CAPTURING AND ANALYZING HUMAN DRIVING BEHAVIOR TO IMPROVE ROAD TRAVEL EXPERIENCE

by

RUILIN LIU

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Badri Nath

And approved by

_____

_____

_____

_____

New Brunswick, New Jersey

OCTOBER, 2017

ABSTRACT OF THE DISSERTATION

Capturing and Analyzing Human Driving Behavior to Improve Road Travel Experience

By RUILIN LIU

Dissertation Director:

Badri Nath

People commute on daily basis and spend a considerable amount of time on road travels. Unfortunately, due to the dense population and concentration of economic activities in metropolitan areas, urban transportation suffers a lot of challenges, e.g. traffic congestion and parking space storage, which also affect each individual's travel experience. Previous studies have looked into the transportation condition sensing solution using various static or mobile sensors and traffic related recommendation services based on the sensing results, e.g. vehicle routing and parking recommendation. However, human behaviors as the most direct reflection as well as the final modifier of the traffic condition, have not been fully explored to close the sensing-and-control loop in the urban transportation system. This dissertation aims at improving drivers' road travel experience by taking human driver's behavior in vehicle routing and parking search processes into consideration.

In the first part of the dissertation, we focus on the traffic condition sensing and routing service design, altering the isolated traffic sensing and the greedy routing strategy based on sensed traffic condition. Instead, we present a participatory system, called Themis, to consider traffic sensing and route planning as a whole: (i) By analyzing time-stamped position reports and route decisions collected from the Themis mobile app, the Themis server estimates both the current traffic rhythm and the future traffic distribution. (ii) The routing requests are then combined with the sensed traffic condition to provide route recommendations that minimize

the travel cost of all drivers to proactively alleviate traffic congestions. Themis has been implemented and its performance has been evaluated in both simulation experiments using real data from over 26,000 taxis and a field study. Results from both experiments demonstrate that Themis reduces traffic congestion and average travel time at various traffic densities and system penetration rates.

In the second part of the dissertation, we look into the crucial part of parking recommendation, i.e., fine-grained parking availability crowdsourcing and propose a solution based on human's parking and ignoring decisions: a parking decision immediately take an available spot while an ignored spot along a driver's parking search trajectory is likely to be already taken. However, complications caused by drivers' preferences, e.g. ignoring the spots too far from the driver's destination, have to be addressed based on human parking decisions. We build a model based on a dataset of more than 55,000 real parking decisions to predict the probability that a driver would take a spot, assuming the spot is available. Then, we present a crowdsourcing system, called ParkScan, which leverages the learned parking decision model in collaboration with the hidden Markov model to estimate background parking spot availability. ParkScan has been evaluated using real-world data from both off-street scenarios (i.e., two public parking lots) and an on-street parking scenario (i.e., 35 urban blocks in Seattle). Both of the experiments show that ParkScan reduces the error of spot-level parking availability estimation compared to the state-of-art solutions. In particular, ParkScan cuts down over 15% of the estimation errors for the spots along parking search trajectory even if there is a single participant driver.

Acknowledgements

I would like to express my deepest gratitude and appreciation to my advisors, Dr. Liviu Iftode and Dr. Badri Nath, who have been leading my way ever since the first day of my Ph.D. study. Dr. Liviu Iftode has been always motivating, guiding me to try new ideas until the last a few days in his life. Now he is resting in heaven, but his knowledge, inspiration, consideration, and sense of humor in both research and daily life will be invaluable treasures for my whole life. Dr. Badri Nath has been a great mentor who always offers insightful feedbacks and protects his students under the wings. The discussions with him have been always constructive and inspiring. Dr. Nath's vision, direction, optimism, and advice have provided a wealth of benefits to me that can never be overstated.

I would like to acknowledge Dr. Desheng Zhang. Although I have been working with him for just a year, I am thankful for his insightful comments, constructive criticisms, selfless help, and thought-provoking encouragement. The experience of working with him has a richness and depth far surpassing any ability of language to express, making it an honor, and a joy.

I would like to thank Dr. Vinod Ganapathy. It is regretful that I do not have the chance to conduct research with him. However, I learned a lot from his insight into research problems during class and group meetings. I also want to thank him for his feedback to my dissertation and qualifying exam, and more importantly for his help, support, and encouragement along the way.

I would also like to thank Dr. Kostas Bekris, Dr. Cristian Borcea (New Jersey Institute of Technology) and Dr. Yong Xiang (Tsinghua University), from whom I got priceless knowledge, insight, support, and feedbacks, which were vital for the completion of this work. Future more, I want to thank Dr. Thu Nyuyen and Dr. Ahmed Elgammal both for the knowledge I learned from them for my study and for their considerations and encouragements when I felt frustrated and helpless.

During my Ph.D. study, I have had the pleasure of working with many excellent people, with whom I have had the privilege of sharing my time. My appreciation goes out to my colleagues, Daehan Kwak, Yu Yang, Srinivas Devarakonda, Hongzhang Liu, Nader Boushehrinejadmoradi, Daeyoung Kim, Hai Nguyen, Susan Pan (New Jersey Institute of Technology), and Yu Zhang (Beijing Institute of Technology), whom I have had the pleasure to work with and learn various things from. I especially would like to thank Daehan Kwak and Yu Yang, who spent days and nights helping me on the research, showing great teamwork. I am also grateful to the staff of the department for assisting me in many different ways, especially, Carol DiFrancesco, Maryann Holtsclaw, Aneta Unrat, Komal Agarwal, and Doris Allen.

Lastly, I would like to thank my family for their unfailing love, encouragement, patience, and unconditional support. They give meaning to my life and work. I dedicate and owe my dissertation to them.

Thank you all!!!

Dedication

To my family, whom I owe everything to.

To Dr. Liviu Iftode, for your inspiration, guidance, and care.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Following the rapid urbanization since the industrial age, half of the human population worldwide now reside in cities, benefiting from the high density of different resources but also bringing out many urban challenges, especially in urban transportation. There has been 145% increase in transport of people in Europe since 1970 [130]. In the United States, the total mileage of passenger vehicles reached 3.17 trillion miles from 1.13 trillion miles in 1971 [11]. Nowadays, U.S. people spend on average 26 minutes traveling over 18 miles for commute per day [138]. When considering the travel for other purposes, e.g. shopping or social events, the average mileage per person increases to 40 miles per day [156, 145].

Among various travel modes, e.g. air, railway, or water transport, road travels, particularly private cars driving, account for the largest share [138, 145, 156]. A two-year survey conducted by the American Automobile Association (AAA) Foundation shows that people drive on average 29.8 miles every day, spending 48.4 minutes [161]. Even higher values are reported by [125]. According to the data released in 2014, the road traffic consumes 82% of the transportation energy consumption in the U.S. and over 20% of the total energy consumption [47]. Road traffic is also attributed for over 22% of greenhouse gas emissions in the U.S. [47] and a major source of other air pollution, e.g. particulate matters (PMs) and volatile organic compounds (VOCs).

The booming road travel demands cast enormous pressures on the road network, causing economic, social, and environmental problems and the degradation of human's travel experience. Since 1985, the total length of public roads in the U.S. increases by about 6% (in lane-mile) until 2009 [12]. However, during the same period, the total annual vehicle mileage traveled has been increased by 72.1% (from 1.72 trillion miles to 2.96 miles) [11]. The contention between explosively increased travel demand and limited road traffic resources builds up the

transportation dilemma worldwide, e.g. traffic congestions and parking difficulty. In 2014, traffic congestions cause urban Americans to spend an extra 22% of travel time and to purchase an extra 3.1 billion gallons of fuel for a total cost of $160 billion ($970 per capita) [148]. The cost has increased by over 40% since 2000 [148]. The INRIX study covering 1,064 cities in 38 countries shows that congestions incur similar per-capita cost in European countries, e.g. UK and Germany, and are also spreading to the developing world [73]. Even worse, urban drivers also spend 3.5 to 14 minutes on average cruising for parking before finding an available spot in downtown areas, which not only waste the parking seekers travel time but also added up to 30% of street traffic [154]. It is reported that the majority of urban/suburban residents are dissatisfactory with the regional transportation system and the dissatisfaction rate is still increasing [42].

1.1 Automatic Road Transport Support System

Due to the challenges and great impact of building smart transportation systems, many intelligent transport systems (ITSs) have been built to improve the efficiency of the transportation network. In this dissertation, we focus on **Automatic Road Transport Support (ARTS) System** [112], the automatic system providing innovative services supporting more informative, coordinated, and efficient road travel.

ARTS System covers the acquisition and analysis of data generated by heterogeneous sensors in the road transportation context to understand the nature of the road travel system, and to estimate or predict the traffic condition. Moreover, based on the analytic models, the ARTS system also enforces interventions on the operation of the road transportation network to improve human road travel experience as well as the efficiency of the operation system.

1.2 The State of the Art

Based on the different functionalities and operation modes, the ARTS systems can be classified into three major categories: (i) traffic sensing & routing systems, (ii) parking information & guidance systems, and (iii) support systems for individual transportation modes.

1.2.1 Traffic Sensing & Routing System

Traffic sensing systems nowadays apply three methods to collect sensing data: dedicated sensors, e.g. induction loop and cameras, passive crowdsourcing, and active crowdsourcing [192]. Managing dedicated sensors, e.g. the installation and maintenance of induction loops, usually incur an expensive cost, though providing reliable data. Therefore, dedicated sensors are deployed in limited areas. Passive crowdsourcing refers to the data collection from existing infrastructures which are initially designed for a different purpose. For instance, GPS equipped vehicles, e.g. taxis, periodically report their positions for security and dispatching purposes. However, these position reports build up mobility trajectories and can be used to learn the traffic volume [17] and travel time information [133, 167] at the city scale. Another example that is often mentioned is telecommunication data [160, 34], as the access of wireless network provides another type of trajectory. Finally, active crowdsourcing uses the similar technology to passive crowdsourcing. However, it requires active input in addition to the trajectory information. For instance, traffic events [9], driving experiences [152], air pollution [49], and fuel consumptions [55] are shared via vehicular social networks [90], which extends both the dimensionality and the coverage of traffic sensing.

The development of in-vehicle navigation system dates back to 1980s [121]. Conventional routing systems attribute a constant weight to edges of a road network and therefore handle the routing problem as the classical shortest path problem [23]. Depending on the focus of the route cost, the weight can be defined as the length of the road, real-time (essentially periodically updated) travel time [61], pollutant intake [15], or fuel consumption [55]. However, a signal value, i.e., weight, is still not powerful enough to represent the variable cost, e.g. travel time. As a result, advanced navigation systems either build up travel time predictions in the form of random variable [82, 169] or learn from human intelligence [181] in order to build smart routing services.

1.2.2 Parking Information & Guidance System

Smart parking system based on human report or dedicated sensors has been developed since last decades [95]. These solutions use human report [62, 21] or various sensors, such as camera

[27, 31], acoustic [120], ultrasonic [84], infrared [118], optical [39], magnetometers [87, 151], piezoelectric [173], or RFID [10, 191, 104] sensors, and inductive loops [84, 46], to monitor the availability status of a single or a fixed set of parking spaces. More recently, the sensors are installed on mobile vehicles to increase the coverage of the parking availability sensing [78, 111, 132]. However, suffering from the cost issues in installation and maintenance, these solutions are deployed in very limited regions.

Crowdsourcing using smartphones or other smart wearable devices, e.g. smart watch, gains more attention as it only uses existing sensors built in the smart devices without dedicated-sensor investment. These solutions use sensors, such as Bluetooth, WiFi signatures, motion sensors, and so on, to automatically detect the transition between driving and walking to estimate the parking/unparking events [124, 170, 123, 155, 105, 165]. Due to the limited penetration of these crowdsourcing systems, sampling theory is used to infer the parking availability of the background spots that are not covered by participant events [170]. For off-street scenario, the background parking availability is also inferred by using the heuristics that a parking lot is very likely to be full if a car goes in and comes out without parking inside [123, 140] and that parking search time (and lower-level vehicle maneuvers, e.g., brakes and turns) is related to parking availability statistics [38].

Due to the lack of parking availability information, the parking guidance system is still in the early developing stage. Most systems assume that parking facilities provide real-time availability [62, 128]. In this case, the preferred parking choice is specified either by the driver [77] or by a utility function [153, 62] based on the driver's trip destination, converting the parking guidance into regular traffic routing problem. Moreover, the parking requests from a group of drivers can also be coordinated to minimize the total cost, e.g., walking distance, of all drivers [65, 57]. For the system that provides probabilistic parking availability, the optimal search path can be solved as the stochastic shortest paths problem with history dependence [159].

1.2.3 Support Systems for Individual Transportation Modes

Besides the routing and parking support systems to facilitate general ground transportation, additional support systems are also proposed and developed to improve individual transportation modes, such as transit buses, taxi and ridesharing, and so on.

Transit Bus

Transit buses play an important role in human transportation, especially in urban areas. Smart public transport systems have been built to predict bus arrival time, to optimize bus route planning, and to understand urban mobility status. Accurate arrival time prediction greatly improves transit riders' travel experiences via public transportation and is offered by many transit agencies, e.g. [142], and online bus information providers, e.g. [61]. However, most of these services rely on the bus tracking module maintained by transit operating companies, e.g. [91]. Recently, crowdsourcing solutions based on riders' cellphone or smart devices are built to reduce the maintenance cost of dedicated tracking module yet still predict the bus arrival time accurately [196, 194, 52]. Bus route planning also becomes attentive because of the advances in human mobility modeling. The mobility data from other transportation modes, e.g. taxi [20, 103], or other resources, e.g. cellphone data [26], have been utilized separately or in combination [103] to build more efficient bus services. Finally, the transit bus records or trajectories also benefit the understanding of human mobility patterns and real-time traffic condition [192]. RFID technology has been widely used for people's commutes in ground public transportation systems, such as transit buses. As passengers swipe their card to get on a bus, transit record can be used to mine the origin and destination, e.g. home and the workplace, which benefits both the individual and city-wide urban planning for mobility [89, 88, 52]. The transit trajectories can be used to estimate traffic conditions [22, 193] to overcome the cost and privacy concerns to expose the sensitive information of private car or taxi passenger.

Taxi and Ridesharing

Taxi and ridesharing, e.g. Uber [8], DiDi [1], and Lyft [2], bridge the gaps between the private driving and the public transit transportation. Since the taxi or ridesharing drivers' income strongly depends on the passenger pickups, several systems have been proposed to suggest the pickup positions[56, 182] or passenger searching paths [137, 183, 186]. From the taxi or ridesharing dispatcher's perspective, they may need a system that assigns a new pickup request to the closest vehicle in terms of driving time to shorten the passengers waiting time. Alternatively, the dispatcher may also want a system that dispatches the participant vehicles in

a way in accordance to the real-time or predicted pickup demand so that the total idle time of the participant vehicles is minimized [113, 114]. From the passengers perspective, they want to minimize the waiting time for taxi especially when taxi reservation is unavailable. Systems, such as [183], are proposed to suggest locations within people's walking distance so that people can find a vacant taxi more likely. Finally, taxi and ridesharing carpool systems are also studies to build the win-win condition which saves both passenger's fare and the social cost, e.g., energy, while satisfying the transportation need [188, 187, 106].

Miscellaneous systems

In addition to the studies to improve transit bus, as well as the taxi and ridesharing services, there are also services presented in the following sub traffic systems:

- Bikesharing systems. The bike-sharing management systems need to focus on the prediction of pick-up and drop-off events because the differences between the two types of events cause bike congestion or starvation at a particular bike station [176, 97]. Accordingly, algorithms controlling the re-balance among different stations [58, 97, 176] or the algorithms that optimize the planning of bike stations [189] are also developed. To improve individual's riding experience, cyclist-annotated route and hazard are crowdsourced to improve fellow bike-riders' experience [136]. Furthermore, the bike availability is predicted to provide recommendations for station choice based on the historical transaction data [79, 29, 54].

- Electric vehicle (EV) charging service. EVs reduce the emission of greenhouse gases and are considered as green transportation mode. Due to the limited battery capacity and limited access to charging stations, EV routing problem has been studied as an extension of regular traffic routing problem, e.g. by taking into account the charging station choice and corresponding charging time [147, 53]. Moreover, the planning of EV stations is also studied by considering both the EV mobility pattern and the travel demand in urban areas [93, 92, 175, 171, 96, 117].

- Special vehicle services. The last kind of prevalence transportation service is related to the special vehicle service, which corresponds to the design of vehicles' routes and

schedules to serve multiple users who specify pickup and delivery requests between origins and destinations [43], e.g. pickup and delivery, dial-a-ride services, and ambulance services. Given the fixed location of the service vehicle and the request set, the problem is modeled and solved as Vehicle Routing Problem with Time Windows (VRPTW) [43, 158, 129, 168]. Another dimension of the study is planning the base location of the service vehicle ahead of any requests, so that the total cost, e.g. service time, can be reduced [146, 80].

1.3 Focus of the Dissertation

Among all the challenges faced by the current road transportation system, traffic congestions and the parking difficulties are recognized as the most prevalent [141]. As a result, the dissertation focuses on building the ARTS system to deal with these two problems. Specifically, for traffic congestion alleviation, we aim to extend the availability of reliable traffic condition estimation and based on the estimation results to develop a social optimal routing system, which minimizes the total travel time of all drivers by coordinating the whole routing request set and relieving traffic congestions. For the parking solution, we believe that the lack of fine-grained parking availability estimation is the main factor that impedes the full exploration of the capacity of the existing parking facility. As a result, we propose to build a cost-effective and reliable crowdsourcing system to enable large-scale and fine-grained parking availability sensing.

1.4 Challenges

Based on the two focus points of the dissertation, i.e., traffic congestion and parking difficulty, we discuss the technical challenges in the two following subsections.

1.4.1 Relieving Congestion through Traffic Routing

The efficiency of traffic routing first depends on sensing and predicting the traffic condition. The state-of-the-art traffic condition mining systems rely on extensive computation power for real-time traffic condition estimation. Due to the gap between the traffic sensor owner, e.g. taxi companies who obsess the real-time taxi trajectory for travel time estimation, and computation

service provider, the real-time traffic condition estimation is available in limited areas from the worldwide point of view. From the prediction perspective, routing system has the knowledge of vehicle's subsequent travel plan based on driver's destination input and current position. However, based on our best knowledge, real-time travel plans in the routing system are still isolated from travel condition estimation systems, hindering the full exploration of the possibility to predict future traffic condition.

The routing strategy greatly impacts the formation of traffic congestions. Nowadays, the commercial routing service provider applies a greedy routing strategy, meaning that the fastest route is suggested by only considering a single driver's routing request. However, due to the high penetration rate of the online routing services, the fastest routes of different drivers may be coupled, generating congestions on the route that is previously fluent due to the low traffic. To solve this problem, routing requests should be coordinated from the global perspective, so that the total travel time is minimized.

1.4.2 Improving Parking Efficiency by Understanding Parking Behaviors

Understanding human's parking behavior is fundamental to smart parking management. In particular, parking behavior models can be used in the simulation to determine the outcome of a planning change in the system, e.g., parking layout and pricing. However, except for some parking choice model, i.e., the model predicting which a driver would choose from the given options, most studies on human's parking decision model, which is primarily used to model a sequence of decisions when looking for parking in a simulation experiment, are not based on large scale real world data. Due to the difficulty to capture the situation that parking seekers experience and the decisions they make, researchers are still unclear how different factors influence the "take" or "ignore" decision when a driver faces an available spot.

In addition, real-time parking availability sensing is considered as the foundation of most smart parking systems, e.g. parking guidance or dynamic pricing. However, the difficulty in achieving such information has been the win-win-win condition among the cost, coverage, and accuracy. Systems relying on human reports or dedicated sensors, no matter with static or mobile deployment, incurs considerable installation or maintenance cost. Therefore, such solutions are suitable for the most competitive and valuable parking resources with limited

coverage, e.g. downtown area. On the other hand, the crowdsourcing solutions, e.g. based on smartphone built-in sensor, are considered low-cost since they automatically detect participant parking/unparking events. However, the body of work in this direction has not overcome the problem of low penetration rate: due to the influence of non-participant parking event, the availability states of the background parking spots, i.e., the parking spots that are not currently taken by any participant vehicles, are unknown. Though existing studies look into the sampling theory or some empirical heuristics for off-street parking scenario, the granularity and the ubiquity of the state-of-the-art crowdsourcing are still limited.

1.5 Thesis

The thesis of the dissertation states that:

(i) Real-time traffic condition can be estimated using low-sampling-rate trajectory data based on limited computation power. Together with the balanced routing strategy, traffic congestion can be proactively alleviated, reducing the average travel time of all drivers. (ii) Parking decision models can be learned from human's parking decisions captured using computer vision technology and can be used to crowdsource fine-grained parking availability, requiring only driver's parking search trajectory and final destination at run-time. These two components work together to improve human's road travel experiences.

1.6 Summary of Dissertation Contributions

This dissertation presents contributions in exploring the challenges of improving the road travel experience in traffic sensing & routing and parking availability estimation, which have been or will be published as follows:

- Themis: A Participatory Navigation System for Balanced Traffic Routing, *in Proceedings of 2014 IEEE Vehicular Networking Conference (VNC)*, pp. 159-166, 2014. [99];
- DoppelDriver: Counterfactual Actual Travel Times for Alternative Routes, *in Proceedings 2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 178-185, 2015. [86].

- Balanced Traffic Routing: Design, Implementation, and Evaluation, *in Ad Hoc Networks 37*, pp. 14-28, 2016. [100];

- Your Search Path Tells Others Where to Park: Towards Fine-Grained Parking Availability Crowdsourcing Using Parking Decision Models, *in Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT) 1.3*, 2017. [101];

- Real-Time Travel Time Estimation Based on Low-Sampling-Rate Vehicle Trajectory Data. [submitted for peer review].

In the dissertation, we present the design, implementation, and evaluation of the following systems: the Themis participatory traffic sensing & routing system and the ParkScan parking availability crowdsourcing system. Themis collects driver's trip information, i.e., origin-destination pair and departure time, and the driving trajectory to estimate the traffic speed and volume at road segment level. Moreover, it also applies several balanced routing algorithms to coordinate the travel plan of different drivers in order to reduce the travel time of all the drivers. The ParkScan system aims to crowdsource the parking availability in both off-street and on-street parking facilities using an infrastructure-free approaching. It is built on a generic parking decision model and based on the detected pass-by or parking/unparking events implied by drivers' parking search/vacating trajectory.

1.6.1 Traffic Sensing & Routing

We present the Themis system as a overall solution package for both traffic sensing and routing. Travel time estimations are computed using low-sampling-rate GPS trajectories through map-matching, travel time allocation, and aggregation processes. Since delayed traffic sensing results are considered as errors for real-time applications, we enforce the timeliness by using an approximation framework, which reduces the computation by abandoning the GPS reports in the spatiotemporal slices, i.e., a spatial region within a short period, where reports from other vehicles have been adequate. In the routing aspect, Themis applies the balanced routing strategy instead of the greedy strategy. In the new routing strategy, the participant traffic and the background traffic are predicted and coordinated so that the total travel time of all participant

drivers can be reduced compared to the situation where everyone uses the fastest path based on latest travel time estimation.

We evaluate the traffic sensing and routing functions of Themis system using the data from 26,000 taxis in Beijing. The results demonstrate that Themis traffic sensing provides accurate travel time estimation in real-time. More importantly, when the arrival rate of the GPS trajectory report exceeds the throughput capacity of computation power, the approximation framework maintain the equivalent accuracy with higher coverage, compared to the baseline without the approximation. The balanced routing strategy is proved to reduce up to 18% of drivers' average travel time. Moreover, the benefit of the balanced routing arises even when the penetration rate is as low as 7%.

1.6.2 Parking Behavior Modeling and Availability Estimation

We present the ParkScan system based on the study of drivers' parking decision behaviors. We collect a large-scale dataset that consists of more than 8,000 vehicle trajectories in the parking lot and 55,000 parking decision events, i.e, either taking or ignoring an available spot during a parking search process. With the features extracted from this valuable dataset, we build a data-driven model that reflects driver's underlying decision strategy. To the best of our knowledge, this is the first dataset at scale that records driver's real parking decisions with rich features and our model is the first data-driven parking decision model generated from large-scale real-world parking decision observations. Moreover, a crowdsourcing system called ParkScan is designed for fine-grained background parking availability estimations. ParkScan takes parking searching trajectories from participant vehicles as input. By combining with historical knowledge and static parking facility layouts, ParkScan leverages the learned parking decision model to generate probabilistic parking availability estimations at the parking spot level.

We evaluate ParkScan in both off-street scenarios (i.e., two public parking lots) and a city-scale on-street parking scenario. In the parking lot experiments, we utilize over 6,000 human parking search trajectories and real-world parking availability to conduct the evaluation. In the city-scale street parking evaluation, we conduct data-driven evaluations using over 63,000 parking requests generated from real parking transactions in an urban area covering 35 blocks

in Seattle, WA. Both of the experiments showed that ParkScan substantially reduces the errors of parking availability estimation compared to state-of-the-art solutions. Moreover, ParkScan shows good robustness for low penetration rates. In particular, even with a single participant driver, ParkScan brings down 15% of estimation errors for the parking spots along drivers' parking search trajectories.

1.7 Contributors to the Dissertation

The following is a list of contributors by chapters who co-authored the papers from which materials are used in this dissertation.

Chapter 2 of the dissertation presents the outcome of the Themis traffic sensing & routing project. This is a collaborative work with Dr. Badri Nath, Dr. Liviu Iftode, Hongzhang Liu, and Dr. Daehan Kwak from Rutgers University, Dr. Cristian Borcea from New Jersey Institute of Technology, Dr. Yong Xiang from Tsinghua University, China, and Yu Zhang from Beijing Institute of technology, China. Hongzhang Liu helped with the implementation of the Themis prototype and the organization of the field experiment. Yu Zhang contributed to the implementation of the approximation framework for travel time estimation as well as its evaluation experiment. Dr. Badri Nath, Dr. Liviu Iftode, Dr. Yong Xiang, Dr. Cristian Borcea and Dr. Daehan Kwak provided many essential insights for the project. Dr. Yong Xiang also contributed to providing the taxi data used for evaluation. Disco Lab members and some volunteer students in Rutgers University participated in the field tests and provided many insightful suggestions.

Chapter 3 presents the outcome of the human parking behavior modeling and parking crowdsourcing project while I was working with my advisors, Dr. Badri Nath, Dr. Liviu Iftode, and Dr. Desheng Zhang, in Rutgers University. My colleague, Yu Yang, contributed in the parking dataset preparation and the organization of the street parking experiments. Dr. Daehan Kwak contributed in parking dataset preparation and the presentation of the work. Disco Lab members provided many insightful suggestions.

1.8 Dissertation Organization

The remaining part of the dissertation is organized as follows. Chapter 2 presents the design, implementation, and evaluation of the Themis system, which covers both traffic condition sensing & prediction component and the balanced traffic routing component. Chapter 3 presents the data acquisition of the real-world parking decision dataset based on computer vision technology, the human parking decision modeling, and the design and evaluation of the ParkScan parking availability crowdsourcing system based on the learned parking decision model. Finally, Chapter 4 presents the conclusion of the dissertation along with the directions of the future work.

Chapter 2

Realtime Balanced Traffic Routing Based on Social Optimum

2.1 Introduction

The widespread deployment of mobile devices has led to many mobile apps for traffic navigation. According to Ericsson ConsumerLab, 29% of smartphone users in the U.S. used Google Maps or other smartphone navigation apps during morning commute in 2011 [51], and this number has increased since then. Given the similar number of dedicated navigation devices [107], the penetration rate of dynamic navigation devices or apps is now considerable.

Modern drivers equipped with GPS-enabled devices not only use the traffic information but also act as traffic information providers. Popular navigation apps, such as Google Maps [61] and Waze [9], have been prevalently applying the location and event reports from smartphone users to compute the estimated time of arrival (ETA) of the alternative routes. Driving experience and fuel consumption are also shared in novel systems [152, 55] to help users select among several route choices.

In spite of the progress, one strategic problem with the current navigators still exists: They greedily route drivers to the fastest path based on the periodically updated traffic conditions. The high penetration of these navigators potentially leads to the Braess's paradox [119] caused by the coupled route choices. For instance, in Figure 2.1, greedy routing leads all of the four vehicles to the same route. Since vehicles need time to reach the bottleneck of the planned route, subsequent drivers may have already made their decisions before the influence of previously routed cars is reflected in the traffic conditions. The coupled "best" route choices lead the drivers to traffic congestion and longer travel time instead of saving their travel time. Note that rerouting is not expected to solve the problem, either because rerouting is too late or because it leads all traffic to new "best" routes and causes new jams. For example, agent based simulation has demonstrated that the average travel time increases when half of the drivers follow the

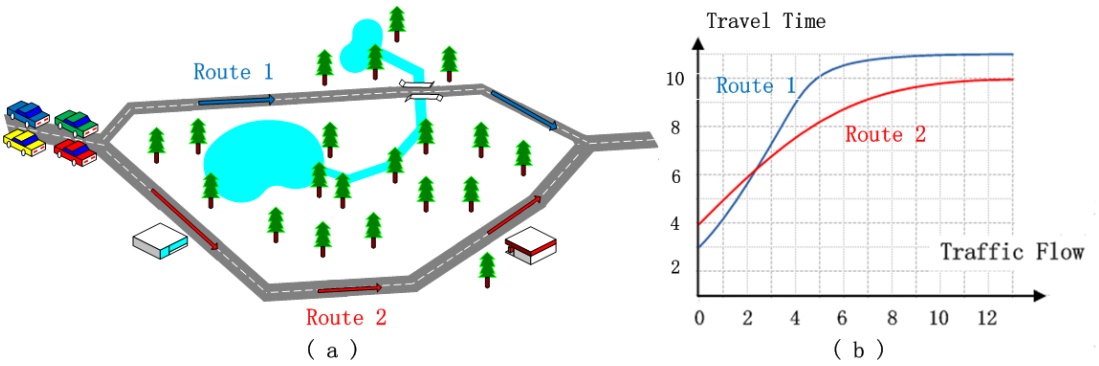

Figure 2.1: An example of different routing strategies. Four cars are about to choose route 1 or route 2 to go from left to right in (a). Due to no traffic, the latest estimated travel times are 3 minutes and 3 minutes for two alternatives, respectively. However, if each car greedily takes route 1, given the travel-delay model of the two routes in (b), the actual travel time for route 1 will be 9 minutes, three times of the estimated value.

dynamic fastest path based on real-time traffic [32]. Roughgarden [143] has also elaborated on the suboptimal global situation caused by the greedy routing without coordination.

