until a passenger is picked up. The initial and online rescheduling are introduced in Sections 3.5.1 and 3.5.2.

3.5.1 Initial Scheduling

When a taxicab becomes vacant, given its current cruising graph, we have many different routes with different lengths and expected numbers of remaining passengers, since no destination is provided. Among all routes, our key objective is to find a cruising route that (i) starts at the current location, (ii) has at least one expected remaining passenger, and (iii) has the shortest length. The rationale behind this

objective is that in the taxicab business, a profitable strategy for a taxicab driver is to select a cruising route to find at least one passenger with the shortest distance from the current location, i.e., minimizing the cruising route. Thus, we select a cruising route based on both the total expected number of remaining passengers on this route (i.e., the total weight on edges of this route) and the total length of this route. For a selected cruising route, the expected number of remaining passengers on this cruising route is obtained by the sum of weights of the cruising route's edges; the total length of this cruising route is obtained by the sum of lengths of road segments in this route, which are given by the cruising graph.

Theoretically, our scheduling can be represented by the following objective function based on a cruising graph $G = (V, A)$ where x_{ij} indicates how many times an edge (i.e., a road segment) $r_{i\to j}$ with the length $|r_{i\to j}|$ and weight $\pi_{i\to j}$ is used in the optimal route

$$\min \quad \sum_{(i,j)\in A} x_{ij}|r_{i\to j}| \tag{2}$$

$$\text{s.t.} \quad x_{ij} \in \{0, 1, 2, \ldots\} \qquad \forall i, j \in V$$

$$\sum_{(i,j)\in A} x_{ij}\pi_{i\to j} \geq 1 \tag{3}$$

$$\sum_{j\in V}(x_{ij} - x_{ji}) \in \{0, 1\} \qquad \text{if } i = s \tag{4}$$

$$\sum_{j\in V}(x_{ij} - x_{ji}) \in \{0, -1\} \qquad \text{if } i \neq s \tag{5}$$

$$\sum_{\forall i,k\in V}\sum_{j\in V}[(x_{ij} - x_{ji}) + (x_{kj} - x_{jk})] \in \{0, -1\}, \tag{6}$$

where (2) ensures that one road segment can be used more than once in the optimal route, e.g., circling around a block with high expected remaining passengers; (3) ensures that the optimal route has at least one expected remaining passenger; (4) ensures that the optimal route starts with the current intersection s and it may end at s (i.e., the outdegree $\sum_j x_{ij}$ is equal to indegree $\sum_j x_{ji}$) or not at s (i.e., the outdegree $\sum_j x_{ij}$ is large than indegree $\sum_j x_{ji}$ by one); (5) and (6) are about the connectivity of the optimal route: (5) ensures that an intermediate intersection has an indegree equal to its outdegree or that an end intersection has an indegree larger than its outdegree by one; (6) ensure that there is only one end intersection, i.e., no two intersections have the indegrees larger than the outdegrees by one.

Since solving the above problem with linear programming is NP-Hard, the running time increases exponentially when the number of intersections increases. Although integer linear programming is sufficient for a small number of intersections, we aim to propose an efficient scheduling algorithm to reduce the running time yet still produce the optimal route. Note that traditional shortest path algorithms, e.g., Dijkstra, would not be suitable for our scheduling, because they do not produce a solution that visits the current intersection s twice, which could be the optimal solution of the scheduling.

Based on the objective of the scheduling, we utilize a traversal based strategy in terms of lengths of road segments.

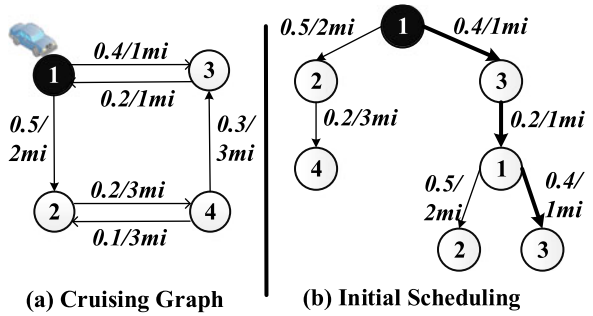


Fig. 17. Initial scheduling.

