A Data-Driven Cyber-Physical System for Urban Mobility Modeling and Their Applications

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Desheng Zhang

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Prof. Tian He

September, 2015

Acknowledgements

Over the last five years, I have had the privilege to collaborate with a number of talented people who have made my Ph.D career at the University of Minnesota extremely rewarding. I would like to thank all of them. First of all, I am grateful to my advisor Prof. Tian He at Computer Science and Engineering Department. Tian is an excellent computer scientist with broad and deep knowledge about systems research. Tian also has a great vision and sharp intuition, which makes him a great mentor. He taught me everything I need to know for a successful Ph.D career. Tian is very patient and encourages me to explore new research directions with great freedom. Without him, this dissertation would not be possible. I am also thankful for the time and advice from Prof. John A. Stankovic, Prof. Shan Lin and Prof. Fan Ye who generously helped me and strongly supported my Ph.D career during our joint research projects. I would like to thank Dr. Hui Lei, Dr. Yu Gu, Dr. Fan Zhang, and Dr. Raghu K. Ganti for their generous supports. Special thanks to my thesis committee members, Prof. Anand Tripathi, Prof. Volkan Isler, and Prof. John Carlsson. I am very grateful for all my coauthors, labmates and colleagues, including Songmin, Shuai, Shi, Yongle, Zhimeng, Wenchao, Ye, Jun, and Juanjuan with whom I have shared hours of discussion, work and laughter. Furthermore, none of this would have been possible without the unwavering support from my parents. Most importantly, my dearest wife, Xiaolin, has shared with me all the sweets and bitters of life since 12 years ago, and has never failed to believe in me. I feel exceptionally favored to have you in my life. Last but not least, I gratefully acknowledge financial support from ADC Fellowship, Doctoral Dissertation Fellowship, Excellent Thesis Research Award, Chinese Government Award, and other travel grants from University of Minnesota, China Scholarship Council, and National Science Foundation.

Dedication

To my parents and Xiaolin.

Abstract

Nowadays, we are in a rapid process of urbanization, which leads to severe mobility challenges, e.g., traffic congestion and gas consumption. To address these challenges, it is essential to model human mobility, and improve urban mobility efficiency with novel applications based on mobility models. Existing human mobility models and resultant applications are mostly driven by data from isolated urban systems, e.g., cellphone networks or transportation systems, which leads to a bias against urban residents not involved and thus inefficiency of resultant applications. In this dissertation, we propose a cyber-physical system called mobileCPS to model the human mobility at fine spatiotemporal granularity and then design novel mobility-driven applications. Specifically, we design a three layer architecture for mobileCPS: (i) a real-time data feed layer where we collect multi-source urban data related mobility from extremely-large urban infrastructures, e.g., cellphone networks and transportation systems, which is one of the largest urban data consolidations for academic research; (ii) a mobility abstraction layer where we design a human mobility model driven by multi-source data we collected with a multi-view learning technique, which is the first work that models human mobility with multi-source data; (iii) an application design layer where we present two mobility driven applications, i.e., a real-time carpooling service called coRide and last-mile transit service called Feeder, to improve urban mobility efficiency. coRide is the first systemic carpooling service with real-world implementation and a dynamic fare model, and Feeder is the first last-mile transit service driven by multi-source urban data. The key intellectual contributions of this work include (i) a human mobility modeling technique iteratively driven by heterogenous multi-source urban data; (ii) a set of optimal, approximation and online algorithms for a mobility-driven carpooling problem; (iii) a data-driven inference technique for last-mile passenger demand. We implement and evaluate mobileCPS based on extremely large datasets in the Chinese city Shenzhen with cellphone and transportation systems including taxis, buses, and subways, capturing more than 27 thousand vehicles and 10 million urban residents. The results show that mobileCPS (i) increases mobility model accuracy by 51%, (ii) reduces mileage by 33% with its carpooling, and (iii) reduces the last mile distance by 68%.

Contents

List of Figures

Chapter 1

Introduction

Cyber-physical systems (CPS) refer to a set of systems featuring coupling between the cyber intelligence and the physical world. With the increasing popularity of could computing, CPS are merging into major systems of our society, e.g., smart grids, medical devices, manufacturing, transportation, and telecommunication [1]. Enabled by the ubiquitous availability of communication, computation, and control capabilities, CPS are envisioned to redefine the way that people interact with the physical world. Researchers have accumulated abundant knowledge for designing CPS for various applications, such as military surveillance, infrastructure protection, scientific exploration, and smart environments, mostly in relatively stationary settings. However, CPS involved mobile elements received little attention from research community. CPS in mobile setting interact with phenomena of interest at different locations and environments, and where spatiotemporal context information are constantly changing. This unique feature calls for new solutions to seamlessly integrate mobile computing with physical-world processes by sharing information among a set of networked CPS.

In this dissertation, we investigate CPS related to mobile urban systems, e.g., cellphone networks, bus networks, subway networks, and taxi networks, to improve urban mobility efficiency. Nowadays, we are in a rapid process of urbanization where more than half of people in the world has moved to urban areas [2]. Such urbanization leads to several sustainability issues, e.g., traffic congestion and gas consumption. To ensure urban sustainability, how to capture human mobility at urban scale is one of the fundamental challenges we need to address. Such human mobility has many real-world applications, e.g., transportation, urban planning, social networking, and location based services [3]. To capture generic human mobility patterns, several theoretical models have been proposed, e.g., the gravity model and the radiation model [4], along with corresponding applications [2]. However, a key drawback of these theoretical models is that they cannot capture human mobility at fine spatiotemporal granularity, e.g., mobility at road segment levels in real time. Therefore, applications based on these models are usually inefficient and unpractical in real-world setting.

Recently, thanks to upgrades of urban infrastructure systems, many real-time location-tracking devices become available, e.g., cellphones, onboard GPS devices and smartcard-s. These devices generate massive real-time mobility data, which hold the key potential to revolutionize real-time human mobility modeling and their applications. Specifically, our work is motivated by two important observations based on the upgrades of urban systems: (i) in addition to macro-level historical statistics, the availability of massive *micro-level* mobility data makes it possible to model human mobility in fine spatiotemporal granularity for novel application design; (ii) integrated information from multi-source mobile urban systems allows more accurate modeling and drives novel applications.

In this dissertation, we propose a cyber-physical system called mobileCPS, which captures real-time human mobility based on multi-source urban system data in order to design real-world applications to address urban mobility issues during urbanization. Interacting with large-scale mobile urban systems in real time, mobileCPS has a three-layer architecture as follows: (i) a real-time data feed layer where mobileCPS collects multi-source mobility data from mobile urban systems, e.g., cellphone networks, taxi networks, bus networks, and subway networks; (ii) a mobility abstraction layer where mobileCPS integrates mobility models driven by the collected multi-source data for a comprehensive rendering of human mobility at urban scale in real time; (iii) an application design layer where mobileCPS enables novel urban applications by the integrated urban mobility model to improve urban mobility efficiency. Most importantly, we implement mobileCPS based on extremely large datasets in the Chinese city Shenzhen with cellphone data and transportation data including taxis, buses, and subways. In particular, the key contributions of the dissertation are as follows:

- To our knowledge, we propose the first cyber-physical system, mobileCPS for human mobility modeling and their applications based on real-time multi-source data from extremely-large urban infrastructures. mobileCPS is the first systematic closed-loop CPS that explores correlated urban systems and their real-time data to design mobility models, which are used to drive practical applications to improve urban systems themselves. Most importantly, we implement our mobileCPS with three layers based on four urban systems, i.e., cellphone, taxi, bus, and subway in the Chinese city Shenzhen, with 10 million cellphone users and 16 million smartcard users involved. To our knowledge, this is one of the largest systems for human mobility modeling driven by real-world datasets.

- In the real-time data feed layer of mobileCPS, we integrate four data feeds from cellphone, taxi, bus and subway systems, and explore their spatiotemporal granularity in terms of human mobility modeling. We present our data maintenance process, including data access, data cleaning, and privacy protection. We release our data for the benefit of the research community, which is the first released multi-source large-scale urban data.

- In the mobility abstraction layer of mobileCPS, we propose a multi-view learning technique to design a human mobility model call coMobile to integrate incomplete yet complementary knowledge from individual urban systems. To our knowledge, the proposed model is the only human mobility model driven by more than one view, which aims to address over-fitting of single view models. It is challenging to apply multi-view learning in human mobility modeling, because data-driven views are mostly incomplete to urban-scale mobility. In this work, (i) we design a single-view learning technique based on context-aware tensor decomposition with both real-time and historical data to improve completeness of single-view mobility models; (ii) we formulate a multi-view modeling problem based on improved single-view models with a joint optimization, which minimizes overall weighted deviation from observed mobility to the ground truth; (iii) we propose an iterative learning process to solve this optimization by alternatively updating the ground truth and view completeness until no further improvement can be made for the objective function. We evaluate coMobile based on an extremely large dataset in the Chinese city Shenzhen, including data about taxi, bus and subway passengers along with cellphone users, capturing more than 27 thousand vehicles and 10 million urban residents. The evaluation results show that coMobile outperforms a single-view model by 51% on average.

- In the application design layer of mobileCPS, we propose the first systematic work to design, implement, and evaluate a carpool service, called coRide, for large-scale taxicab networks intended to reduce total mileage for less gas consumption. Our coRide consists of three components: a dispatching cloud server, passenger clients, and onboard customized devices, called TaxiBox. In the coRide design, in response to the delivery requests of passengers, dispatching cloud servers calculate cost-efficient carpool routes for taxicab drivers and thus lower fares for the individual passengers. To improve coRide's efficiency in mileage reduction, we formulate a NP-hard route calculation problem under different practical constraints. We then provide (i) an optimal algorithm using Linear Programming, (ii) a 2 approximation algorithm with a polynomial complexity, and (iii) its corresponding online version with a linear complexity. To encourage coRide's adoption, we present a win-win fare model as the incentive mechanism for passengers and drivers to participate. We test the performance of coRide by a comprehensive evaluation with a real-world trial implementation and a data-driven simulation with 14,000 taxi data from the Chinese city Shenzhen. The results show that compared with the ground truth, our service reduces 33% of total mileage; with our win-win fare model, we lower passenger fares by 49% and simultaneously increase driver profit by 76%.

- In the application design layer of mobileCPS, we propose another application Feeder, which is a transit service to tackle the last-mile problem, i.e., passengers' destinations lay beyond a walking distance from a public transit station. Feeder utilizes ridesharing-based vehicles (e.g., minibus) to deliver passengers from existing transit stations to selected stops closer to their destinations. We infer real-time passenger demand (e.g., exiting stations and times) for Feeder design by utilizing extreme-scale urban infrastructures, which consist of 10 million cellphones, 27 thousand vehicles, and 17 thousand smartcard readers for 16 million smartcards in a Chinese city Shenzhen. Regarding these numerous devices as pervasive sensors, we mine both online and offline data for a two-end Feeder service: a back-end

Feeder server to calculate service schedules; front-end customized Feeder devices in vehicles for real-time schedule downloading. The key novelty of Feeder is that it is transparent to passengers by utilizing historical and real-time streaming data from extremely large urban infrastructures. We implement Feeder using a fleet of vehicles with customized hardware in a subway station of Shenzhen by collecting data for 30 days. The evaluation results show that compared to the ground truth, Feeder reduces last-mile distances by 68% and travel time by 52% on average.

We organize the dissertation as follows. Chapter 3 gives the architecture of mobileCPS and its the real-time data feed layer. Chapter 2 introduces related work. Chapter 4 presents the our human mobility model coMobile on the mobility abstraction layer. For the application design layer, Chapter 5 describes our first mobility-driven application coRide for real-time carpooling, and Chapter 6 introduces our second mobility-driven application Feeder for last-mile transit. Chapter 7 presents our future work. Chapter 8 concludes this dissertation.

Chapter 2

Related Work

Analyzing the human mobility in urban scales has many real world applications, e.g., urban planning [2], transportation [5] and social networks [6]. In this chapter, we briefly introduce three types of work related to this dissertation, i.e., mobility modeling, real-time carpooling services and last-mile transit services.