A few routing algorithms have been proposed to overcome the drawback of greedy routing [172, 169, 94, 81, 67, 163]. These algorithms, employing *cooperative routing*, plan routes based on anticipated traffic volume (ATV) and corresponding predicted travel time (PTT) by assuming previously routed cars follow their suggested routes. For example, in Figure 2.1, two cars may anticipate the future congestion on route 1 and take route 2 instead, even if route 2 has longer ETA based on the real-time traffic. In our prior work [131], we also presented a cooperative routing algorithm, EBkSP, to route traffic based on both ETA and the popularity of the candidate routes.

Despite plenty of algorithmic studies, there are still no practical cooperative routing services deployed in real life. We believe this is due to two types of challenges. First, there are a number of practical aspects that have to be solved when building a cooperative routing service: (1) What routing algorithm should be used and how can it work at low penetration rates? (2) How to estimate the realtime travel time accurately? (3) How to predict the future traffic flow? (4) Most importantly, how to incorporate the above components in one system to provide cooperative routing services? Second, the service has to be evaluated in a meaningful way before deployment: (1) How to realistically evaluate the system without large-scale real-life experiments? (2) How to evaluate the performance under different penetration rates? This chapter

addresses both aspects.

In this chapter, we present a participatory system, Themis, which utilizes the data collected from road vehicles, such as location samples and route choice decisions, to estimate the traffic speed as well as the future traffic flow at road segment level. By learning and combining the multi-dimensional traffic information, Themis applies balanced routing algorithms to route the participating drivers to the routes that are good for both the drivers and the traffic ecosystem to proactively alleviate congestion.

Furthermore, we present a method to investigate how the performance of a participatory navigation system scales at different penetration rates by meaningfully expanding the real data collected from probe vehicles. We apply this method to the trajectory data from over 26,000 taxis and demonstrate that Themis outperforms greedy navigation systems. We find that the benefits of Themis emerge even when the penetration rate is as low as 7%.

Finally, we prototype Themis as a vehicular routing server and an Android smartphone app. The prototype system is validated in a field study with 16 cars and it illustrates Themis's benefits in terms of traffic distribution and travel time reduction in real world when the penetration rate is almost 100%.

The remainder of the chapter is structured as follows: Section 2.2 describes the Themis architecture and elaborates on the algorithms used by each component. Section 2.5 covers the implementation details of our Themis prototype system. The evaluation using city-scale synthetic experiments and neighborhood-scale field experiments are presented in Sections 2.6 and 2.7, respectively. Section 2.8 discusses the lessons learned and the future work. Section 2.9 reviews the related work. The chapter concludes in Section 2.10.

2.2 Themis Overview

The Themis app is assumed to be installed in a mobile device, which is equipped with GPS and wireless communication such as DSRC or cellular to connect with the Themis server. The route computation is carried out at the Themis server using a cooperative routing algorithm. While the route suggestions are retrieved by the Themis app to provide users with turn-by-turn directions, the app also uploads route confirmations and time-stamped positions to help deal

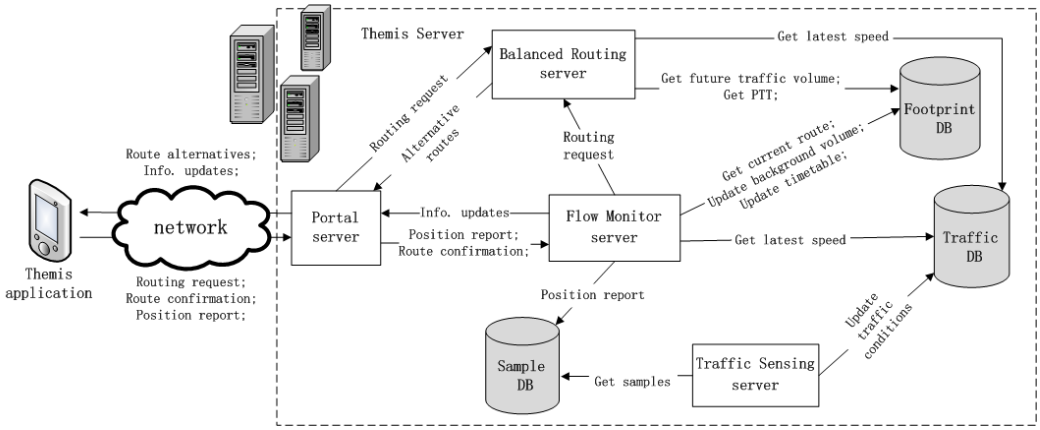

Figure 2.2: Themis system architecture

with subsequent navigation requests.

2.2.1 System Architecture

As illustrated in Figure 2, Themis consists of five executable entities: the Themis app on mobile devices, the Portal server, the Traffic Sensing server, the Flow Monitor server, and the Cooperative Routing server. While logically centralized, each server can be implemented in a distributed fashion to provide scalability. The Themis app allows the drivers to select from suggested routes, presents turn-by-turn driving directions, and uploads time-stamped position reports as well as the route selection decisions. The Portal server ensures the interaction between the Themis server and the Themis app; at the same time, it performs request dispatching and load balancing. The Traffic Sensing server estimates the travel time at road segment level. The Flow Monitor server supervises the cars' movement along their selected routes, triggers rerouting if needed, and estimates the future traffic that is scheduled to travel through each road segment. In addition, it is responsible to update the traffic information (e.g., ETA) for drivers. The Cooperative Routing server computes the route candidate(s) based on real-time speed, PTT, and ATV.

The information used by the Themis server is stored in three databases. The Traffic database stores the static road map, the traffic-delay model, and the latest estimated travel time of each road segment; this database is updated by the Traffic Sensing server. The Footprint database maintains the routes taken by the drivers and their status, such as the timetables containing

the ETA to each road segment included in the confirmed routes. It also stores the short-term predictions of the traffic flow on each road segment (i.e., how many cars will go through a road segment in the future) based on the routes being taken and the latest traffic conditions. The Footprint database is updated by the Flow Monitor server. The Sample database is only used to cache the position reports before they are periodically processed by the the Traffic Sensing Server.

Navigation Process

When a user issues a new navigation request, the Themis app contacts the Portal server with the origin-destination information. The Portal server forwards the routing request to the Cooperative Routing server, where route candidates are calculated using data from the Traffic and Footprint databases. The routing results are returned to the Themis app by the Portal server to generate alternative route previews. The user can choose one of the alternatives, and the Themis app translates the selected route into turn-by-turn directions. A confirmation of the selected route is meanwhile sent back to the Flow Monitor server to update the Footprint database.

Position Report Process

During regular driving, the Themis app periodically reports time-stamped positions (also called samples) to the Portal server, which are then sent to the Flow Monitor server. If the position is successfully matched to the previously confirmed route, the Flow Monitor server will provide the user with the latest ETA and earned route score (see Section 2.4) and update the timetable belonging to this route in the Footprint database; otherwise, a detour is detected and the Flow Monitor server will end the current confirmed route and issue a new routing request to the Cooperative Routing server. Note that, no matter whether the route matching process is successful or not, each position report handled by the Flow Monitor server will then be stored in the Sample database, from which the Traffic Sensing server retrieves samples to estimate the travel time of each road segment periodically.

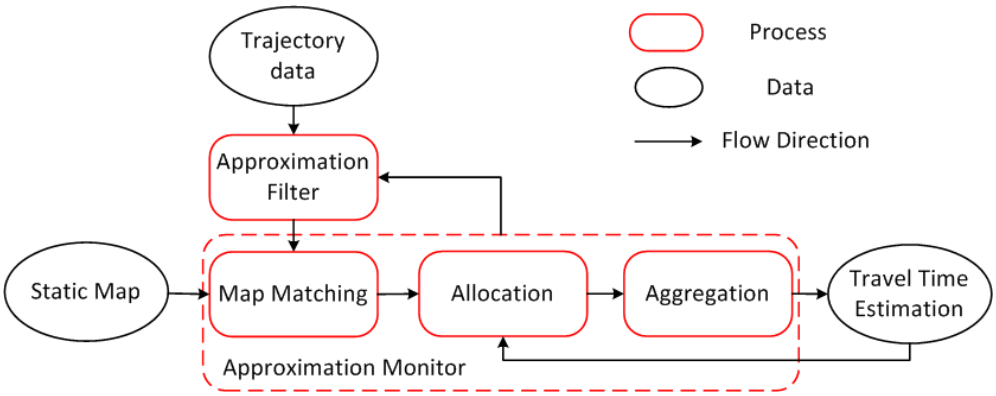

Figure 2.3: Travel time estimation framework.

2.2.2 Challenges

The efficiency of traffic routing first depends on sensing the real-time traffic condition. The state-of-the-art traffic condition mining systems rely on extensive computation power for real-time traffic condition estimation. Due to the gap between the traffic sensor owner, e.g. taxi companies, and computation service provider, the real-time traffic condition estimation is available only in limited areas from the worldwide point of view.

The routing strategy greatly impacts the formation of traffic congestions. Nowadays, the commercial routing service provider applies a greedy routing strategy, meaning that the fastest route is suggested by only considering a signal driver's routing request. However, due to the high penetration rate of the online routing services, the fastest routes of different drivers may be coupled, generating congestions on the route that is fluent due to the low traffic. To solve this problem, routing requests should be coordinated from the global perspective, so that the total travel time is minimized.

From the prediction perspective, routing system has the knowledge of vehicle's subsequent travel plan based on driver's the destination input and the current position. However, based on our best knowledge, real-time travel plans in the routing system are still isolated from travel condition estimation systems, hindering the full exploration of the possibility to predict future traffic condition.

2.3 Participatory Traffic Sensing

The participatory traffic sensing system takes the low-sampling-rate vehicle trajectory data (time-stamped GPS points) and a static map database as input and finally outputs travel time estimations for individual road segments. The raw GPS point reports first pass the preprocess which uses a slide time window with distance filters to abandon abnormal samples due to hardware failures or stopping for non-traffic reasons. After that, the Traffic Sensing server takes the steps shown in Figure 2.3 to estimate the travel time on each road segment.

Each trajectory episode needs to pass an approximation filter. After that, it will be processed by the map-matching component, where trajectories from different vehicles will be processed separately by different threads to infer the paths between consecutive GPS points. Since the time elapsed between each pair of consecutive GPS points is associated with the path connecting them, the travel time allocation module distributes the total time to each covered components. Finally, the distributed time on a road segment or an intersection based on different vehicle trajectories will be aggregated to provide one value as the output of travel time estimation system. Considering the spatial-temporal correlation of the road conditions, our system uses the historical estimations and the peer estimations, i.e., estimation of the road segments that are of the same type, to supplement the aggregation process.

Note that, each thread that runs the process from map-matching until the travel time aggregation is tracked by an approximation monitor. The tracking results will be used to configure the approximation filter to regulate filter threshold for future trajectory episodes. Depending on the temporal-spatial feature of a trajectory, a higher filter threshold means that a trajectory is less likely to be abandoned without further process. The filter will ensure that the limited computation capacity of the hardware resource can be wisely used to those trajectories that are considered more informative.

2.3.1 Map Matching

During the map matching process, samples are matched to the road map (i.e., to the most likely position on road segments), and the possible episode routes (i.e., partial routes) linking consecutive samples from each car are also inferred. Figure 2.4 shows how the episode routes are

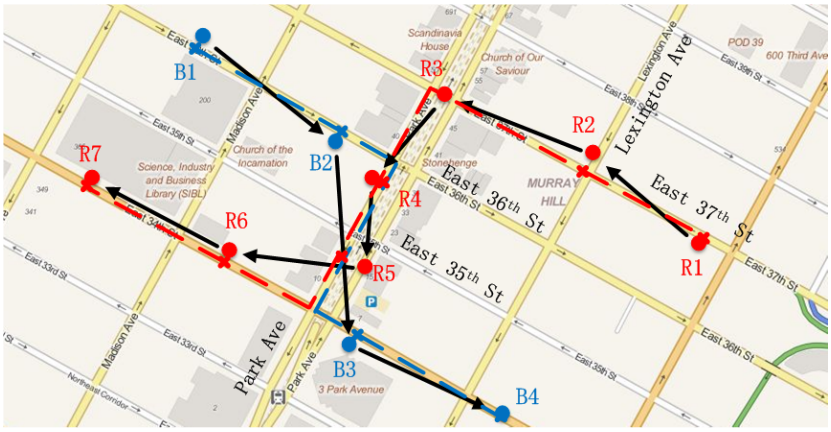

Figure 2.4: An example of participatory traffic sensing. Red and blue dots represent the trajectories from two different cars in the same interval. Their movement directions are illustrated using arrows. After the map matching process, the samples in each trajectory are matched to points along the roads shown with crosses. Meanwhile, the inferred episode routes that connect the matched points (shown in dash lines) are used for travel time allocation and aggregation.

constructed. Themis map matching is based on the Hidden Markov Map Matching (HMMM) method in [126], which considers both the distance to nearby roads and the context of each sample. For example, although sample R2 in Figure 2.4 is closer to Lexington Ave, it should be matched to East 37th St because it is unlikely for a driver to travel from R1 to R3 through Lexington Ave.

In [126], the context information used to compute the transition probability in HMMM is the length of the episode routes connecting adjacent samples. Themis enhances it by defining the transition probability based on Weighted Route Length (WRL). The motivation for this change is the observation that people tend to take main roads instead of lower-level (i.e., smaller) roads even if the length of low-level roads is shorter. Suppose p_i is the ith possible episode route between two matching samples, then:

$$WRL(p_i) = \sum_{e \in p_i} len_e * weight_{l(e)}, \tag{2.1}$$

where len_e is the length of the road segment e and $weight_{l(e)}$ is the weight associated with the level of the road segment e. The HMMM method essentially computes the probability of several episode route candidates, each of which is a function that uses the weights as its parameters. Therefore, the $weight$ of each road level can be learned from real data (i.e., training data) for different cities. In our case, we set the objective function of the training process so that the

sum of the probability differences between the ground truth path and all the other paths are maximized. Then, we determine the value of $weight$ using over 15,000 manually matched samples as the training set. Though this value is learned in the offline context using the data from Beijing, it can also be used for online map-matching algorithms, e.g. OHMMM [60], or for other cities.

2.3.2 Travel Time Allocation and Aggregation

The episode routes inferred through the map matching process may consist of multiple road segments and even partial road segments. The travel time allocation process divides the travel time observed on an episode route to the road segments covered by this route using the estimated travel time in the previous interval. For example, the travel time from B2 to B3 in Figure 2.4 is distributed to the two fully covered road segments and the two partially covered road segments.

Given an episode route p_i and the travel time observation τ_i, the travel time allocation process computes a travel time estimation for each road segment covered by p_i, defined as $R_{p_i}(e_{i,1}, e_{i,2}, ..., e_{i,n})$. The travel time on the road segment $e_{i,j}$ estimated from the episode route p_i, denoted as $\tau^n_{i,j}$, is computed as follows:

$$\tau^n_{i,j} = \frac{\bar{t}^{\,n-1}_{i,j}}{\sum\limits_{e_{i,k} \in R_{p_i}} \rho_{i,k} * \bar{t}^{\,n-1}_{i,k}} * \tau_i. \tag{2.2}$$

In this formula, $\rho_{i,j}$ is the fraction of the total length of $e_{i,j}$ covered by p_i. The parameter $\bar{t}^{\,n-1}_{i,j}$ is the travel time estimation on the road segment $e_{i,j}$ in the previous interval (i.e., interval $n-1$).

Suppose $p\,(p_1, p_2, ..., p_n)$ is the collection of episode routes covering a specific road segment within interval n. The aggregation process utilizes the time estimations for this segment drawn from each $p_i \in p$ and aggregates them into one travel time expectation value. As shown in Figure 2.4, by allocating the travel time of episode route (R3, R4), episode route (R4, R5), and episode route (B2, B3), respectively, we have three travel time estimations for the road segment from East 36th St to East 35th St on Park Av. The estimations are aggregated to get

$\bar{t}_e^n$, the travel time estimation of edge e in interval n:

$$\bar{t}_e^n = \frac{\sum_{p_i \in p, e_{i,j}=e} \tau_{i,j}^n * \rho_{i,j}/\tau_i}{\sum_{p_i \in p, e_{i,j}=e} \rho_{i,j}/\tau_i} \tag{2.3}$$

The aggregation process smooths the influence of the non-traffic factors, such as the driving style. Equation (2.3) calculates the weighted average value of individual estimations, which biases the estimation in favor of the episode routes with longer coverage and the episode routes with higher sampling rates.

Note that, when conducting the aggregation process, we group the travel time estimations along a same road segment grouped by their turning directions. Therefore, the Themis travel time estimation results are able to differentiate intersection transition time even though it is not explicitly isolated.

2.3.3 Approximation Framework

When the volume of the trajectories is too big to be handled by the provided computation resources in real time, GPS points overstock before the map-matching module can process them. As a result, the estimation results will be significantly delayed compared to snapshot when the trajectory points are captured. This produces inaccurate results for real-time applications and actually waste the computation resources.

In our travel time estimation system, we envision that the vehicle trajectory data shows temporal-spatial homogeneity, which can be explored to enable fast travel time estimation through approximation. For example, in the morning rush hours, most vehicles commute from the residential area to the office area. As a result, trajectory reports overlap a lot at the road segments connecting these two areas. Since the accuracy of the travel time estimation tends to be stable when the number of trajectories covering the road segment has been relatively large, the redundant trajectories can be abandoned at the approximation filter.

To achieve a fine-grained control on the approximation, the best consideration unit is the road segment, i.e., abandon redundant trajectories covering the same road segment. However, the approximation module is designed to reduce unnecessary computation and redundant points should be discarded as early as possible before too much computation is wasted. As a result,

the module is implemented before map-matching, where we do not know the road segment on which the vehicle is traveling. In our system, we divide the map into 1.5 * 1.5 square kilometer regions and we filter the trajectory based on these regions. Moreover, traffic conditions are split by time intervals so it is reasonable to count density in different time intervals separately. In summary, trajectories in different regions and time intervals would be filtered independently.

When a thread processes a trajectory, it checks the density of the region where the first point of the trajectory is located in the corresponding time interval. The part of the trajectory within the region would be discarded if the number of points in that region has exceeded the discard threshold in the given time interval. The discard threshold is dynamically adjusted by comparing the monitored actual processing delay to the user specified delay requirement.

2.4 Balanced Routing Routing

The transportation network in a city is represented as directed graph $G = (V, E)$, where E and V stand for the road segments and intersections, respectively. Depending on various road conditions, e.g. number of lanes, each road $e \in E$ has its cost $c_e(f_e)$, e.g. travel time, given that the traffic flow running on edge e is f_e. Therefore, we define $e = (u, v, c_e)$, where u and v are starting and ending vertices (i.e., intersection) of the road segment, i.e., directional road segment. In the graph, we define $\mathcal{N}_u^{in}$ as the vertices from which there is a directional edge to vertex u. Similarly, $\mathcal{N}_u^{out}$ represents the vertices which vertex u has a directional edge to.

Assuming the total travel demand $D = (d_1, ..., d_i, ..., d_n)$ is a vector, each dimension of which defines the travel demand between a particular pair of source and destination: $d_i = (s_i, t_i, r_i)$ represents that there are r_i vehicles travelling from vertex s_i to vertex t_i in G. Depending on the actually number of vehicles that have the source-destination pair info, r_i could be any non-negative integer. To define the routing problem, we propose an multiple-path routing schema, with each vehicle having a stochastic choice among different alternatives at an intersection: we define $p_i(u, v)$ as the flow partition of d_i on edge (u, v). Given the stochastic multi-path routing assumption, $p_i(u, v)$ essentially is the probability that the traffic flow defined by $d_i = (s_i, t_i, r_i)$ would go through road segment $e = (u, v, c_e)$.

2.4.1 Routing Cost

For regular greedy shortest-path routing strategy, the travel cost cared by a driver is the travel time experienced by itself. Therefore the travel cost on each edge e included in the driver's routing path π is defined as :

$$C_e^{greedy}(f_e) = c_e(f_e), \tag{2.4}$$

where f_e is the traffic flow on edge e. In reality, most online routing services use ETA under the current traffic condition as an approximation of $c_e(f_e)$ assuming the traffic volume would not change in a short term.

In the balanced routing problem, since we want to minimize the total travel time of all participant drivers, we need to formally define the cost as:

$$C_e^{social}(f_e^{(p)}) = c_e(f_e) + f_e^{(p)} c_e(f_e)', \tag{2.5}$$

$$f_e = f_e^{(p)} + f_e^{(bk)}, \tag{2.6}$$

where $f_e^{(p)}$ and $f_e^{(bk)}$ denote the flow of the participant traffic and background traffic (i.e., non-participatory traffic) on road segment e. Note that, the term $f_e^{(p)} c_e(f_e)'$ defines the additional cost incurred to other participant drivers on edge e. In fact, the greedy shortest path is the special scenario of our problem where $f_e^{(p)} = 0$, because the social benefit group only contains one driver. Meanwhile, the system optimal path is the other special scenario where every driver in the city is assumed to be included in the social benefit group.

2.4.2 Problem Formulation

It is proved in the prior study [37] that when all the traffic are under control, the equilibrium state for a cost function definition could be achieved by the Nash flow in the following convex

programme:

$$
\begin{aligned}
\text{Min} \quad & \sum_{(u,v)\in E} \int_0^{f_{(u,v)}} C_{(u,v)}(x)dx \\
\text{s.t.} \quad & \sum_{w\in\mathcal{N}_u^{out}} p_i(u,w) - \sum_{w\in\mathcal{N}_u^{in}} p_i(w,u) = 0 \quad u \neq s_i, t_i \\
& \sum_{w\in\mathcal{N}_{s_i}^{out}} p_i(s_i,w) - \sum_{w\in\mathcal{N}_{s_i}^{in}} p_i(w,s_i) = 1 \\
& \sum_{w\in\mathcal{N}_{t_i}^{in}} p_i(w,t_i) - \sum_{w\in\mathcal{N}_{t_i}^{out}} p_i(t_i,w) = 1 \\
& f_{(u,v)} = \sum_{i=1}^{n} p_i(u,v)\cdot r_i \\
& p_i(u,v) \geq 0
\end{aligned}
\tag{2.7}
$$

In the social routing scenario, when only a portion of the traffic is under control, the convex programme is modified to include some background traffic f^{bk} (i.e, non-participatory traffic) in the definition of the social cost (i.e., Eq. 2.5) as constant. Moreover, the optimization is done with respect to the participatory traffic f^p in the objective function and constraints.

To make the algorithm description easier, we rewrite the optimization problem into its general mathematical form:

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{i=1}^{N} g_i(x_i) \\
\text{Subject to} \quad & Ax = b \\
& x_i \geq 0, \quad i = 1, ..., N
\end{aligned}
\tag{2.8}
$$

In the above definition, $g_i(x_i)$ is a convex function; $N = |E|(n+1)$, representing the total number of variables that need to be determined. The first $|E|n$ variables, i.e., x_1 to $x_{|E|n}$, represent the traffic volumes of each travel demand on each edge, i.e., $p_i(u,v)$, and rest $|E|$ variables stand for the total participant traffic volume on each edge, i.e., $f_{(u,v)}$. A is a $M \times N$ matrix, where $M = |V|(n+1)$. x and b are two column vectors of dimensionality as $|E|\cdot(n+1)$ and $|V| \cdot (n+1)$, respectively.

2.4.3 Themis Routing Algorithms

The problem 2.8 essentially defines a multi-commodity flow problem [70] with separable non-linear convex optimization functions. As implied by the dimensionality of the constraint and variables, the computation time to solve the programme problem will be too long to provide real time routing services to a considerable amount of drivers in a large-enough road network, e.g. a city. This is also the reason why the equilibrium states based on various cost functions are so far only used to evaluate the global travel cost in off-line simulation studies [41, 37].

Two routing algorithms are implemented in the Themis routing server, a sequential balanced routing algorithm, and a heuristic balanced routing algorithm.

Sequential Balanced Routing

In problem 2.7, we essentially need to find a combination of routes for all participant vehicles so that the sum of their expected travel times is minimized. Strictly speaking, the expected travel time of a vehicle on a road segment e is determined by the cost function c_e, and anticipated traffic flow f_e, i.e., the sum of $f_e^{(p)}$ and $f_e^{(bk)}$, when the vehicle actually passes through road segment e. Since this problem is computationally hard to be solved in real-time. We make the following approximation:

- Instead of exploring all combinations of routes for all participant vehicle at the same time, we use a sequential search strategy: we compute the social optimal path for the participants one by one. When computing the route for the request of sequence number i, we assume all previously determined routes, i.e., the routing results for sequence number 0 to $i-1$, are fixed. This approximation reduces the search space in order to achieve real-time routing results.

- To make the compensation for reduced search space, we periodically reroute all traffic using random sequence numbers. This step will reduce the possibility that the sequential solution of a certain setting is substantially degraded from the social optimum. In addition, the rerouting process also iteratively considers new routing requests on the fly, which might have changed the optimality property of the previous routing results.

Algorithm 1 Sequential Balanced Routing Algorithm

Input: routing request set $\mathcal{R}$, traffic database $\mathcal{T}$, footprint database $\mathcal{F}$
Output: the social optimal route

```
repeat
    R = update_request_set(R);
    for all origin O, destination D ∈ R do
        π = A*_shortest_path(O, D, T, F);
        F = update_footprint(π, T);
    end for
    wait for next rerouting time
until R is ∅
```

Alg. 1 presents the framework of the sequential balanced routing algorithm. The framework employs a periodical rerouting framework. During each rerouting process, we first update the routing request set: we need to (i) add new requests proposed when the last round of rerouting has been finished and (ii) update the origin of each request since the rerouting should start from the vehicles' current position. For the updated routing request set, we sequentially compute the social optimal path using an A^* algorithm with the cost defined as the social cost, i.e., Eq. 2.5, considering only background traffic and previously routed traffic. Finally, the current rerouting result will update the distribution of the controlled traffic, which will be used for subsequent route computation. Note that, the free-flow travel time is used as heuristic in the A^* algorithm as it is the lower bound of the social cost. Since the sequential balanced routing algorithm uses social cost to compute the routing suggestion, we also call the sequential balanced routing as social routing.

Heuristic Balanced Routing

The sequential balanced routing algorithm requires the background traffic volume estimation and the learned cost function c_e for all the road segments. However, the information may not be available for certain cities. Therefore, Themis system also implements a heuristic algorithm as a light-weight version.

The basic workflow of the heuristic balanced routing algorithm is presented in Alg. 2. Themis first computes the alternative routes based on real-time traffic, and then executes a popularity scoring algorithm to evaluate the popularity of each possible route.

For the computation of alternatives (i.e., KSPs), we choose the Iterative Penalty Method

Algorithm 2 Heuristic Balanced Routing Algorithm

Input: origin $\mathcal{O}$, destination $\mathcal{D}$, traffic database $\mathcal{T}$, footprint database $\mathcal{F}$, similarity threshold θ, number of paths k, max iteration ϵ

Output: alternative routes with scores

1: $\pi = \varnothing$; {initialize the set of alternative routes}
2: π = get_dissimilar_KSPs ($\mathcal{O}$, $\mathcal{D}$, $\mathcal{T}$, θ, k, ϵ);
3: **for all** $\pi_i \in \pi$ **do**
4: π_i.score = compute_score(π_i, $\mathcal{T}$, $\mathcal{F}$);
5: **end for**

(IPM) [13] as simulation results showed that IPM could provide dissimilar KSPs with small computational cost. We define the similarity between two paths as the ratio of the length of the overlapped road segments out of the total length of the shorter path. By setting a threshold of similarity, IPM terminates either when enough dissimilar paths have been computed or when there have been too many iterations.

After computing the route alternatives, Themis associates a score with each alternative, which reflects the popularity of that route. The score is inversely related to the route popularity. Given that traffic conditions are periodically updated, routing cars to a route π_i results in lower score for any route that has overlapping road segments with π_i in a traffic condition update interval. Consequently, taking a route with a higher score does not create congestion on this route, and this decision is expected to alleviate congestion elsewhere. Themis leaves the route selection decision to the user, who is presented with the estimated travel time and the score of each alternative; it is up to the user to find a balance between these two parameters. By selecting an alternative with a higher score and with a good, but perhaps not the best ETA, a user will contribute to the global traffic ecosystem (i.e., the system's state will move closer to the system optimum).