To find the optimal cruising route, i.e., the shortest route with the total expected number of remaining passengers larger than 1, a straightforward solution with linear programming is to find all the routes with the total expected number of remaining passengers larger than 1 by a complete search, and then to select the shortest one among them. But it will take a long time for a taxicab to find the optimal route, although the number of vertices in a route is limited. Thus, to enable a real-time scheduling, we utilizes a hill-climbing based traversal that quickly finds a potential solution first and then continues to try other possible solutions to verify that the current potential solution is the optimal solution or not.

The key advantage of this scheme is that it can quickly stop to expand on a route to save the running time for an agile scheduling if this route will never be the optimal solution. This is performed by comparing the expanding route with the current potential solution, and if this route is longer than the current potential solution we stop the expanding on this route even though the cumulative weight on this route is smaller than 1. This is because this route will never be the optimal solution, since the current potential solution is shorter than it and also has the total weight larger than 1. Specifically, during a traversal, the hill-climbing scheme always greedily expands the vertex on the current shortest route, and stops expanding on a route if the total expected number of remaining passengers of this route (i.e., the cumulative weight) is larger than 1. It replaces the current potential solution with a new route, if this route is shorter and has the cumulative weight is larger than 1. After expanding on all the possible routes, the current potential solution is the optimal solution for our scheduling, because it is the shortest route among all the routes with cumulative weight larger than 1.

In Fig. 17, we give an example about the initial scheduling. A taxicab T_1 becomes vacant at intersection 1, and its weighted cruising graph is in subfigure (a) where both the weights (numbers of remaining passengers) and the lengths (in terms of miles) of road segments are given in associated edges.

i) $pCruise$ starts at intersection 1 where T_1 is currently located, and the current road segment candidates for the traversal are $r_{1\to 2}$ and $r_{1\to 3}$. Since $r_{1\to 3}$ is shorter than $r_{1\to 2}$, $pCruise$ first traverses $r_{1\to 3}$.
ii) The current segment candidates for the traversal are $r_{1\to 2}$ and $r_{3\to 1}$ shown by subfigure (b). A traversal on

$r_{3\to1}$ leads to a route $r_{1\to3}+r_{3\to1}$ with the same length of $1+1=2$ miles, comparing to a traversal on $r_{1\to2}$ leading to a route $r_{1\to2}$. So we have to compare the expected numbers of remaining passengers between them. Because a traversal on $r_{3\to1}$ leads the $0.4+0.2=0.6$ expected passenger, more than the 0.5 expected passenger obtained by a traversal on $r_{1\to2}$, $pCruise$ secondly traverses $r_{3\to1}$.

iii) Based on the similar strategy, $pCruise$ finds the first cruising route ($r_{1\to3}+r_{3\to1}+r_{1\to3}$) with the length of 3 and the expected number of remaining passengers larger than or equal to 1 ($0.4+0.2+0.4=1$) as the current solution. $pCruise$ continues to find other possible solutions, but $pCruise$ will not traverse the route with the length longer than the current solution to save running time. This is because a route (e.g., route $r_{1\to2}+r_{2\to4}$ with length of 5) longer than the current solution would not be the optimal solution. Finally, the route $r_{1\to3}+r_{3\to1}+r_{1\to3}$ is the optimal solution, since other routes shorter than it do not have the expected number of remaining passenger larger than or equal to 1.

3.5.2 Online Rescheduling

When a taxicab is cruising a selected route, several passengers may be picked up and some competing taxicabs may arrive on the following road segments of this cruising route, which are indicated by GPS records of nearby taxicabs. These situations change weights on road segments and thus change the attractiveness of the corresponding cruising route. Therefore, when a taxicab is on a selected cruising route, $pCruise$ employs an online rescheduling to agilely select a new cruising route at every intersection. The objective to reschedule a cruising route is the same as in the initial scheduling: to find a route (i) starts at the current location, (ii) has at least one expected remaining passenger, and (iii) has the shortest length. The algorithm for the online rescheduling is also similar to the algorithm for the initial scheduling.

4 Trace-Driven Evaluation

To test $pCruise$ in a real-world scenario, we utilize a real-world dataset about 6 months of GPS traces of more than $14{,}000$ taxicabs belonging to different taxicab companies in Shenzhen, a Chinese city with 10 million population. The dataset is obtained by letting every taxicab upload its GPS records (the format as in Section 2) to report its traces to a base station. Based on the dataset of GPS records, we obtain location and time distributions of pick-up events, which are used to evaluate the performance of $pCruise$.