2.1 Mobility Modeling

Modeling the human mobility in urban scales is crucial for mobile applications, urban planning and social networks [2]. However, almost all existing models are driven by single views. We made the first attempt to model the human mobility with multi-source data [7], but our previous work was to use transportation data to adjust the modeling process based on cellphone data, and did not treat these two kinds of data equally as two views. As follows, we summarize the related work by different views.

2.1.1 Cellphone View

Modeling from the cellphone view based on call detail records (CDR) is the most common method, e.g., modeling how residents move around the cities [8]; estimating cellphone users' travel range [9]; predicting where cellphone users will travel next [10]. However, the models from cellphone views are mostly biased against a certain group of residents, leading to inaccurate analyses. To our knowledge, we are the first to combine data from more than one carrier to model the human mobility.

2.1.2 Transportation View

Transportation data are another important data source for human mobility, e.g., bus data [11], subway data [12], taxicab data [13], and private vehicle data [14]. However, the models driven by data from one kind of transportation are mostly biased against the passengers using other transportation. To our knowledge, there is no model driven by more than one transportation mode, and we are the first to combine data from three kinds of transportation for mobility modeling.

2.1.3 Other Views

Other data generated by urban residents have also been used to study human mobility, e.g., social networks or mobile *ad hoc* networks, i.e., with check-in data [6] and proximity data [15]. However, the number of residents captured by these views is often extremely limited compared to the cellphone data and transportation data, which leads to a bias that cannot be quantified.

2.1.4 Summary

In short, almost all human mobility modeling is based on single views, which are often incomplete in terms of capturing the human mobility at urban scale in real time. Such a shortcoming motivates us to take a multi-view approach, which uses incomplete yet complementary views to model the human mobility. As a result, the fundamental difference between related work and our mobility modeling is that our model is driven by multi-source real-time urban data from heterogenous urban systems.

2.2 Real-time Carpooling

The premise of taxicab carpooling is not new, but in real world it is normally negotiated privately by drivers and passengers in an *ad hoc* manner. We lack a systematic design to balance benefits of all the involved parties, e.g., drivers, passengers, and taxicab operators. Two types of previous work are directly related to our carpooling work: existing carpooling applications and taxicab system applications.

2.2.1 Existing Services for Carpooling

Limited *ad hoc* taxicab carpools exist in both developed and developing countries. For example, in New York City, up to four passengers can carpool together in a single taxicab ride during 6 AM to 10 AM on a weekday, along three preset routes in Manhattan at a flat fare of $3 or $4 per passenger, significantly less than the regular metered rates [16]. In Beijing, *ad hoc* taxicab carpooling is also allowed with the consent of both passengers and drivers, and every passenger pays 60% of the regular fare. Further, some door to door shuttle services are also available in major airports, and can enable shared rides to or from airports [17]. However, in the aforementioned carpool services, both time and locations are preset and the services are arranged on the spot by passengers or drivers in a small-scale *ad hoc* manner, and no infrastructure is provided to improve the efficiency of carpooling.

Several papers have been proposed after our conference paper [18] to explore carpool services, e.g., real-time carpooling [19], slugging form of carpooling [20], and social effects of carpooling [21]. Uber recently proposes a new service called Uberpool [22], which uses real-time passenger requests to group several passengers together with similar origin and destination with a heuristic solution. In contrast, our work is from system aspects to explore carpool issues with a hardware-software co-design with implementation.

There are several pieces of theoretical work about carpools for private cars. The carpools for private cars is with regular passengers and fixed routes to or from work, which leads to a private classic carpool problem where one has to assign drivers to subsets of carpool participants to reduce the detour of drivers [23]. It is different from our carpool where the drivers do not be considered. Some work has been proposed to find the opportunities for carpool by finding the shared common routes by tracking personal mobile devices [24] [25], but the whole system architecture and how to calculate the carpool route is not given. Fagin *et al.* [23] first present a simple carpool scheduling algorithm in which no penalty is assessed to a carpool member who does not ride on any given day. The proposed algorithm, the FW share, is shown to be fair, in a certain reasonable sense. Naor *et al.* [26] provide an axiomatic characterization of the fair share and indicates that the FW share is the unique one satisfying these requirements. Recently, Coppersmith *et al.* [27] show that the greedy algorithm is optimal among online algorithms for the private carpool problem. Different from the classic carpool problem, our service focuses on how to pool passengers in different taxicabs to optimize the total mileage. In our service, passengers do not have a regular and fixed route, and drivers are not like the passengers needed to be considered.

2.2.2 Taxi System Applications

The increasing availability of GPS devices has encouraged a surge of research intended to improve the efficiency of large-scale taxicab networks. First, several systems are proposed for the benefit of passengers or drivers, e.g., allowing passengers to query the expected duration and fare of a planed taxicab trip based on the history of previous trips [28] and query real-time taxicab availability to make informed transportation choices [29], as well as recommending optimal pickup locations or routes [30] [31] [32]. Second, taxicab traces can also help taxicab network operators better oversee taxicabs and provide efficient service to passengers, e.g., discovering spatial and temporal causal interactions to provide timely and efficient service in certain areas with disequilibrium [33] [34], and detecting anomalous taxicab trips to discover driver fraud or road network changes [35]. Third, traces from experienced taxicab drivers can help other drivers improve their driving performance, e.g., navigating newer drivers to smart routes based on those of experienced taxicab drivers [36], and assisting other drivers to improve their driving performance with GPS records from experienced taxicab drivers [37]. Fourth, large scale taxicab traces enable us to better understand traffic conditions of cities, e.g., semantics of origin-destination flows [38], traffic congestion and volumes [5], and traffic patterns between regions with different functions [39]. Finally, large scale traces also can help in city planning, e.g., detecting flawed urban planning [40] or improving map inference [41] [42].

Yet existing research on taxicab systems focuses on scheduling individual taxicabs, assuming that one taxicab can accommodate only a single delivery request at a time. In contrast, our carpooling service allows shared delivery. Technically, we focus on carpool route calculation and a win-win fare model, neither of which has been investigated before.

2.3 Last-mile Transit

The last-mile problem is how to deliver passengers from existing urban transit stations to their final destinations. To tackle the last-mile problem, several services have been proposed as last-mile transit with different focuses. Two types of the work are related to our last-mile transit services: existing services for last-mile transit and vehicular system applications.

2.3.1 Existing Services for Last-Mile Transit

In addition to obvious solution, e.g., walking, bikes, taxicabs, and personal vehicles, taxicab ridersharing and minibus services are two major efforts for the last-mile problem. Some cities, e.g., New York City [16], Beijing [19] and Shenzhen [43], introduce taxicab ridersharing services for passengers to share taxicabs for *ad hoc* rides, but both time and locations are preset and no infrastructure support is provided. Some cities, e.g., Hong Kong [44], use minibuses to deliver passengers closer to their destinations, but they have fixed routes and schedules.

The key difference between our work and ridersharing is that it learns passenger demand automatically, while ridersharing assumes demand is given in advance. Our work is also different from the above services in terms of low infrastructure costs, flexible network coverages, and real-time supports from our server with online data from urban infrastructures.

2.3.2 Transit System Applications

Another type of related work is urban data-driven vehicular applications [45] [46] [47] [42]. Many novel applications are proposed to assist urban residents or city officials, e.g., assisting mobile users to make transportation decisions, such as taking a taxicab or not [29], finding parking spots for drivers [48], inferring real-world maps based on GPS data [41], predicting bus arrival times [49], enabling passengers to query taxicab availability to make informed transit choices [28], informing drivers with smart routes based on those of experienced drivers [36], predicting passenger demand for taxicab drivers [31], recommending optimal pickup locations [30], modeling the urban transit [45], suggesting profitable locations for taxicab drivers by constructing a profitability map where the nearby regions of drivers are scored serving as a metric for a taxicab driver decision making process [50], detecting the taxicab anomaly [51], navigating new drivers based on GPS traces of experienced drivers [37], and enabling us to better understand region functions of cities [39].

Yet existing research on these systems has not focused on the last-mile problem, and typically utilizes only one type of datasets. But Feeder utilizes streaming data from several urban infrastructures to tackle the last-mile problem without the burden on the passenger side. Such a unique combination has not been investigated before.

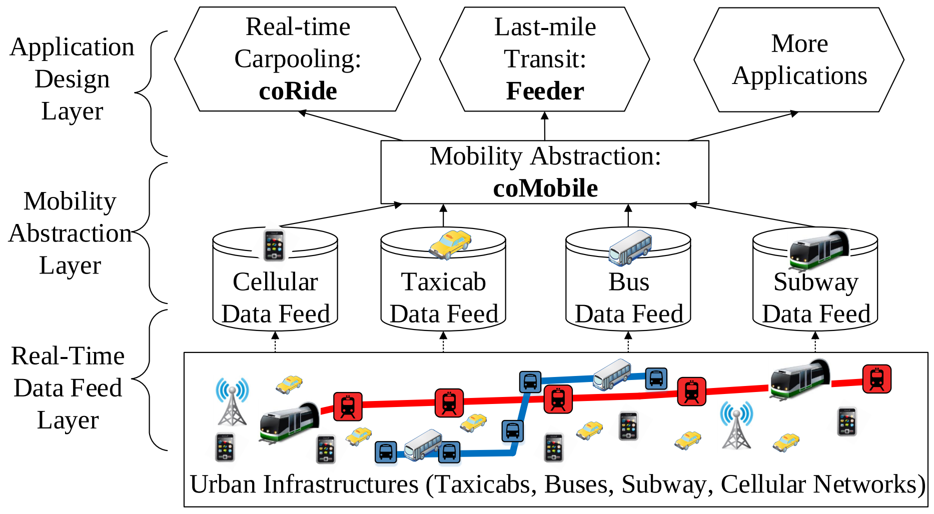
Chapter 3

mobileCPS Architecture

In this chapter, we first introduce the architecture of mobileCPS, and then introduce data management.

3.1 Architecture

As in Figure 3.1, we present mobileCPS's architecture, which consists of three layers.



Application Design Layer
Real-time Carpooling: **coRide**
Last-mile Transit: **Feeder**
More Applications
Mobility Abstraction Layer
Mobility Abstraction: **coMobile**
Cellular Data Feed
Taxicab Data Feed
Bus Data Feed
Subway Data Feed
Real-Time Data Feed Layer
Urban Infrastructures (Taxicabs, Buses, Subway, Cellular Networks)

Figure 3.1: MobileCPS Architecture

- **Real-Time Data Feed Layer** ensures a secure and reliable feeding mechanism to establish multi-source data feeds through urban infrastructures in a privacy-preserving method. At a macro level, mobileCPS establishes the data feed for anonymous cellphones in the cellular networks; at a micro level, mobileCPS establishes the data feeds in the transit networks including taxicabs, buses and subways. The details are given in the later part of this chapter.
- **Mobility Abstraction Layer** transparentizes heterogeneous features in our multi-source data to enable an effective mobility abstraction with mobility information. As a result, mobileCPS enables novel mobility modeling by integrating individual mobility models based on single-source data. The details are given in Chapter 4.
- **Application Design Layer** bridges our mobility model to real-world applications to improve urban efficiency, e.g., increasing urban transit ridership and reducing travel and waiting time for urban residents by novel transit services. In this dissertation, we design two mobility-driven transit services, i.e., coRide for carpooling and Feeder for last-mile transit. The details are given in Chapters 5 and 6, respectively.

Similar to the IP layer, i.e., the narrow waist of the Internet, the mobility abstraction layer essentially serves as the narrow waist of mPat, allowing a separation between data feeds and applications. Based on the real-time input from the data feeds, the mobility abstraction provides appropriate service interfaces for accurate rendering of human mobility, which are utilized by the applications to improve performance. coMobile's three-layer architecture suggests a horizontal view of building high-performance applications based on correlated mobile systems, but traditional stand-alone CPS, e.g., systems based on cellular networks or transportation networks, do not have such capacities. The narrow waist allows fellow researchers to add more transit modes (e.g., bicycles) or applications without redesigning the whole architecture.