To compute the popularity score, we applied the following equation based on the popularity computed in EBkSP [131]. The score of an alternative route π_i is defined as:

$$score_{\pi_i} = \frac{ETA_{avg}}{pop(\pi_i)}, \tag{2.9}$$

where ETA_{avg} is the average ETA for all the alternative routes and $pop(\pi_i)$ is the popularity of route π_i. $pop(\pi_i)$ is determined by both the ETA of the route and ATV that is scheduled to fully or partially travel through route π_i. Including ETA_{avg} in the equation does not influence the route choice. We add this term only to scale the magnitude of the scores.

2.4.4 Anticipated Traffic Volume Estimation

Assuming the penetration rate of Themis cannot be 100%, the ATV includes both the controlled traffic and the unexposed background traffic. In this subsection, we explain how the controlled traffic volume and the background traffic volume are estimated in the Flow Monitor Server.

The controlled traffic represents the cars that use Themis to plan paths and navigate to destinations. This part of traffic can be determined easily. Once a route is confirmed by the Themis app, the Flow Monitor server translates the planned path into a timetable, in which the ETA to each road segment in the path is sequentially estimated starting from the current location using the latest travel time estimations. The time-table is updated when a new route confirmation or location report is received. Based on the time-table, the controlled traffic volume can be estimated given a road segment and a time-stamp.

The method used to estimate the background traffic in Themis is based on [17], which proposed to extrapolate unexposed traffic volume based on naturally distributed probes in real time. In Themis system, we use the controlled traffic as the probes. As we can also get the total traffic volume on some baseline road segments (i.e., the roads where the governments have already deployed sensors to measure the total traffic volume), our system dynamically estimates the ratio of the controlled traffic volume to the background traffic volume on each road segment based on the known ratios of baseline road segments and the similarity between the road segments.

This algorithm has been proven to have good performance given accurately estimated real-time speed and controlled traffic volume generated by a large number of natural probes. In Themis, the estimation of real-time speed is discussed in Section 2.3. However, we cannot directly use the controlled traffic volume to infer the background traffic because the paths of the controlled traffic are not "natural" (as expected by the method in [17]) but influenced by the cooperative routing system. Currently, we assume the natural path to be the fastest path and estimate the natural traffic volume by assuming cars move using the latest estimated speed along the natural path. Although there may be exceptions when users would not take the fastest path in the absence of Themis, we leave more accurate natural path inference for future work.

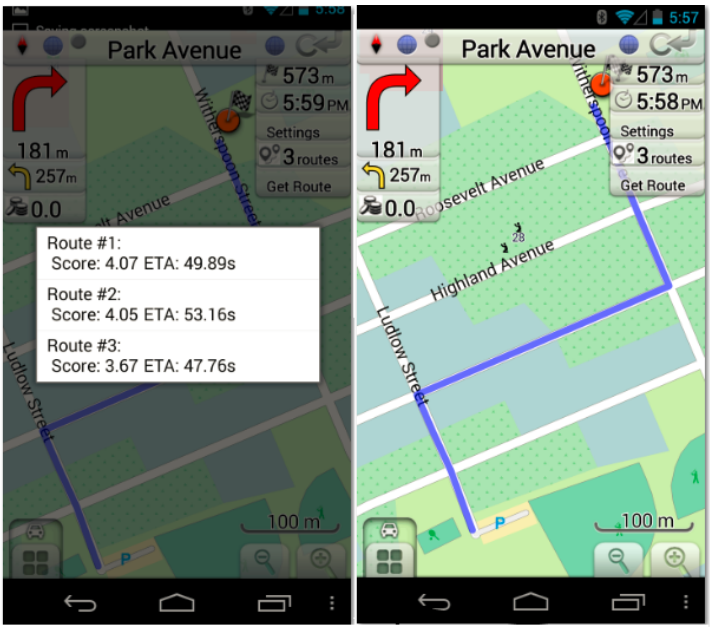

Figure 2.5: Themis application interface. The left screenshot shows three possible routes ordered by their scores and ETAs. After choosing one route from the three alternatives, a user enters the navigation mode (right screenshot), which provides turn-by-turn driving directions. During the navigation mode, the upper left part of the screen displays the driving directions for the next two steps. The score for taking the current route is shown below the driving directions. The labels in the upper right part show the distance to the destination, estimated arrival time, and other route choices.

2.5 Prototyping

We implemented the Themis app on Galaxy Nexus (Android 4.3) based on an open-source GPS navigator, OsmAnd [5]. The Themis app is implemented as a plugin, together with a customized map layer, such that it can easily be switched on or off. We inherit the interfaces of turn-by-turn driving assistance provided by OsmAnd and move the route calculation to the Themis server. The user interface of Themis is shown in Figure 2.5. Themis computes three alternative routes based on real-time traffic conditions and associates each route candidate with the popularity score and its ETA. During the navigation process, the app listens to position change events and routing requests from the user. The former triggers a potentially new position update request, and the latter directly starts generating a routing request. These requests are sent to and replied by the Portal server via a JSON Interface.

The Themis server is physically built on a Dell PowerEdge T110 server equipped with Intel

Xeon E3-1230 v2 3.30GHz Quad Core processor, 4×8GB 1600MHz DIMMs, and 2T 7200RPM SATA 3Gbps hard drive. The Portal server, Cooperative Routing server, and Flow Monitor server are implemented using PHP scripts. The Cooperative Routing server and Flow Monitor server expose their core functions to the Portal server. When a route confirmation, routing request, or location report request arrives, the Portal server directly calls the corresponding functions in either the Cooperative Routing server or the Flow Monitor server to handle the request. The Traffic Sensing server is implemented as three separate Java programs (i.e., map matching, travel time allocation, and travel time aggregation). As the location reports are stored in the Sample DB, the Traffic Sensing server just takes the records from the database for processing. For scalability, in each interval, the server launches four instances for map matching and travel time allocation; for the aggregation process, one instance is enough in our experiments due to the low complexity of the task.

The implementation of the Themis server combines several open-source software packages to provide cooperative routing. OpenStreetMap (OSM) [3] is imported into PostgreSQL[7] database as the source of static map data by using osm2pgrouting [4]. The traffic information estimations are also stored in the PostgreSQL database as properties of each road segment. The two algorithms presented in Section 2.4 are implemented based on the open-source routing library pgRouting [6].

2.6 Simulation Evaluation

This section analyzes the performances of Themis in synthetic scenarios modeling a city-scale deployment in Beijing using trajectory data from 26,000 taxis. Specifically, this experimental evaluation addresses the following questions:

1) How accurate and how fast is the Themis participatory sensing in estimating the traffic characteristics?

2) How does balanced routing compare with greedy routing in terms of traffic distribution and average travel time?

3) How do the traffic distribution and travel time vary with the system penetration rate and the total traffic amount?

The baseline routing algorithm in the evaluation is greedy routing (i.e., fastest route based on the latest ETA) and, for balanced routing, we assume that drivers choose the alternative route with the highest score.

2.6.1 Experimental Methodology

Dataset

We use GPS trajectories from a 26,000 fleet of taxis in Beijing during three consecutive Tuesdays starting from April 6, 2010, which amount to approximately 58,000,000 valid data points. Each sample contains the taxi ID, the timestamp, the geo-location, and the passenger status (i.e., free or occupied). They are sampled at intervals between 30 seconds and five minutes. To derive the total traffic, we use two other datasets: the daily percentage of taxi traffic out of total traffic on 3981 main road segments, and the citywide half-hour variation of taxi traffic out of total traffic. By checking both the covered area and the taxi penetration rate, we decided to carry out our experiment using the data in a rectangle area covering the Beijing second ring area, which is about 65 square kilometers. In this area, our taxi data account for almost 7% of the total traffic.

During the synthetic experiments, we imported the above datasets into the databases used by the Themis sever and processed them using our Themis server described in Section 2.5, including the Traffic Sensing server, the Flow Monitor server and the Cooperative Routing server. As a result, the Themis server is thoroughly evaluated with large amounts of real data even though this was historical data.

Modeling Trips and Penetration Rates

The first problem we solved in the synthetic evaluation is how to extract routing requests from the dataset and manipulate them to get different penetration rates. First, we identify passenger-on-board (POB) trips. The time-stamped location where the passenger status of the taxi changes to POB is recognized as the origin of a POB trip. The time-stamped location where the POB status is reset to a different status is considered as the destination. Since the number of POB trips is limited, especially during late-night hours, we also include non-passenger (NP) trips,

which are those trips happening between POB trips. We limit the max duration of NP trips to five minutes to avoid the influence of vacant wandering taxis. The routing requests generated in this way could be used to evaluate Themis when the penetration rate is no more than the original percentage of the taxis in traffic.

For the sensitivity analysis at higher penetration rates, we built a threefold and a sixfold routing request set. The threefold routing request set is generated by proposing each original routing request three times and adding a random in-between delay within five minutes. The sixfold routing request set is generated likewise. Therefore, the penetration rate can be increased to approximately 20% and 40%, respectively.

Note that the global penetration rate is carefully bounded by 40% such that, on any road segment, the traffic generated by the amplified routing request set does not exceed the total traffic estimated using the original routing request set (this will be presented in Section 2.6.1). Different penetration rates only change the proportion of the controlled traffic but do not influence any global traffic features such as traffic volumes and average speeds. The essence of amplifying the request set is that we extract some background traffic on consecutive road segments and use it to generate new trips based on real previous trips. Compared to methods that randomly choose the routing requests and expand them, our method preserves the traffic conditions implied in the taxi trajectories dataset. Therefore, it is a more persuasive setup to show the realistic influence of balanced routing in the targeted city. There are other alternatives to adjust the penetration rate such as modeling the traffic demand based on the original routing request set and then generating the amplified request set based on the traffic demand model. Due to the complexity of such methods, we leave them as future work.

Background Traffic Estimation

At each penetration rate, the background traffic volume equals the original total traffic volume minus the controlled traffic determined by different routing request sets (origin, threefold, or sixfold) and the original traffic condition. Based on the method from [17] (also described in Section 2.4.4) and our taxi ratio statistical data, we can derive the background traffic volume if we know the movement of the controlled traffic on the road segment.

Given any routing request set with original routes, the movement of the controlled traffic

can be simulated at the original average speed, which can be estimated by applying our traffic sensing algorithm (Section 2.3) to the taxi trajectory dataset. After each 15-min interval, we directly estimate the average travel speed on a road segment if there are at least two taxi trajectories passing through the segment during the interval. For the road segments that do not have direct estimations, we derive the estimated average speed using the speed of their joint road segments. As discussed in Section 2.6.1, the estimated speed when the penetration rate is 7% can also be used for the other two penetration rates (i.e., 20% and 40%). We call the speed estimated using participatory traffic sensing the *sensed speed*.

Learning the Traffic-Delay Model

If every car takes its original route, the average travel speed on each road segment would be the original sensed speed. However, to evaluate the travel time of the trips after rerouting, this original sensed speed is not useable as reroutings dramatically change the traffic distribution. Therefore, we use a traffic-delay model [28] to infer the travel time of a road segment based on the traffic volume on it during a given time interval, which is determined as described in Section 2.6.1. The speed derived based on the traffic-delay model is called the *inferred speed*, with the inferred travel time derived from it. The function used for the traffic-delay model is:

$$t_e(f_e) = T_e^0(1 + \alpha(\frac{f_e}{C_e})^{\beta}) \tag{2.10}$$

T_e^0 is the free-flow speed (i.e., speed limit). C_e is the capacity of the road segment e and f_e is the average traffic volume. $t_e(f_e)$ is the inferred speed based on the traffic volume f_e.

Using a similar method as [18], we learn the parameters α and β for 7,411 road segments from the total of 11,450 road segments. These segments were used because they have good direct travel time estimation. Fortunately, these models cover most main roads. During the evaluation, we only route traffic over road segments that have traffic-delay models such that we are able to measure travel time changes. For the comparison between the actual travel time of the original routes and the inferred travel time using the traffic-delay models, we also only use the routes fully covered by the traffic-delay models.

Traffic Movement Simulation

Given the route and the average speeds in each interval, we simply simulate the trip by assuming a car travels through each road segment included in the route using the average speed of the current interval. As we know the starting time of the trip, we can compute the starting time of each road segment step-by-step. When the time reaches a new interval, we use the average speeds for this new interval.

One difficult issue is how to simulate the rerouted trips. After rerouting, the movement simulation must be done using the speeds inferred based on the traffic-delay model. However, the traffic-delay model needs the traffic volumes to derive speeds. This is a chicken-and-egg problem because the traffic volume is determined by the movement of the traffic within the 15-min interval. To solve it, we use an iterative method to compute the speeds within each interval. In the first iteration, we use the sensed speeds to simulate the traffic movement. This step will end with new inferred speeds on the road segments. In the next iteration, we used the inferred speeds from the first iteration to update the speeds in the same interval. The process goes on iteratively until no speed changes on any road segment. Fortunately, the process converges very quickly because small speed changes do not influence the traffic volume distribution significantly.

2.6.2 Evaluation of Travel Time Estimations

Accuracy Test

The accuracy of either the sensed speed or the inferred speed is evaluated by a comparison between the average ground truth trip travel time and the average simulated trip travel time based on the corresponding speed estimations. To get the sensed speed, we apply the participatory traffic sensing algorithm (Section 2.3) to our taxi trajectory dataset. For the inferred speed, we apply the method in Section 2.6.1 to the 7% penetration routing requests set based on the learned traffic-delay model.

In the evaluation, we choose 100 random trips from the original request set in each 15-minute interval from 1:00 to 13:00 on April 6, 2010. We choose this period as it contains both peak hours and low-traffic hours. In addition, the number of taxis in operation changes

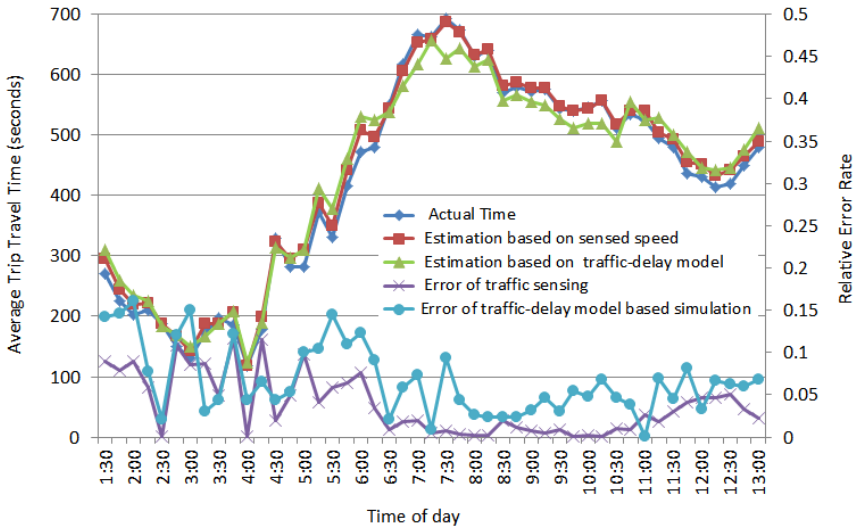

Figure 2.6: The accuracy of trip travel time estimation.

greatly during this period so that we can analyze the sensitivity of Themis participatory traffic sensing. At 4:00, only 7647 taxis (30% in our dataset) are running, while over 95% taxis operate between 9:00 and 9:30. For sensed speed evaluation, we use "leave-one-out" validation by not including the estimations from the car under evaluation into the aggregation step. However, the "leave-one-out" validation is not used for the inferred speed evaluation because the traffic-delay models are built using the data from three days, not specifically optimized for our evaluation set.

Figure 2.6 shows that the actual time and the simulated travel time computed based on sensed speed are closely matched, which means that the accuracy of our participatory sensing is high. During the period from 6:30 to 10:30, the relative error is less than 3%. The highest error comes during late-night hours when the participants are extremely sparse. However, even in this case, the relative error is below 12%. The results show that our participatory sensing algorithm has good accuracy and is robust to varying numbers of participants.

The travel time simulated using the inferred speed also matches the ground truth well. During the period from 6:30 to 13:00, the relative error is less than 10%. Similar to participatory sensing, the inferred speed also has a higher error rate during late night, which is less than 15%. These results prove that our traffic-delay model is acceptable.

In order to get the accurate evaluation of the balanced navigation system, we only carry out

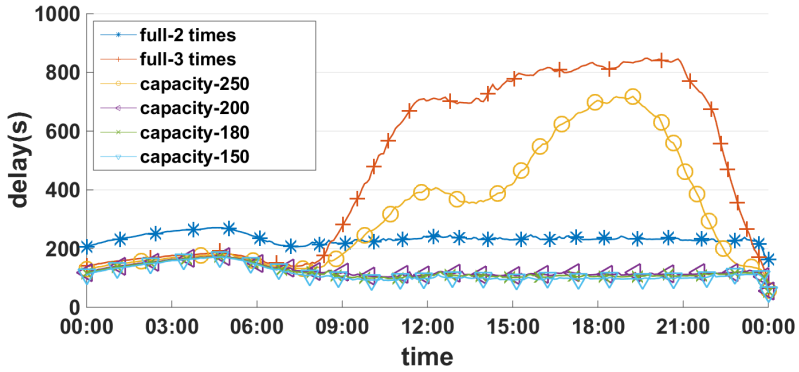

Figure 2.7: System delay at different capacity

the following evaluation experiments between 6:30 and 13:00. Another reason to abandon the period between 1:00 and 6:30 is that during this period the traffic load is too low to generate traffic congestion.

Timeliness test

In order to ensure the timeliness of the travel time estimation in our system, we apply an online algorithm for map-matching and introduce an approximation filter. In this subsection, we primarily focus on total process delay (including the travel time estimation computation time and the waiting time for GPS points) and the performance of the approximation framework using exaggerated trajectory volume to test its performance under data processing congestion.

Table 2.1: Performance of approximation framework in different settings

ID	replay rate	capacity (filter threshold)	discard ratio (%)	number of road conditions	error rate (%)
1	2	infinity	0	8876456	15.8
2	3	infinity	0	7008530	15.1
3	3	250	4.00	8039238	16.5
4	3	200	12.38	8663924	16.2
5	3	180	15.00	8568120	17.3
6	3	150	20.17	8350833	15.6

To simulate the congestion of trajectory points at the server, GPS point reports are replayed faster than their actual arriving rates. We conduct 6 groups of experiments using different approximation filter thresholds and replay rates. The settings and corresponding reliability

measures of the results are shown in Table 2.1. The overall processing delay, i.e., from receiving a GPS point report until the travel time estimation using this GPS report is computed, is illustrated in Fig. 2.7.

Fig. 2.7 shows that the processing delay under setting 1 (reply rate = 2) is stable even if no trajectory data is discarded, implying that our system throughput can be two times as high as the trajectory data generation rate of a 27,000 fleet of taxis. However, when we set the replay rate as 3, GPS point reports overstock, starting from the morning rush hour, and the processing delay increases significantly. When the approximation filter is initiated, i.e., the filter threshold is set ranging from 250 to 150, the system delay gradually returns to a stable level. Finally, the 3-time reply rate settings maintain an even lower system delay compared to the 2-time replay rate setting, because the time used to wait for next GPS points is shorter when replay rate is 3.

The overall processing delay is reduced at the expense of discarding GPS reports when replay rate is set as 3. Accordingly, we need to check the reliability metrics in Table 2.1. Interestingly, when the discard ratio grows from 0% to 20%, the coverage first increases in the beginning and then drops a little while the accuracy rate remains mostly stable. Since there is no data congestion and the same copy of the trajectory database is processed by our system regardless of the replay-rate setting, we can use the experiment performance when the replay rate is 2 as the oracle baseline (performance upper bound). The results in Table 2.1 demonstrate that when a proper filter threshold is set, the reliability of the estimation results is barely influenced compared to the oracle case. Specifically, the coverage is reduced by less than 2.5% and the error rate is increased by 0.4% when the filter threshold is set as 200. Compared to the situation where no approximation filter is enforced (setting 2), the approximation filter increases the coverage increases by 23% and the error rate by only 1.1%.

Note that, in this evaluation, we manually set different filter thresholds (i.e., discard ratio) to show the resulting performance. However, this value can also be adjusted dynamically when the system is running.

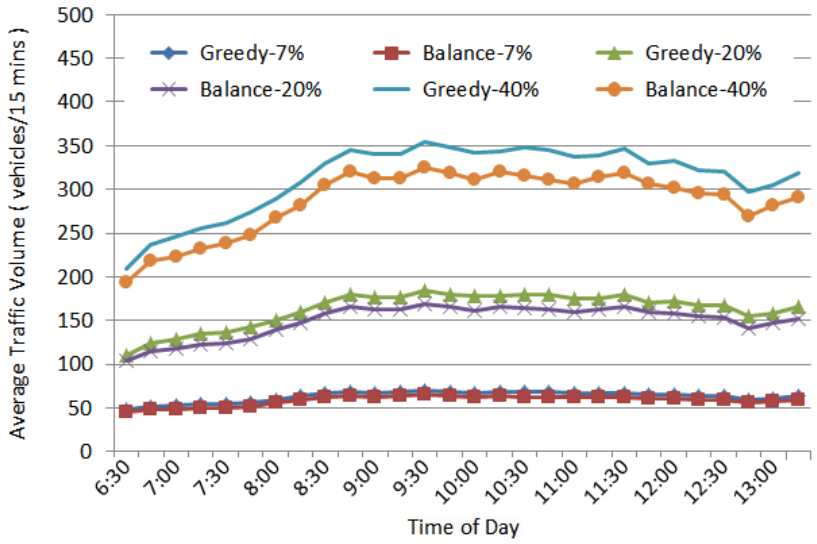

Figure 2.8: Traffic volume comparison at different penetration rates.

2.6.3 Evaluation of the Heuristic Balanced Routing Algorithm

Evaluation of Traffic Distribution

The average traffic volume is used as a global measure of congestion in the road network and is computed over all road segments that are traversed by at least one car. The higher the traffic volume, the less distributed the traffic; consequently, it is more likely to experience congestion in the network.

Figure 2.8 presents the comparison of the navigated traffic volume between the heuristic balanced routing algorithm and greedy routing algorithm (see details in Section 2.4) at different penetration rates. Each data point in Figure 2.8 represents the average traffic volume over every experimental road segment when the the controlled traffic is determined by the corresponding penetration rate and is routed using the given routing algorithm in the specified interval (computed as suggested in Section 5.1.5). The balanced routing results in a lower traffic volume at each interval for each penetration rate. These results demonstrate that balanced routing distributes the traffic better than the greedy routing. Furthermore, the traffic volume is reduced more substantially for higher penetration rates. We also observe that, during the experimental period, the relative traffic volume reduction is steady at each penetration rate regardless of congestion levels. Another interesting finding is that the relative traffic volume reduction tends to increase more slowly as the penetration rate increases. This implies that, as

the penetration rate rises above a certain threshold, the balanced routing's function to reduce global traffic congestion will linearly increase with the penetration rate.

Evaluation of Trip Travel Time

One danger of the heuristic balanced routing algorithm used in Themis system is that it could lead to longer trips when distributing traffic to unpopular routes. We define two criteria to compare the performances of balanced routing and greedy routing in terms of trip travel time in each 15 min interval:

$$FR = \frac{Number(A's\ travel\ time < B's\ travel\ time)}{Number(all\ the\ routing\ requests)} \tag{2.11}$$

$$RTTR = \frac{B's\ travel\ time - A's\ travel\ time}{B's\ travel\ time} \tag{2.12}$$

Faster rate (FR) reflects the percentage of routes suggested by A that are faster than those suggested by B. Relative travel time reduction (RTTR) reflects to what extent the routes suggested by A are faster than those suggested by B. In both definitions, A refers to Themis balanced routing while B is the greedy routing (i.e., fastest path based on real-time traffic sensing).

Figure 2.9 shows that over 50% of routes suggested by the heuristic balanced routing have shorter travel time than the greedy routing at any interval for any penetration rate in our experiment. Specifically, the average FRs for 7%, 20%, and 40% penetration are 55%, 58%, and 61%, respectively. The results demonstrate that the balanced routing provides users with a higher chance to achieve shorter travel time. As the penetration rate increases, FR also rises. Thus, the more users adopt Themis, the higher the probability that users will reduce their trip travel time. Another finding is that even at 7% penetration rate, Themis users could still expect lower travel time, which could be a motivation to change the greedy routing behavior during the bootstrapping stages of Themis.

Figure 2.10 shows that the heuristic balanced routing reduces the average trip travel time substantially, e.g., as much as 15% at 40% penetration rate. As expected, the higher the penetration rate is, the lower the trip travel time. Another significant trend of travel time reduction is that RTTR is determined by both the traffic density and the penetration rate. During moderate-traffic hours (10:00 to 12:00), Themis averagely reduces travel time by 8.2% for 40% penetration, 4.0% for 20% penetration, and 2.7% for 7% penetration. These results illustrate that the

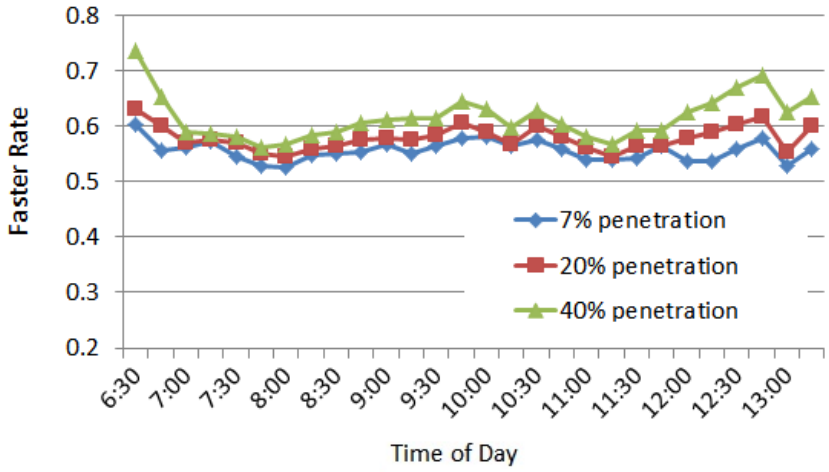

Figure 2.9: Comparison of FRs at different penetration rates.

RTTR achieved by Themis over greedy routing is substantial and increases with the penetration rate for moderate traffic. In addition, the relative travel time reduction increases faster when the penetration rate rises from 20% to 40%. Therefore, better RTTR can be expected when the penetration keeps rising beyond 40%.

During morning commute (6:30 to 9:30) and lunch time (12:30 to 13:00), though the heuristic balanced routing still leads to lower average travel time, it is less significant than that in moderate-traffic hours. Moreover, the penetration rate does not have much influence during these intervals. One reason for this result is that these periods represent rush hours in Beijing when most road segments are crowded and there are not many better options to choose even for the balanced routing algorithm. Another reason is that the number of taxis in operation is small at 6:30 and gradually increases to normal level until 9:00, which implies the actual penetration during this period can be lower than daily average value.

2.6.4 Evaluation of the Sequential Balanced Routing Algorithm

Travel Time Comparison

Fig. 2.11 demonstrates the average trip travel time by using either greedy routing or the sequential based balanced routing (also called social routing) at different time of day. One can clearly find that sequential balanced routing provided by Themis system leads to shorter travel times than greedy routing. More specifically, Fig. 2.12 presents the ratio of reduced trip travel time

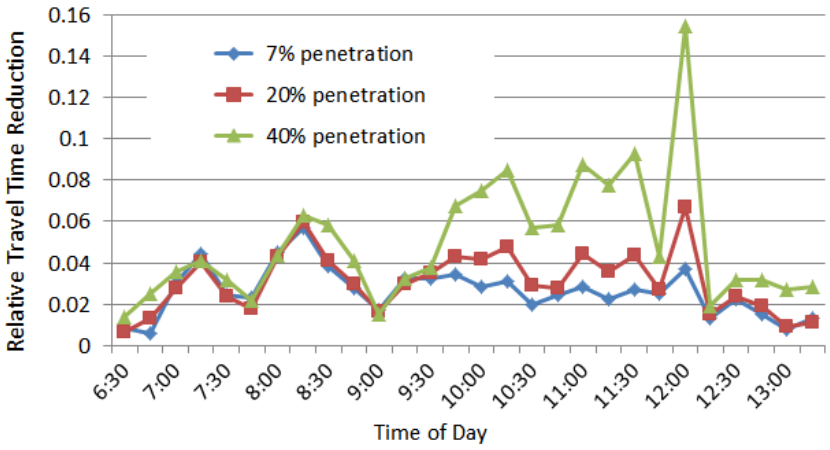

Figure 2.10: Comparison of RTTRs at different penetration rates.

by using social routing compared to the greedy routing. The results show that social routing achieves better performance in general during the morning rush hours and it can reduce up to 10.8% of the drivers' travel time compared to greedy routing. Note that this performance gain is achieved when only 7% of the total traffic adopts social routing. If the penetration rate increases, we expect the performance gain to also rise. Unfortunately, due to lack of information, e.g. we do not have realistic ratio between the increased participant traffic and the total traffic to predict background traffic volume, we cannot verify our expectation.

Response Time

Fig. 2.13 demonstrates the average response time for new routing request during the taxi dataset evaluation. The majority of the routing computation response times are found within 10 milliseconds. The trend that early hours tend to have higher response time is because during those intervals the total number of routing requests is small. Since we update the routing data structure based on latest ETA at the beginning of each interval, the preparation time is distributed to less routing requests on the early intervals.