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

We evaluate the performance of $pCruise$ under three scenarios: one driver using $pCuirse$, the drivers from one taxicab company using $pCuirse$ (accounted for 10 percent of total taxicabs); all drivers using $pCuirse$. We first report the results of only one taxicab and 10 percent of the total taxicabs using $pCruise$, where we manipulate the randomly selected taxicabs using $pCruise$ with routes decided by pick-up events and $pCruise$ scheduling together, and manipulate other taxicabs with the original traces. It is difficult to accurately model the impacts of the routes of the taxicabs with $pCruise$ on the routes of the taxicabs without $pCruise$. Alternatively, to take the feedback of the system into consideration, we investigate the system performance where all taxicabs are using $pCruise$. In this scenario, we essentially create a simulator where every taxicab cruises streets based on $pCruise$ (instead of the original traces), and picks up or drops off passengers according to pick-up events (in terms of trips' pick-up location, drop-off location, and beginning time) obtained by the GPS dataset. But the beginning time in a pick-up event is not the time when a passenger starts to wait for taxicab, so for every passenger, we use a random number w between 0 and 10 to show the waiting time in terms of minutes, where the 10-minutes waiting time as a upper bound is based on a taxicab survey [2]. In the simulation, if a taxicab arrives at a pick-up location within a w-minute time window before the corresponding beginning time, then this taxicab picks up the passenger, and then goes to the same route as the taxicab that delivers the same passenger in the original pick-up event. After arriving at the drop-off location, this taxicab cruises the cruising graph based on the scheduling of $pCruise$, until arriving at the next pick-up location.

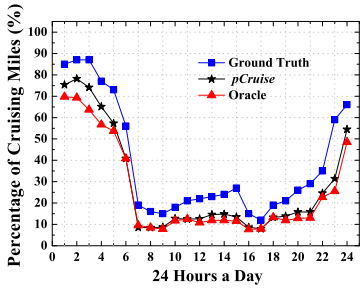


Fig. 18. Miles vs. time (1 taxi).

To show the effectiveness of $pCruise$, we first compare the performance of $pCruise$ under the explicit coordinations with *Ground Truth*, which is the original GPS traces from the dataset without any modifications. Second, since $pCruise$ only employs the pick-up spots, instead of the entire trajectories, to assign weights on a cruising graph, we also compare $pCruise$ with an *Oracle* scheme, which stores and processes all the collected trajectories of nearby taxicabs. The Oracle is used to verify the effect of the approximation about the triangle-like shape in $pCruise$. Therefore, Oracle utilizes triangle-like shapes to calculate a more accurate edge weight based on the completed trajectories about nearby taxicabs as introduced in Fig. 8. These two comparisons are given in Sections 4.1, 4.2 and 4.3. Third, we compare two versions of $pCruise$ (called $pCruise$-Explicit and $pCruise$-Implicit) under the explicit and implicit coordinations as introduced in Section 3.4.2 to show the effectiveness of the explicit coordinations in Section 4.4.

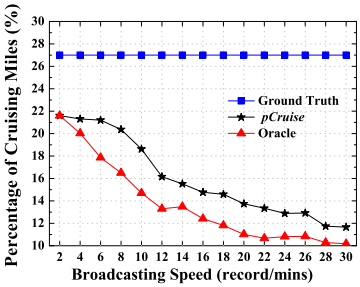


Fig. 19. Miles vs. speed (1 taxi).

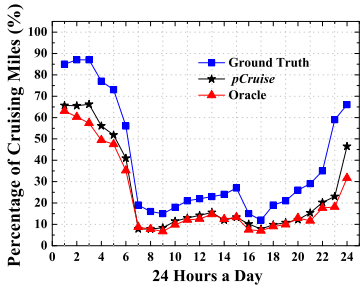


Fig. 21. Miles vs. time (10 percent taxi).

4.1 One Taxicab with *pCruise*

Fig. 18 plots the percentage of cruising miles in every one hour time window of a day. In the rush hour of a day, e.g., $06:00-10:00$, the percentages of cruising miles for all three schemes are below 20 percent. In contrast, in the non-rush hour of a day, e.g., $00:00-6:00$, the percentages of cruising miles for all three schemes are over 60 percent. But for every one hour time window of a day, both *pCruise* and Oracle outperform Ground Truth with an average gain of 35 and 42 percent, which verifies the effectiveness of taking nearby taxicab's activities into consideration. Oracle outperforms *pCruise* by 14 percent on average in the non-rush hour, and by 5 percent on average in the rush hour, indicating that even though in the rush hour Oracle considers the entire trajectories, the effect is limited. It implies during the rush hour, the pick-up spots alone used by *pCruise* effectively reduce the cruising miles, and thus considering the entire trajectories in the rush hour is not as effective as in the non-rush hour.