3.2 Data Management

In this section, we first introduce the Real-Time Data Feed Layer in terms of data feeds, storage and management.

3.2.1 Urban Data Feeds

We have been collaborating with several Shenzhen government agencies and service providers, and establishing a reliable feeding mechanism that feeds mobileCPS various data collected within Shenzhen infrastructures without impacting data sources in service providers. For security and efficiency purposes, such a mechanism is facilitated with a commercial solution called Shenzhen Transmission Standard, which is customized for Shenzhen infrastructures and operates with high performance and low overhead, even at high volume of streaming data, e.g., vehicular GPS. This mechanism enables continuous capture and delivery of data from service providers to coMobile's data feed layer with end-to-end sub-second latency. Further, it has the ability to provide the data in a variety of formats, e.g., binary or text files, enabling direct real-time analyses. Since the data are already being collected to help the service providers to operate their services, our feeding mechanism incurs little marginal cost. We briefly introduce the established feeds in the layer as follows.

- **Cellphone Data Feed** is established for 10.4 million users in Shenzhen. The total records of data (including call detail records [CDR] among 17859 cell towers) are more than 5 million per day.
- **Taxicab Data Feed** is established through Shenzhen Transport Committee, to which all taxicab companies upload their taxicab status (GPS and occupancy) in real time by a cellular network used by all taxicabs in Shenzhen. The temporal granularity for this feed is extremely high, i.e., the uploading period is less than 30s. The daily size of all taxicab status data is 2 GB.
- **Subway Data Feed** is established by streaming entering and exiting records in smart card transactions. Such a feed accounts for more than 16 million smart cards, leading to 10 million daily transactions. A total of 2,570 fixed smartcard readers in 127 subway stations capturing 60 thousand subway passengers per hour.
- **Bus Data Feed** consists of two parts: a GPS feed for all buses in real time (2 GB per day), and a transaction record feed from 16 million smartcards. A total of 14,270 onboard smartcard readers in 13 thousand buses capturing 168 thousand bus passengers per hour.

The heat map of their spatial granularity is given by Figure 3.2 with an area of 14×5 km^2. The lighter the icon, the high the passengers captured by the data feeder.

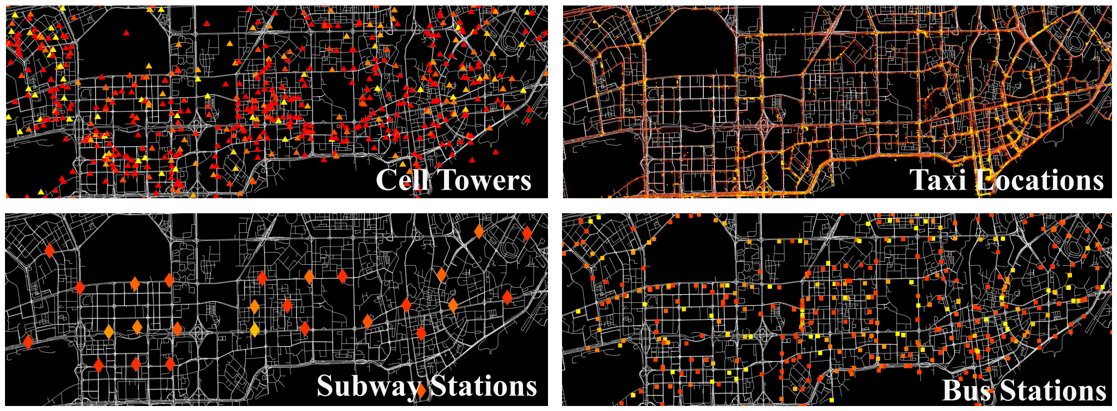
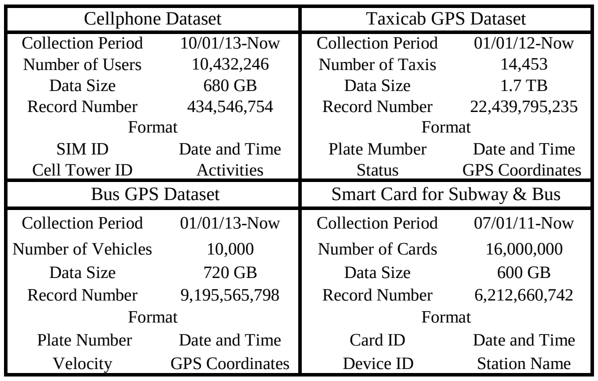


Figure 3.2: Data Spatial Granularity

Our endeavor to consolidate the above feeds enables an extremely fine-grained mobility tracking that is unprecedented in terms of both quantity and quality. To facilitate mobility analyses based on real-time and historical data, we have stored the data from these feeds as in Figure 3.3. Due to large sizes of location updates (90 GB per day) in the cellphone feed, we only store the activity data.



Cellphone Dataset		Taxicab GPS Dataset	
Collection Period	10/01/13-Now	Collection Period	01/01/12-Now
Number of Users	10,432,246	Number of Taxis	14,453
Data Size	680 GB	Data Size	1.7 TB
Record Number	434,546,754	Record Number	22,439,795,235
Format		Format	
SIM ID	Date and Time	Plate Mumber	Date and Time
Cell Tower ID	Activities	Status	GPS Coordinates
Bus GPS Dataset		**Smart Card for Subway & Bus**	
Collection Period	01/01/13-Now	Collection Period	07/01/11-Now
Number of Vehicles	10,000	Number of Cards	16,000,000
Data Size	720 GB	Data Size	600 GB
Record Number	9,195,565,798	Record Number	6,212,660,742
Format		Format	
Plate Number	Date and Time	Card ID	Date and Time
Velocity	GPS Coordinates	Device ID	Station Name

Figure 3.3: Datasets from Real-Time Feeds

3.2.2 Data Storage and Maintenance

Such big amounts of mobility data require significant efforts for the efficient storage and daily maintenance. We utilize a 34 TB Hadoop Distributed File System (HDFS) on a cluster consisting of 11 nodes, each of which is equipped with 32 cores and 32 GB RAM. For daily maintenance, we use the MapReduce based Pig and Hive. Pig is a high-level data-flow execution framework for parallel computation and Hive is a data warehouse infrastructure for data summarization and *ad hoc* querying.

Due to the extremely large size of our data, we found three main kinds of errant data. (i) Missing Data: e.g., a taxicab's GPS data were not uploaded within a given time period. Such missing data are detected by monitoring the temporal consistence of incoming data for every data source, e.g., a taxicab. (ii) Duplicated Data: e.g., the smart card datasets show two identical records for the same smart card. Such duplicated data are detected by comparing the timestamp of every record belonging to the same data source, e.g., the same smart card. (iii) Data with Logical Errors: e.g., GPS coordinates show that a vehicle is off the road. Such data with logical errors are detected later when we analyze the data. To detect these errors, we utilize a digital map of Shenzhen to verify if a GPS location is plausible or not. This is performed by checking the previous location and the duration between the timestamps of these two records. The above errors may result from various reasons, e.g., hardware malfunctions, software issues, and communications.

To address the above errors, for all incoming data, we first filter out the duplicated records and the records with missing or errant attributes. Then we correct the obvious numerical errors by various known contexts. We next store the data by dates and categories. Finally we compare the temporal consistence of the data to detect the missing records. Admittedly, the missing or filtered out data (which accounted for 11% of the total data) may impact the performance of our later analyses, but given the long time period, we believe we are still be able to provide insightful analyses as follows.

While the streaming data for the human mobility study have the potential for great social benefits, we have to protect the privacy of the residents involved for wider applications. We took four active steps for privacy protections. (i) Anonymization: All data analyzed are anonymized by the service providers who were not involved in this project, and all identifiable IDs, such as SIM card IDs, are replaced by a serial identifier during the analyses. (ii) Minimal Exposure: We only store and process the information which are useful for our mobility analysis, and drop other information for the minimal exposure, e.g., we store the cell tower IDs to infer locations in the cellphone data, but not durations of calls. (iii) Aggregation: The mobility patterns obtained by coMobile are given at aggregated results with a mobility graph in a large spatiotemporal partition. We do not focus on individual residents during the analyses. (iv) Nature of Data: The nature of different data also provides a certain level of privacy protections, e.g., the taxicab and bus GPS data do not involve any identity about passengers; the smartcard data only show passengers' locations at transit station levels.

Chapter 4

coMobile: Human Mobility Modeling

In this chapter, as the mobility abstraction layer of MobileCPS, we present our human mobility model called coMobile based on a generic human mobility modeling technique.

4.1 Introduction

Human mobility is of great importance for various urban applications, e.g., urban planning, transportation, social networking, and location based services [3]. Recently, thanks to upgrades of urban infrastructures, many real-time location-tracking devices become available, e.g., cellphones, onboard GPS devices and smartcards. These devices generate massive real-time location data, which hold the key potential to revolutionize real-time human mobility modeling. Based on these real-time data, several data-driven models have been proposed, e.g., driven by data from cellphones [52], smartcards [53], taxis [13], buses [11], or subways [14]. However, a common feature of these models is that they capture mobility only from one view, e.g., a cellphone view or a transportation view. These single-view models are sufficient if single-view data are complete, but in reality this is not the case. From the cellphone view, the models driven by cellphone data cannot capture residents without cellphone data, e.g., residents who do not have cellphones and residents who have cellphones but do not use their cellphones during our modeling time; similarly, from the transportation view, the models driven by one kind of transportation data, e.g., taxi, cannot capture the passengers who use other transportation modes, e.g., bus and subway, and further there is no urban infrastructure that can capture private vehicles at urban scale. To our knowledge, no data-driven urban human mobility models are based on a complete view so far. As a result, these single-view human mobility models essentially use residents captured by these single views as a sample to study all residents, which inevitably leads to a bias and thus over-fitting of their models, as shown in Section 4.2.

To address this issue, we aim to combine different views for multi-view modeling. Each view is incomplete to capture mobility by itself, but one view is often complementary to others, e.g., the cellphone view can capture some private-vehicle passengers, whereas the transportation view can capture some inactive cellphone users. But a view's ability to capture human mobility is unknown *a priori* and is highly dynamic based on spatiotemporal contexts. As a result, such dynamic view completeness makes multi-view human mobility modeling extremely challenging.

In this chapter, serving as the mobility abstraction layer, we propose coMobile, a generic framework to capture human mobility with a multiple-view learning technique. In coMobile, we first design a single-view learning technique based on context-based tensor decomposition to improve completeness of single-view models. Then, we integrate those improved single-view models together by formulating a convex optimization to obtain the ground truth of urban mobility. Most importantly, we implement coMobile based on extremely large datasets in the Chinese city Shenzhen with cellphone data and transportation data including taxis, buses, and subways. In particular, the key contributions of the chapter are as follows:

- We propose the first multi-view learning framework for human mobility to integrate incomplete yet complementary knowledge from individual views. To our knowledge, the proposed model is the only human mobility model driven by more than one view, which aims to address over-fitting of single view models. It is challenging to directly apply multi-view learning in human mobility modeling, because data-driven views are mostly incomplete to urban-scale mobility.

- We design a single-view learning technique based on context-aware tensor decomposition with both real-time and historical data to improve completeness of single-view models. This technique addresses data sparsity challenges of particular views to improve their completeness. In particular, we use a cellphone-view model as an example to show how we extract three contexts, i.e., cellphone user density, calling location patterns, and calling time patterns, based on historical data for joint tensor decomposition.

- Based on improved single-view models, we formulate a multi-view modeling problem by designing a joint optimization, which minimizes overall weighted deviation from observed mobility to the ground truth. To solve this optimization, we propose an iterative learning process to alternatively update the ground truth and view completeness until no further improvement can be made for the objective function. We formally prove the convexity of the joint optimization and the convergence of our iterative learning.

- We implement our multi-view human-mobility model based on two datasets in the Chinese city Shenzhen, with 10 million cellphone users and 16 million smartcard users involved. To our knowledge, this is one of the largest human mobility models driven by real-world datasets. We evaluate our model by comparing it to a single-view model, and results show that we reduce error rates by 51% on average.