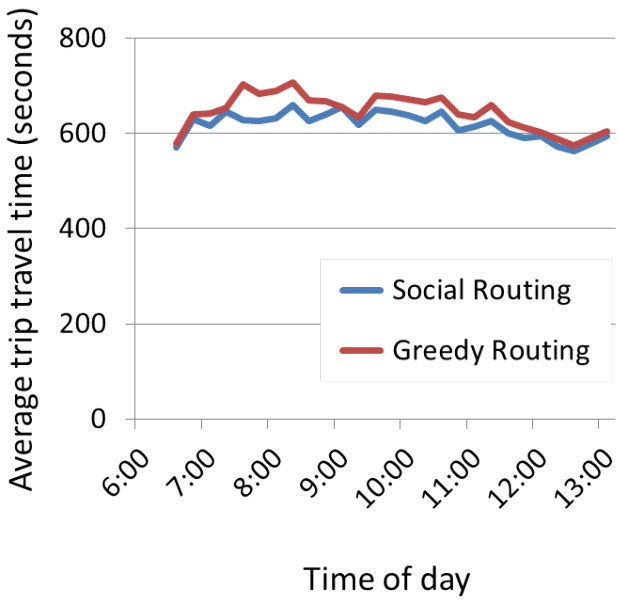

Figure 2.11: Average trip travel time under different routing strategies

2.7 Field Evaluation

We carried out the field study for balanced routing by scaling down the experimental area to a small neighborhood with sparse background traffic. Compared with the simulation experiments, the field experiments are capable of evaluating the real influence on traffic rather than the estimated one. The field experiments also have the ability to test the Themis app when users are driving on the roads. However, since we do not have enough data to build the travel cost function, we only evaluate the routing result for heuristic balanced routing algorithm.

2.7.1 Experimental Setup and Methodology

During our field study, we recruited 16 volunteers to drive their cars using the Themis app installed on their Android phones. Due to the limited control traffic, one critical issue was how to design the experiment scenarios such that routing with different algorithms (i.e., heuristic balanced routing vs. greedy routing) would show representative results. We chose a neighborhood shown in Figure 2.14 to solve this issue: First, the neighborhood provided multiple alternative routes with similar distances, i.e., four possible routes existed between positions A and B. Second, the background traffic was low such that the background traffic conditions were

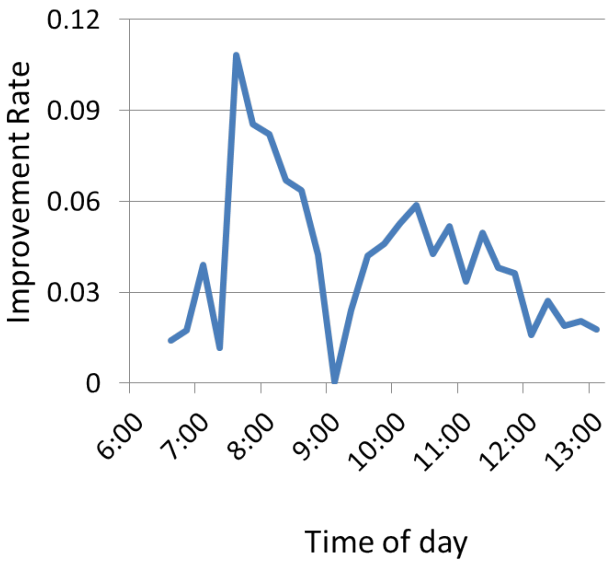

Figure 2.12: RTTR of approximated balanced routing algorithm

almost identical during the experiments for balanced routing and greedy routing. Finally, the majority of road segments in this neighborhood were one lane and around 50 meters long with stop signs at all intersections. This layout made the average travel delay on the road segments highly depend on the traffic volume even if the number of cars was small. As a result, the experiments conducted in this neighborhood allowed us to observe the performance differences between balanced routing and greedy routing for both traffic distribution and average trip travel time.

We note that traffic density is essentially determined by the vehicles' arrival rate at a road segment. Therefore, we designed two groups of experiments, synchronous and asynchronous, to mimic high-traffic scenarios and low-traffic scenarios, respectively. During synchronous experiments, all participants submitted routing requests simultaneously to get to B from A. After all the cars had arrived at B, they requested to come back to A simultaneously. The synchronization ensured higher vehicle arrival rate even if the total number of vehicles was small. In the asynchronous experiments, all the setups were the same except that the drivers freely drove from A and B and then came back without synchronization at either positions.

For both groups of experiments, we started with the balanced navigation mode and after 15 minutes switched to greedy routing mode. Out of the 16 experimental cars, 14 submitted routing requests and followed the navigation while the other two traveled around the neighborhood

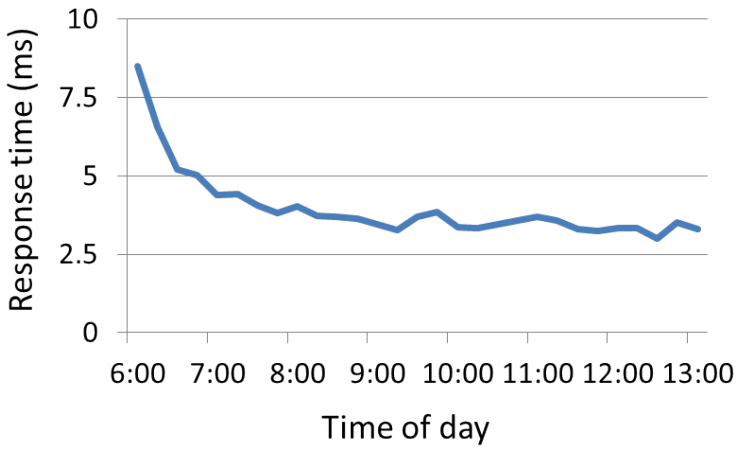

Figure 2.13: Average response time for new routing requests

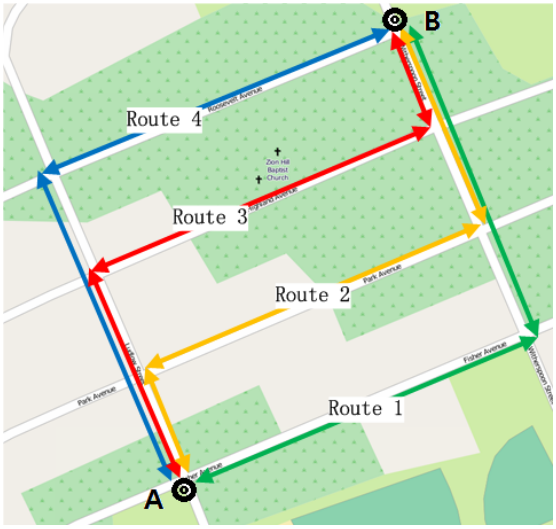

Figure 2.14: The map of the neighborhood for field experiments.

to collect traffic reports from the road segments where no experimental vehicle traveled. The position report frequency was set to 30 seconds. The Traffic Sensing server updated the traffic information every 5 minutes (i.e., traffic estimation interval was 5 minutes).

We defined the trip from A to B as to-trip, and the trip from B to A as return-trip. For both to-trips and return-trips, there were four possible routes (as shown in Figure 2.14). We named the eight routes using the route id and direction; for example, “route_1_to” referred to the trip from A to B taking route 1.

	Synchronous Experiments	Asynchronous Experiments
Number of Trips	83	86
Mean of the relative errors	-0.025	0.003
SD of the relative errors	0.039	0.013

Table 2.2: The statistic summary of the relative sensing error in the field study

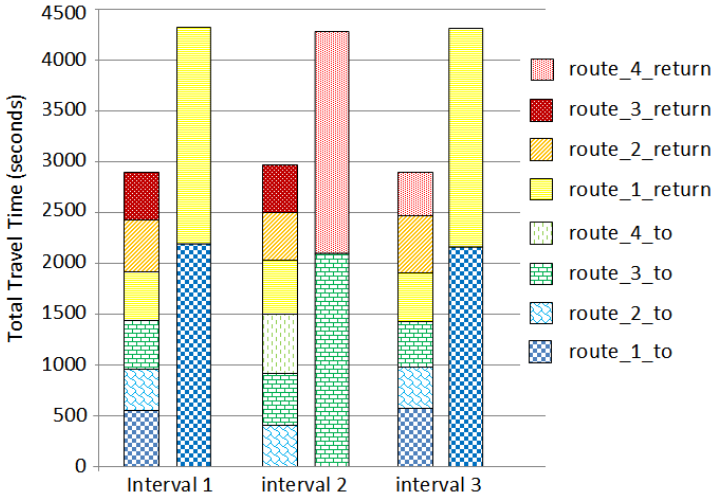

Figure 2.15: Distribution of total travel time in the synchronous experiment

2.7.2 Sensing Accuracy

Similar to the synthetic experiments, we first tested the accuracy of the participatory traffic sensing. We compared the ETA of a path shown to the drivers with the actual travel time (ATA) of the same path in the previous interval, as ETA was always computed based on the location reports from the previous interval. For each pair of ATA and ETA, we computed the relative error.

The statistics summary for each group of experiments (i.e., synchronous and asynchronous) is presented in Table 2.2. These results confirm the results from the synthetic experiment: a low relative error is observed, which proves the high accuracy of the Themis participatory sensing algorithm. Note that the synchronous experiments show a slightly higher error rate. This is due to the fact that the variation of the actual travel times of different cars is larger in synchronous experiments because of the stronger queueing effect at each intersection or stop sign.

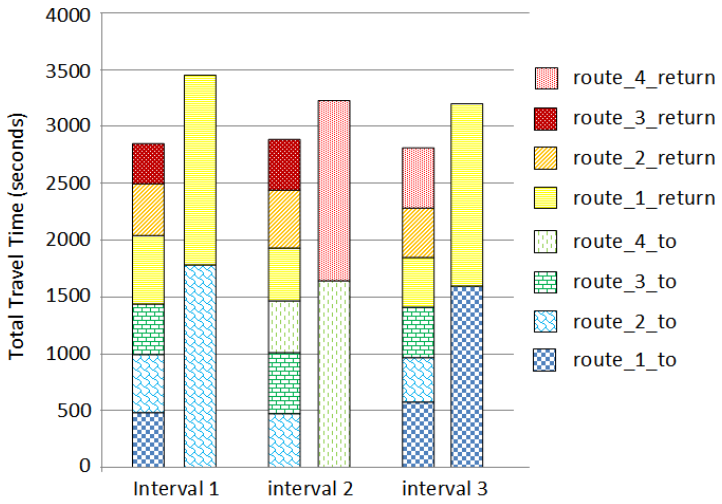

Figure 2.16: Distribution of total travel time in the asynchronous experiment

2.7.3 Traffic Distribution and Trip Travel time

Figure 2.15 and Figure 2.16 compare the total travel time of all the cars as well as the travel time distribution on each route under balanced routing and greedy routing for the synchronous and asynchronous experiments, respectively. In each interval, the first bar indicates the results of balanced routing, while the second bar represents the results of greedy routing.

In each figure, since traffic conditions are not recomputed within each interval, greedy navigation assigns traffic to only one to- or return- route with the lowest ETA. On the other hand, balanced navigation reduces the travel time significantly by balancing the traffic load on different alternatives. By checking the ETA of each route in Themis, we also find the balanced navigation to favor the routes with shorter ETA or the routes sharing less road segments with the other alternatives. Compared with the results from the synthetic experiments, both groups of field experiments get higher time savings under relatively low traffic density. This is partly due to the fact that the neighborhood we choose has the ideal layout for balanced routing, i.e., it has multiple, almost equivalent routes between the origin and the destination. In addition, the high penetration rate also help to bring about better time savings.

The comparison between Figure 2.15 and Figure 2.16 shows that the ratio of reduced travel time shrinks from 32% to 13.5%. This can be explained by the fact that the congestion induced by the greedy routing in the asynchronous experiments is alleviated by the lower traffic density

(i.e., lower traffic arrival rate than that in the synchronous experiments).

2.8 Discussion

2.8.1 Scalability

The Themis system is designed to work in real time. The position samples are processed in every time interval to update the travel speed estimations. The flow-estimation algorithm re-estimates the traffic volume when cars change their travel plans. The cooperative routing module performs the route planning in real time once it receives a new routing request. As a result, the scalability of the Themis system deserves careful discussion.

In the Themis prototype system, since there was only one physical server, the request dispatching of routing and flow monitoring tasks relied primarily on the PHP concurrency. For the traffic sensing task, we launched multiple map-matching and travel time allocation instances to parallelize these processes and applied the approximation framework to ensure the timeliness. During the synthetic experiment, the server was barely powerful enough to provide real time computation for traffic sensing and traffic routing at the same time in the experimental area. When the penetration rate was 40%, the average computation time of the heuristic balanced routing was almost 0.8 s. Meanwhile, the average computation time for traffic sensing in an interval is 878 s, slightly shorter than the length of the interval, i.e., 900 s. While these performance results have proven that city-scale balanced routing is not intractable on a single server, the settings in our prototype system will face scalability issues, especially for the traffic sensing tasks, if the covered area or the penetration goes beyond the values we have tested (i.e., 65 square kilometers with 7% penetration rate).

Therefore, we plan to explore a parallel implementation for Themis and dispatch the tasks onto multiple machines. The participatory traffic sensing is an ideal "map-reduce" model, where the map-matching and travel time allocation processes for each car are the "map" tasks and the travel time aggregation processes for groups of cars are the "reduce" tasks. The flow estimation is done on each road segment, and thus, the map could be partitioned and processed in parallel. In addition, each component is isolated, with data being exchanged only through the shared databases. For instance, the cooperative routing algorithm executes merely on the data

from the traffic database and the footprint database. Therefore, the scalability of the balanced routing system mainly relies on the concurrency of the databases, which has been extensively studied (e.g., NoSQL databases).

2.8.2 Balanced Routing Algorithms

The Themis system implemented the sequential balanced routing algorithm and the heuristic algorithm based on EBkSP as a first step, and we used this algorithm as an example to compare the performance of the balanced routing and the greedy routing. As to be discussed in Section 2.9, most cooperative routing algorithms require similar information. Consequently, they can also be implemented into Themis easily and evaluated using the same methods described in Section 2.6.

The sequential balanced routing algorithm currently can only compute one route suggestion for each pair of source and destination. However, most state-of-the-art navigation systems provides alternative routes for users to choose from. The heuristic balanced routing algorithm provides alternatives: by incentivizing the least popular route, the algorithm based on EBkSP increases the entropy of the traffic subsystem (i.e., the partial road map) only containing the planned alternative routes, and consequently, the entropy of the whole traffic system, which relates to the system-wide degree of balance. However, during our field study, we found it difficult to translate the meaning of each score into a simple metric understood by drivers. Our test drivers believed this was critical in the successful adoption of the system. In addition, the optimality of the heuristic balanced routing algorithm has not been proven even if 100% penetration rate is assumed.

We plan to investigate more human acceptable solutions to solve the balanced routing problem. For example, the additional delay incurred to other drivers could be defined as a more meaningful score to minimize the global travel time. Moreover, frequent taxi trajectories could be used as alternatives to incorporate taxi drivers' intelligence.

2.8.3 Bootstrapping the System

During the initial deployment of Themis, there might not be enough users to sense the city-scale traffic conditions or to collect plenty of routing requests. However, the participatory traffic sensing algorithm in Section 2.3 and the background traffic estimation algorithm in Section 2.4.4 can also be used in conjunction with external datasets. For example, many taxi companies sell their real-time trajectory data. For the traffic sensing task, the Themis system can also utilize popular online services such as Google Maps [61] or Waze [9] to get their ETAs.

Since Themis reduces the global traffic volume and travel time, it could be useful to prevent congestion and reduce pollution. The governments may want to incentivize drivers to participate by rewarding the users who contribute to a better traffic ecosystem. For instance, drivers who usually take high-scored routes may receive discounts for their vehicle registration. We believe this approach will help build a friendly traffic ecosystem and ideally lead the traffic system to its system optimum.

2.9 Related Work

The Themis system estimates the road segment level travel time to support the balance traffic routing. Therefore, both travel time estimation systems and the cooperative traffic routing systems are related to this study.

2.9.1 Travel Time Estimation Systems

Traffic condition estimation using dedicated sensors, e.g. loop detectors [40], RFID sensors [14], and cameras [45], is prevalent for highway systems based on classical models. [166, 184] both adopted extend Kalman filter to estimate and predict traffic conditions on the highway. [195] proposed a two-stage stochastic model to estimate travel time with only sensor data. The models mentioned above provide high accuracy but are not suitable for dense urban road networks because of the complex urban traffic patterns and the investment of installing dedicated sensors.

Vehicle trajectory data has been widely utilized for large-scale traffic condition estimation [134, 59] and prediction [122, 174] in urban areas. While trajectory data with high sampling

frequency [48, 177, 178] and rich features [116] can be easily use for travel time/speed estimation, due to cost concerns, most trajectory data contains only time-stamped GPS point at a low sampling frequency for travel time estimation [19, 44, 100, 68]. In addition, travel time estimation based on low-sampling-rate trajectory is so far discussed mostly in the offline processing context or through cloud computing platforms with extensive computation resources [19, 44]. Finally, since a large portion of urban travel time is spent at intersections, e.g. waiting for a traffic light, intersection transition time gains attentions recently to improve the travel time estimation accuracy. Although existing studies isolate the intersection transition time from road segment travel time, e.g. by either determining the range of intersection [185, 102], these methods work only for high-sampling-rate trajectories.

The system presented in this chapter focuses on processing low-sampling-rate vehicle trajectory data to provide real-time travel time estimation over limited computation resources. Moreover, we explicitly aggregate the travel time along a road segment grouped by their turning direction, therefore are able to differentiate travel time estimations for different intersection transition time.

2.9.2 Cooperative Routing Systems

Several algorithms were designed to solve the cooperative routing problem, and they can be divided into three categories.

The first category of algorithms focuses on user equilibrium [66], which computes the PTT of road segments and plans the fastest path for a driver based on PTT. Since PTTs increase with ATVs, which include the previously routed cars, the algorithms of this category automatically route subsequent traffic to the alternative paths if previously routed traffic have made the current fastest path (i.e., fastest path based on real-time traffic) suboptimal. Yamashita et al. [172] used Greenshield's model [63] to relate PTT to ATV and designed the Passage Weight heuristic to generate the contribution of each planned path toward ATV. The work in [169] used a similar model to relate PTT and ATV except that it assumed the traffic volume to be stochastic and determined by both historical traffic and previously assigned traffic. In [67], the authors proposed to compute a few alternative routes based on real-time traffic and then route the car to the path with the shortest PTT based on encounter prediction.

Other studies aim to route the traffic to achieve the system optimum of the transportation network, which does not necessarily minimize the individual driver's travel time but minimizes the average travel time of a group of users, e.g., all the drivers in a city or the users of a certain navigation system. The routing algorithms based on system optimum are also called social navigation [163]. Bosch et al. [163] proposed to handle a routing request by searching a path minimizing the total PTT of all previously assigned drivers. Lim et al. [94] proposed to compute a few route candidates based on real-time traffic and investigate the mutual timing influence of users' route choices based on the BPR flow-delay model [28]. This work assigns a group of drivers to the combination of paths that optimize the total travel time, and the algorithm was evaluated using taxi trajectory data from Singapore [18].

Finally, Pan et al. proposed several heuristics to plan or to choose from the first-k shortest paths (KSPs) based on previously assigned traffic [131]. The basic workflow of these approaches is similar to aforementioned two categories. However, the criteria used to compute or choose from KSPs in these approaches are not precisely computed PTT but several heuristic functions. For example, the EBkSP algorithm computes the KSPs according to real-time traffic and assigns the traffic to the least popular route among KSPs to balance the traffic volume distribution. The popularity heuristic is defined based on both current traffic conditions and previously routed traffic (i.e., ATV). Simulation result shows that the heuristic methods achieve substantial travel time reduction while using very limited computation resources.

In this chapter, we address the challenges of implementing these algorithms in real life. We present a participatory system, using up-to-present data collected from cars to determine the traffic conditions, based on which cooperative routing algorithms make decisions (i.e., real-time traffic and PTT or ATV). Compared with the taxi data evaluation in [18], our synthetic evaluation method investigates the performance of cooperative navigation at different penetration rates by meaningfully expanding the trajectory data. In addition, we also include real world evaluation results from field studies.

2.10 Summary

This chapter presents the design, implementation, and evaluation of Themis, a practical balanced traffic routing system that supports cooperative routing algorithms. To the best of our knowledge, Themis is the first stand-alone navigation system and phone app providing cooperative routing services. Themis collects traffic data from the participating drivers and accurately estimates the real-time traffic conditions and the anticipated traffic volume.

We build a Themis prototype, consisting of an Android app and a backend server, and carry out field studies to validate its feasibility. More importantly, a simulation evaluation method is presented to investigate the performance of Themis at different penetration rates based on real-life trajectory dataset. The city-scale simulation experiments using data from 26,000 taxis demonstrate that balanced routing can reduce the average travel time in the road network and alleviate congestion. The benefits of balanced routing over greedy routing emerge even at low penetration rates. In addition, the performance variations at different penetration rates consolidate the hypothesis that balanced routing provides more benefits when the penetration rates are higher.

Chapter 3

Parking Availability Crowdsourcing Based on Parking Decision Models

3.1 Introduction

Parking availability sensing is an important topic in the context of Smart City. According to a recent survey [154], cars seeking for parking on average spend 3.5 to 14 minutes on cruising before finding an available spot in downtown areas, which account for 30% of street traffic. Moreover, many cities are experimenting dynamic parking price policies, which are based on real-time parking availability [135, 197]. To date, most parking sensing systems are focused on detecting where and when a parking or unparking event happens using various sensors. Since the early 2000s, researchers have been using dedicated sensors (e.g. infrared sensor at each spot) [71, 139] or human reports [21] to detect the state of monitored spots. These solutions are reported to incur high installation or maintenance cost (hundreds of dollars per unit) and thus are only deployed in limited area [111]. More recently, sensors in smartphones were explored to identify the transition between driving and other mobility patterns to infer parking/unparking events of participant drivers [124, 170, 105].

However, most existing sensing systems are not designed to efficiently estimate the availability of background spots, i.e., those not covered by dedicated sensors or by the parking/unparking events of participant drivers. These background spots, due to low penetration rates of existing parking sensing systems, account for the vast majority of urban parking resources. Even worse, in many application scenarios, we need the fine-grained availability estimation at the spot level. For instance, when looking for a parking spot in a busy airport parking lot, a driver needs to know which spots are more likely to be available. Existing crowdsourcing approaches extrapolate statistics generated from sensed spots to the overall parking availability using the random sampling theory [139]. Since the sampling theory is primarily suitable for a large

population random process, the current method to estimate background parking availabilities is intrinsically inefficient for fine-grained background parking availability estimations.

To address this issue, we design ParkScan, a crowdsourcing system estimating the availability of the parking spots along drivers' parking search trajectories. The key idea of ParkScan is based on the empirical observation that passing by a parking spot that is of interest to the driver's destination implies that the spot is unavailable. For instance, if a driver cruises around her/his workplace for a long time and parks at a spot far away, we can infer that most spots, if not all, along the cruising path should be unavailable. Otherwise, the driver would take one of them. Given that people's travel destinations for daily commutes can be highly predictable along with the ubiquity of trajectory data, ParkScan leverages the participating drivers as human sensors to scan the spots along their parking search trajectories and then infer the spots that have possibly been taken. Since a driver's search trajectory usually covers many spots [154], ParkScan has a great potential to improve the parking sensing coverage with little cost incurred, compared to solutions based on dedicated sensors. Moreover, as we use the driver's search trajectory to directly estimate the parking availability along the trajectory, we can overcome the drawbacks of extrapolating participant parking events to global parking statistics using only sampling theory. As a result, ParkScan tolerates a much lower penetration rate, e.g. even with a single driver.

The key technical challenge we address in ParkScan is how to model drivers' general parking preferences in heterogeneous parking facilities and to merge the estimates from different parking search trips at different times to build the best estimation. In particular, the contributions of this chapter are as follows:

- To study drivers' parking decisions, we collect a large-scale dataset that consists of more than 8,000 vehicle trajectories in the parking lot and 55,000 parking decision events, i.e, either taking or ignoring an available spot during a parking search process. With the features extracted from this valuable dataset, we build a data-driven model that reflects driver's underlying decision strategy. To the best of our knowledge, this is the first dataset at scale that records real driver's parking decisions with rich features. Based on this dataset, our model is the first data-driven parking decision model generated from large-scale real-world parking decision observations. The sample dataset is shared with the

research community in [98].

- We design a crowdsourcing system called ParkScan to complement the state-of-the-art parking crowdsourcing systems particularly for fine-grained background parking availability estimations. ParkScan takes parking searching trajectories from participant vehicles as input. By combining with historical knowledge and static parking facility layouts, ParkScan leverages a learned parking decision model to generate probabilistic parking availability estimations at the parking spot level.

- We evaluate ParkScan in both off-street scenarios (i.e., two public parking lots) and a city-scale on-street parking scenario. In the parking lot experiments, we utilize over 6,000 human parking search trajectories and real-world parking availability to conduct the evaluation. In the city-scale street parking scenario, we conduct data-driven evaluations using over 63,000 parking requests generated from real parking transactions in an urban area covering 35 blocks in Seattle, WA. Both of the experiments showed that with a 5% penetration rate, ParkScan reduces at least 12.9% of availability estimation errors for all the spots in the experimental scenario during the peak parking hours, compared to a state-of-the-art solution based on historical data. More importantly, even with a single participant driver, ParkScan cuts off 15% of estimation errors for the parking spots along drivers' parking search trajectories.

The rest of this chapter is organized as follows: Section 3.2 provides the main idea of the ParkScan system. Section 3.3 presents the architecture of ParkScan and discusses the technical challenges. Section 3.4 elaborates on our parking search dataset and how we build our data-driven parking decision model. Section 3.5 discusses in detail how to use the pre-trained parking decision model to estimate spot availability and how to merge estimations of different qualities into one availability probability estimation. Sections 3.6 and 3.7 present the evaluation experiments and results for off-street parking and on-street parking scenarios. Finally, we discuss the limitations and some insights in Section 3.8 and related work in Section 3.9, followed by the conclusion of the chapter in Section 3.10.

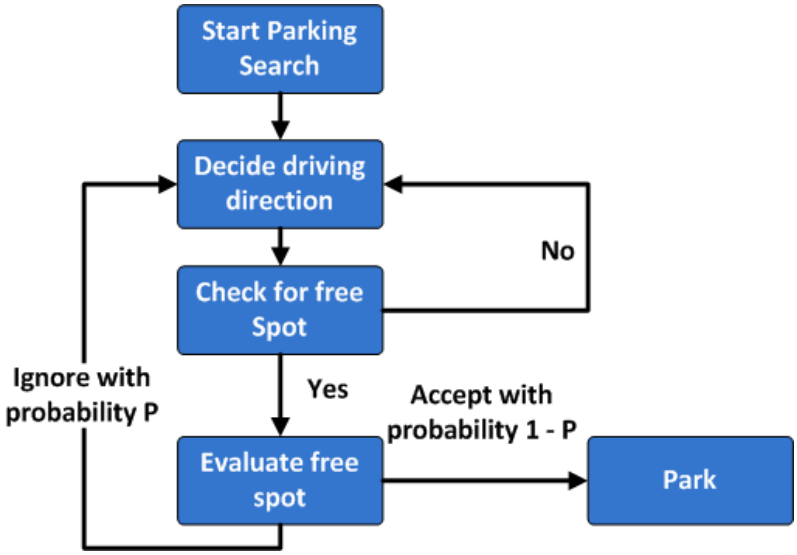

Figure 3.1: Parking search process

3.2 Motivation

To motivate the design of ParkScan, we analyze the human behavior during the parking search process, i.e., the process from a driver starts looking for parking until the driver parks the car.

As shown in Fig. 3.1, a typical parking search process can be modeled as a sequence of decisions. During a parking search trip π, a driver u needs to make routing decisions at each intersection. More frequently, u faces a decision moment when u decides whether to take an available spot i. Based on the state-of-the-art work [179, 157, 24, 110], this decision process is probabilistic and determined by the driver's **parking decision model** where with the probability $p_{u,\pi,i}\left(\neg park \,|\, empty\right)$, u chooses to ignore the spot i and continues to search for the next available spot; with the probability of $1 - p_{u,\pi,i}\left(\neg park \,|\, empty\right)$, u chooses to take the spot i for parking.

An example of this parking process is illustrated in Fig. 3.2. A driver who is looking for a parking spot in a parking lot visited several spots, i.e., from spot 1 to spot 4. But since these spots were already taken, the driver continues to search until finding an available spot, i.e., spot 5. The driver evaluated this spot 5 by implicitly computing a probability, $p_{u,\pi,i}\left(\neg park \,|\, empty\right)$ based on the distance to his/her trip destination, i.e., the building on the right side of the figure. In this case, the probability $p_{u,\pi,i}\left(\neg park \,|\, empty\right)$ for ignoring

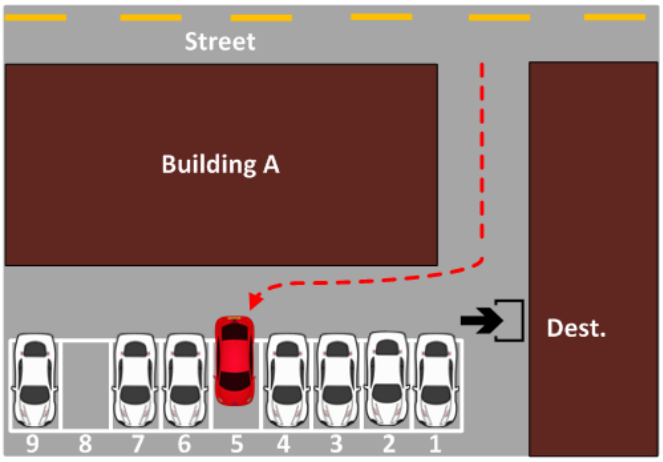

Figure 3.2: ParkScan use case

spot 5 is not quite high, so the driver selects spot 5 and parks there.