Fig. 19 plots the effect of different broadcasting speeds on the percentage of cruising miles. The 27 percent of the cruising miles for Ground Truth are obtained by the average value of all day. We observe that with the increase of the speed, the percentages of cruising miles in *pCruise* and Oracle decrease significantly as much as 39 percent when the speed is larger than 6 records per min, but the decreasing slows down when the speed is larger than 12 records per min. This is because when the speed first becomes larger, *pCruise* and Oracle have more frequently updated GPS records to obtain the optimal route, but when the speed is larger than 12 records per min, 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 low, e.g., 6 records per min.

Fig. 20 plots the effect of different broadcasting ranges on the percentage of cruising miles. We observe that with the increase of ranges from 0.3 to 3.0 KM, the percentages of cruising miles in *pCruise* and Oracle continuously decrease. When the range is larger than 0.6 KM, the decrease becomes more significant, but when the range is larger than 1.5 KM, it becomes more stable. This is because when the ranges become larger, *pCruise* and Oracle select a route based on more nearby taxicabs' information. But when the range is larger than 1.5 KM, the GPS records about the far taxicabs may not provide any related information about nearby road segments. Thus, the decrease of cruising miles becomes less obvious.

4.2 Ten Percent of the Total Taxicabs with *pCruise*

Fig. 21 plots the performance of 10 percent of the total taxicabs in terms of the percentage of cruising miles. We observe that in the rush hour 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 percent. In contrast, in the non-rush hour 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 in the rush hour. In addition, both *pCruise* and Oracle outperform Ground Truth with an average gain of 39 and 45 percent, 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 percent. This performance gain decreases compared to the scenario where one taxicab uses *pCruise*, which implies that *pCruise* performs better when more taxicabs employ *pCruise*.

Figs. 22 and 23 plot the effects of different broadcasting speeds and ranges on the percentage of cruising miles. In both figures, 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/min and 0.75 KM,

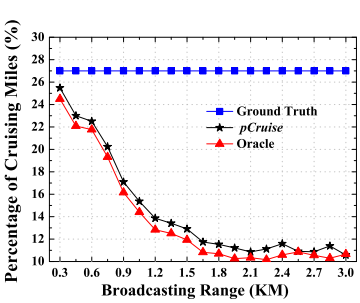


Fig. 20. Miles vs. range (1 taxi).

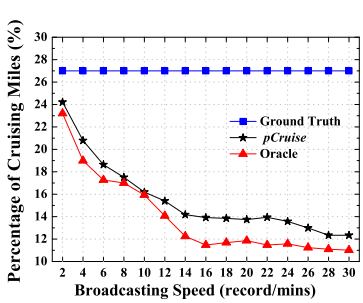


Fig. 22. Miles vs. speed (10 percent taxi).

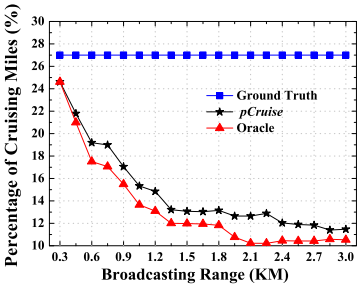


Fig. 23. Miles vs. range (10 percent taxi).

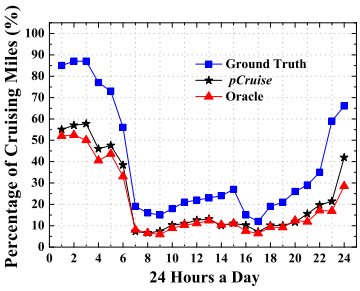


Fig. 24. Miles vs. time (all taxi).

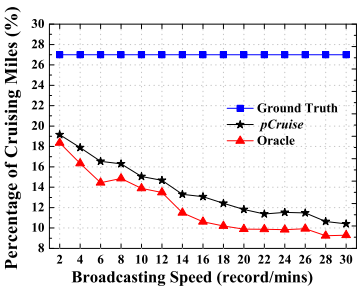


Fig. 25. Miles vs. speed (all taxi).

respectively. Compared to the results in Figs. 19 and 20, the results in Figs. 22 and 23 have the similar tendency yet the better performance. But the values from which the percentages of cruising miles decrease significantly are smaller.