We organize this chapter as follows. Section 4.2 gives our motivation. Section 4.3 presents the model overview. Section 4.4 depicts our single-view modeling. Section 4.5 describes our multi-view modeling. Sections 4.6 and 4.7 give the implementation and evaluation of our method, followed by the discussion in Section 4.8. Section 4.9 concludes the chapter.

4.2 Motivation

We first show the drawback of single-view models and the opportunity of multi-view models.

4.2.1 Drawback of Single-View Models

We give two comparisons: (i) models driven by two different cellphone views; (ii) models driven by a cellphone view and a transportation view.

As in Figure 4.1, we first compare models driven by two one-day CDR (call detail records) datasets from two carriers in Shenzhen. This kind of models driven by single-carrier data is mostly used for human mobility modeling [8]. A point indicates a spatial unit covered by a cell tower, and an edge linking two points together indicates the mobility between them. We only show the major mobility for the clarity of the figure. As shown by the circles, we found that each model can capture some unique mobility that cannot be captured by the other, which leads to over-fitting of these models driven by CDR data from single cellphone carriers.

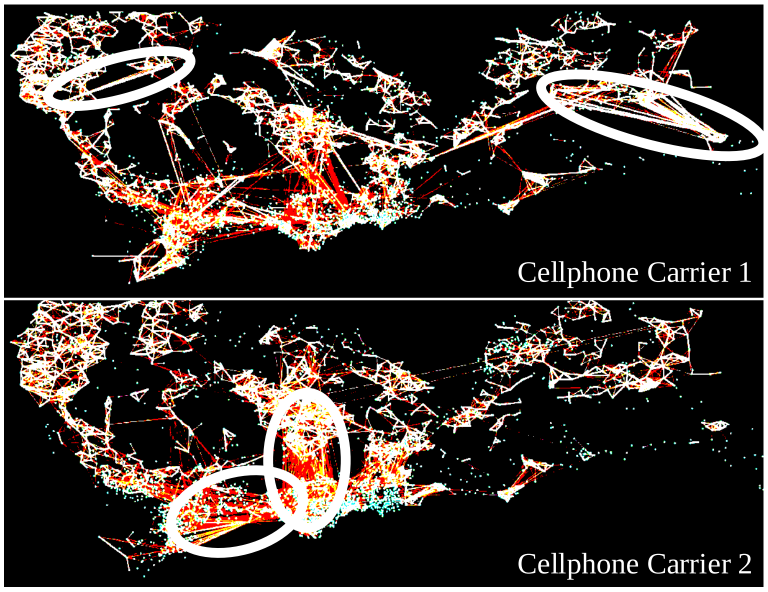


Figure 4.1: Models Driven by Two Carrier's CDR Data

We combine the CDR data from different carriers, and obtain a model driven by combined CDR data, i.e., a model driven by the cellphone view. Similarly, we combine data from different urban transportation, i.e., taxi, bus and subway, together, and then obtain a model driven by the transportation data, i.e., a model driven by the transportation view. Due to different spatial granularity, we use a urban-region-based model to show captured mobility in the morning rush hour.

As in Figure 4.2, every point indicates a region in the Shenzhen urban area; every edge linking two regions together indicates the mobility volume between them. The size of a vertex indicates associated mobility, and the different color indicates urban districts. As shown by the circles, we also found that each model can capture some mobility that cannot be captured by the other.

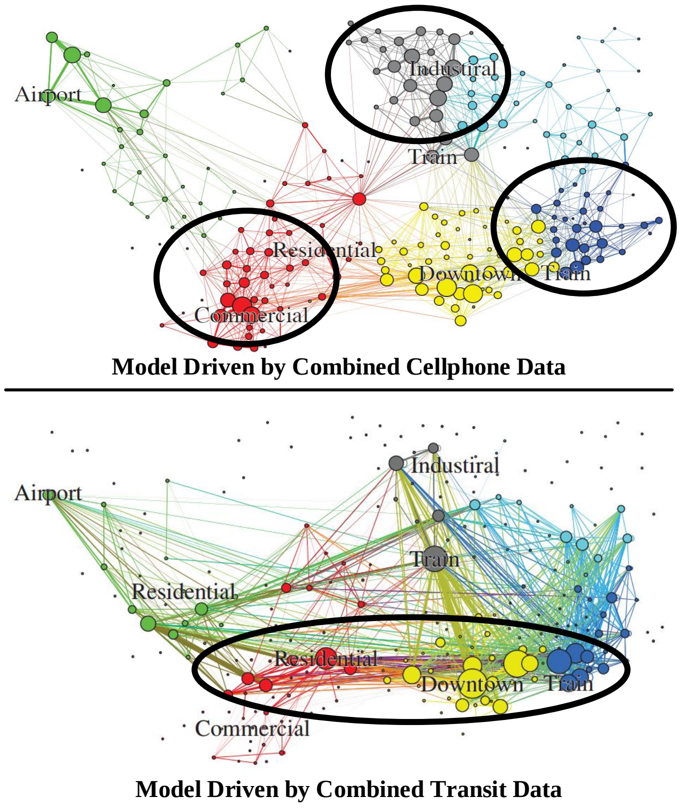


Figure 4.2: Models Driven by Cellphone and Transit Data

4.2.2 Opportunity for Multi-View Models

Due to the limitation of the single-view models, we are motivated to combine two separate views together in order to design a multi-view model for human mobility.

As shown by Figure 4.3, from the transportation view, we aim to combine four independent models (i.e., four triangles) driven by data from taxis, buses, subways, and private vehicles for human mobility modeling. But currently there is no urban infrastructure that can capture private transportation in real time at urban scale. Some efforts have been made by the research community to install GPS devices in the private vehicles to study human mobility [54], but only limited private vehicles are involved.

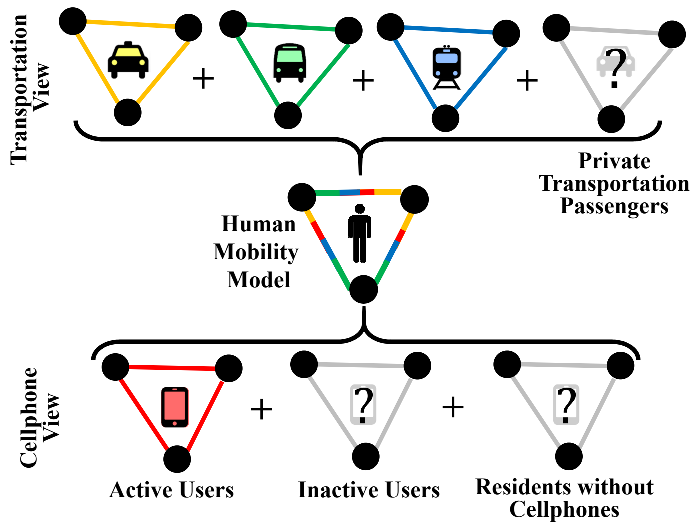


Figure 4.3: Multi-View Modeling

Alternatively, we can design a model driven by cellphone CDR data as in Figure 4.3. But there are two challenges. (i) Some cellphone users would not use their cellphones (i.e., being inactive) during the time we perform modeling. To address this issue, we design a technique based on tensor decomposition with correlated contexts to infer locations of inactive cellphone users in Section 4.5. (ii) Some urban residents who opt out of allowing their CDR data used for other purposes or do not have cellphones at all. Therefore, for these residents, we cannot capture their mobility.

As a result, neither the transportation view nor the cellphone view is complete by itself, but one view is often complementary to another. For example, the model driven by cellphone data can provide some mobility about residents using private transportation; whereas the model driven by transportation data can provide some mobility about residents without cellphone CDR data. It motivates us to design an effective modeling technique to combine these two views for better mobility modeling.

4.3 Model Overview

In this section, we first introduce views we used for multi-view modeling, and then we present a concept called mobility graph to capture the real-time human mobility, and finally we give the architecture of coMobile.

4.3.1 Multi-View Data

We have been working with several service providers and the Shenzhen Transport Committee (hereafter STC) for data access of urban infrastructures as introduced in Section 3.2. We consider two kinds of data, i.e., cellphone data and transportation data, as two individual views to model human mobility.

Cellphone View: Cellphone CDR (call detail records) data are used to infer cellphone users' locations at cell tower levels. We utilize CDR data through two major operators in Shenzhen with more than 10 million users. The CDR data give 220 million locations per day.

Transportation View: Data from three kinds of transportation modes, i.e., taxi, subway and bus, are used to detect transportation passengers' locations. We study transportation data through STC to which taxicab, bus and smartcard companies upload their data in real time.

- **Taxi data** are used to infer taxi passengers' origins and destinations based on status (i.e., pickups and dropoffs) at GPS location levels. They account for 14 thousand taxis, each of which generates 2 records/min.

- **Smartcard data** are used to infer origins and destinations of residents with smartcards used to pay bus and subway fares, which capture more than 10 million rides and 6 million passengers per day. Smartcard data and subway map data are used together to detect subway passengers' origins and destinations at subway station levels.

- **Bus data** are used to infer bus passengers' origins and destinations along with smartcard data (showing that a passenger uses a smartcard at a bus station) at 4849 bus station levels. They account for all 13 thousand buses, each of which generates 2 records/min.

Our endeavor of consolidating the above data-driven views enables extremely large-scale real-time urban phenomenon rendering, e.g., human mobility, which is unprecedented in terms of both quantity and quality.

4.3.2 Mobility Graph

In this work, we use Mobility Graph to capture human mobility in real time at urban scale, which is a time-varying graph where a vertex indicates a spatial unit (e.g. a urban region or a street block) and a weight of an edge linking two vertices indicates the mobility volume between them. Due to its time-varying nature, a mobility graph G_t is associated with a time period t (e.g., 4-5PM), which shows the mobility during this particular time period.

Figure 4.4 gives a simplified example of a mobility graph with only 3 vertices. The number of people moving between different spatial units, i.e., weights of edges, should include people associated with a particular view, e.g., the cellphone view or the transportation view. In this work, our main objective is to obtain mobility graphs based on single-view modeling, and then to combine them together by multi-view modeling for a comprehensive human mobility graph.

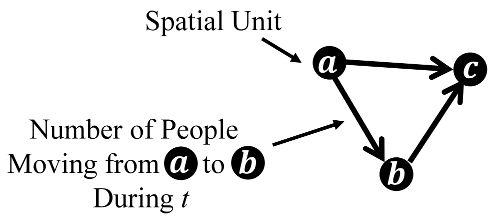


Figure 4.4: Mobility Graph

4.3.3 coMobile Framework

We introduce our coMobile Framework by Figure 4.5. From the bottom, we have urban data generated by urban infrastructures, e.g., cellphone data and transportation data, which are introduced in Section 3.2. Based these two kinds of data, we design two single-view models capturing mobility patterns of cellphone users and urban transportation users by two mobility graphs, which are introduced in Section 4.4. Then, we present our multi-view learning to integrate single-view models for more complete human mobility modeling, which is introduced in Section 4.5. Finally, the obtained human mobility model can be used in many applications, e.g., ridesharing and last-mile transit as introduced in Chapters 5 and 6.