Given the driver's parking search trajectory in Fig. 3.2 (the dotted line) and the driver's final destination, intuitively, we can easily find that spot 1 to spot 4 are "better" than spot 5 for this driver. Since the driver has by-passed these spots and parked at a suboptimal spot, we can infer that spot 1 to spot 4 were taken when the driver visited them. During this intuitive process, this inference is actually a Bayesian modeling process defined by the following equation:

$$p_{u,\pi,i}\left(empty \,|\, \neg park\right) = \frac{p_{u,\pi,i}\left(\neg park \,|\, empty\right) \times p_i\left(empty\right)}{p_{u,\pi,i}\left(\neg park\right)}, \tag{3.1}$$

where $p_{u,\pi,i}\left(\neg park \,|\, empty\right)$ is the probability for a driver u to ignore spot i when i is available, $p_i\left(empty\right)$ is the probability that spot i is available, $p_{u,\pi,i}\left(\neg park\right)$ is the probability that u does not park in spot i for trip π. Note that, only $p_{u,\pi,i}\left(\neg park \,|\, empty\right)$ is estimated using an observer's parking decision model, so both $p_i\left(empty\right)$ and $p_{u,\pi,i}\left(\neg park\right)$ are prior knowledge that can be learned using historical data.

The above discussion essentially suggests that if we are given a parking search trajectory, it implies that all spots except for the last one along the trajectory are ignored by the driver. If we can estimate $p_{u,\pi,i}\left(\neg park \,|\, empty\right)$ by feeding necessary features into a machine learning model, the observation that the driver u does not choose to take spot i leads to an estimation of the posterior probability that spot i is available. Since this process combines the historical knowledge and real-time observations, it could potentially provide a better availability estimation for the background spots along a driver's parking search trajectory. For instance,

when a spot i is extremely attractive for driver u's trip π, $p_{u,\pi,i}\left(\neg park \mid empty\right)$ will approach zero. In this case, when the driver u ignores this attractive spot, we will accordingly have $p_{u,\pi,i}\left(empty \mid \neg park\right)$ to be very small, indicating that spot i is not likely to be available.

3.3 ParkScan Overview

Based on the discussion in Section. 3.2, we present an overview of the ParkScan crowdsourcing system.

3.3.1 Key Idea

When a participating driver starts a searching process for a parking spot, the key objective of ParkScan is to provide parking availability estimations of nearby parking spots to facilitate this search process, e.g. by influencing their search routing, with a frontend app and a backend cloud server. The key feature of ParkScan is its transparency to its background participating drivers whose searching processes contribute to parking availability estimations for drivers using ParkScan for parking. While a driver is looking for a parking spot near her/his final destination, our ParkScan frontend app will show locations of potential parking spots with different probabilities indicating parking availability estimations to the driver. Based on these availability estimations, the driver will select a route to find an available parking spot with a high probability. Then, the key question becomes how to obtain parking availability estimations.

In this chapter, we utilize a driver's search trajectory along with his/her final destination to infer and update parking availability estimations for the parking spots along the search trajectory. As follows, we introduce these three components, respectively. (i) Search Trajectories: While showing parking availability estimations to the driver, the ParkScan app simultaneously tracks his/her search trajectory. After a parking event is detected for this vehicle, e.g., when a transition from an in-vehicle mode to an on-foot mode is detected by Google activity recognition API [50], the search trajectory of this parking event near the final parking spot will be recorded by the ParkScan app. (ii) Final Destinations: Driver's final destinations are important for ParkScan to improve its performance. Some drivers may have their final destinations input in the navigation system, which can be utilized by our ParkScan app. In such case, when the

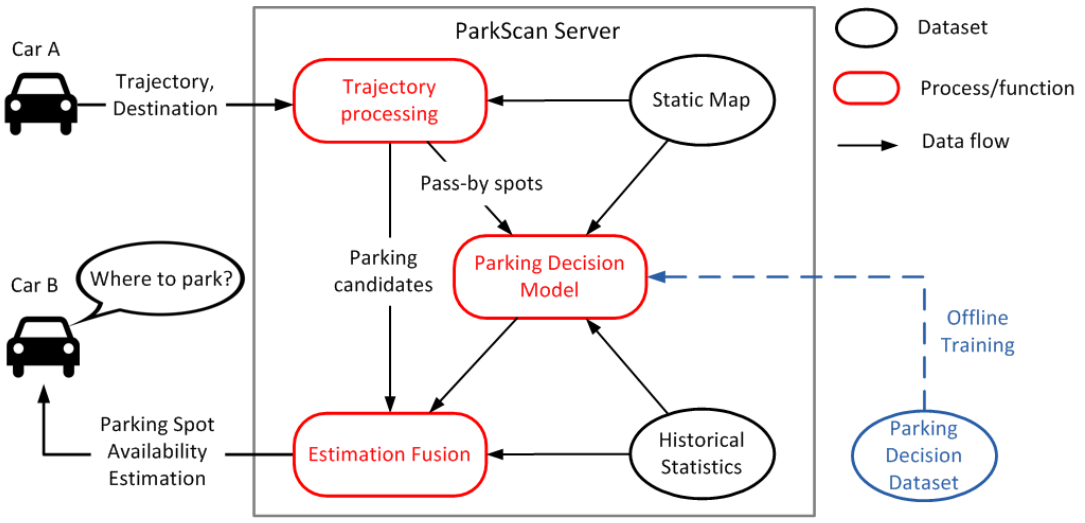

Figure 3.3: ParkScan system architecture

driver occupies a parking spot and starts walking to his/her final destination, the ParkScan app will upload the recorded trajectory and the final destination to a backend ParkScan cloud server. However, other drivers may drive without using a navigation system. In this case, our ParkScan system can estimate the drivers' final destinations by using locations of the first building that drivers entered [123]. (iii) Parking Availability Estimations: Upon receiving a search trajectory (ending with the parked position) together with a final destination, the ParkScan cloud server extracts a few features needed for a pre-built parking decision model. For each of the spots covered by a search trajectory, ParkScan estimates its posterior availability probability (details are in Section 3.2) to update parking estimations. Finally, these updated parking estimations are then utilized by subsequent drivers during their parking search processes.

In this chapter, we focus on the key idea of ParkScan, which is to utilize search trajectories and final destinations to estimate parking availabilities of parking spots because parking/unparking detections and trajectory tracking techniques have been extensively studied, e.g. [105, 123, 190, 155].

3.3.2 Architecture

Based on the previous discussion, we present the architecture of ParkScan in Fig. 3.3. The parking decision model is the key component used in the ParkScan server, which is trained

offline using a large-scale parking decision dataset. However, as this model describes the parking decision strategy of generic drivers, it does not have to be re-trained for each individual deployment. To justify this design choice, we present a default model and test it using different settings in Section 3.6 and Section 3.7.

For online parking availability estimations, ParkScan uses both real-time data (i.e., parking search trajectory and drivers destination estimated from the ParkScan app), and off-line data (i.e., historical parking statistics and stationary parking facility map) as input. When the real-time input data of a driver are received, ParkScan first processes the trajectory and generates two types of parking spots: (i) parking spot candidates (the spots that have been potentially taken by this driver based on the ending position of the search trajectory) and (ii) pass-by spots (the rest spots along the search trajectory).

Since locations reported in the trajectory, especially the ending point, are noisy, we rectify the trajectory and estimate the localization errors by using static map of the parking facility. The parking spot candidates define probabilistic observations to fill those spots. For the pass-by spots, we generate the features by using both a static parking layout and historical data, and then feed them into the parking decision model to compute the probabilistic posterior availability, i.e., using Equation 3.1. Since the estimates from a single parking candidate or pass-by spot observation are error-prone, we apply an estimation fusion process to combine the estimates from heterogeneous resources at different moments with the historical data to build the latest parking spot availability estimation. This process generates the final results for participant drivers and other potential applications, e.g. a routing service.

3.3.3 Challenges

In order to build ParkScan, there are two key challenges to address. This subsection elaborates on these challenges and our main idea to address them.

Modeling Parking Decision with Scenario-independent Features

The first challenge faced in ParkScan is building a generic model that predicts the likelihood that a driver would take spot i when it is available, i.e., $1 - p_{u,\pi,i}\left(\neg park \mid empty\right)$. As prior parking behavioral studies showed that people apply similar parking search strategies in various

environments [33, 64], we applied a data-driven approach to build the model from a dataset of extensive real parking trajectories and driver's parking decisions from surveillance videos of a public parking lot. More importantly, the proposed parking decision model uses only features independent of individual parking facilities and obtainable from search trajectories, static data resources, and historical data. In this way, the model is robust for different scenarios, and does not have to be re-trained for every new deployment. We will explain parking decision modeling in more detail in Section 3.4.

Generating and Fusing Heterogeneous Estimations

Assuming the parking decision model has been built, we need to extract the spot visits along the search trajectories and infer the parked spots according to the reported noisy parked position. For the spot visited along the search trajectory, we need to generate the features in realtime so that Equation 3.1 can be leveraged to get the spot availability estimates. Note that, the individual estimates from the parked position or pass-by spot along the search trajectory may be inaccurate due to the GPS noise or only unpredictable causes, e.g. the driver might just neglect an available spot. Therefore, after we get estimates from heterogeneous resources, i.e., parked position, pass-by visits from different vehicles at various times and historical data, we need to fuse them together to build the latest estimates when a driver requests for the estimation map. Moreover, all of these considerations have to work well when the penetration rate of ParkScan is low. To solve this problem, we discuss the realtime process in the ParkScan server and present a framework based on hidden Markov model for estimation fusion (details are shown in Section 3.5).

3.4 Model Parking Decision

In this section, we present our parking decision model based on large-scale surveillance video data we collected in a parking lot. Compared to GPS-based data, surveillance video data provide detailed information about trajectories of all vehicles and empirical parking decisions without installing GPS devices or smartphones in each individual vehicle. Note that surveillance video data are only used to train an offline model beforehand, and ParkScan does not

require surveillance video data to function in real time.

This section first starts by describing our method to obtain the parking decision dataset from surveillance videos of a public parking lot, then explains how we extract scenario-independent and accessible features from these videos, and finally builds our generic parking decision model.

3.4.1 Video Dataset Generation

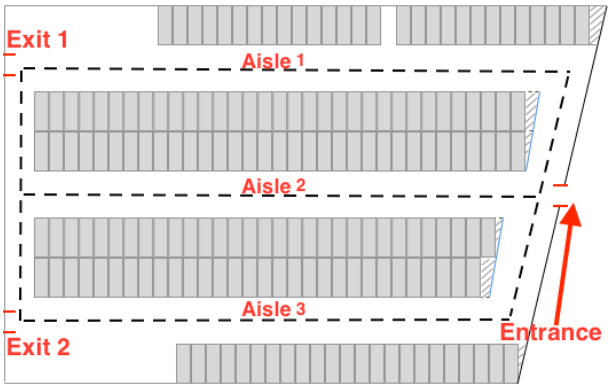

Figure 3.4: Layout of the data source lot

Our data source is a public parking lot on a university campus, where we placed a surveillance camera to video record the parking lot from an approximately 45-degree birds-eye view. We produced a parking decision dataset by extracting realistic parking events along with driver's final parking decisions from the video. The parking lot has 192 available spots including 12 spots for handicap parking, and drivers usually use this parking lot when they visit two buildings located at two exits shown in Fig. 3.4.

Vehicle Trajectory

We take three steps to process the raw video data to generate the parking trajectories as shown in Fig. 3.5. (i) Calibration: The calibration step builds a translation matrix from a video frame to the map of the physical world by using affine transformation. (ii) Mobile Blob Detection: Each video frame is fed into a Pixel Based Adaptive Segmenter [69] to detect moving objects and remove shadows using LR Textures based detection [144]. (iii) Blob Matching: Finally, we

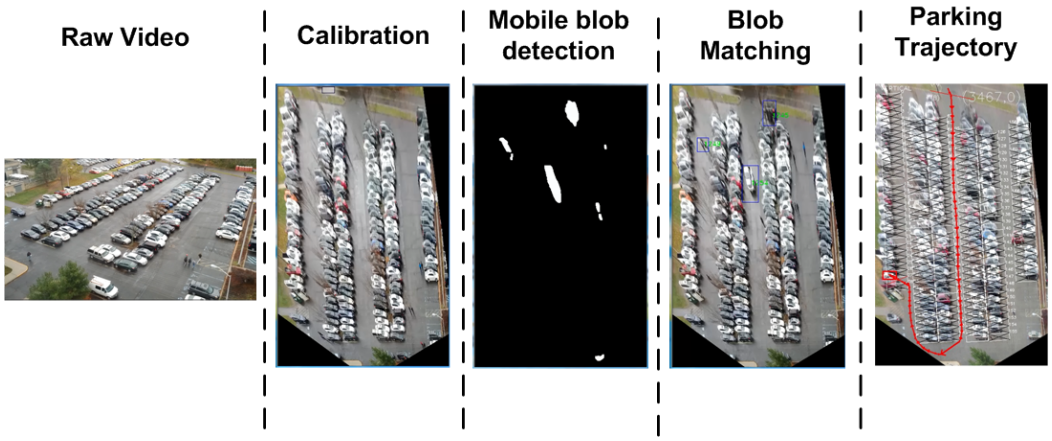

Figure 3.5: Video processing

track the moving blobs and differentiate vehicles from pedestrians using the model presented in [149]. After these steps, we recover the trajectory of a vehicle in the parking lot.

Spot State Ground-truth and Parking Decision

The position where the parking trajectory terminates defines the taken spot. Similarly, the position where the leaving trajectory starts defines the leaving spot. The accumulation of the taken spot and leaving spots generates the ground-truth of the parking lot availability state at any given time. Using the parking trajectory collected in Section 3.4.1, we set up a distance threshold to determine spot visibility of all drivers. If a visible spot is available, it defines a parking decision event for a timestamped location. The driver's choice for each parking decision event is generated such that only the last visit to its taken spot is a "take" decision and all the rest are labeled as "ignore".

Final Destination Extraction

Since the drivers in the experimental parking lot have only two final destination options (i.e., two buildings next to the exits), we build a binary classifier to infer their final destinations using turning directions, when they are searching for parking spots, e.g. the direction of a drivers first turn after entering the parking lot shows a very strong correlation to his/her final destination. Our cross-validation results show that this classifier predicts drivers' destinations with 92% accuracy. Therefore, we use the prediction result as the ground-truth destination for

each parking event.

Summary

From the video records in the workdays between Oct. 30, 2015 and Nov. 25, 2015, we extracted 8,181 vehicle trajectories, among which 5,988 trajectories are parking trajectories. In total, over 55,000 parking events are generated, with 5,754 decisions labeled as "take" and the rest labeled as "ignore". Fig. 3.6 depicts the cumulative distribution function (CDF) of the parking availability based on the collected spot availability ground truth. Fig. 3.7 illustrates the CDF of the number of visited spots per parking trajectory and that of visited available spots per trajectory. Fig. 3.6 highlighted the needs for estimating spot-level parking availability, because during some time, e.g. 8:00 or 10: 00, the CDF grows smoothly, meaning there is much variance among different spots' availability rates. Based on Fig. 3.7, the mean of the number of all visited spots and that of visited available spots are 47.3 and 14.7, respectively. The large mean of the number of all visited spots implies that the posterior parking availability estimation has a higher potential to obtain high coverage. The high mean of the number of all visited available spots, on the other hand, suggests the requirement of an efficient parking decision model, because we need to filter out the influence of many spots that are ignored by drivers but in fact available.

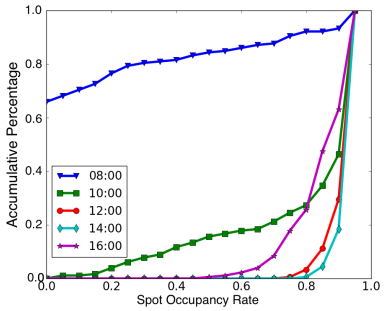

Figure 3.6: CDF of parking spot occupancy rate at different times of day

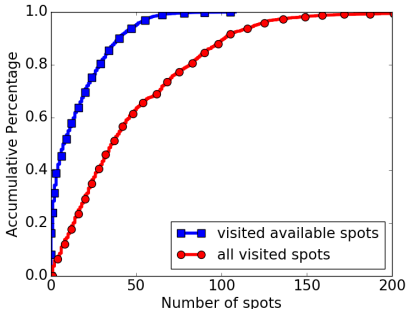

Figure 3.7: CDF of the number of visited (available) spots per trajectory

3.4.2 Feature Engineering

After the training dataset is obtained, including historical parking availability statistics, parking trajectories, final destinations of drivers, and parking decisions, we use this information to define the features that can be used in a parking decision model and accessible in real time for ParkScan.

Spot Related Features

Since a parking decision model essentially reflects human's evaluation on the properness of a spot for the trip destination, we define a set of features to evaluate a particular spot.

Percent of better spots (PBS): This feature depicts the rank of the current spot s_c among all spots S in the parking lot, according to their total travel cost $cost_{total}$, i.e., the driving time to a spot plus the walking time from that spot to the driver's final destination. We defined a "superior" spot set S_s for the current spot s_c, which includes all the spots whose total travel time is shorter than s_c. As a result, the PBS for a current spot s_c is defined as $|S_s(s_c)|/|S|$, where $|.|$ denotes the size of a set.

Expected cost saving (ECS): ECS is designed to complement PBS to describe how much cost the spots in $S_s(s_c)$ is expected to save for a driver including both the magnitude of the saving for each spot and the available probability for that spot. ECS is defined as:

$$ECS(s_c) = \frac{\sum_{s \in S_s(s_c)} p(s) * (cost_{total}(s_c) - cost_{total}(s))}{|S_s(s_c)|}, \tag{3.2}$$

where $p(s)$ is the historical probability estimation that a spot s is available at present. ECS is designed for the people who would optimistically seek to minimize their total travel costs.

Expected cost overhead (ECO): In contrast to ECS, we compute the ECO for those who value the risk of missing the current spot as:

$$ECO(s_c) = \frac{\sum_{s \in S - S_s(s_c)} p(s) * (cost_{total}(s) - cost_{total}(s_c))}{|S - S_s(s_c)|}. \tag{3.3}$$

Conceptually, the larger the ECO value, the more likely a driver would choose to park at the current spot.

Historical availability (HA): It is well believed that the parking availability influences people's parking decision [25, 36, 164, 64, 83]. For instance, when the parking condition is critical,

people tend to take a spot even if the spot is not that "good". As we do not have access to people's real visibility, we use HA for a spot i at the same time of day as the decision moment as one feature dimension for the parking decision event for that spot i.

Added walking distance (WD): Intuitively, the further a spot is from the driver's final destination, the less likely a driver would park at that spot. Since we essentially need to choose a spot from a consideration set, e.g. a parking lot, we normalize WD by subtracting the minimum walking distance of any parking spot in the consideration set from the actual walking distance of the current spot.

Driving direction (DD): This is represented as the angle between the current driving direction and the straight line segment connecting driver's current position and the final destination. Ideally, when a driver is heading to the final destination, she/he should have a lower probability to take a spot compared to when she/he is moving further from the destination.

Trip Related Features

In addition to the spot related features, there are some trip related features that also influence driver's parking decisions.

Parking search time (PST): This feature is calculated by subtracting the search start time from the current time. Ideally, the larger the value, the more likely a driver should choose to park, once he/she finds an available spot.

Number of visited spot (NVS): This is the total number of visited spots during a driver's parking search trajectory. This feature overlaps with parking search time since a longer search time generally means more search spots. However, we found within a same search duration, exploring more spots reflects a driver's risk-taking trend, therefore it might predict a lower probability if he/she finds an available spot.

Summary

All spot related features defined above can be built, given historical availabilities, the parking facility layout, the driver's destination, the current spot's position and the driving direction (binary along with the street or aisle in a parking lot). Trip-related features can also be easily

obtained in a parking lot scenario because a driver's parking search trip starts when the vehicle enters the parking lot. The determination of the starting point of an on-street parking searching trajectory will be discussed in Section 3.5. Note that, we articulately normalized most features for individual parking facilities instead of using absolute costs, so that the model is generic for different parking facilities without having the model re-trained for every deployment.

We ran the logistic regression and based on the absolute value of features' coefficients, we get the rank list of importance for the features: HA $>$ WD $>$ ECO $>$ PST $>$ NVS $>$ DD $>$ PBS $>$ ECS. More specifically, the first three features in the rank list have dominated weights than the rest. Since HA, WD, and ECO are fully defined by the driver's final destination, the parking facility historical availability, and the parking facility layout, we expect that the driver's parking decision can be highly predictable, which eventually leads to good parking availability estimations using equation 3.1.

3.4.3 Model Training

To guarantee the quality of the training data, we filtered the events whose features may not be calculated incorrectly, e.g. some trajectories are detected to have started from a non-entrance position. With the features generated in Section 3.4.2, we trained a boosting tree model over the data after filtering, because it is more robust to heterogeneous features. We tested the performance of our model over all experimental time and the interval only includes busy hours, and then we compare it with a model that uses more features, which are not accessible by ParkScan in real time but are believed to improve the accuracy of parking decision models. For example, they include the availability of adjacent parking spots, the actual spot states along driver's search trajectory, and comparison between the current spot and the best spot the driver has seen so far and some properties associated with the spot, e.g., obstacles or shadows.

The results of the 10-fold cross-validation results are shown in Table 3.1. The model used in ParkScan to estimate posterior parking availabilities is called the estimation model. The model with additional features is labeled as the simulation model, because this model simulates driver's actual decisions by including the features that only onboard drivers can see but ParkScan cannot access.

From Table 3.1, we obtain three key findings as follows. (i) The estimation model does

Table 3.1: Model training with different feature sets

	All Time (42202 decision events)		Busy Hours (3953 decision events)	
Model	Precision	Recall	Precision	Recall
Estimation Model	0.734	0.852	0.937	0.856
Simulation Model	0.738	0.854	0.944	0.869

Note: Busy Hours is defined from 10:30 am to 3:30 pm for cross-validation. Precision is defined as the number of correctly predicted "take" decisions / the number of predicted "take" decisions. Recall is defined as the number of correctly predicted "take" decisions / the number of all ground truth "take" decisions.

not show a largely degraded performance, even though the model only includes the features generated using historical data, map data, and driver's search trajectories and destination. This observation essentially suggests that estimating parking availabilities based on the estimation model is feasible. (ii) The parking decision model shows better performance during busy hours, which implies that ParkScan should have better performance in busy parking time, i.e., exactly when drivers need parking availability estimation the most. (iii) Either in busy hours or free hours, there are about 15% of driver's parking decisions that cannot be predicted (suggested by the recall). While these false negative parking decisions might come from the potential errors contained in our dataset, it motivates us to combine the estimates from different parking search trips so that we can mitigate the errors from a single observation.

3.5 Real-Time Parking Availability Estimation

Based on the parking decision model constructed from human parking decision observations, this section discusses how to use noisy parking search trajectories to generate probabilistic parking availability estimation and how to fuse estimations from heterogeneous sources to mitigate potential errors in the estimations from individual search trajectories.

To simplify the discussion, for a parking lot scenario, we envision the start of a driver's search trajectory (used in Section 3.5.1) is the entrance of the lot and a driver's consideration set includes (needed in Section 3.5.2) all spots of the parking lot. Then, we present the estimation fusion framework in Section 3.5.3. Finally, we discuss how to extend our parking lot scenario to a street parking scenario in Section 3.5.4, i.e., the choice of the start of the search trajectory and all candidate spots for street parking.

3.5.1 Trajectory Preprocessing

After receiving the parking search trajectory, ParkScan applies the map matching algorithm [127] to translate the trajectory to a list of points along road segments or aisles in the parking lot and the paths connecting the points. Since parking spots are along the streets or aisles, all spots on the generated path are considered visited by the given search trajectory. Note that, even though each individual point in the search trajectory may not be accurate, the path and visited spots generated using map matching are accurate enough for ParkScan, except for a handful number of spots at the end of the search trajectory. In ParkScan, we need to identify these spots because they are the candidates where the car actually parked (called parking candidates). In the current ParkScan implementation, we set up a round area centered at the trajectory ending position and consider all spots within that area parking candidates.

3.5.2 Estimating Posterior Availability for Visited Spots

For each of the pass-by spots (i.e., all the visited spots except for the parking candidates), the parking search trajectory defines a decision event when the car passes by the spot and the human's decision is to ignore the spot.

Therefore, ParkScan computes the features in the way used in Section 3.4.2 and then leverages the learned parking decision model to compute the likelihood of $p_{u,\pi,i}\left(\neg park \,|\, empty\right)$. However, the difficulty of using Equation 3.1 is to estimate $p_{u,\pi,i}\left(\neg park\right)$, the probability that a driver u does not park in the spot i for the trip π. This probability needs to be estimated per driver per timestamped destination, requiring extensive fine-grained historical data. We decompose $p_{u,\pi,i}\left(\neg park\right)$ using the total probability theorem:

$$p_{u,\pi,i}\left(\neg park\right) = p_{u,\pi,i}\left(\neg park \,|\, empty\right) \times p_i\left(empty\right) + p_{u,\pi,i}\left(\neg park \,|\, full\right) \times p_i\left(full\right). \tag{3.4}$$

Since $p_{u,\pi,i}\left(\neg park \,|\, full\right)$ is 1 for everybody (i.e., nobody can park at a spot that is taken), by substituting Equation 3.4 into Equation 3.1, we obtain the following formula:

$$p_{u,\pi,i}\left(empty \,|\, \neg park\right) \;=\; \frac{p_{u,\pi,i}\left(\neg park \,|\, empty\right) \times p_i(empty)}{p_{u,\pi,i}\left(\neg park \,|\, empty\right) \times p_i(empty) + p_i(full)} \tag{3.5}$$

where $p_i\left(empty\right)$ and $p_i\left(full\right)$ are driver/trip invariant and can be estimated easily from the

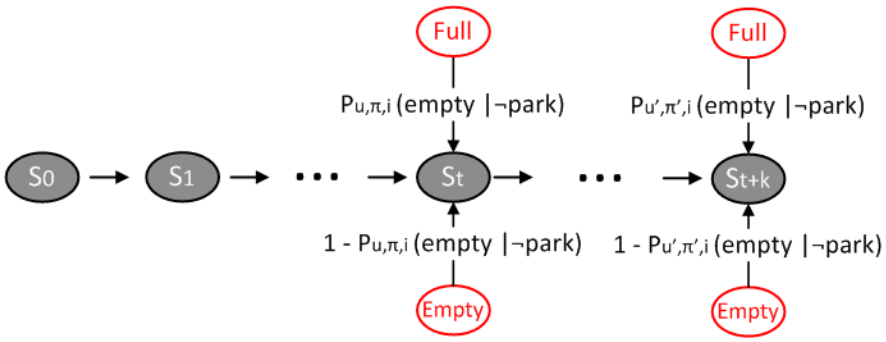

Figure 3.8: Hidden Markov process for probabilistic estimation updates

historical availability data. Once the likelihood $p_{u,\pi,i}\,(park \mid empty)$ is generated by the parking decision model, based on the equation 3.5 we can compute the posterior availability probability $p_{u,\pi,i}\,(empty \mid \neg park)$ for a visited spot that is not taken.

The parking candidates, instead of reflecting the state when the car passes by, define possible transitions from empty to full state, since there is a new car parked in that area. We use the prior knowledge that GPS errors are in general Gaussian noise [127]. Therefore, we distribute the transition probability 1 (for one newly parked car) to all parking candidates proportional to $exp(-d^2/\sigma^2)$, where d is the distance from the spot to the reported parked position, σ can be estimated using the method provided in [127].

3.5.3 Estimation Fusion

As discussed previously, the parking search trajectories provide the transition probability and availability estimates at different times of day, which individually may contain considerable noises. In this section, we fuse estimations for the same spot together to obtain more robust estimations.

We regard the state estimation for a spot i as a hidden Markov process (HMM) shown in Fig. 3.8. We partition the time into a sequence of intervals, within which the state of the spot will remain unchanged, e.g. 30 seconds. This is because parking a car or leaving from a spot usually takes from seconds to minutes. We define the true state of the spot in the interval t as the hidden variable S_t, which is either full or empty. At some intervals, e.g., interval t and interval $t + k$ in Fig. 3.8, parking search trajectories from participant drivers may cover this spot and generate two state probability estimates, $p_{u,\pi,i}\,(empty \mid \neg park)$ and $p_{u',\pi',i}\,(empty \mid \neg park)$.

Due to parking or vacating events of both participant and non-participant drivers, the true state may change with some transition probability, e.g. change from empty to full by a parking event. Since the transition probability can be achieved from driver's parking candidates or historical parking/vacating rates, the HMM process targets to find the best combination of states, whose accumulative probability has the highest value among all possible accumulation paths including transition probability and state probability. In our case, we only focus on the latest state of a spot, i.e., full or empty, so we normalize the full and empty probabilities at the last step, according to the sum of all probability accumulation paths ending with the current full or empty states.