4.3 All Taxicabs with *pCruise*

Figs. 24, 25 and 26 plot the performance of all taxicabs with $pCruise$. Fig. 24 shows the similar tendency, compared to Fig. 21. In the rush hour of a day, the percentages of cruising miles for all three schemes are below 20 percent. In contrast, in the non-rush hour of a day, the percentages of cruising miles for all three schemes are higher. Further, both $pCruise$ and Oracle outperform Ground Truth with the similar average gains, compared to the situation where 10 percent of taxicabs using $pCruise$, which implies that when all taxicabs are using $pCruise$, the performance is not significantly improved. In both Figs. 25 and 26, we see that with the increases of broadcasting speeds and ranges, the percentage of cruising miles in Ground Truth has the similar tendency, while those in $pCruise$ and Oracle decrease when the broadcasting speed or range is increasing from 2 records/min or 0.3 KM, respectively.

4.4 Impacts of Coordinations on *pCruise*

Fig. 27 gives the impacts of two coordinations on the performance of $pCruise$, i.e., the explicit coordinations and

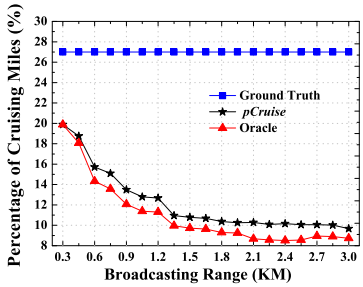


Fig. 26. Miles vs. range (all taxi).

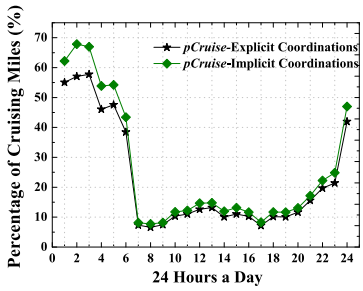


Fig. 27. Impacts of coordinations.

implicit coordinations as in Section 3.4.2. Fig. 27 shows the similar tendency between two coordinations, but the $pCruise$ under explicit coordinations outperforms the $pCruise$ under the implicit coordinations, especially in the early morning. This is because a taxicab under the implicit coordinations cannot accurately predict the future route of other taxicabs due to limited taxicab services in the morning. In contrast, in the rush hour of a day, the percentages of cruising miles for $pCruise$ under two coordinations are similar, though the $pCruise$ under explicit coordinations still outperforms the $pCruise$ under implicit coordinations by 10 percent on average.

5 Related Work

The premise of GPS based taxicab dispatching is not new, and many studies have focused on taxicab scheduling [9], [10], [11], [12], [13] or novel systems taking advantage of taxicab traces [4], [5], [6], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23]. Two types of the previous work are directly related to our work, namely dispatching systems and recommendation systems.

5.1 Dispatching Systems

Taxicab dispatching systems are proposed along with the development of intelligent transportation systems and the popularization of GPS [24]. Currently, in the 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. [9] employ the naive Bayesian classifier with an error-based learning approach, which obtains the number of vacant taxicabs at a given time and a location to enhance dispatching systems. Yang et al. [10] propose a model 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. [11] present an adaptive routing scheme and a clustering

scheme to enhance dispatching systems via assigning vacant taxicabs to the locations with a high expected potential passenger number adaptively. Liu et al. [12] propose a method to discover spatio-temporal causal interactions in traffic data streams. Huang and Powell [13] present a system to detect the insufficient taxicab services in certain areas for dispatching centers.

5.2 Recommendation Systems

Recommendation systems are proposed to provide useful business intelligence for drivers, passengers, or taxicab companies to maximize their benefits. Based on GPS data from a metropolitan area, Zheng et al. [15], [16], [17] present several novel methods to model the transportation. Given the traces of taxicabs, Balan et al. [18] propose a system for passengers to inquire the travel time and taxicab fare. Wu et al. [28] present a system to assist mobile users to make transportation decisions, such as taking a taxicab or not. Wei et al. [19]] employ uncertain trajectories to construct popular routes of taxicabs. Ge et al. [20] present a model to recommend a taxicab driver with a sequence of pick-up points to maximize profits via a centralized solution. Powell et al. [21] propose an approach to suggest profitable grid-based locations for taxicab drivers by constructing a profitability map where the nearby regions of drivers are scored serving as a metric for a taxicab driver decision making process.