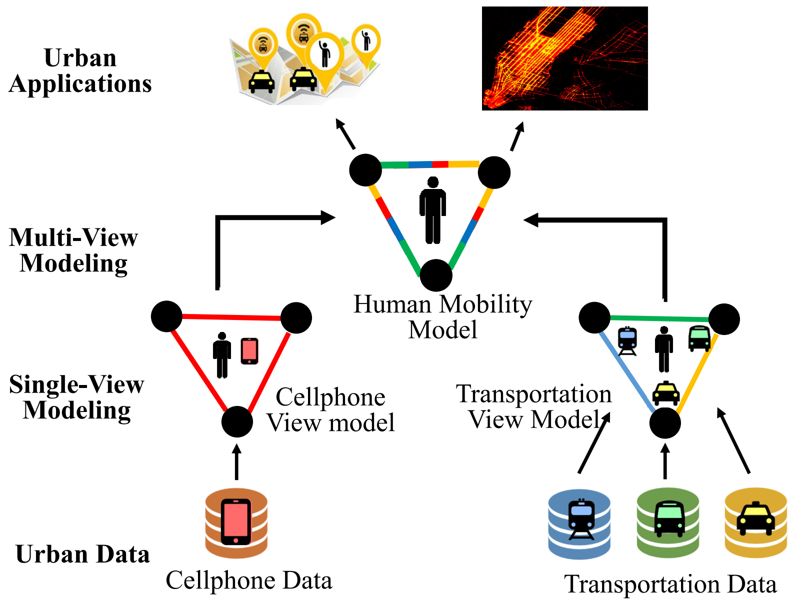


Figure 4.5: coMobile Overview

Note that we only consider two specific views in coMobile but it can be generalized to more views if more data are available. In coMobile, we first generate single-view models and then combine them together at model levels, instead of raw data levels (e.g., using multi-source raw data to directly design a multi-view model). This is because in many applications due to privacy issues, raw data are not available, and only high-level single models can be used as input. Our coMobile is still applicable to this situation.

4.4 Single-View Mobility Modeling

We introduce how to model urban mobility based on two single views, i.e., a cellphone view and a transportation view.

4.4.1 Cellphone-View Modeling

As introduced earlier, the key challenge to model human mobility based on cellphone data is that inactive cellphone users or residents without cellphones do not generate any CDR data. As a result, we cannot model their mobility to obtain mobility graph. For residents without cellphones, the solution is limited although the model based on transportation can capture some of them. In this subsection, we focus on inactive cellphone users to infer their mobility by an observation that inactive cellphone users who did not use their cellphones today may use their cellphones before during the similar trips [55]. Accordingly, we formulate a tensor decomposition problem to infer mobility of both active and inactive users based on real-time and historical data.

Tensor Construction

We infer locations of cellphone users for specific time slots by a three dimensional tensor $\mathcal{A} \in \mathbb{R}^{N \times K \times M}$.

- A cellphone user dimension indicates individual cellphone users differentiated by SIM IDs: $[u_1, ..., u_N]$.
- A time slot dimension indicates specific time windows (e.g., one hour window from 5PM to 6PM): $[t_1, ..., t_K]$.
- A spatial unit dimension indicates specific spatial units (e.g., a urban region): $[s_1, ..., s_M]$.
- An entry $\mathcal{A}(n, k, m)$ indicates the number of CDR records a user n has in a spatial unit m during a slot k.

With our cellphone data, we fill this tensor $\mathcal{A}$, and then obtain all cellphone users' locations with a specific spatiotemporal partition. However, a key challenge is that the tensor $\mathcal{A}$ is sparse because for inactive cellphone users, their corresponding entries are empty due to lacking CDR data.

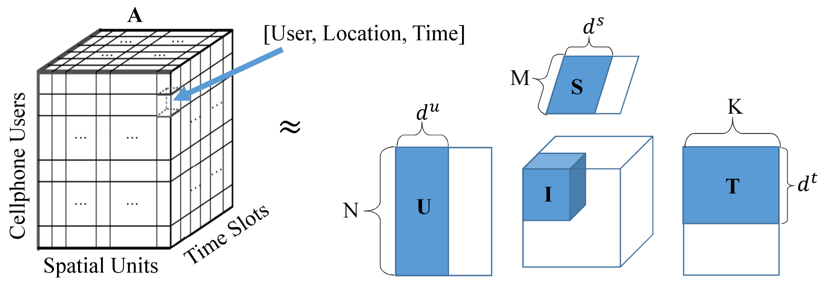


Figure 4.6: Tensor Decomposition

A common approach to address this issue is to use tensor decomposition. As in Figure 4.6, we have a tensor with three dimensions indicating cellphone users, spatial units, and time slots. An entry denotes a tuple [user, location, time]. But this tensor is sparse due to inactive cellphone users. Based on the classic Tucker decomposition model [56], we decompose $\mathcal{A}$ into a core tensor $\mathcal{I}$ along with three matrices, $\mathcal{U} \in \mathbb{R}^{N\times d^u}$, $\mathcal{S} \in \mathbb{R}^{M\times d^s}$, and $\mathcal{T} \in \mathbb{R}^{K\times d^t}$. $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{T}$ infer correlations between different cellphone users, different spatial units, and different time slots, respectively. d^u, d^s and d^t are the number of latent factors and very small.

The following objective function is used to optimize the decomposition.

$$||\mathcal{A} - \mathcal{I} \times \mathcal{U} \times \mathcal{S} \times \mathcal{T}||^2 + \lambda(||\mathcal{I}||^2 + ||\mathcal{U}||^2 + ||\mathcal{S}||^2 + ||\mathcal{T}||^2)$$

where the first term is to measure the error of decomposition and the second term is a regularization function to avoid over-fitting. $||\cdot||^2$ denotes the l_2 norm and λ is the parameter to control the contribution of the regularization function. By minimizing this objective function, we obtain the optimized $\mathcal{I}$, $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{T}$ by the sparse tensor $\mathcal{A}$, which is given by cellphone data. As a result, we use $\mathcal{I} \times \mathcal{U} \times \mathcal{S} \times \mathcal{T} = \mathcal{A}'$ to approximate $\mathcal{A}$ where $\times$ is the tensor-matrix multiplication.

However, a key challenge for this decomposition is that $\mathcal{A}$ is over sparse especially under fine spatiotemporal partition, which leads to poor performance of decomposition.

To address this issue, in this work, we propose a technique to use historical cellphone data to establish correlated contexts that improve the performance of the decomposition.

Context Extraction

To provide additional information for the decomposition, we use the historical cellphone data to extract three contexts, i.e., cellphone user density, calling location pattern, and calling time pattern. We use three matrices to denote these three contexts as in Figure 4.7.

- Cellphone User Densities are given by a matrix $\mathcal{B}$ where a row denotes a spatial unit; a column denotes a time slot; an entry denotes the average CDR record count in this spatial unit for this time slot over a period of historical time.
- Calling Location Patterns are given by a matrix $\mathcal{C}$ where a row denotes a spatial unit; a column denotes a cellphone user; an entry denotes a cellphone user's CDR record count in this spatial unit given a period of historical time.
- Calling Time Patterns are given by a matrix $\mathcal{D}$ where a row denotes a time slot; a column denotes a cellphone user; an entry denotes a cellphone user's CDR record count in this time slot given a period of historical time.

All the matrices $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ can be obtained by a set of historical cellphone data.

Context-based Tensor Decomposition

We present a joint tensor decomposition based on the three extracted context matrices. In particular, we design the objective function as follows.

$$\begin{aligned}\min_{\mathcal{I},\mathcal{U},\mathcal{S},\mathcal{T}} \mathbf{L}(\mathcal{I},\mathcal{U},\mathcal{S},\mathcal{T}) = ||\mathcal{A} - \mathcal{I} \times \mathcal{U} \times \mathcal{S} \times \mathcal{T}||^2 \\ +\lambda_1||\mathcal{B} - \mathcal{S} \times \mathcal{T}||^2 + \lambda_2||\mathcal{C} - \mathcal{S} \times \mathcal{U}||^2 + \lambda_3||\mathcal{D} - \mathcal{T}^T \times \mathcal{U}||^2 \\ +\lambda_4(||\mathcal{I}||^2 + ||\mathcal{U}||^2 + ||\mathcal{S}||^2 + ||\mathcal{T}||^2).\end{aligned} \tag{4.1}$$

where the first term is to measure the error of decomposing $\mathcal{A}$; the second, third, and forth terms are to measure the error of factorizing matrix $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$, respectively; the

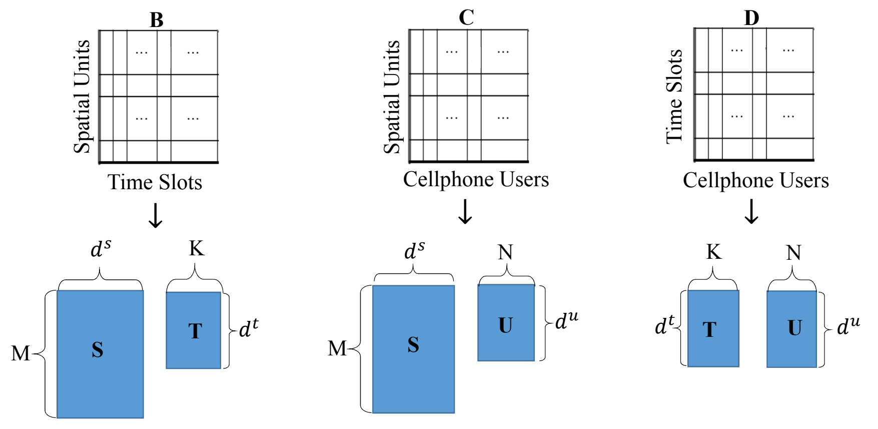


Figure 4.7: Context Matrix Factorization

last term is to avoid over-fitting. In our setting, $d^u = d^s = d^t$. λ_1, λ_2, λ_3, and λ_4 are preset parameters to indicate term weights. We normalized all values to [0, 1] for the decomposition.

In this objective function, $\mathcal{A}$ and $\mathcal{B}$ share $\mathcal{S}$ and $\mathcal{T}$; $\mathcal{A}$ and $\mathcal{C}$ share $\mathcal{S}$ and $\mathcal{U}$; $\mathcal{A}$ and $\mathcal{D}$ share $\mathcal{U}$ and $\mathcal{T}$. Since $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ are not sparse, they lead to accurate $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{U}$, which increases the performance of decomposing $\mathcal{A}$. As a result, the historical cellphone user calling patterns are transferred into the decomposition of $\mathcal{A}$, which leads to an accurate tensor decomposition.

Because this objective function does not have a closed-form solution to find the global optimal $\mathcal{I}$, $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{T}$, we use an element-wise optimization algorithm as a numeric method [57] to obtain a local optimal solution. Finally, after we obtain $\mathcal{I}$, $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{T}$, we use $\mathcal{I} \times \mathcal{U} \times \mathcal{S} \times \mathcal{T} = \mathcal{A}'$ to obtain cellphone mobility graph G^C of all cellphone users.

4.4.2 Transportation-View Modeling

Based on our transportation data, we model human mobility by three transportation modes, i.e., taxi, bus and subway. Given attributes of our transportation data, we directly obtain origins and destinations of taxi, bus and subway passengers at GPS, bus station, and subway station levels. In this work, we use a space alignment technique where we assign taxi GPS locations, bus stations, and subway stations into corresponding spatial units based on a specific spatial partition of urban areas. Thus, for a pair of spatial units, e.g., from an airport to a train station, we aggregate all the above passengers who traveled between these two spatial units to obtain a mobility volume during a particular time period, because these three kinds of transportation modes are independent from each other. Thus, from the transportation view, obtaining transportation mobility graph G^T is straightforward.

Our context-aware tensor decomposition can also be used to improve completeness of the transportation-view model since we have missing data issues (e.g., GPS records) as well. The process is conceptually similar to the tensor decomposition for the cellphone-view model, which is omitted due to the space limitation.

Further, we did not consider private vehicles in our transportation view due to lack of private vehicle data. However, some urban residents using private transportation would be captured by multi-view learning, which is introduced as follows.

4.5 Multi-View Mobility Modeling

In this section, based on single-view modeling, we introduce multi-view modeling in coMobile. Even though our data can only form two views to obtain two mobility graphs, i.e., the cellphone mobility graph G^C and the transportation mobility graph G^T, we aim to tackle a more generic problem, i.e., multi-view modeling, and thus double-view modeling is a concrete example of multi-view modeling.

We first formulate a joint optimization problem for multi-view human mobility modeling, and then we develop an iterative learning processing to solve this problem, and finally we theoretically analyze the performance of modeling in terms of convexity and convergence.