Finally, the remaining problem is to estimate the state transition probabilities between two consecutive intervals. From the historical data, we can obtain the historical transition probability, e.g. parking probability can be computed by dividing the total number of parking events by the total length of an interval. Since the interval time is short, the historical transition probability will be very small. Therefore, when there is a transition probability generated by a parking spot in an interval, this transition probability will be used to update the historical value and dominate the transition probability during that interval.

Note that, the estimation fusion process, i.e., HMM process, does not necessarily end at an interval when a pass-by event or transition observation from a participant driver is detected. In most of the time, HMM changes the state estimation of a spot using the historical transition probability, which in fact fades the previously estimated state and gradually makes it converge to the historical availability rate. In this sense, the ParkScan estimation fusion algorithm naturally solves the spot availability dynamism under a limited system penetration rate.

3.5.4 Street Parking Extension

The discussion in the previous section is based on two assumptions: (i) the start of the searching trajectory is known; (ii) the consideration set, i.e., candidate spots that a driver would like to consider for parking, are also known. These two assumptions are easy to validate for a parking lot scenario: the entrance is the start and all spots in this parking lot are included in the consideration set. However, it is not straightforward to obtain such information for a street parking scenario.

In this work, ParkScan utilizes the intuition that when a driver approaching its destination from distance, the likelihood $p_{u,\pi,i}(\neg park \mid empty)$ gradually decreases, because closer spots to the final destination provide a shorter walking time. Based on this intuition, the start of the search trajectory can be inferred using the first point along the driving path where $p_{u,\pi,i}(\neg \hat{park} \mid empty)$ goes above a pre-defined threshold. Note that the start of the search trajectory does not need to be very accurate, because $p_{u,\pi,i}(\neg \hat{park} \mid empty)$ for those spots are usually approaching 1, providing no more information than the historical data. Moreover, the features closely related to the start of the search trajectory, e.g. parking searching time, are not the most impactful ones in the parking decision model according to the discussion in Section 3.4.2. For the consideration set, if we have enough historical data, we find the maximum distance from the spots ever taken for the same destination and then use that as a radius to consider candidate spots; if not, we use the distance from the start of the trajectory estimated above to infer the candidate spots.

3.6 Parking Lot Evaluation

In this section, we test ParkScan with the data collected from two real-world parking lots to investigate its performance in the parking lot scenario.

3.6.1 Experiment Methodology

We conducted the parking lot evaluation in two different parking lots. The first parking lot (called parking lot 1 hereafter) is the one mentioned in Section 3.4.1. Since our parking decision model is learned from parking lot 1, we also include a second parking lot (called parking lot 2) to validate the applicability of ParkScan. As shown in Fig. 3.9, parking lot 2 has 40 parking spots (including two handicapped spots). People come from both exits of the parking lot and go to a grocery store. Since people visit the grocery store at will and they spend on average 15 mins for each visit, the occupancy rate of parking lot 2 is generated by continuous parking and leaving events.

We use the parking search trajectory and ground truth availability collected in Section 3.4.1 as the request set for the parking lot 1. Similarly, for the parking lot 2, we also video recorded

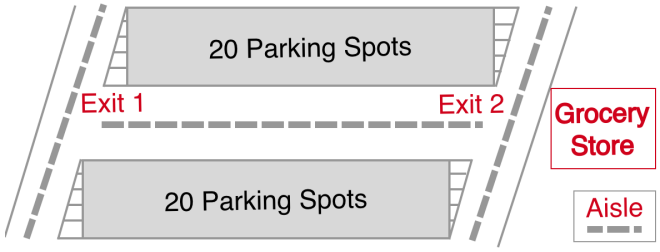

Figure 3.9: Layout of parking lot 2

the real parking trajectories and parking lot availability ground truth in 3 weekdays during the busy hours of the grocery store (from 3:15 pm to 5:15 pm).

For experiments in both of the parking lots, we use the parking decision model learned from parking lot 1 in ParkScan. Since parking lot 2 has a different layout and a different visit pattern, the results for parking lot 2 validate the generalization of the parking decision model learned from parking lot 1. For the experiments on parking lot 1, we use a 10-fold cross-validation (based on a chronological partition by dates) to avoid the over-fitting problem for the parking decision model.

3.6.2 Baseline Solution and Evaluation Metrics

Since no prior study has looked into the spot-level background parking availability estimations (details are given in the related work section), we use the historical availability rate at each spot as the baseline solution for the comparison. We measure the deviation of the probability estimations from the ground truth availability as the estimation error, defined as:

$$error_i = \begin{cases} 1 - p_i(\hat{empty}), & if\ spot\ i\ is\ empty \\ 1 - p_i(\hat{full}), & otherwise. \end{cases} \tag{3.6}$$

where $error_i$ represents the error of spot i; $p_i(\hat{empty})$ and $p_i(\hat{full})$ are the estimated probabilities for the spot i to be empty or full, respectively.

Estimation errors in equation 3.6 demonstrate the accuracy for a particular estimation. To evaluate the estimation performance for a group of estimations, we defined the mean absolute

deviation (MAD) as follows:

$$MAD = \frac{1}{N}\sum_{k=1}^{N} |\, error_k \,|, \tag{3.7}$$

where $error_k$ stands for the error of the k-th estimation among all N estimations.

Finally, we defined another metric, i.e., relative error reduction (RER), to measure the accuracy improvements produced by ParkScan and comparing it to the baseline method:

$$RER = \frac{error_{baseline} - error_{parkscan}}{error_{baseline}}, \tag{3.8}$$

where $error_{baseline}$ and $error_{parkscan}$ denote the errors for the baseline estimations and errors for the estimation using ParkScan. The RER for MAD can also be defined similarly. Note that, a positive RER value illustrates that the ParkScan estimation accuracy outperforms the baseline estimation.

3.6.3 Evaluation Results

In this section, we present the results for (i) the spots visited by a single parking search trajectory (called one-pass estimation), and (ii) all the spots within the parking lot considering multiple visits from different drivers at different times (called estimation fusion experiment).

One-pass Estimation

In this evaluation, we apply the method presented in Section 3.5.1 and Section 3.5.2 to estimate the probabilistic availabilities for the spots that are along any search trajectory. We compute the MAD for all estimations and then calculate the RERs for the MAD using Equation 3.8.

Fig. 3.10 and Fig. 3.11 show how the RERs change in two experimental parking lots while we generate estimations for the only those visited spots whose $p_{u,\pi,i}\,(park \mid empty)$ are bigger than the given likelihood threshold. Moreover, we also added the ratio of visited spots which produce estimations out of the total number of spots visited. The results show that even considering the estimation using only a single pass-by visit from one driver, ParkScan provided over 15% estimation accuracy improvements in parking lot 1 and 40% in parking lot 2 compared to the historical estimations. Moreover, as we raise the $p_{u,\pi,i}\,(park \mid empty)$ threshold, the estimation accuracy continuously rises. In particular, one-pass estimation could reduce the MAD

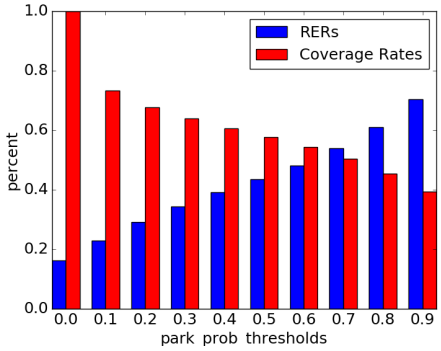

Figure 3.10: One-pass estimation results in parking lot 1

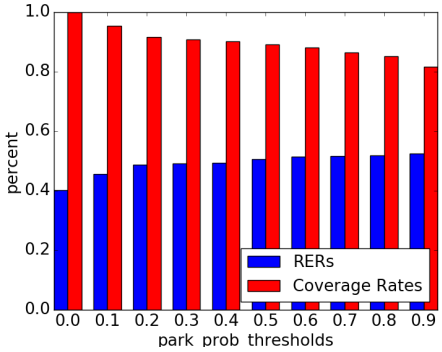

Figure 3.11: One-pass estimation results in parking lot 2

by over 70% for the spots whose $p_{u,\pi,i}\,(park \mid empty)$ are larger than 0.9 in parking lot 1 and over 50% in parking lot 2 for the same likelihood threshold.

By comparing Fig. 3.10 and Fig. 3.11, we found that different layouts and visit patterns in the two parking lots change the coverage rates and RERs defined by the same likelihood threshold. For instance, since the walking distance feature for the spots in parking lot 2 is smaller than that in parking lot 1, the results for parking lot 2 show larger $p_{u,\pi,i}\,(park \mid empty)$ likelihood value in general. However, if we consider all spots visited by the parking search trajectory (left-most bar in either figure), parking lot 2 actually shows improved RER, which suggested that ParkScan system has a good generalization property even when using the default parking decision model.

Estimation Fusion

In this evaluation, we apply an estimation fusion algorithm presented in Section 3.5.3 to provide the parking availability estimations for any spot each minute. Since the number of participant drivers influences the spatiotemporal coverage of spot visits along detected search trajectories, we also test different penetration rates to show the scalability of the ParkScan system. For the performance measurement, we calculate the MAD for all spots in each minute if there is at least one on-going parking search process. The MADs from ParkScan are also compared to the historical estimation MADs to present the RERs.

RERs of MAD for different penetration rates. Fig. 3.12 demonstrated the mean RERs

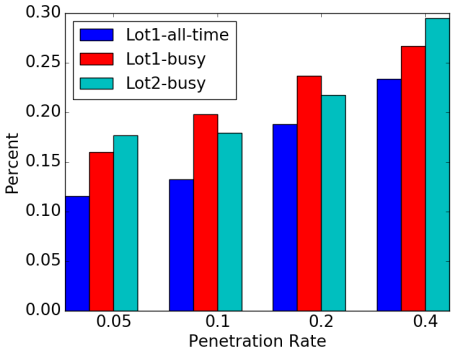

Figure 3.12: Mean RERs for various penetration rates under different experiment settings

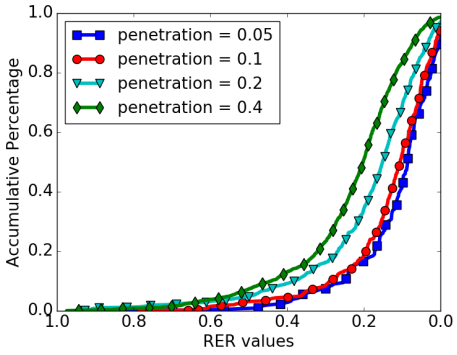

Figure 3.13: CDF of RER for various penetration rates in lot 1 (all time)

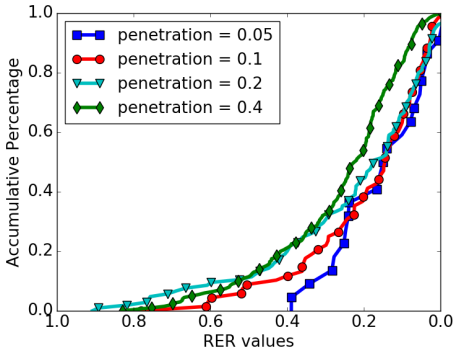

Figure 3.14: CDF of RER for various penetration rates in lot 1 (busy time)

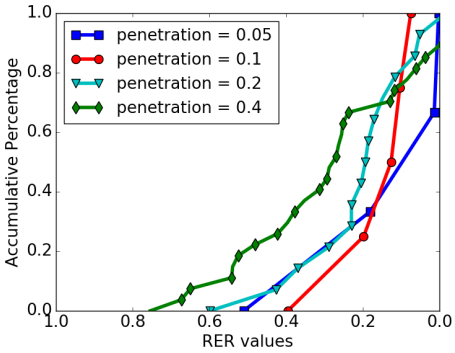

Figure 3.15: CDF of RER for various penetration rates in lot 2 (busy time)

under different system penetration rates for different parking lots or time intervals, i.e., all time or only busy hours. One can find from the figure that ParkScan reduces over 11% of MAD at a 5% penetration rate, and the performance gains grow when the penetration rate increases. By comparing the all time results (i.e., the first bar in each cluster) and the busy time results (the second bar in each cluster) in parking lot 1, we find that the parking availability estimation performance of ParkScan is obviously better during busy parking hours. This is caused by two reasons: (i) the searching trajectory is usually long during busy hours, which increases the coverage of the visited spots along the search trajectories; (ii) a drivers parking decision during the busy parking hours is more predictable, shown by the high prediction model accuracy presented in Section 3.4.3. These two reasons bring about the better performance for ParkScan during busy hours, exactly when people need the system the most. Finally, by comparing the RERs in parking lot 1 and those in parking lot 2, we confirmed that the default parking decision model works well in the new parking lot setting (i.e., parking lot 2), leading to even better RERs during the busy hours.

To understand the distribution of performance improvement among the measurements at different times of day (e.g. whether it is influenced by only heavier striker), we present the CDF of RERs in different settings in Fig. 3.13 to Fig. 3.15. All three figures demonstrate that the RERs are normally distributed. Moreover, the CDF of RER curves are generally pushed to the up-left side when the penetration rate increases, implying that higher penetration rates raise both estimation accuracy and coverage.

Comparison of MAD by using only one-pass estimation and estimation fusion. The RER values in the estimation fusion experiment are numerically less impressive than those from the one-pass estimation, e.g. by comparing Fig. 3.12 to Fig. 3.10. However, this is because in the estimation fusion experiment, we consider all parking spots in each one-minute interval instead of evaluating only the covered spots at the visited time. To show the benefits of estimation fusion algorithms presented in Section 3.5.3, we computed the MAD of the estimation fusion results and that using the one-pass estimation from the latest spot visit to calculated the RER between these two MADs. Fig. 3.16 demonstrated positive RERs of estimation fusion results with respect to the one-pass estimation at all penetration rates, meaning that our estimation fusion algorithm generates better spot availability estimations. In particular, it could reduce the

MAD of one-pass estimation by up to 55% in the parking lot 1 experiments. Since we only have 3 days historical data in parking lot 2, the statistics, especially the fill/vacating rate statistics, are not reliable. As a result, parking lot 2 shows less improvement in Fig. 3.16. However, the parking lot 2 experiments to some extent demonstrated the robustness of our estimation fusion method over the noise of historical statistics, because the estimation fusion results still show improvements compared to the one-pass estimation (positive RERs) for parking lot 2.

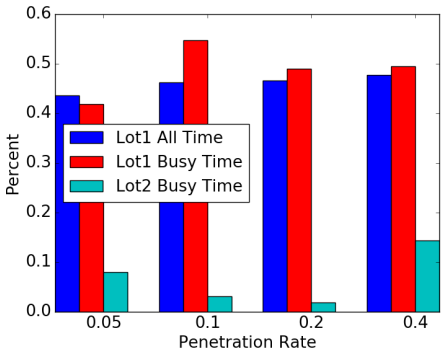

Figure 3.16: RERs of estimation fusion with respect to the one-pass estimation for various penetration rates in different settings

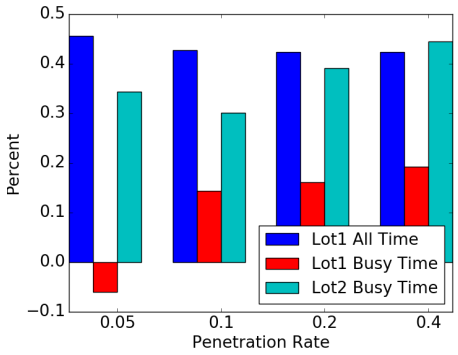

Figure 3.17: RERs of ParkScan estimation with respect to the coarse-grained solution for various penetration rates in different settings

Comparison of MAD by using ParkScan and the coarse-grained solution. Though the spot-level historical availability rate is used as the baseline to conduct most comparisons, we finally compute the MAD of ParkScan and that generated using PhonePark [170] to compare ParkScan with prior coarse-grained solutions. Note that, PhonePark also combines the detected realtime parking events and historical statistics as ParkScan does but uses Kalman filter to estimate the parking availability rate of a street parking section or a parking lot. Moreover, PhonePark additionally assumes that the penetration rate of the participant drivers is known in the targeted parking section. Since PhonePark cannot differentiate the spot-level parking availability within the targeted section, for the parking lot scenarios, we use the parking availability of the whole lot as that of each spot to compute the MAD. Fig. 3.17 demonstrates that in most cases ParkScan provides better spot-level availability estimation than PhonePark (by positive RER values). The only exception happens in parking lot 1 when the penetration rate is 0.05 during the busy time and it is caused by two reasons: (i) We provide PhonePark with

the ground-truth penetration rate (not used in ParkScan), which enables PhonePark to very accurately estimate the overall parking availability rate of the parking lot. (ii) The vast majority spots in parking lot 1 are usually taken during the busy time, meaning that the spot-level parking availabilities are all close to the overall availability of the parking lot. However, ParkScan outperforms PhonePark by up to over 40% when the penetration rate increases, or when looking at parking lots with different availability distribution pattern (e.g., parking lot 2 has fluctuant availability since people frequently come to and leave the store).

Evaluation Summary

During the parking lot evaluation, we collected driver's real-world parking search trajectories and tested ParkScan in two different parking facilities under different penetration and time interval settings. The key findings from the parking lot evaluation are as follows: (i) ParkScan reduces the estimation errors, compared to historical estimations for visited spots by a single parking search trajectory (shown in Fig. 3.10 and Fig. 3.11). This implies that ParkScan has the ability to work at an extremely low penetration rate, e.g. as low as only a single participant driver. (ii) ParkScan effectively reduces the estimation errors for all spots at the moment when drivers need the estimation map (shown by Fig. 3.12 to Fig. 3.17). (iii) During the experiment, we learned the parking decision model from parking lot 1 and applied it to a different parking lot (i.e., parking lot 2) with a different layout and different visit patterns. The good performance of ParkScan in parking lot 2 suggested that the parking decision model has good generalizability and does not need to be retrained for every new deployment.

3.7 Street Parking Evaluation

As mentioned in Section 3.5.4, we treat on-street parking as a special case, where each individual driver has her/his own start of a search trajectory and a consideration set of all possible spots. In this section, we investigate the performance of ParkScan in the street parking scenario.

3.7.1 Methodology

We leverage the on-street parking facility map and the parking transaction dataset [74] from Seattle, WA in the street parking evaluation. The parking facility map has the information of all the parking payment stations, including the station ID, the type description, the GPS coordinates, the effective time and so on. Based on the station coordinates, we associate the parking station with a road segment using OpenStreetMap and the nearby parking spots based on data collected from Google Street View.

Figure 3.18: Hottest parking area of Seattle

For the evaluation setup, we first select the experiment area. After visualizing the parking transactions (shown in Fig. 3.18), we find that Belltown area is the hottest parking region in the city. Since a parking crowd-sensing system is most needed in a busy parking area, we choose the region that is bordered by 5th Ave, Vine St., Elliott Ave., and Stewart St. as our experiment area.

We filter the city-wide parking transaction data to only preserve the data from the experiment area (its statistics is shown in Table 3.2 and its temporal distribution of transactions is shown in Fig. 3.19). Note that, only paid on-street parking spaces are available for public use in the experimental area, so the parking transactions cover the vast majority of all parking requests in this area.

Table 3.2: Seattle dataset description

Pay Stations	121 stations (road segments)
Parking Transactions	63960 parking requests
Parking spots	1276 spots
Time period	weekdays from 2015-03-02 to 2015-03-27

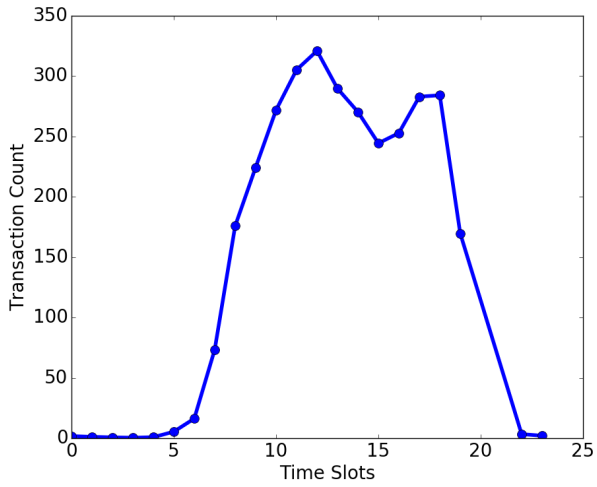

Figure 3.19: Average parking transaction count in a day

Total Parking Requests and Parking Availability

The experiment needs the parking requests, i.e. the time-destination pairs, to initiate parking search trips. Since parking transaction is associated with a payment station, in this experiment, we assume the driver's destination is a random point along the street where the payment station is located. Moreover, considering potential parking violations, we use the model between the paid parking time and the total parking time presented in [150], to estimate the ratio of paid parking requests among all parking requests on each road segment. We use the estimated ratio to super-sample the parking request set generated by the parking transactions, and obtain the extrapolated parking requests representing all parking demands. Once the total parking request set are identified, we also get the ground-truth of the parking availability at any time of a day (statistics are shown in Fig. 3.20), because it is essentially a parking/leaving event that causes the parking availability to change.

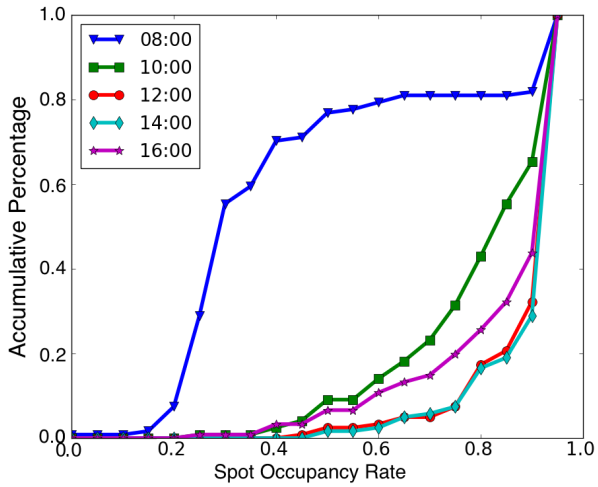

Figure 3.20: Occupancy rate distribution

Parking Search Simulator

Since there is no parking search trajectory in the transaction dataset, we rely on a simulator to generate the trajectory and the final parked spot. Given a final destination and a starting point far enough from the destination, the parking search simulator assumes that the driver is rational and always turns to the direction that minimizes the expected total travel time (driving time plus walking time) based on the historical availability. When an available spot is found, the simulator utilizes our parking decision simulation model, which takes extensive features than the estimation model (refer to Section 3.4.3 for the simulation/estimation model), to decide whether that spot should be taken. This process iterates until a spot is taken eventually. Note that a distance threshold is defined as a parameter for the parking simulator to determine the consideration candidate spots. This parameter also defines the start of the search trajectory, which is the first point located within the distance threshold when the vehicle approaches its destination. We tuned this parameter in Section 3.7.2.

Experiment Process

After the experiment starts, the parking requests are picked one by one from the total parking request set to simulate the parking search process, which includes the parking search trajectory and the final parked spot using the parking search simulator (Section 3.7.1). The state of a

taken spot will be updated in the ground truth availability map so that subsequent drivers will not park there until the parked car's paid parking time expires.

During the experiment, the ParkScan system obtains only the whole movement trajectory and the final destination. ParkScan infers the start point of the search trajectory and the driver's consideration set and then estimates the spot-level parking availability. Since the ground truth availability map is maintained by the simulator, the estimation results from ParkScan can be evaluated.

We extracted the parking request set for all workdays in March 2015 to generate historical knowledge information, i.e., historical parking availability rates and historical park/vacate probabilities (discussed in Section 3.5.3) at the road segment level. However, due to the extensive computation cost to simulate the step-by-step vehicle movements, we chose the parking requests for one day (March 6, 2015) from 8:00 am to 5:00 pm to evaluate the parking estimation performances of the proposed ParkScan system.

3.7.2 Validation of Experiment Settings

To verify our simulation design and tune the parameter of distance thresholds in the parking search simulator, we run the parking search simulator presented in Section 3.7.1 for March 6, 2015, starting from 8:00 am to 5:00 pm to simulate the cars' final parking positions. As the baseline, we also test another simulator which assumes drivers will take the first available spot while searching for parking. We compare the resulting occupancy map from the two simulators to the ground-truth occupancy generated using the parking transaction. An ideal simulator with a good parameter setting should generate realistic parking search trajectory and thus incur smaller occupancy difference.

The comparison result demonstrates that our simulation method relying on parking decision simulation model substantially outperforms the baseline simulator when we set the distance threshold as 150 meters. Fig. 3.21 shows the road segment level occupancy differences after 2 hours of the simulation. In particular, 95% of the road segments have an occupancy difference of fewer than 3 cars. Moreover, since quite a few car's final parking positions are just one road segment shifted from the actual parking position, the actual occupancy map shows only subtle differences, e.g. Fig. 3.22 and Fig. 3.23 visualize the heap map of the two resulting occupancy

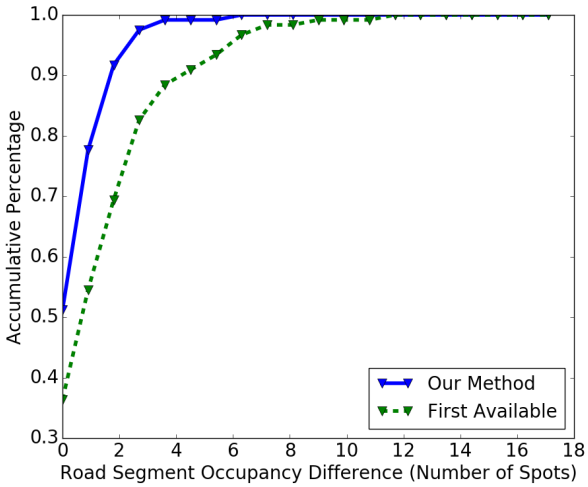

Figure 3.21: The CDF of occupancy differences generated by different simulation algorithms

maps at 9:00 am. These results demonstrate that our way to simulate the driver's parking search trajectory is realistic.

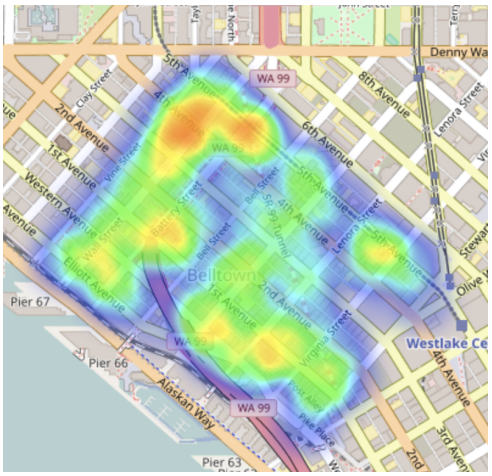

Figure 3.22: Parking occupancy at 9:00 am generated by our simulation algorithm

Figure 3.23: Parking occupancy at 9:00 am generated by real parking transactions

3.7.3 Evaluation Results

In this section, we use the same metrics and evaluation process as in Section 3.6.3 to test the performance of ParkScan in the street parking scenario.

One-pass Estimation

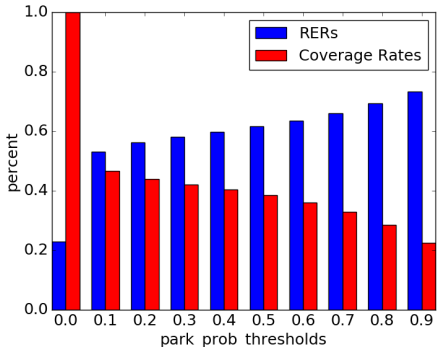

Figure 3.24: One-pass estimation results in the Seattle street parking experiment

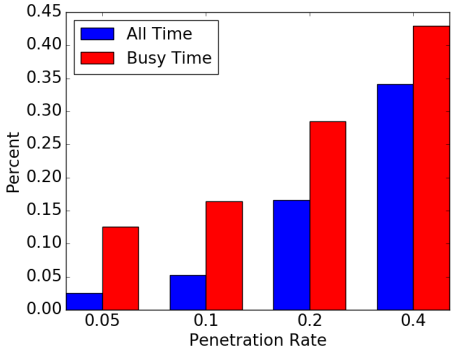

Figure 3.25: Mean RERs for various penetration rates in the Seattle street parking experiment

Fig. 3.24 illustrates the RER and coverage rate results from the one-pass experiment in the Seattle street parking scenario. One-pass estimation computes the availability prediction for all spots along the reported search trajectory. Similar to the parking lot results, the RER values in the street parking one-pass experiment also increase as $p_{u,\pi,i}\,(park \mid empty)$ threshold values increase. However, the coverage rate drops sharply from the threshold value $p_{u,\pi,i}\,(park \mid empty) = 0.2$. This phenomenon is reasonable because the drivers may visit a lot of spots along the reported driving trajectory, however, they only consider the "good" parking candidates limited to a smaller number of spots that are close to the drivers' final destinations. Our parking decision model naturally differentiates the "good" candidates and the rest by generating different $p_{u,\pi,i}\,(park \mid empty)$ likelihood values. Motivated by this finding, we set the likelihood threshold of $p_{u,\pi,i}\,(park \mid empty)$ as 0.2 in the estimation fusion process to determine the start position of each search trajectory (refer to Section 3.5.4 for details).

Estimation Fusion

We also applied the estimation fusion algorithm presented in Section 3.5.3 in the street parking scenario. For the performance measurement, for each minute we calculate the MAD for all spots in the simulation urban area, if there is at least one ongoing parking search process. The MADs from ParkScan are compared with the historical estimation MADs to compute the RERs.