5.3 Novelty of *pCruise*

Compared with the above systems, the novelty of *pCruise* lies in three aspects. (i) *pCruise* does not rely on the assumption that passengers provide their locations, and automatically draws the passenger distribution based on real-time status of nearby taxicabs. (ii) *pCruise* only employs "on/off" information, i.e., pick-up or drop-off events, for the model. Therefore, we significantly reduce the size of data needed to be processed, making *pCruise* suitable for the implementation on low-cost devices, e.g., smart phones. (iii) *pCruise* utilizes an efficient online scheduling based on the events of their own nearby taxicabs, instead of the entire GPS dataset of all taxicabs. Since every taxicab has different nearby taxicabs and starting times, *pCruise* introduces unpredictability and randomness in the system to compensate for more taxicabs heading to the same area with the same cruising route.

6 Conclusion

In this work, based on a half-year dataset, we design and evaluate *pCruise*, a cruising system to reduce cruising miles for taxicab networks. Our work provides a few valuable insights, which are hoped to be useful for applying *pCruise* in a real-world taxicab network. Specifically, (i) for the different time of the day, a cruising system has different performance gains, and it is more efficient during the non-rush hour; (ii) a better networking setting to efficiently exchange information among nearby taxicabs improves the performance of a cruising system; (iii) increasing the number of taxicabs using the same cruising system further improves the overall system performance; (iv) explicitly coordinating with nearby taxicabs by sharing the future routes increases the effectiveness of a cruising system.

Acknowledgments

This research was supported in part by the US National Science Foundation (NSF) grants CNS-0845994, CNS-0917097, CNS-1239226, CNS-1239108, CNS-1239483 and UMN Thesis Travel Grant. A preliminary work has been presented in IEEE RTSS 2012 [30].

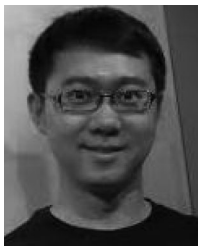
References

[1] Schaller Consulting. The New York city taxicab fact book [Online]. Available: http://www.schallerconsult.com/taxi/taxifb.pdf, 2006.
[2] NYC Taxi Commission, "Taxi of tomorrow survey," 2011.
[3] Wikipedia. Emissions trading [Online]. Available: http://en.wikipedia.org/wiki/Emissionstrading, 2014.
[4] J. Aslam, S. Lim, X. Pan, and D. Rus, "City-scale traffic estimation from a roving sensor network," in *Proc. 10th ACM Conf. Embedded Netw. Sensor Syst.*, 2012, pp. 141–154.
[5] J. Yuan, Y. Zheng, and X. Xie, "Discovering regions of different functions in a city using human mobility and POIs," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 186–194.
[6] Y. Zheng, Y. Liu, J. Yuan, and X. Xie, "Urban computing with taxicabs," in *Proc. 13th Int. Conf. Ubiquitous Comput.*, 2011, pp. 89–98.
[7] J. Biagioni and J. Eriksson, "Map inference in the face of noise and disparity," in *Proc. 20th Int. Conf Adv. Geographic Inf. Syst.*, 2012, pp. 79–88.
[8] X. Liu, J. Biagioni, J. Eriksson, Y. Wang, G. Forman, and Y. Zhu, "Mining large-scale, sparse GPS traces for map inference: Comparison of approaches," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 669–677.
[9] S. Phithakkitnukoon, M. Veloso, C. Bento, A. Biderman, and C. Ratti, "Taxi-aware map: Identifying and predicting vacant taxis in the city," in *Proc. 1st Int. Joint Conf. Ambient Intell.*, 2011, pp. 86–95.
[10] H. Yang, C. Fung, K. Wong, and S. Wong, "Nonlinear pricing of taxi services," *Transportation Res. Part A: Policy Practice*, vol. 44, pp. 337–348, 2010.
[11] 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*. New York, NY, USA: Springer, 2010.
[12] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xing, "Discovering spatio-temporal causal interactions in traffic data streams," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 1010–1018.
[13] Y. Huang and J. W. Powell, "Detecting regions of disequilibrium in taxi services under uncertainty," in *Proc. 20th Int. Conf. Adv. Geographic Inf. Syst.*, 2012, pp. 139–148.
[14] W. Zhang, S. Li, and G. Pan, "Mining the semantics of origin-destination flows using taxi traces," in *Proc. ACM Conf. Ubiquitous Comput.*, 2012, pp. 943–949.
[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, pp. 1:1–1:36, 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 *Proc. 17th Int. Conf. World Wide Web*, 2008, pp. 247–256.
[17] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on GPS data," in *Proc. 10th Int. Conf. Ubiquitous Comput.*, 2008, pp. 312–321.
[18] R. K. Balan, K. X. Nguyen, and L. Jiang, "Real-time trip information service for a large taxi fleet," in *Proc. Int. Conf. Mobile Syst., Appl. Serv.*, 2011, pp. 99–112.
[19] L.-Y. Wei, Y. Zheng, and W.-C. Peng, "Constructing popular routes from uncertain trajectories," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 195–203.
[20] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani, "An energy-efficient mobile recommender system," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 899–908.
[21] J. Powell, Y. Huang, F. Bastani, and M. Ji, "Towards reducing taxicab cruising time using spatio-temporal profitability maps," in *Proc. 12th Int. Symp Advances Spatial Temporal Databases*, 2011, pp. 242–260.
[22] X. Liu, J. Biagioni, J. Eriksson, Y. Wang, G. Forman, and Y. Zhu, "Mining large-scale, sparse GPS traces for map inference: Comparison of approaches," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 669–677.