4.5.1 Joint Optimization

The main objective of our multi-view modeling is to obtain a comprehensive human mobility graph G^H for a given time period based on several single-view mobility graphs, e.g., G^C and G^T. Because we have the same spatial partition for different mobility graphs, they have the same number of edges and vertices, and the key difference is edge weights. Since different edges are independent in a human mobility graph, we use one edge ab in a human mobility graph G^H as an example to show how we obtain the human mobility from one spatial unit a to another spatial unit b by our multi-view technique, and combine different edge weights together to obtain a complete human mobility graph G^H.

For a specific edge ab in G^H, the volume of passengers traveling from a spatial unit a (e.g., an airport) to b (e.g., a train station) during a time period t (e.g., 4-5PM) is $x^*_{ab\cdot t}$, which is the unknown ground truth we want to infer. Assuming we have V different views, which leads to V different mobility graphs that are incomplete by themselves yet complementary to each other. For a specific view $v \in [1, V]$, we use $x^v_{ab\cdot t}$ to indicate the weight of the edge ab during t in the mobility graph G^v; for a specific view $v \in [1, V]$, we use $w^v_{ab\cdot t}$ to indicate the completeness degree of this view during a time period t from this edge ab of G^v. The completeness degree of a view quantifies its capability to capture human mobility. The stronger the capability is, the higher the degree is. Under different spatiotemporal contexts, the completeness degree of the same view is different. We use a vector $\mathcal{W}_{ab\cdot t} = \{w^1_{ab\cdot t}, ..., w^v_{ab\cdot t}, ..., w^V_{ab\cdot t}\}$ to indicate completeness degrees for all V views.

In coMobile, based on the above definitions, V and $x^v_{ab\cdot t}$ are given in advance by the datasets; whereas $x^*_{ab\cdot t}$ and $\mathcal{W}_{ab\cdot t}$ are unknown. Therefore, we present a joint optimization to obtain optimal $x^*_{ab\cdot t}$ and $\mathcal{W}_{ab\cdot t}$ together. The basic idea behind our multi-view learning is that a view with a higher completeness degree provides more comprehensive information, so the ground truth should be close to mobility observed by a view with a higher completeness degree. As a result, we should minimize the deviation from mobility observed by a view v to the ground truth $x^*_{ab\cdot t}$ (unknown), proportionally to its completeness degree $w^v_{ab\cdot t}$ (also unknown). Therefore, we develop the following objective function for multi-view learning.

$$\begin{aligned} \min_{x^*_{ab\cdot t}, \mathcal{W}_{ab\cdot t}} \mathbf{F}(x^*_{ab\cdot t}, \mathcal{W}_{ab\cdot t}) = \sum_{v=1}^{V} [w^v_{ab\cdot t} \cdot \mathbf{D}(x^*_{ab\cdot t}, x^v_{ab\cdot t})], \\ \text{s.t.}, \mathbf{R}(\mathcal{W}_{ab\cdot t}) = 1. \end{aligned} \tag{4.2}$$

$\mathbf{D}(x^*_{ab\cdot t}, x^v_{ab\cdot t})$ is a distance function that describes the distance between $x^*_{ab\cdot t}$ and $x^v_{ab\cdot t}$. Therefore, the term $\sum_{v=1}^{V}[w^v_{ab\cdot t} \cdot \mathbf{D}(x^*_{ab\cdot t}, x^v_{ab\cdot t})]$ indicates the overall weighted distance between the observed mobility and the ground truth. We aim to find the optimal $x^*_{ab\cdot t}$ and $\mathcal{W}_{ab\cdot t}$ that minimize this overall weighted distance under a constraint.

$\mathbf{R}(\mathcal{W}_{ab\cdot t})$ is a constraint function, which gives the distribution of view completeness. Without this constraint, the optimization problem is unbounded. For the sake of simplicity, we set $\mathbf{R}(\mathcal{W}_{ab\cdot t}) = 1$. Other constraint functions can also be used since we can divide $\mathbf{R}(\mathcal{W}_{ab\cdot t})$ by a constant.

The rationale behind this function is that for a more-complete view, we have a high penalty if the mobility observed from this view has a longer distance to the ground truth. In contrast, for a less-complete view, we have a low penalty if the mobility observed from this view has a longer distance to the ground truth. Thus to minimize the objective function, ground truth relies on the more complete views.

4.5.2 Iterative Learning

We develop an iterative learning technique based on the block coordinate descent [58] to solve this optimization. Since in our objective function we have two sets of variables, i.e., both the ground truth $x^*_{ab\cdot t}$ and the view completeness degree $\mathcal{W}_{ab\cdot t}$, we aim to iteratively yet alternatively optimize these two sets of variables until the result converges. In particular, we optimize the value of one set to minimize the objective function while keeping the value of the other set fixed, and then we swap the fixed variable and the optimized variable to continue this process until the result converges. Figure 4.8 gives the description of our iterative technique.

In Step 1, we first initialize $x^*_{ab\cdot t}$ and $\mathcal{W}_{ab\cdot t}$ based on the average value of $x^*_{ab\cdot t}$, because the initialization does not affect the final results based on the property of the block coordinate descent [58]. In Step 2, we first fix the initialized $x^*_{ab\cdot t}$, and then find the optimal $\mathcal{W}_{ab\cdot t}$ that minimizes the objective function. In Step 3, with this optimized $\mathcal{W}_{ab\cdot t}$, we fix it and then find the optimal $x^*_{ab\cdot t}$ that minimizes the objective function again. Then, with this optimized $x^*_{ab\cdot t}$, we go back to Step 2 to fix $x^*_{ab\cdot t}$ again, and then to further optimize $\mathcal{W}_{ab\cdot t}$. This is an iterative process to alternatively optimize $x^*_{ab\cdot t}$ and $W_{ab\cdot t}$ until the result converges.

Based on the property of the block coordinate descent [58], the convergence of the above iterative process is based on the distance function and constraint function used.

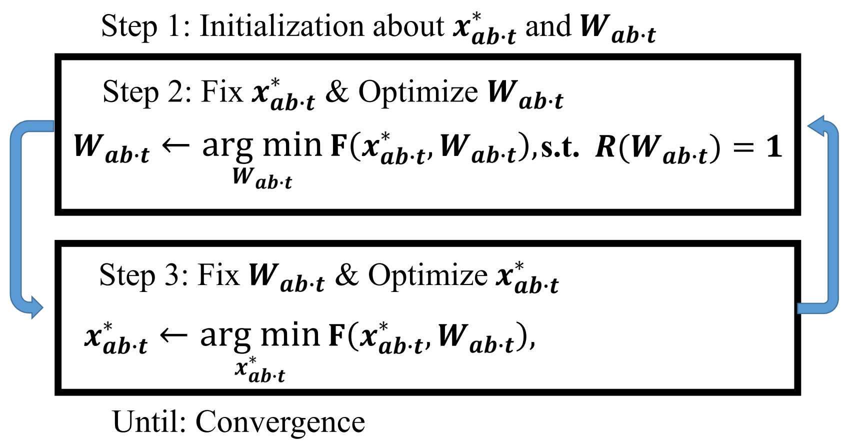

Step 1: Initialization about $\boldsymbol{x}^*_{ab\cdot t}$ and $\boldsymbol{W}_{ab\cdot t}$
Step 2: Fix $\boldsymbol{x}^*_{ab\cdot t}$ & Optimize $\boldsymbol{W}_{ab\cdot t}$
$\boldsymbol{W}_{ab\cdot t} \leftarrow \arg\min_{\boldsymbol{W}_{ab\cdot t}} \mathbf{F}(\boldsymbol{x}^*_{ab\cdot t}, \boldsymbol{W}_{ab\cdot t})$, s.t. $\boldsymbol{R}(\boldsymbol{W}_{ab\cdot t}) = \mathbf{1}$
Step 3: Fix $\boldsymbol{W}_{ab\cdot t}$ & Optimize $\boldsymbol{x}^*_{ab\cdot t}$
$\boldsymbol{x}^*_{ab\cdot t} \leftarrow \arg\min_{\boldsymbol{x}^*_{ab\cdot t}} \mathbf{F}(\boldsymbol{x}^*_{ab\cdot t}, \boldsymbol{W}_{ab\cdot t})$,
Until: Convergence

Figure 4.8: Iterative Multi-View Learning

As follows, we theoretically analyze the performance of our technique in terms of convexity and convergence.

4.5.3 Theoretical Analyses

We use Negative Log Function as our constraint function:

$$\mathbf{R}(\mathcal{W}_{ab\cdot t} = \{w^1_{ab\cdot t}, ..., w^v_{ab\cdot t}, ..., w^V_{ab\cdot t}\}) = \sum_{v=1}^{V} \exp(-w^v_{ab\cdot t}).$$

This negative log function maps a number between 0 and 1 to a number from 0 to ∞, which enlarges the difference between different view completeness degrees for better modeling.

Further, we use Normalized Squared Loss function as our distance function given as

$$\mathbf{D}(x^*_{ab\cdot t}, x^v_{ab\cdot t}) = \frac{(x^*_{ab\cdot t} - x^v_{ab\cdot t})^2}{\mathrm{STD}(x^1_{ab\cdot t}, ..., x^v_{ab\cdot t}, ..., x^V_{ab\cdot t})}.$$

This normalized squared loss is an effective method to measure the distance between two variables and consider the distribution of $x^v_{ab\cdot t}$ at the same time.

As follows, we prove the convexity and convergence of our iterative learning with the above two functions.

THEOREM: If the negative log function and the normalized squared loss function are used, then convergence of our iterative process in Figure 4.8 is guaranteed.

PROOF: Based on the convergence proposition on the block coordinate descent [58], the iterative process converges to a stationary point, if the optimizations in Steps 2 and 3 are convex. Thus, the rest of our proof has 2 steps: (i) in Step 2, if $x^*_{ab\cdot t}$ is fixed, the optimization for $\mathcal{W}_{ab\cdot t}$ is convex; (ii) in Step 3, if $\mathcal{W}_{ab\cdot t}$ is fixed, the optimization for $x^*_{ab\cdot t}$ is convex.

To prove the convexity of Step 2, we use another variable $y_v = \exp(-w^v)$. Therefore, the optimization problem becomes a new function with only one variable of y_v.

$$\min_{y_1,...,y_v,...,y_V} \mathbf{F}(y_1, ..., y_v, ..., y_V) = \sum_{v=1}^{V}[-\log(y_v) \cdot \mathbf{D}(x^*_{ab\cdot t}, x^v_{ab\cdot t})],$$

$$\text{s.t.,} \sum_{v=1}^{V} y_v = 1.$$

With this new variable y_v, we have a linear constraint function and a linear objective function (i.e., a linear combination of negative logarithm functions). Therefore, both the constraint function and objective function are convex, which leads to the fact that any local optimal solution is also the global optimal solution for Step 2.

To prove the convexity of Step 3, we treat the objective function as an unconstrained optimization with only one variable. In Step 3, since the normalized squared loss function is convex, the objective function is a linear combination of convex functions, which makes it convex. $\square$

Note that other constraint and distance functions can also be used in our iterative process but may not lead to the convexity of the optimization problem, and thus the convergence of the iterative process cannot be guaranteed.

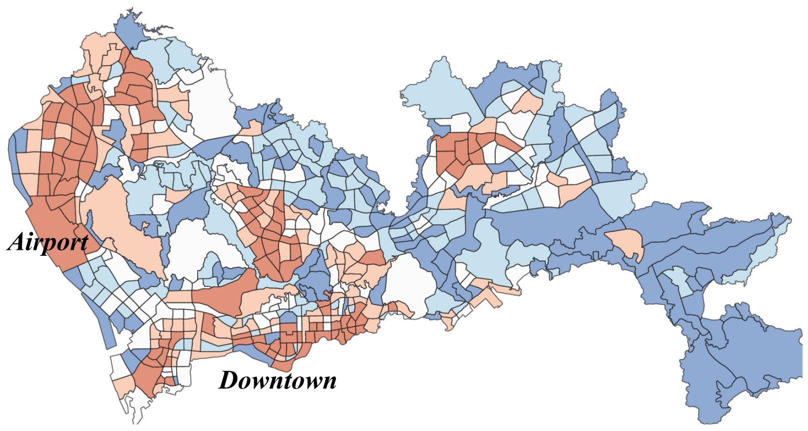


Figure 4.9: Shenzhen Urban Partition

4.6 Real-World Implementation

We implement coMobile based on cellphone and transportation data in Shenzhen introduced in Section 3.2. Since this chapter concentrates on modeling, we briefly introduce our data-related issues during our implementation. We establish a secure and reliable transmission mechanism, which feeds our server the data collected by STC and service providers with a wired connection. As shown in Section 3.2, we have been storing a large amount of data, requiring significant efforts for the daily management. We utilize a 34 TB Hadoop Distributed File System (HDFS) on a cluster consisting of 11 nodes, and each of them is equipped with 32 cores and 32 GB RAM. For daily management, we use the MapReduce based Pig and Hive. Because of the extremely large size of our data, we have been finding several kinds of errant data, e.g., duplicated data, missing data, and data with logical errors. To address these issues, we conduct a detailed cleaning process to filter out errant data.