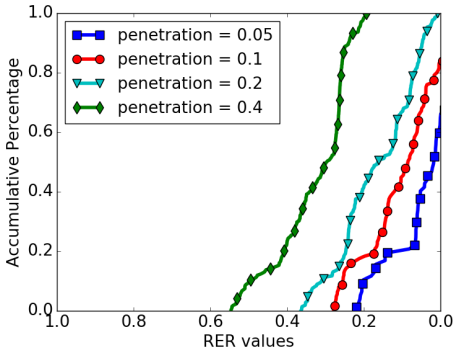

Figure 3.26: CDF of RERs for various penetration rates in the Seattle street parking experiment (all time)

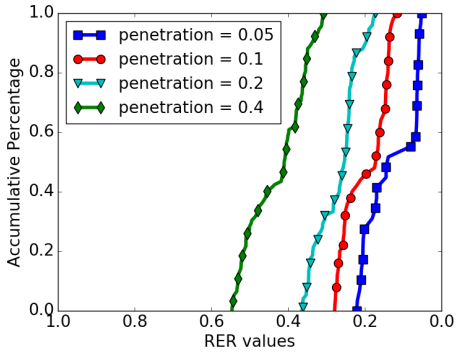

Figure 3.27: CDF of RERs for various penetration rates in the Seattle street parking experiment (busy time)

RERs of MAD for different penetration rates. Fig. 3.25 demonstrates the mean RERs under different system penetration rates over different time interval settings. In the street parking experiment, when the penetration rate is 5%, we achieve 2.5% RER. However, when the penetration rate rises to 40%, the RER of MAD grows to 34.1%. During the busy parking hours, the RERs increase from 12.9% at a 5% penetration to 42.8% at a 40% penetration. The trends are in accordance with the findings in the parking lot experiment. Fig. 3.26 and Fig. 3.27 further illustrate the CDF of RERs for the street parking experiment for all intervals throughout the experiment and for busy parking hours, respectively. Similar to the parking lot experiment, the benefits in the street parking experiment are also normally distributed among most time. In addition, a higher penetration rate pushes the CDF of RER curve to the up-left side, implying increased benefits in both estimation accuracy and coverage. In the street parking all time results (Fig. 3.26), low penetration rates sometimes lead to worse estimation accuracies comparing to the historical data. This is because we use a stochastic process to make parking decisions in the parking search simulator: even if the predicted $p_{u,\pi,i}\,(park)$ is higher than 0.5 for an available spot, there are still chances that a driver will ignore that spot. When the penetration rate is low, this random decision process generates considerable noise, which causes wrong predictions at some visited spots. Fortunately, this type of errors is mitigated as the penetration increases or during the busy parking hours.

Comparison of MAD by using only one-pass estimation and estimation fusion. Fig. 3.28 d-emonstrates positive RERs of estimation fusion results in street parking scenario with respect

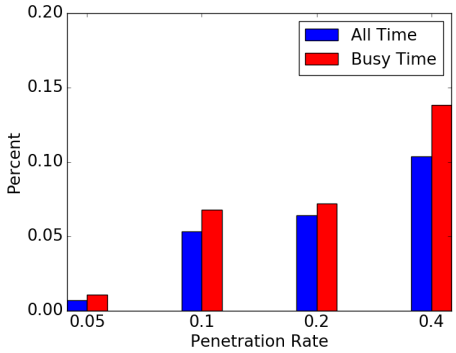

Figure 3.28: RERs of estimation fusion with respect to the one-pass estimation for various penetration rates

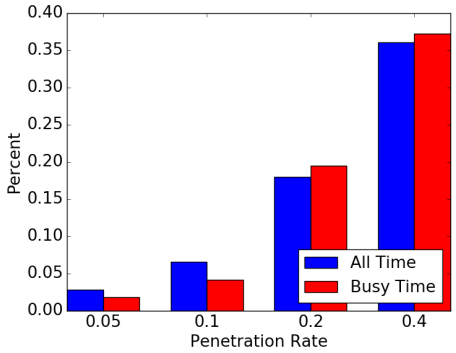

Figure 3.29: RERs of ParkScan estimation with respect to the coarse-grained solution for various penetration rates

to the one-pass estimation at all penetration rates, meaning that estimation fusion produces better spot availability estimations in the street parking scenario. The estimation fusion shows less improvement in the street parking scenario than the parking lot 1 experiment. This is because the park/vacate probabilities estimated from the historical data in the Seattle parking transaction are different from the real rate generated from our vehicle parking simulation. However, the positive RER values also demonstrate the robustness of our estimation fusion algorithm over noise park/vacate rate estimations.

Comparison of MAD by using ParkScan and the coarse-grained solution. Similar to the pa-rking lot evaluation, we also compare the spot-level availability estimation using ParkScan and the coarse-grained availability estimation using PhonePark [170] in the street parking scenario. We provide the PhonePark with ground-truth penetration rates and use Kalman filter to combine the detected real-time parking events and historical statistics. The section unit used in PhonePark is each street parking section, i.e., road segment. Fig. 3.29 demonstrates that ParkScan provides more accurate spot-level availability estimation than PhonePark (by positive RER values). Moreover, by comparing the RER values at different penetration rates, we find that the performance improvement of ParkScan increases with rising penetration rate in the street parking scenario.

RERs of MAD for different likelihood thresholds. Section 3.7.3 determined the likelihood threshold to detect the start of a search trajectory and the consideration candidates set. In this evaluation, we tested the threshold values from 0.0 to 0.8 under different penetration

rates to evaluate the likelihood threshold configuration. Fig. 3.30 and Fig. 3.31 show the CDF of RERs for different likelihood settings when the penetration rate is 0.1. In both figures, a threshold of 0.2 provides the best performance, which validates our setting of the likelihood thresholds in Section 3.7.3. This result is consistent with all penetration rates in both experimental interval settings, though, due to the limit of space, we do not present the figures for other penetration rates. Interestingly, it is also shown in Fig. 3.30 and Fig. 3.31 that different thresholds achieve similar results. This implies that ParkScan is not very sensitive for likelihood thresholds, so this value does not need to be precisely tested for each real-world deployment.

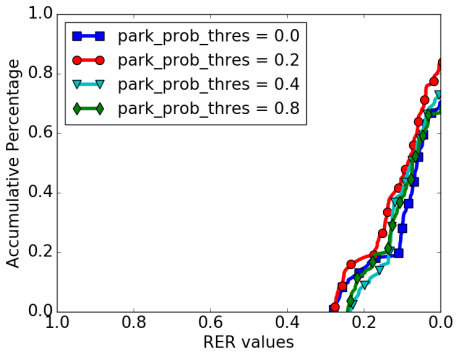

Figure 3.30: CDF of RER for various park probabilities (all time, penetration rate = 0.1)

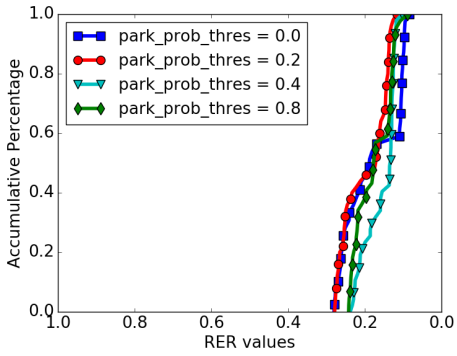

Figure 3.31: CDF of RER for various park probabilities (busy time, penetration rate = 0.1)

3.7.4 Evaluation Summary and Limitations

Our findings from street parking experiments are as follows: (i) ParkScan reduces the estimation errors compared to historical estimations, when considering only visited spot at the visit moments (shown in Fig. 3.24). (ii) ParkScan reduces the estimation errors for all spots at the moment when drivers need the estimation map (shown in Fig. 3.25 to Fig. 3.29). (iii) The method to estimate the parking search start and consideration set improves the performance on ParkScan for the street parking scenario (as shown in Fig. 3.30 and Fig. 3.31). Moreover, ParkScan is not very sensitive to threshold values, which determine the park search start. So we can start from a default value (e.g. 0.2) for the real world deployment.

In the street parking experiment, since we do not have as detailed data as the parking lot experiment. We made a few assumptions (e.g., a driver's final destination is a random point

along the parked road segment; a simulator starts the parking simulation from a random point 150 meters from the final destination) to build the experiment setting. Moreover, we made an approximation to estimate the historical statistics in the unit of road segments instead of spots. The assumptions on the experiment setting mainly influence the factuality of the ground truth in our simulation towards the real situation in Seattle. When assumptions change, the inputs of ParkScan (i.e., search trajectory and final parked position) also change. Therefore, we believe the influence of these assumptions on the ParkScan performance measurement is low, especially after we have validated the results of these settings in Section 3.7.2. The approximation of the historical statistics is due to the fact that we only have road segment level information (i.e., parking transactions are associated with payment stations at each road segment). We acknowledge that this approximation may influence the evaluation results, because the historical availability rate is the baseline algorithm. However, since our ParkScan is also fed with the historical statistic data with the same unit, the improvement on this approximation will also improve the performance of ParkScan.

3.8 Discussion

3.8.1 Acquisition of Driver's Destination and Search Trajectory

ParkScan parking availability estimation leverages the driver's destination and the parking search trajectory, which is not uncommon in other parking crowdsourcing systems, e.g., [123]. The final destination can be obtained from vehicle navigation system or estimated using place visit detections and indoor localization technologies [123, 75, 85]. Note that, ParkScan may work with any parking/unparking detection technique to detect the parked positions and can tolerate location noises. Given that vehicle tracking becomes more accurate in both outdoor [76, 16] and indoor environments [190, 155], estimating the parking trajectory is increasingly feasible.

3.8.2 Generalization of the Parking Decision Model

Section 3.4 builds the parking search model for a group of people that use a particular parking lot. Since the estimates from multiple sources are fused to eliminate the individual errors,

ParkScan works well even if the model is not tailored for each participant driver. However, we do acknowledge that the parking decision model of individuals may be different, for instance, some driver may prefer shadow spots during the summer. Once the ParkScan system has been installed in the driver's smartphone, we could use opportunistic sensing to generate samples to refine driver's decision model, e.g., parking at a spot generates a positive parking decision sample, and ignoring a spot, which according to a dedicated parking status sensor or our estimation is almost 100% sure available, generates a negative parking decision sample. We believe this will further improve the performance of ParkScan.

3.8.3 Historical Data and Parking Layout Map

ParkScan leverages the historical data when estimating the background parking availability. Without the initial historical data, ParkScan may need a long time to converge. Fortunately, given the development of the internet of things, we can retrieve historical statistics from many government resources (e.g., we compute historical statistics using the parking transaction dataset in Seattle) and online parking information providers (e.g. ParkMe [72] provides the historical parking availability of many cities). Parking layout maps are also available from map providers or government open datasets. Inspired by the recent advances to build road and building maps from satellite images [180, 115], parking layout maps can also be potentially extracted using similar technologies.

3.8.4 Enhancement with Richer Data and More Powerful Models

Park-Scan currently relies on a Bayesian inference model to estimate spot-level parking availability. The Bayesian model has minimum deployment requirements, i.e., it only needs the parking search trajectory, trip destination, historical statistics, and facility layout to estimate parking availability. However, the Bayesian model also has limitations. For instance, the current ParkScan system may provide biased estimations, because based on the Bayesian model, passing by a spot without taking it never makes the availability probability higher than the historical availability rate. While some applications, e.g. routing, just need the relative availability rate instead of the absolute values, we argue that ParkScan can be enhanced by using dedicated parking sensor data, coarse-grained crowdsourcing approaches, and more complex

models. For example, the ParkScan estimation result of a spot can always be over-written by the sensor monitoring the same spot. Even if only the coarse-grained parking availability can be measured using sensors, e.g. gate counter, or coarse-grained crowdsourcing approaches, we can use the coarse-grained availability statistics to renormalize the ParkScan results. Finally, if adequate data is accessible for a parking facility, more powerful models with additional features can be explored. For instance, traffic demand (e.g. taxi pickup/drop-offs), POIs data, and social network check-ins can be used in addition to the $P(empty\,|\neg\, park)$ as new features. We expect these features can be combined together (e.g. by tensor decomposition) to improve the accuracy of the parking availability estimation.

3.9 Related Work

ParkScan is a crowdsourcing system relying on a parking decision model to compute probabilistic posterior estimations of spot-level parking availability. So both smart parking systems and parking search process modeling are related to our work.

3.9.1 Smart parking systems

Table. 3.3 surveyed the smart parking systems, which can be divided into solutions based on dedicated sensors and dedicated-sensor free solution. Smart parking systems using dedicated sensors have been excessively studied back to the last decade [71]. These solutions can be used to detect the parking/unparking event for a particular vehicle [104] or detect the availability of spots [151, 87]. Recently, the sensors are deployed in a mobile fashion to increase the coverage [111]. However, suffering from the cost issues in installation and maintenance, these solutions are deployed in very limited regions. The dedicated-sensor-free solutions rely on the sensors already built in smartphones or other wearable devices. They use various sensors to detect the transition between driving and walking to estimate the parking/unparking events [124, 170, 123, 155, 105]. Systems use either sampling theory [170], or use some special heuristics for off-street parking, e.g. a parking lot is very likely to be full if a car goes in and comes out without parking inside [123, 140] or parking search time (and lower-level vehicle maneuvers, e.g., brakes and turns) is related to parking availability statistics [38], to infer

Table 3.3: Smart parking system survey

Scenarios		Dedicated Sensors	Dedicated Sensor Free
Park/Unpark Event Detection		Spark[104], RFID or computer vision solutions	ParkSense[124], Updetector[105], PhonePark[170], PocketParker[123], BluePark[155]
Background Availability Detection	Coarse-grained	ParkNet[111], Parking Gate Counter†, etc.	PhonePark[170], PocketParker[123]†, ParkGauge[38]†, SPIRE[140]†
	Fine-grained	ParkNet[111], SFPark[151], LA Express Park[87], QuickSpot[109]	ParkScan (This work)

Note: † solutions only for off-street parking usage.

background parking availabilities under low penetration rates.

Our work looks into the human's probabilistic parking decision model, which is dedicated sensor free and can be used both for on-street and off-street parking scenarios. Moreover, the system provides fine-grained and reliable estimations under a low penetration rate rather than only coarse-grained statistics.

3.9.2 Parking Behavior Modeling

The modeling of drivers' parking behavior in the transportation domain can be divided into parking choice modeling and parking decision modeling. The parking choice model is a decision maker that selects a parking space from given options. Many data-driven parking choice models have been proposed, e.g. [25, 36, 164, 64]. Prior studies on the parking decision model (the decision maker to decide whether to take the current spot or not) mostly use ad-hoc models without using real-world data, e.g. [179, 157, 24, 110], or assume that drivers are fully rational when facing parking decisions or routing choices [162, 159, 108, 30]. The only data-driven study is based on lab questionnaires [33]. Nevertheless, the behavioral studies in both categories show homogeneity of the models among various users and parking facilities [64, 33].

Our work applied a data-driven approach to learning the parking decision model from real-world parking decision events with human decisions. We automate the data collection process

through computer vision techniques and thus make the calibration of parking decision models practically feasible. To the best of our knowledge, this is the first work that extracts fine-grained trajectory data and individual parking decisions to build a parking decision model.

3.10 Summary

This chapter presented a parking availability crowdsourcing system, ParkScan, based on the key idea that the behavior of a parking seeker's ignoring a visited spot probabilistically implies the unavailability of that spot. We collected an extensive dataset containing 8,000 vehicle trajectories and over 55,000 human parking decisions, and we utilized a data-driven approach to building a parking decision model. Using the learned model, we provided an efficient way to estimate the priors used in the Bayesian rule and presented an approach to combine estimations from multiple spot visits covered by different parking search trajectories. Finally, we evaluated the ParkScan system using real-world data both from the parking lot and the street parking scenarios. Both of the experiments show that ParkScan improves the fine-grained parking availability estimations, even under extremely low penetration rates.

Chapter 4

Conclusions and Future Directions

In this dissertation, we have identified the practical and theoretical challenges of vehicular traffic routing and parking availability crowdsourcing from the human behavior's perspective. We have presented novel approaches to address these challenges and demonstrated their feasibility and effectiveness through both simulation experiments and field studies.

The main contribution of this dissertation is to explore the design of cost-effective systems that utilize driver's smart devices to solve the road transportation challenges and to enable better travel experience. To solve several challenges in relieving traffic congestions, we propose the Themis system, which applies participatory sensing techniques for efficient traffic condition sensing and balanced routing strategy to proactively alleviate the congestions caused by drivers' coupled routing decisions. To expand our knowledge on human's parking behavior, we collect a large-scale parking trajectory dataset and accordingly build a generic model to describe and predict driver's parking decision. Moreover, the model is used for a novel parking crowdsourcing system that enables fine-grained parking availability estimation of high reliability without using dedicated parking sensors.

4.1 Summary of the Key Findings

The dissertation draws the following key findings based on the design, implementation, and design of the Themis system and the ParkScan system:

- Low-sampling-rate trajectory data can be used to estimate real-time traffic conditions using limited computation power through approximation. The spatiotemporal information implied in the low-sampling-rate vehicle trajectory can be recovered through the map-matching, travel time allocation, and aggregation processes, considering driver's travel

preferences, i.e., drivers prefer to use shorter and easier driving paths. Moreover, the computational requirement for travel time estimations can be compensated by removing redundant trajectory data through an approximation framework. This finding is supported by Section 2.3, Section 2.6.2, and Section 2.7.2.

- Balanced traffic routing strategy can significantly reduce the average travel time of the participant driver by proactively alleviating the traffic congestions. Given the high penetration of the navigation systems based on real-time traffic condition, people's driving paths are likely to be coupled, generating unnecessary congestions while using a greedy routing strategy. Balanced traffic routing, by considering the expected outcome of routing decision towards other travelers, proactively reduces the chance to build congestions and thus substantially cuts down the participant drivers' trip travel time. The finding is supported by Section 2.4, Section 2.6.3, Section 2.6.4, and Section 2.7.3.

- Human seekers' parking decisions can be modeled using generic features and are therefore highly predictable given the layout of the parking facility, the historical parking availability statistics, the driver's parking search trajectory, and the driver's final destination. Based on the extensive parking trajectory dataset collected using the computer vision techniques, we train a generic model that is independent of different parking facilities and can be used to accurately simulate/predict a parking seeker's decision when finding an available spot. The finding is supported by Section 3.4, Section 3.7, and Section 3.6.

- Driver's parking decision models can be used to infer the availability of the spots along the driver's parking search trajectory, resulting in a parking availability crowdsourcing system that produces fine-grained parking availability estimation with high reliability and low cost. Using the learned parking decision model, we first employ a Bayesian model to infer the availability of the pass-by spot to generate the one-time estimation. By combining the one-time estimations at different times as well as the participant parking/unparking transition events using Hidden Markov Model, we produce the reliable parking availability estimation in different parking scenarios with various penetration rates. The finding is supported by Section 3.5, Section 3.7, and Section 3.6.

All the above key findings support the thesis of the dissertation that human behaviors can be captured and analyzed to build road travel support systems to improve drivers' travel experience.

4.2 Future Directions

Looking forward, we believe the success and development of road travel support system rely heavily on the infrastructure, performance, and functionalities. Future work on road travel support system will range from infrastructure improvement, studies in the human machine interaction, and novel applications.

4.2.1 Infrastructure Improvements

The road travel support system relies on efficient sensing and powerful computation infrastructures. The recent advances in both directions are likely to promote the future work of the road travel support system.

GPU based Routing computation. GPU computing gains great advances in large-scale machine learning and parallel optimization. As the balanced routing problem is essentially an constrained optimization problem with a separable convex objective function [35], it is highly possible that the computation can be accelerated using GPU infrastructure. Since GPU processors have much higher computation throughput, it is possible that we can explore larger search space than the proposed sequential balanced routing algorithm and thus achieve even better routing efficiency.

Parking availability crowdsourcing based on dash cameras. With the development of IoTs, the dash camera with the dual lens is becoming popular. It is possible to use the video to crowdsource parking availabilities. Computer vision components can differentiate the parked car with other obstacles. Moreover, the depth information provided by the dual lens together with the mobility status of the vehicle can be used to estimate the position of the parked vehicle. The new infrastructure can potentially generate accurate parking availability by direct observations.

4.2.2 Human Machine Interaction

Road travel support systems impact the transportation system through drivers' compliance to the system suggestions. As a result, the human machine interaction needs to be studied.

Understand the human route choice model. While we design the balanced routing algorithm to provide routing suggestions, human's route choice model needs to be explored to solve the following two problems: (i) the route choice model defines driver's compliance rate to the suggested route. As a result, it also affects the prediction of participant traffic; (ii) understanding the human route choice model helps the system to determine when to initiate the rerouting services. This will help resolve driver's confusion and uncertainty and improve the driving experiences.

Personalized Parking Decision Model. While we have built the generic model to estimate/predict the parking decision of the public, the models for different persons may be slightly different. For instance, some people may prefer to longer walking distance for a healthy lifestyle while others may want as short walking distance as possible to save travel time. We believe the personalized parking decision model can be refined through opportunistic sensing when the ParkScan system is installed in a user's smartphone. For instance, parking at a spot generates a positive sample, while ignoring a spot that is estimated to be empty produces a negative sample.

4.2.3 Novel Applications

Apart from the Themis system and the ParkScan system proposed in the dissertation, novel applications can be built to extend the functionality or the usage scenarios.

Smart parking planning. The ParkScan system utilizes the parking decision model to crowdsource parking availability. However, the parking decision model can also be used for smart parking planning. For instance, when designing the layout of a parking facility, the planner can test different designs using simulation experiments to investigate many metrics, such as the expected searching time, the total walking distance and so on. The experiment result will help make the best planning.

Routing in autonomous driving. With the advent of the autonomous car, we believe the

routing algorithm may also need to be changed. Since people are released from boring driving, they may just want to go through a path that has a better scene or network connectivity for entertainment. In other cases, people may also want the car to be driven in a path that is easier for the autonomous car, e.g. highways. These new rerouting objectives require richer crowdsourcing content and are the chances for future routing applications.

On-the-fly parking route suggestion. Nowadays, most parking guidance systems work in the deterministic availability situation. The problem is hard for probabilistic settings because the parking guidance system does not have the equivalent observation as the onboard driver. For instance, when passing by an empty spot, the human driver may remember it and directly come back if he/she later finds no better spot. However, the guidance system does not have the availability information of that spot and thus may believe the spot is unavailable. With the help of new sensors, e.g. the dash camera mentioned in 4.2.1, and the human decision models, on-the-fly parking suggestion may help the driver determine the best parking spot even when the availability estimation is probabilistic.

References

[1] DiDi. http://www.didichuxing.com/en/.

[2] Lyft. https://www.lyft.com/.

[3] OpenStreetMap. http://www.openstreetmap.org/.

[4] osm2pgrouting. https://github.com/pgRouting/osm2pgrouting/.

[5] OsmAnd Maps and Navigation. http://osmand.net/.

[6] pgRouting. http://pgrouting.org/.

[7] PostgreSQL. http://www.postgresql.org/.

[8] Uber. https://www.uber.com/.

[9] Waze. http://www.waze.com/.

[10] Samihah Abdullah, Widad Ismail, Zaini Abdul Halim, and Che Zalina Zulkifli. Integrating zigbee-based mesh network with embedded passive and active rfid for production management automation. In *RFID-Technologies and Applications (RFID-TA), 2013 IEEE International Conference on*, pages 1–6. IEEE, 2013.

[11] Federal Highway Administration. Annual vehicle miles traveled in the u.s. https://www.afdc.energy.gov/data/10315/.

[12] Federal Highway Administration. Our nation's highways: 2011. https://www.fhwa.dot.gov/policyinformation/pubs/hf/pl11028/chapter1.cfm/.

[13] Vedat Akgun, Erhan Erkut, and Rajan Batta. On finding dissimilar paths. *European Journal of Operational Research*, 121(2):232 – 246, 2000.

[14] Fawzi M Al-Naima and Hassan A Hamd. Vehicle traffic congestion estimation based on rfid. *International Journal of Engineering Business Management*, 4:30, 2012.

[15] Dominik Allemann and Martin Raubal. Towards health-optimal routing in urban areas. In *Proceedings of the Second ACM SIGSPATIAL International Workshop on the Use of GIS in Public Health*, pages 56–59. ACM, 2013.

[16] Heba Aly and Moustafa Youssef. Dejavu: an accurate energy-efficient outdoor localization system. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 154–163. ACM, 2013.

[17] Javed Aslam, Sejoon Lim, Xinghao Pan, and Daniela Rus. City-scale traffic estimation from a roving sensor network. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 141–154. ACM, 2012.

[18] Javed Aslam, Sejoon Lim, and Daniela Rus. Congestion-aware traffic routing system using sensor data. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 1006–1013. IEEE, 2012.

[19] Jie Bao, Ruiyuan Li, Xiuwen Yi, and Yu Zheng. Managing massive trajectories on the cloud. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 41. ACM, 2016.

[20] Favyen Bastani, Yan Huang, Xing Xie, and Jason W Powell. A greener transportation mode: flexible routes discovery from gps trajectory data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 405–408. ACM, 2011.

[21] Alessio Bechini, Francesco Marcelloni, and Armando Segatori. A mobile application leveraging QR-codes to support efficient urban parking. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2013*, pages 1–3. IEEE, 2013.

[22] Andrei Iu Bejan, Richard J Gibbens, David Evans, Alastair R Beresford, Jean Bacon, and Adrian Friday. Statistical modelling and analysis of sparse bus probe data in urban areas. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1256–1263. IEEE, 2010.

[23] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

[24] Itzhak Benenson, Karel Martens, and Slava Birfir. Parkagent: An agent-based model of parking in the city. *Computers, Environment and Urban Systems*, 32(6):431–439, 2008.

[25] Angela Stefania Bergantino, Angela De Carlo, Andrea Morone, et al. Individuals' behaviour with respect to parking alternatives: a laboratory experiment. Technical report, University Library of Munich, Germany, 2015.

[26] Michele Berlingerio, Francesco Calabrese, Giusy Di Lorenzo, Rahul Nair, Fabio Pinelli, and Marco Luca Sbodio. Allaboard: a system for exploring urban mobility and optimizing public transport using cellphone data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 663–666. Springer, 2013.

[27] Zhang Bin, Jiang Dalin, Wang Fang, and Wan Tingting. A design of parking space detector based on video image. In *Electronic Measurement & Instruments, 2009. ICEMI'09. 9th International Conference on*, pages 2–253. IEEE, 2009.

[28] Board of Public Roads. Traffic assignment manual for application with a large, high speed computer. Technical report, 1964.

[29] Pierre Borgnat, Eric Fleury, Céline Robardet, and Antoine Scherrer. Spatial analysis of dynamic movements of vélo'v, lyon's shared bicycle program. In *ECCS'09*. Complex Systems Society, 2009.

[30] Stephen D Boyles, Shoupeng Tang, and Avinash Unnikrishnan. Parking search equilibrium on a network. *Transportation Research Part B: Methodological*, 81:390–409, 2015.

[31] Orhan Bulan, Robert P Loce, Wencheng Wu, YaoRong Wang, Edgar A Bernal, and Zhigang Fan. Video-based real-time on-street parking occupancy detection system. *Journal of Electronic Imaging*, 22(4):041109–041109, 2013.

[32] Daniele Buscema, Matteo Ignaccolo, Giuseppe Inturri, Alessandro Pluchino, Andrea Rapisarda, Corrado Santoro, and Salvatore Tudisco. The impact of real time information on transport network routing through intelligent agent-based simulation. In *Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference*, pages 72–77. IEEE, 2009.

[33] Michal Chalamish, David Sarne, and Sarit Kraus. Mass programmed agents for simulating human strategies in large scale systems. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 135. ACM, 2007.

[34] Gayathri Chandrasekaran, Tam Vu, Alexander Varshavsky, Marco Gruteser, Richard P Martin, Jie Yang, and Yingying Chen. Tracking vehicular speed variations by warping mobile phone signal strengths. In *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pages 213–221. IEEE, 2011.

[35] Tsung-Hui Chang, Mingyi Hong, and Xiangfeng Wang. Multi-agent distributed optimization via inexact consensus admm. *IEEE Transactions on Signal Processing*, 63(2):482–497, 2015.

[36] Emmanouil Chaniotakis and Adam J Pel. Drivers' parking location choice under uncertain parking availability and search times: A stated preference experiment. *Transportation Research Part A: Policy and Practice*, 82:228–239, 2015.

[37] Po-An Chen and David Kempe. Altruism, selfishness, and spite in traffic routing. In *Proceedings of the 9th ACM conference on Electronic commerce*, pages 140–149. ACM, 2008.

[38] Jim Cherian, Jun Luo, Hongliang Guo, Shen-Shyang Ho, and Richard Wisbrun. Parkgauge: Gauging the occupancy of parking garages with crowdsensed parking characteristics. In *Mobile Data Management (MDM), 2016 17th IEEE International Conference on*, volume 1, pages 92–101. IEEE, 2016.

[39] Jatuporn Chinrungrueng, Udomporn Sunantachaikul, and Satien Triamlumlerd. Smart parking: An application of optical wireless sensor network. In *Applications and the Internet Workshops, 2007. SAINT Workshops 2007. International Symposium on*, pages 66–66. IEEE, 2007.

[40] Benjamin Coifman. Estimating travel times and vehicle trajectories on freeways using dual loop detectors. *Transportation Research Part A: Policy and Practice*, 36(4):351–364, 2002.

[41] Serdar Çolak, Antonio Lima, and Marta C González. Understanding congested travel in urban areas. *Nature communications*, 7, 2016.

[42] LDA Consulting. Commuter connections state of the commute survey. Technical report, 2016.