[23] J. Biagioni and J. Eriksson, "Map inference in the face of noise and disparity," in *Proc. 20th Int. Conf. Adv. Geographic Inf. Syst.*, 2012, pp. 79–88.
[24] D. H. Lee, H. Wang, R. L. Cheu, and S. H. Teo, "A taxi dispatch system based on current demands and real-time traffic conditions," in *J. Trans. Res. Board*, no. 1882, pp. 193–200, 2004.
[25] 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, pp. 3–18, Dec. 2010.
[26] 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 *Proc. 33rd Int. Conf. Very Large Data Bases*, 2007, pp. 794–805.
[27] 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 *Proc. 10th Int. Conf. Ubiquitous Comput.*, 2008, pp. 322–331.
[28] W. Wu, W. S. Ng, S. Krishnaswamy, and A. Sinha, "To taxi or not to taxi? - enabling personalised and real-time transportation decisions for mobile users," in *Proc. IEEE 13th Int. Conf. Mobile Data Manage.*, 2012, pp. 320–323.
[29] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 316–324.
[30] D. Zhang and T. He, "pcruise: Reducing cruising miles for taxicab networks," in *Proc. IEEE 33rd Real-Time Syst. Symp.*, 2012, pp. 85–94.

Desheng Zhang (M'10) is working toward the PhD degree in the Department of Computer Science and Engineering, University of Minnesota-Twin City. His research includes big data analytics, mobile CPS, wireless sensor networks, intelligent transportation systems. He is a member of the IEEE.

Tian He (M'03-SM'12) is currently an associate professor with the Department of Computer Science and Engineering, University of Minnesota Twin Cities. He is the coauthor of more than 100 papers in premier journals and conferences with more than 12,000 citations. He received the US National Science Foundation CAREER Award' 09, and was a program chair in few international conferences. His research includes wireless sensor networks, intelligent transportation systems, and distributed systems. He is a member of the IEEE.



Shan Lin (M'03) received the PhD degree in computer science from the University of Virginia. He is currently an assistant professor with the Department of Electrical and Computer Engineering, Stony Brook University. His research is in the area of networked systems, with an emphasis on feedback control based design in cyber physical systems. He works on wireless network protocols, medical devices, first responder systems, and smart transportation systems. He is a member of the IEEE.

Sirajum Munir (M'13) is working toward the PhD degree in the Computer Science Department, University of Virginia. His research interest lies in the area of cyber physical systems, ubiquitous computing, and data mining. He has been working in several projects involving smart home management, human-in-the-loop systems, and routing protocol design for real-time applications. He is a member of the IEEE.

John A. Stankovic (F'94) received the PhD degree from Brown University. He is the BP America professor in the Computer Science Department, University of Virginia. In the past, he was a chair of the Department for eight years. He also won the IEEE Real-Time Systems Technical Committees Award for Outstanding Technical Contributions and Leadership. He also won the IEEE Technical Committee on Distributed Processings Distinguished Achievement Award (inaugural winner). He has won six Best Paper awards including for ACM SenSys 2006. He was the editor-in-chief for *IEEE Transactions on Distributed and Parallel Systems*. His research interests are in cyber physical systems, distributed computing, real-time systems, and wireless sensor networks. He is a fellow of the IEEE.