For real-world implementation, we have to decide the spatiotemporal partition for the mobility graph, which decides the spatiotemporal granularity of our model. For example, we have more than 110 thousand road segments, 496 urban regions, and 10 urban districts in Shenzhen, and we can capture the mobility with one of those three spatial partitions for every 15 mins, 30 mins, 60 mins, or even longer. Due to the spatial

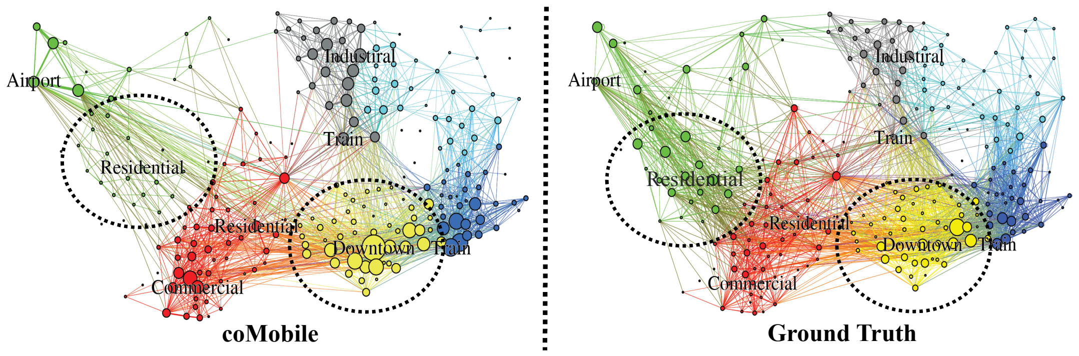


Figure 4.10: Mobility Graphs in Shenzhen

resolutions of our data (especially for bus, subway, and cellphones), we use a urban-region partition proposed by the Shenzhen government as our spatial partition, which is given by Figure 4.9. Different colors indicate different population density. Based on this partition, we implement our multi-view mobility modeling technique coMobile based on two views. A human mobility graph obtained by coMobile for major urban areas during the evening rush hour at region levels is given by the left of Figure 4.10.

4.7 Data-Driven Evaluation

4.7.1 Evaluation Methodology

Based on our implementation, we compare coMobile with a single-view human mobility model called WHERE. WHERE [8] is a model driven by cellphone data, and it is based on spatial and temporal probability distributions of human mobility and produces synthetic cellphone records as the inferred mobility. We compare these two models with the inferred ground truth. In this project, to infer the ground truth, we introduce another new cellphone related dataset for the evaluation. Different from regular CDR data, this dataset logs locations of all cellphone users at cell tower levels for every 15 mins even without activities. We use the mobility graph obtained from this dataset as the ground truth, which is given in Fig. 4.10. By a visual comparison, we found that we underestimate the mobility at residential areas and overestimate the mobility at downtown areas.

We utilize three months of data to evaluate these two models. We use Mean Average Percent Error (MAPE) in a time slot as a metric to test those two models MAPE $= \frac{100}{n}\sum_{i=1}^{n}\frac{|\bar{\mathbf{T}}_i-\mathbf{T}_i|}{\bar{\mathbf{T}}_i}$, where $n = 496 \times 496 = 246016$ is the total number of region pairs, i.e., the total number of edges in a mobility graph; $\mathbf{T}_i$ is the inferred mobility between a region pair i; $\bar{\mathbf{T}}_i$ is the ground truth of the mobility between a region pair i. An accurate model yields a small MAPE, and *vise versa*. We use 90 days of data, leading to 90 experiments. The average results were reported.

We investigate the impact of different contexts by adjusting three model parameters, i.e., λ_1, λ_2, and λ_3, which control contributions of different contexts in our tensor decomposition with Eq.(1). The default setting is $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{4}$ where we consider all contexts and the regularization term equally. Further, we investigate the impact of historical cellphone data on the model performance in terms of extracting correlated contexts.

We compare two models' inferring accuracy in terms of MAPE values by (i) a low level comparison on five particular region pairs, (ii) a high level comparison on all 246016 region pairs, (iii) different lengths of slots, and (iv) different amount of historical data.

4.7.2 Evaluation Results

Figure 4.11 plots the MAPE under one hour slots with the two-way mobility between a residential region and five other regions. We found that coMobile outperforms WHERE in general. This is because WHERE only uses the cellphone data to model the human mobility from the cellphone view alone; whereas coMobile uses two views to model the human mobility, which leads to better performance. We also found that the performance gain between coMobile and WHERE is lower during the rush hour. One of the possible explanations is that the repeatable mobility patterns are higher during the rush hour, so all models have better performance. Comparing the five region pairs, we found that for the commuting region pairs (e.g., between the residential region and the industrial, commercial or downtown regions), all models have better performance than the region pairs on which the residents go for travel (i.e., between the residential region to the airport or train station regions). This is due to the fact that repeatable pattern for travel is limited.

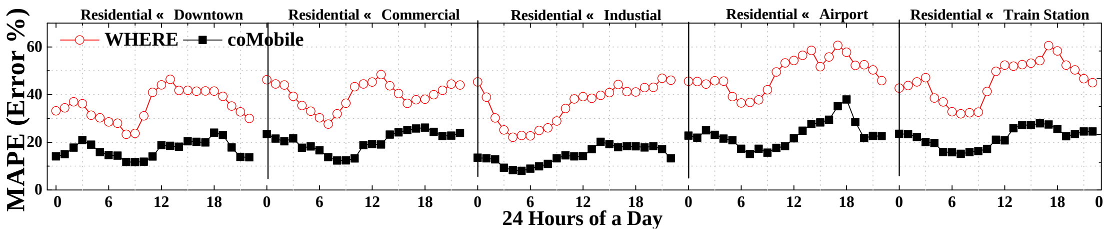


Figure 4.11: MAPE under One Hour Slot for 24 Hours of a day

Figure 4.12 gives the MAPE on all region pairs under one hour slots. We found that all two models have higher MAPE than the MAPE we observed in Figure 4.11. This is because the urban mobility is highly dynamic between various regions pairs, many region pairs have very limited mobility, which leads to high MAPE. But we also found that relative performance between these two models is the same as in Figure 4.11. coMobile is better than WHERE, which shows the advantage of using multi-view learning to model the human mobility. coMobile outperforms WHERE by 51% in terms of MAPE, resulting from its multi-view learning from both cellphone data and transportation data.

Figure 4.13 plots the MAPE of coMobile and WHERE with different slot lengths from 15 mins to 12 hours. Basically, the MAPE of both models reduces with the increase

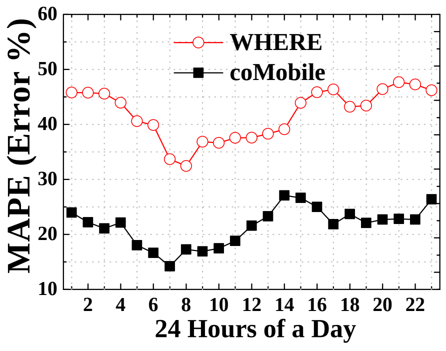


Figure 4.12: Hourly MAPE

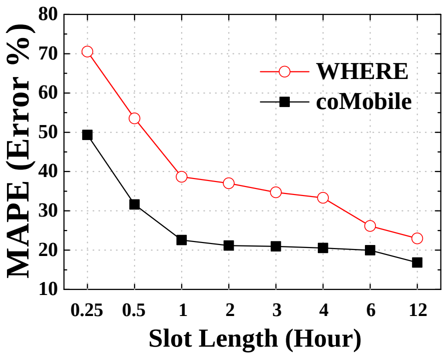


Figure 4.13: Effects of Lengths

of the modeling lengths. This is because the mobility in a longer time slot is much more stable. coMobile significantly outperforms WHERE when the slot length is short. This is because the transportation data can capture lots of mobility during a short time period. We notice that the slot length becomes longer than 6 hours, both coMobile and WHERE have the similar performance, because in a long time slot, the cellphone data alone is capable of inferring mobility.

Figure 4.14 shows how much historical information is necessary for coMobile and WHERE. As expected, the longer the time, the lower the MAPE for both models, the better the performance. But a too long history does not help much, especially for coMobile whose MAPE became stable when the historical data are longer than 4 weeks. It shows that coMobile does not reply on long-term historical cellphone data, thanks to the transportation view. But WHERE needs a longer historical period of data, i.e., 9 weeks, before its MAPE becomes stable.

Figure 4.15 shows the impact of two contexts, i.e., cellphone user densities and calling time&location patterns as introduced in Section 4.4.1. In particular, we set $\lambda_2 = \lambda_3 = 0$ and $\lambda_1 = \lambda_4 = \frac{1}{2}$ to obtain a model called coMobile w/ User Density, which only considers the cellphone user density as a context. Similarly, we set $\lambda_1 = 0$ and $\lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{3}$ to obtain a model called coMobile w/ Time&Location Patterns, which only considers time&location patterns as contexts. We compare them with coMobile, which considers all contexts. In generally, coMobile outperforms the other two models. We found that for the early morning, considering time&location patterns is better than

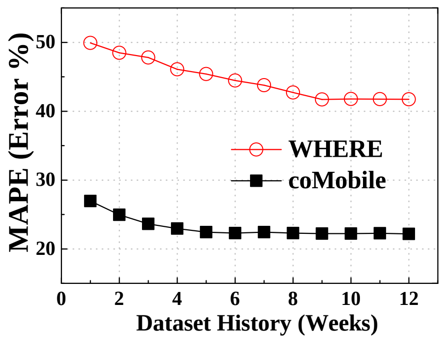


Figure 4.14: Historical Data

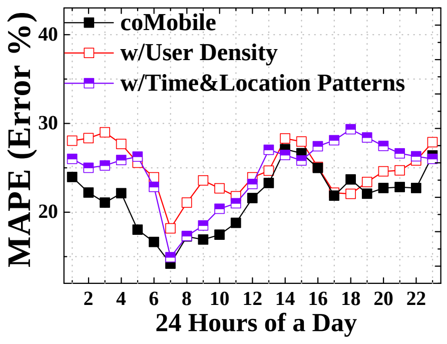


Figure 4.15: Impacts of Contexts

considering user density; while for the late night, considering user density is better than considering time&location patterns. Also, during some slots in the afternoon or evening, e.g., 14:00, 15:00 and 18:00, it leads to better performance if we do not consider certain contexts.

4.7.3 Evaluation Summary

In short, we have the following observations. (i) As in Figure 4.11, the accuracy of human mobility modeling is highly depended on both locations and time of day. (ii) As in Figure 4.12, both models have better performance in the morning rush hour in general due to the predicability of morning commutes, and coMobile outperforms WHERE during all times. (iii) As in Figure 4.13, the length of slots has significant impacts on performance of all models. (iv) As in Figure 4.14, how much historical data to be used by coMobile does not significantly affect the performance of coMobile. (v) As in Figure 4.15, the same contexts have different effects according to the time of day, but considering them together leads to better average performance.

4.8 Discussion

We provide some discussion about coMobile as follows.