[43] Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem: models and algorithms. *Annals of operations research*, 153(1):29, 2007.

[44] Philippe Cudre-Mauroux, Eugene Wu, and Samuel Madden. Trajstore: An adaptive storage system for very large trajectory data sets. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 109–120. IEEE, 2010.

[45] Daniel J Dailey, Fritz W Cathey, and Suree Pumrin. An algorithm to estimate mean traffic speed using uncalibrated cameras. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):98–107, 2000.

[46] Andrea David, Klaus Overkamp, and Walter Scheuerer. Event-oriented forecast of the occupancy rate of parking spaces as part of a parking information service. In *Proceedings of the 7th World Congress on Intelligent Systems*, 2000.

[47] Stacy Cagle Davis, Susan E Williams, and Robert Gary Boundy. Transportation energy data book: Edition 35. Technical report, Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States), 2016.

[48] Corrado De Fabritiis, Roberto Ragona, and Gaetano Valenti. Traffic estimation and prediction based on real time floating car data. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pages 197–203. IEEE, 2008.

[49] Srinivas Devarakonda, Parveen Sevusu, Hongzhang Liu, Ruilin Liu, Liviu Iftode, and Badri Nath. Real-time air quality monitoring through mobile sensing in metropolitan areas. In *Proceedings of the 2nd ACM SIGKDD international workshop on urban computing*, page 15. ACM, 2013.

[50] Google Developers. ActivityRecognitionApi, 2017. `https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi/`.

[51] Ericsson ConsumerLab. From apps to everyday situations - an ericsson consumer insight study. Technical report, 2011.

[52] Stefan Foell, Santi Phithakkitnukoon, Gerd Kortuem, Marco Veloso, and Carlos Bento. Catch me if you can: Predicting mobility patterns of public transport users. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 1995–2002. IEEE, 2014.

[53] Matthew William Fontana. *Optimal routes for electric vehicles facing uncertainty, congestion, and energy constraints*. PhD thesis, Massachusetts Institute of Technology, 2013.

[54] Jon Froehlich, Joachim Neumann, Nuria Oliver, et al. Sensing and predicting the pulse of the city through shared bicycling. In *IJCAI*, volume 9, pages 1420–1426, 2009.

[55] Raghu K Ganti, Nam Pham, Hossein Ahmadi, Saurabh Nangia, and Tarek F Abdelzaher. GreenGPS: a participatory sensing fuel-efficient maps application. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 151–164. ACM, 2010.

[56] Yong Ge, Hui Xiong, Alexander Tuzhilin, Keli Xiao, Marco Gruteser, and Michael Pazzani. An energy-efficient mobile recommender system. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 899–908. ACM, 2010.

[57] Yanfeng Geng and Christos G Cassandras. Dynamic resource allocation in urban settings: A smart parking approach. In *Computer-Aided Control System Design (CACSD), 2011 IEEE International Symposium on*, pages 1–6. IEEE, 2011.

[58] Supriyo Ghosh, Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. Dynamic redeployment to counter congestion or starvation in vehicle sharing systems. In *Eighth Annual Symposium on Combinatorial Search*, 2015.

[59] Győző Gidófalvi and Can Yang. Scalable detection of traffic congestion from massive floating car data streams. In *Proceedings of the 1st International ACM SIGSPATIAL Workshop on Smart Cities and Urban Analytics*, UrbanGIS'15, pages 114–121, New York, NY, USA, 2015. ACM.

[60] Chong Yang Goh, Justin Dauwels, Nikola Mitrovic, Muhammad Tayyab Asif, Ali Oran, and Patrick Jaillet. Online map-matching based on hidden markov model for real-time traffic sensing applications. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 776–781. IEEE, 2012.

[61] Google Maps. `http://www.google.com/mobile/maps/`.

[62] Alessandro Grazioli, Marco Picone, Francesco Zanichelli, and Michele Amoretti. Collaborative mobile application and advanced services for smart parking. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 2, pages 39–44. IEEE, 2013.

[63] BD Greenshields, JR Bibbins, Ws Channing, and Hh Miller. A study of traffic capacity. In *Highway research board proceedings*, volume 1935. National Research Council (USA), Highway Research Board, 1935.

[64] Liya Guo, Shan Huang, Jun Zhuang, and Adel W Sadek. Modeling parking behavior under uncertainty: a static game theoretic versus a sequential neo-additive capacity modeling approach. *Networks and Spatial Economics*, 13(3):327–350, 2013.

[65] Abeer Hakeem, Narain Gehani, Xiaoning Ding, Reza Curtmola, and Cristian Borcea. On-the-fly curbside parking assignment. In *Proceedings of the 8th EAI International Conference on Mobile Computing, Applications and Services*, pages 1–10. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016.

[66] Patrick T Harker. Multiple equilibrium behaviors on networks. *Transportation Science*, 22(1):39–46, 1988.

[67] Pei-Jin He, Kuo-Feng Ssu, and Yu-Yuan Lin. Sharing trajectories of autonomous driving vehicles to achieve time-efficient path navigation. In *VNC*, pages 119–126, 2013.

[68] Aude Hofleitner, Ryan Herring, Pieter Abbeel, and Alexandre Bayen. Learning the dynamics of arterial traffic from probe data using a dynamic bayesian network. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1679–1693, 2012.

[69] Martin Hofmann, Philipp Tiefenbacher, and Gerhard Rigoll. Background segmentation with feedback: The pixel-based adaptive segmenter. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 38–43. IEEE, 2012.

[70] T Chiang Hu. Multi-commodity network flows. *Operations research*, 11(3):344–360, 1963.

[71] MYI Idris, YY Leng, EM Tamil, NM Noor, and Z Razak. Car park system: A review of smart parking system and its technology. *Information Technology Journal*, 8(2), 2009.

[72] ParkMe Inc. ParkMe, 2016. `https://www.parkme.com/`.

[73] INRIX. Inrix global traffic scorecard. Technical report, 2016.

[74] Seattle IT. Seattle Parking Transactions, 2015. `https://data.seattle.gov/Transportation/Seattle-Parking-Transactions/updn-y53g/data`.

[75] Yifei Jiang, Xin Pan, Kun Li, Qin Lv, Robert P Dick, Michael Hannigan, and Li Shang. Ariel: Automatic wi-fi based room fingerprinting for indoor localization. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 441–450. ACM, 2012.

[76] Yurong Jiang, Hang Qiu, Matthew McCartney, Gaurav Sukhatme, Marco Gruteser, Fan Bai, Donald Grimm, and Ramesh Govindan. Carloc: Precisely tracking automobile position. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 253–265. ACM, 2015.

[77] Yang Jun. A system framework of active parking guidance and information system. In *Information Engineering (ICIE), 2010 WASE International Conference on*, volume 2, pages 150–154. IEEE, 2010.

[78] Ho Gi Jung, Young Ha Cho, Pal Joo Yoon, and Jaihie Kim. Scanning laser radar-based target position designation for parking aid system. *IEEE Transactions on Intelligent Transportation Systems*, 9(3):406–424, 2008.

[79] Andreas Kaltenbrunner, Rodrigo Meza, Jens Grivolla, Joan Codina, and Rafael Banchs. Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system. *Pervasive and Mobile Computing*, 6(4):455–466, 2010.

[80] Chennakesava Reddy Kamireddy, BN Keshavamurthy, et al. Efficient routing of 108 ambulances using clustering techniques. In *Computational Intelligence and Computing Research (ICCIC), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[81] Ryo Kanamori, Jun Takahashi, and Takayuki Ito. Evaluation of anticipatory stigmergy strategies for traffic management. In *VNC*, pages 33–39, 2012.

[82] Evangelos Kanoulas, Yang Du, Tian Xia, and Donghui Zhang. Finding fastest paths on a road network with speed patterns. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 10–10. IEEE, 2006.

[83] Merkourios Karaliopoulos, Konstantinos Katsikopoulos, and Lambros Lambrinos. Bounded rationality can increase parking search efficiency. In *Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing*, pages 195–204. ACM, 2014.

[84] Amin Kianpisheh, Norlia Mustaffa, Pakapan Limtrairut, and Pantea Keikhosrokiani. S-mart parking system (sps) architecture using ultrasonic detector. *International Journal of Software Engineering and Its Applications*, 6(3):55–58, 2012.

[85] Donnie H Kim, Younghun Kim, Deborah Estrin, and Mani B Srivastava. Sensloc: sensing everyday places and paths using less energy. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 43–56. ACM, 2010.

[86] Daehan Kwak, Daeyoung Kim, Ruilin Liu, Badri Nath, and Liviu Iftode. Doppeldriver: Counterfactual actual travel times for alternative routes. In *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*, pages 178–185. IEEE, 2015.

[87] LADoT. LA Express Park, 2017. `http://www.laexpresspark.org/`.

[88] Neal Lathia and Licia Capra. How smart is your smartcard?: measuring travel behaviours, perceptions, and incentives. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 291–300. ACM, 2011.

[89] Neal Lathia and Licia Capra. Mining mobility data to minimise travellers' spending on public transport. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1181–1189. ACM, 2011.

[90] Iván Lequerica, Miguel Garcia Longaron, and Pedro M Ruiz. Drive and share: efficient provisioning of social networks in vehicular scenarios. *Communications Magazine, IEEE*, 48(11):90–97, 2010.

[91] Feng Li, Yuan Yu, HongBin Lin, and WanLi Min. Public bus arrival time prediction based on traffic information management system. In *Service Operations, Logistics, and Informatics (SOLI), 2011 IEEE International Conference on*, pages 336–341. IEEE, 2011.

[92] Kai Li and Shuai Wang. Electric vehicle charging station deployment for minimizing construction cost. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 471–485. Springer, 2017.

[93] Yanhua Li, Jun Luo, Chi-Yin Chow, Kam-Lam Chan, Ye Ding, and Fan Zhang. Growing the charging station network for electric vehicles with trajectory data analytics. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 1376–1387. IEEE, 2015.

[94] Sejoon Lim and Daniela Rus. Stochastic distributed multi-agent planning and applications to traffic. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2873–2879. IEEE, 2012.

[95] Trista Lin, Hervé Rivano, and Frédéric Le Mouël. A survey of smart parking solutions. *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[96] Chen Liu, Ke Deng, Chaojie Li, Jianxin Li, Yanhua Li, and Jun Luo. The optimal distribution of electric-vehicle chargers across a city. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 261–270. IEEE, 2016.

[97] Junming Liu, Leilei Sun, Weiwei Chen, and Hui Xiong. Rebalancing bike sharing systems: A multi-source data smart optimization. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1005–1014. ACM, 2016.

[98] Ruilin Liu. Parking Trajectory Dataset, 2017. http://www.cs.rutgers.edu/~rl475/parkscan.html#dateset.

[99] Ruilin Liu, Hongzhang Liu, Daehan Kwak, Yong Xiang, Cristian Borcea, Badri Nath, and Liviu Iftode. Balanced traffic routing: Design, implementation, and evaluation. *Ad Hoc Networks*, 37:14–28, 2016.

[100] Ruilin Liu, Hongzhang Liu, Daehan Kwak, Yong Xiang, Cristian Borcea, Badri Nath, and Liviu Iftode. Balanced traffic routing: Design, implementation, and evaluation. *Ad Hoc Networks*, 37:14–28, 2016.

[101] Ruilin Liu, Yu Yang, Daehan Kwak, Desheng Zhang, Liviu Iftode, and Badri Nath. Your search path tells others where to park: Towards fine-grained parking availability crowdsourcing using parking decision models. *ACM Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*, 1(78), 2017.

[102] Xiliang Liu, Feng Lu, Hengcai Zhang, and Peiyuan Qiu. Estimating beijing's travel delays at intersections with floating car data. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, IWCTS '12, pages 14–19, New York, NY, USA, 2012. ACM.

[103] Yanchi Liu, Chuanren Liu, Nicholas Jing Yuan, Lian Duan, Yanjie Fu, Hui Xiong, Songhua Xu, and Junjie Wu. Exploiting heterogeneous human mobility patterns for intelligent bus routing. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 360–369. IEEE, 2014.

[104] Rongxing Lu, Xiaodong Lin, Haojin Zhu, and Xuemin Shen. Spark: a new vanet-based smart parking scheme for large parking lots. In *INFOCOM 2009, IEEE*, pages 1413–1421. IEEE, 2009.

[105] Shuo Ma, Ouri Wolfson, and Bo Xu. Updetector: Sensing parking/unparking activities using smartphones. In *Proceedings of the 7th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, pages 76–85. ACM, 2014.

[106] Shuo Ma, Yu Zheng, and Ouri Wolfson. T-share: A large-scale dynamic taxi ridesharing service. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 410–421. IEEE, 2013.

[107] Andr Malm. Mobile navigation services and devices - 6th edition. Technical report, Berg Insight, 2013.

[108] Ayub Mamandi, Saleh Yousefi, and Reza Ebrahimi Atani. Game theory-based and heuristic algorithms for parking-lot search. In *Computer Science and Software Engineering (CSSE), 2015 International Symposium on*, pages 1–8. IEEE, 2015.

[109] Elena Màrmol and Xavier Sevillano. Quickspot: a video analytics solution for on-street vacant parking spot detection. *Multimedia Tools and Applications*, 75(24):17711–17743, 2016.

[110] Karel Martens and Itzhak Benenson. Evaluating urban parking policies with agent-based model of driver parking behavior. *Transportation Research Record: Journal of the Transportation Research Board*, (2046):37–44, 2008.

[111] Suhas Mathur, Tong Jin, Nikhil Kasturirangan, Janani Chandrasekaran, Wenzhi Xue, Marco Gruteser, and Wade Trappe. Parknet: drive-by sensing of road-side parking statistics. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 123–136. ACM, 2010.

[112] Thomas Leo McCluskey, Apostolos Kotsialos, Jörg P Müller, Franziska Klügl, Omer Rana, and René Schumann. *Autonomic Road Transport Support Systems*. Springer, 2016.

[113] Fei Miao, Shuo Han, Shan Lin, John A Stankovic, Desheng Zhang, Sirajum Munir, Hua Huang, Tian He, and George J Pappas. Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach. *IEEE Transactions on Automation Science and Engineering*, 13(2):463–478, 2016.

[114] Fei Miao, Shuo Han, Shan Lin, Qian Wang, John Stankovic, Abdeltawab Hendawi, Desheng Zhang, Tian He, and George J Pappas. Data-driven robust taxi dispatch under demand uncertainties. *arXiv preprint arXiv:1603.06263*, 2016.

[115] Volodymyr Mnih and Geoffrey Hinton. Learning to detect roads in high-resolution aerial images. *Computer Vision–ECCV 2010*, pages 210–223, 2010.

[116] Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336. ACM, 2008.

[117] Marjan Momtazpour, Patrick Butler, M Shahriar Hossain, Mohammad C Bozchalui, Naren Ramakrishnan, and Ratnesh Sharma. Coordinated clustering algorithms to support charging infrastructure design for electric vehicles. In *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*, pages 126–133. ACM, 2012.

[118] Kyriacos Mouskos, Maria Boile, and Neville Anthony Parker. Technical solutions to overcrowded park and ride facilities. Technical report, New Jersey Department of Transportation, 2007.

[119] John D Murchland. Braess's paradox of traffic flow. *Transportation Research*, 4(4):391–394, 1970.

[120] Keewook Na, Yungeun Kim, and Hojung Cha. Acoustic sensor network-based parking lot surveillance system. In *European Conference on Wireless Sensor Networks*, pages 247–262. Springer, 2009.

[121] Koichi Nagaki. Evolution of in-car navigation systems. In *Handbook of Intelligent Vehicles*, pages 463–487. Springer, 2012.

[122] Takayuki Nakata and Jun-ichi Takeuchi. Mining traffic data from probe-car system for travel time prediction. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 817–822, New York, NY, USA, 2004. ACM.

[123] Anandatirtha Nandugudi, Taeyeon Ki, Carl Nuessle, and Geoffrey Challen. Pocketparker: Pocketsourcing parking lot availability. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, Ubicomp '14, pages 963–973. ACM, 2014.

[124] Sarfraz Nawaz, Christos Efstratiou, and Cecilia Mascolo. Parksense: A smartphone based sensing system for on-street parking. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 75–86. ACM, 2013.

[125] Garrett Dash Nelson and Alasdair Rae. An economic geography of the united states: from commutes to megaregions. *PloS one*, 11(11):e0166083, 2016.

[126] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 336–343. ACM, 2009.

[127] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 336–343. ACM, 2009.

[128] Andreea I Niculescu, Mei Quin Lim, Seno A Wibowo, Kheng Hui Yeo, Boon Pang Lim, Michael Popow, Dan Chia, and Rafael E Banchs. Designing ida-an intelligent driver assistant for smart city parking in singapore. In *Human-Computer Interaction*, pages 510–513. Springer, 2015.

[129] Noraimi Azlin Mohd Nordin, Norhidayah Kadir, Zati Aqmar Zaharudin, and Nor Amalina Nordin. An application of the a* algorithm on the ambulance routing. In *Humanities, Science and Engineering (CHUSER), 2011 IEEE Colloquium on*, pages 855–859. IEEE, 2011.

[130] Raymond W Novaco and Oscar I Gonzalez. Commuting and well-being. *Technology and well-being*, 3:174–4, 2009.

[131] J Pan, Mohammad A Khan, Iulian Sandu Popa, Karine Zeitouni, and Cristian Borcea. Proactive vehicle re-routing strategies for congestion avoidance. In *Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on*, pages 265–272. IEEE, 2012.

[132] Kevin Petsch, Phillip Dotzlaf, Charles Daubenspeck, Nathan Duthie, and Adam Mock. Automated parking space locator: Rsm. In *ASSE North Central Section Conference*, 2012.

[133] Dieter Pfoser, Sotiris Brakatsoulas, Petra Brosch, Martina Umlauft, Nektaria Tryfona, and Giorgos Tsironis. Dynamic travel time provision for road networks. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, page 68. ACM, 2008.

[134] Dieter Pfoser, Sotiris Brakatsoulas, Petra Brosch, Martina Umlauft, Nektaria Tryfona, and Giorgos Tsironis. Dynamic travel time provision for road networks. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '08, pages 68:1–68:4, New York, NY, USA, 2008. ACM.

[135] Gregory Pierce and Donald Shoup. Getting the prices right: an evaluation of pricing parking by demand in san francisco. *Journal of the American Planning Association*, 79(1):67–81, 2013.

[136] Reid Priedhorsky, Mikhil Masli, and Loren Terveen. Eliciting and focusing geographic volunteer work. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 61–70. ACM, 2010.

[137] Meng Qu, Hengshu Zhu, Junming Liu, Guannan Liu, and Hui Xiong. A cost-effective recommender system for taxi drivers. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 45–54. ACM, 2014.

[138] Melanie A Rapino and Alison K Fields. Mega commuters in the us: time and distance in defining the long commute using the american community survey. Technical report, 2013.

[139] G. Revathi and V.R.S. Dhulipala. Smart parking systems and sensors: A survey. In *Computing, Communication and Applications (ICCCA), 2012 International Conference on*, pages 1–5, Feb 2012.

[140] Mikko Rinne, Seppo Törmä, and D Kratinov. Mobile crowdsensing of parking space using geofencing and activity recognition. In *10th ITS European Congress, Helsinki, Finland*, pages 16–19, 2014.

[141] Jean-Paul Rodrigue. The geography of transport systems. chapter 6. Taylor & Francis, 2016.

[142] Francisca M Rojas. Transit transparency: Effective disclosure through open data. *Transparency policy project*, 2012.

[143] Tim Roughgarden. *Selfish routing and the price of anarchy*. MIT press, 2005.

[144] Andres Sanin, Conrad Sanderson, and Brian C Lovell. Shadow detection: A survey and comparative evaluation of recent methods. *Pattern recognition*, 45(4):1684–1695, 2012.

[145] Adella Santos, Nancy McGuckin, Hikari Yukiko Nakamoto, Danielle Gray, and Susan Liss. Summary of travel trends: 2009 national household travel survey. Technical report, 2011.

[146] Verena Schmid and Karl F Doerner. Ambulance location and relocation problems with time-dependent travel times. *European journal of operational research*, 207(3):1293–1303, 2010.

[147] Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520, 2014.

[148] David Schrank, Bill Eisele, Tim Lomax, and Jim Bak. 2015 urban mobility scorecard. 2015.

[149] Andrew Senior, Arun Hampapur, Ying-Li Tian, Lisa Brown, Sharath Pankanti, and Ruud Bolle. Appearance models for occlusion handling. *Image and Vision Computing*, 24(11):1233–1243, 2006.

[150] SFMTA. Sensor Independent Rate Adjustments Methodology and Implementation Plan, 2014. http://sfpark.org/wp-content/uploads/2014/05/SIRA-methodology-and-implementation-plan_2014_05-14.pdf.

[151] SFMTA. SFpark, 2017. http://www.sfpark.org/.

[152] Wenjie Sha, Daehan Kwak, Badri Nath, and Liviu Iftode. Social vehicle navigation: integrating shared driving experience into vehicle navigation. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, page 16. ACM, 2013.

[153] Jong-Ho Shin and Hong-Bae Jun. A study on smart parking guidance algorithm. *Transportation Research Part C: Emerging Technologies*, 44:299–317, 2014.

[154] Donald C Shoup. Cruising for parking. *Transport Policy*, 13(6):479–486, 2006.

[155] Sonia Soubam, Dipyaman Banerjee, Vinayak Naik, and Dipanjan Chakraborty. Bluepark: tracking parking and un-parking events in indoor garages. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, page 33. ACM, 2016.

[156] BUREAU OF TRANSPORTATION STATISTICS.

[157] Thérèse Steenberghen, Karel Dieussaert, Sven Maerivoet, and Karel Spitaels. Sustapark: an agentbased model for simulating parking search. *URISA Journal*, 24(1):63–77, 2012.

[158] Luca Talarico, Frank Meisel, and Kenneth Sörensen. Ambulance routing for disaster response with patient groups. *Computers & Operations Research*, 56:120–133, 2015.

[159] Shoupeng Tang, Tarun Rambha, Reese Hatridge, Stephen Boyles, and Avinash Unnikrishnan. Modeling parking search on a network by using stochastic shortest paths with history dependence. *Transportation Research Record: Journal of the Transportation Research Board*, (2467):73–79, 2014.

[160] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98. ACM, 2009.

[161] Tim Triplett, Rob Santos, Sandra Rosenbloom, and Brian Tefft. American driving survey: 2014–2015. 2016.

[162] Oliver Ullrich, Benjamin Brandt, Daniel Lückerath, and Naphtali Rishe. A simple model of cruising for garage parking. In *Proceedings of ASIM-Workshop STS/GMMS-ARGESIM Report*, volume 51, pages 10–11.

[163] Alje van den Bosch, Bart van Arem, Mohamed Mahmod, and James Misener. Reducing time delays on congested road networks using social navigation. In *Integrated and Sustainable Transportation System (FISTS), 2011 IEEE Forum on*, pages 26–31. IEEE, 2011.

[164] Peter van der Waerden, Aloys Borgers, and Harry Timmermans. Travelers microbehavior at parking lots: a model of parking choice behavior. In *Processing of the 82nd Annual Meeting of the Transportation Research Board*, volume 1212, 2003.

[165] Félix Jesús Villanueva, David Villa, Maria José Santofimia, Jesus Barba, and Juan Carlos Lopez. Crowdsensing smart city parking monitoring. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, pages 751–756. IEEE, 2015.

[166] Yibing Wang and Markos Papageorgiou. Real-time freeway traffic state estimation based on extended kalman filter: a general approach. *Transportation Research Part B: Methodological*, 39(2):141–167, 2005.

[167] Yilun Wang, Yu Zheng, and Yexiang Xue. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 25–34. ACM, 2014.

[168] Felix Wex, Guido Schryen, Stefan Feuerriegel, and Dirk Neumann. Emergency response in natural disaster management: Allocation and scheduling of rescue units. *European Journal of Operational Research*, 235(3):697–708, 2014.

[169] David Wilkie, Jur P van den Berg, Ming C Lin, and Dinesh Manocha. Self-aware traffic route planning. In *AAAI*, 2011.

[170] Bo Xu, Ouri Wolfson, Jie Yang, Leon Stenneth, S Yu Philip, and Peter C Nelson. Real-time street parking availability estimation. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 1, pages 16–25. IEEE, 2013.

[171] Longlong Xu, Wutao Lin, Xiaorong Wang, Zhenhui Xu, Wei Chen, and Tengjiao Wang. Chargemap: An electric vehicle charging station planning system. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data*, pages 337–340. Springer, 2017.

[172] T. Yamashita, K. Izumi, and K. Kurumatani. Car navigation with route information sharing for improvement of traffic efficiency. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 465–470, Oct 2004.

[173] Gongjun Yan, Weiming Yang, Danda B Rawat, and Stephan Olariu. Smartparking: A secure and intelligent parking system. *IEEE Intelligent Transportation Systems Magazine*, 3(1):18–30, 2011.

[174] Bin Yang, Chenjuan Guo, and Christian S. Jensen. Travel cost inference from sparse, spatio temporally correlated time series using markov models. *Proc. VLDB Endow.*, 6(9):769–780, July 2013.

[175] Jie Yang, Jing Dong, and Liang Hu. A data-driven optimization-based approach for siting and sizing of electric taxi charging stations. *Transportation Research Part C: Emerging Technologies*, 77:462–477, 2017.

[176] Zidong Yang, Ji Hu, Yuanchao Shu, Peng Cheng, Jiming Chen, and Thomas Moscibroda. Mobility modeling and prediction in bike-sharing systems. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 165–178. ACM, 2016.

[177] Song Ying and Yu Yang. Study on vehicle navigation system with real-time traffic information. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 4, pages 1079–1082. IEEE, 2008.

[178] Jungkeun Yoon, Brian Noble, and Mingyan Liu. Surface street traffic estimation. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 220–232. ACM, 2007.

[179] William Young. A model of vehicles movements in parking facilities. *Mathematics and computers in simulation*, 28(4):305–309, 1986.

[180] Jiangye Yuan. Automatic building extraction in aerial scenes using convolutional networks. *arXiv preprint arXiv:1602.06564*, 2016.

[181] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, pages 99–108. ACM, 2010.

[182] Jing Yuan, Yu Zheng, Liuhang Zhang, XIng Xie, and Guangzhong Sun. Where to find my next passenger. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 109–118. ACM, 2011.

[183] Nicholas Jing Yuan, Yu Zheng, Liuhang Zhang, and Xing Xie. T-finder: A recommender system for finding passengers and vacant taxis. *IEEE Transactions on Knowledge and Data Engineering*, 25(10):2390–2403, 2013.

[184] Yufei Yuan, Hans Van Lint, Femke Van Wageningen-Kessels, and Serge Hoogendoorn. Network-wide traffic state estimation using loop detector and floating car data. *Journal of Intelligent Transportation Systems*, 18(1):41–50, 2014.

[185] Yang Yue, Hai-xiang Zou, and Qing-quan Li. Urban road travel speed estimation based on low sampling floating car data. In *Proceedings of the 9th International Conference of Chinese Transportation Professionals (ASCE), Harbin, China*, pages 1719–1725, 2009.

[186] Desheng Zhang and Tian He. pcruise: Reducing cruising miles for taxicab networks. In *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd*, pages 85–94. IEEE, 2012.

[187] Desheng Zhang, Tian He, Yunhuai Liu, and John A Stankovic. Callcab: A unified recommendation system for carpooling and regular taxicab services. In *Big Data, 2013 IEEE International Conference on*, pages 439–447. IEEE, 2013.

[188] Desheng Zhang, Ye Li, Fan Zhang, Mingming Lu, Yunhuai Liu, and Tian He. coride: Carpool service with a win-win fare model for large-scale taxicab networks. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 9. ACM, 2013.

[189] Jiawei Zhang, Xiao Pan, Moyin Li, and Philip S Yu. Bicycle-sharing systems expansion: station re-deployment through crowd planning. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 2. ACM, 2016.

[190] Mingmin Zhao, Tao Ye, Ruipeng Gao, Fan Ye, Yizhou Wang, and Guojie Luo. Vetrack: Real time vehicle tracking in uninstrumented indoor environments. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 99–112. ACM, 2015.

[191] Wan Chen Zhao. Parking management system with rfid and sensor networks. In *Applied Mechanics and Materials*, volume 198, pages 1690–1696. Trans Tech Publ, 2012.

[192] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):38, 2014.

[193] Pengfei Zhou, Shiqi Jiang, and Mo Li. Urban traffic monitoring with the help of bus riders. In *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*, pages 21–30. IEEE, 2015.

[194] Pengfei Zhou, Yuanqing Zheng, and Mo Li. How long to wait?: predicting bus arrival time with mobile phone based participatory sensing. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 379–392. ACM, 2012.

[195] Ning Zhu, Shoufeng Ma, and Liang Zheng. Travel time estimation oriented freeway sensor placement problem considering sensor failure. *Journal of Intelligent Transportation Systems*, 2017.

[196] John Zimmerman, Anthony Tomasic, Charles Garrod, Daisy Yoo, Chaya Hiruncharoenvate, Rafae Aziz, Nikhil Ravi Thiruvengadam, Yun Huang, and Aaron Steinfeld. Field trial of tiramisu: crowd-sourcing bus arrival times to spur co-design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1677–1686. ACM, 2011.

[197] Onno Zoeter, Christopher Dance, Stéphane Clinchant, and Jean-Marc Andreoli. New algorithms for parking demand management and a city-scale deployment. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1819–1828, New York, NY, USA, 2014. ACM.