Privacy Protections. While the data for the human mobility study have the potential for great social benefits, we have to protect the privacy of the residents involved for wider applications. We took three active steps for privacy protections. (i) Anonymization: All data analyzed are anonymized by the service providers who were not involved in this project, and all identifiable IDs, such as SIM card IDs, are replaced by a serial identifier during the analyses. (ii) Nature of Data: The nature of different data also provides a certain level of privacy protections, e.g., the taxicab and bus G-PS data do not involve any identity about passengers; the smartcard data only show passengers' locations at transit station levels. (iii) Aggregation: the mobility patterns obtained by coMobile are given at aggregated results with a mobility graph in a large spatiotemporal partition. We do not focus on individual residents during the analyses.

Public Data Access. Accessing empirical datasets is vital to the geographic information system research, but such datasets are usually not available for the fellow researchers due to the privacy issues. As an initiative step, the partial aggregated data used in this work have been made for public access in the website of Transport Committee of Shenzhen Municipality [59]. Most importantly, we release the first big urban data [7], which include the large-scale Shenzhen data including taxi, bus, subway, s-martcard, and cellphone data. This is the first time that such comprehensive urban data are released for the benefit of research community. Moreover, we will release more detailed data with privacy protection schemes after this paper is accepted.

Implementation in Different Cities. The residents in different cities typically have different mobility patterns due to geographic and demographic features. It is therefore extremely important to implement the proposed architecture in different cities to validate its generalizability. Currently, we have access to partial taxicab and cellphone data in Shanghai, the second largest city in China, and are negotiating with other service providers for potential implementation.

4.9 Conclusions

In this chapter, as the mobility abstraction layer for mobileCPS, we design, implement and evaluate a human mobile modeling technique called coMobile based on context-aware tensor decomposition and iterative multi-view learning. It is the first human mobility model based on both a cellphone view and a transportation view. Our endeavors offer a few valuable insights: (i) the human mobility modeling based on single-view data introduces biases, which can be addressed by using historical data and multi-view data; (ii) to model human mobility, every view itself is incomplete but they are often complementary to each other, and thus it is essential to model the completeness degree of a view before inferring the mobility; (iii) multi-view learning for human mobility requires an iterative optimization process to improve the accuracy of modeling, and thus how to select an objective function and constraint function to ensure the convergence is essential for real-time applications.

Chapter 5

coRide: Real-time Carpooling

In this chapter, we introduce a taxicab carpooling application as a component on the application design layer for mobileCPS. This carpooling service is built upon the mobility model we proposed in the last chapter. In particular, based on mobility modeling, we found urban regions with high human mobility demand yet low taxi supply, and then provide the carpooling service to increase taxi supply for a balanced relationship between passenger demand and taxi supply.

5.1 Introduction

Among all transportation modes, taxicabs play a particularly prominent role in residents' daily commutes in many metropolitan areas [60] [61]. Based on a recent survey in New York City [62], over 100 taxicab companies operate more than $13,000$ taxicabs, with stable demand of $660,000$ passengers per day, and transport more than 25% of all transit passengers, accounting for 45% of all transit fares paid. To fulfill such delivery requests, these taxicabs travel a total of roughly 800 million miles per year [61]. Unfortunately, with 25 MPG, these taxicabs consume about 32 million gallons of gas every year, more than the total annual gas consumption in some middle-sized countries (e.g., Central African Republic [63]), therefore leading to severely harmful tailpipe emissions and energy consumption. On the other hand, in carbon emissions trading under the Kyoto Protocol [64], governments will provide economic incentives for achieving reductions in the emissions of carbon pollutants. Thus, for both environmental and economic purposes, it is imperative to find a practical initiative to support the same delivery requests for taxicab transportation from passengers with lower total mileage and less carbon emissions.

In this chapter, we argue that a *taxicab carpool service* is a promising solution. The key advantage of a carpool service is that it can pool groups of several passengers heading in the similar direction into *one* rather than *several* taxicabs. In other words, a carpool service provides a valid solution for delivering the same number of passengers with lower total mileage and thus less gas. The economic incentive for drivers is that groups of passengers can pay a higher aggregated fare, whereas the incentive for passengers is that every passenger will pay less than in a non-carpool situation. With an effective fare model, we can achieve a win-win situation. Furthermore, carpools can also improve the availability of taxicab service during rush hours and after major events.

Admittedly, taxi carpool is not a new concept and has been around for years. But they are mostly negotiated by individual drivers and passengers in an *ad hoc* manner without a facilitating infrastructure. Until now, we have lacked a systemic study of carpooling in large scale taxicab networks. We note that many studies have focused on taxicab scheduling [28] [31] [33] [34] [29] [32] [35] or novel systems taking advantage of taxicab mobile traces [5] [41] [42] [36] [39] [37] [65] [38] [40], but little research has been done on taxicab carpool services with a software and hardware co-design. In this chapter, we present the first systematic study of how to design, evaluate, and implement taxicab carpool services in real-world scenarios. Specifically, the key contributions of this chapter are as follows:

- To the best of our knowledge, we conduct the first carpool service that considers the mutual benefits for passengers and drivers in large-scale taxicab networks and provide a comprehensive study of how to fill the same passenger delivery requests with less total mileage and thus less gas consumption. To achieve our goal, we develop customized hardware, TaxiBox, with multiple sensors and onboard devices (such as CDMA communication module, MIC, camera, and carpool fare meters). Further, we develop a passenger client app, which are used to locate on-duty taxis and to send carpool requests. Using these frontend devices, we design a taxicab carpool system, coRide, with a backend server to gather requests from passengers, to inform drivers of carpool requests, to calculate carpool routes for drivers, and to estimate carpool fares for passengers.

- In coRide, we introduce a mathematical concept *delivery graph* to represent a carpool route schedule for delivering passengers. Given requests provided by passengers, we seek an optimal delivery graph to achieve the minimum total mileage. We show that this optimization is NP-hard by linking it to the classic traveling salesman problem, and provide (i) an optimal solution with integer programming, (ii) a 2-factor approximation solution with a polynomial complexity, and (iii) an online algorithm to accommodate online streaming requests with a linear complexity. In addition, we consider different real-world constraints, e.g., passenger travel periods, number of available taxicabs, and taxicab capacities.

- Based on the carpool route, we propose a win-win carpool fare model to encourage both drivers and passengers to participate in carpooling. In this model, given a carpool benefit due to mileage reduction, passengers and a driver will share this benefit based on a ratio dynamically adjusted by the supply and demand relationships in a taxicab network.

- To test the performance of coRide in real-world setting, we implement coRide with a small-scale real-world trial in Shenzhen with 3 taxicabs and 12 passengers for 31 days. The results show that we reduce mileage by 49% comparing to the ground truth without carpool services.

- To test the performance of coRide at large-scale systems, We use a real-world dataset consisting of GPS traces from more than $14,000$ taxicabs in a Chinese city Shenzhen with a population of 13 million. The evaluation results show that compared with the ground truth, our carpool service reduces the total mileage by as much as 33%, and our win-win fare model lowers passenger fares by 49% and increases driver profit by 76% at the same time.

The rest of the chapter is organized as follows. Section 5.2 proposes the motivation for the design. Section 5.3 presents the coRide system overview. Section 5.4 depicts the passenger clients. Section 5.5 describes our customized device, TaxiBox. Section 5.6 explains the algorithms for carpool route calculation. Section 5.7 presents a win-win carpool fare model. Section 5.8 shows real-world small-scale implementation about our coRide system. Section 5.9 validates our service with a big dataset. Finally, we conclude this chapter in Section 5.10.

5.2 Motivation

In this section, based on two datasets about traces and fares of $14,000$ taxicabs in the Chinese city Shenzhen with a population of 13 million, we first introduce the basic properties of large scale taxicab networks in dense urban areas. Then, we present evidence about the inefficiencies of current taxicab networks, demonstrate opportunities for carpools to address these inefficiencies, and identify challenges to facilitate carpooling in current taxicab networks. Details of the datasets appear in Section 3.2.

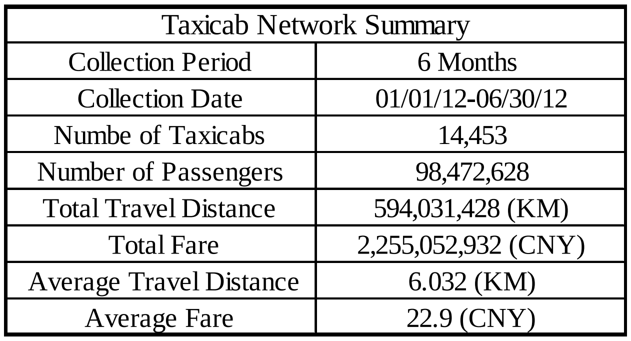
5.2.1 Drawback of Taxicab Networks

In developed countries such as United States, taxicabs are usually used to serve passengers to airports, and personal vehicles are used for other activities, excepting extreme large cities such as New York City. But in developing countries, due to high costs of owing personal vehicles, taxicabs and other public transportation are popular for daily activities. In dense urban areas such as Beijing, taxicabs are affordable for local traveling with an initiate fare about 2 USD for a 3 KM trip, and are more comfortable than other public transportation with cheaper fares (such as buses or subway). Due to the popularity and the affordability of taxicab services, the number of taxicabs in a taxicab network of a large city is typically more than 10,000. Thus, these taxicabs can be easily found on streets at the most of time and locations (except in rush hours or in hot pickup spots) and are commonly used for shopping, traveling to and from the work or schools, and other daily activities.

We discuss the inefficiencies from three perspectives, i.e., society, drivers and passengers, based on a taxicab dataset shown in Figure 5.1.

For society, the key inefficiency of taxicab networks is the large gas consumption of *a long-travel distance*. We observe that these taxicabs travel a total of 1.2 billion kilometers per year, consuming about 100 million liters of gas to deliver 200 million passengers and causing harmful tailpipe emissions. Figure 5.2 gives the total travel distance of all taxicabs in the network on an hourly basis.

For drivers, the key inefficiency is *low profits*, which are decided mainly by delivery distances (i.e., the mileage with paying passengers). Intuitively, drivers should earn more profit in rush hours, but this is not the case in large cities with severe congestion.



Taxicab Network Summary	
Collection Period	6 Months
Collection Date	01/01/12-06/30/12
Numbe of Taxicabs	14,453
Number of Passengers	98,472,628
Total Travel Distance	594,031,428 (KM)
Total Fare	2,255,052,932 (CNY)
Average Travel Distance	6.032 (KM)
Average Fare	22.9 (CNY)

Figure 5.1: Taxi Data Statistics

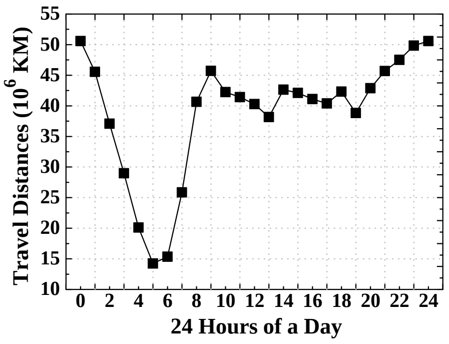


Figure 5.2: Travel Distance

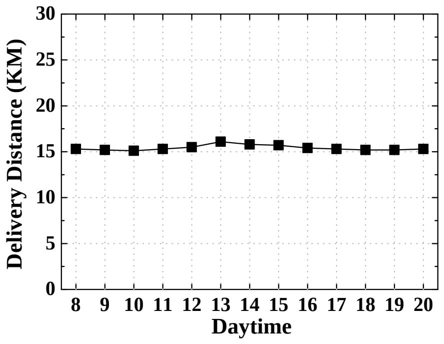


Figure 5.3: Delivery Distance

In regular hours without congestion, the total distance (i.e., also including total mileage without paying passengers) is high, but the percentage of delivery distance is low, since it is not easy to find a passenger. In rush hours with congestion, the percentage of delivery distance is high (i.e., easy to find a passenger), but the total distance is low due to the slow pace of traffic. Figures 5.3 shows the average delivery distance. It shows delivery distances at different times of day (i.e., rush and non-rush hours) are not significantly different.

For passengers, key inefficiencies are *high fares* and *low availability.* According to statistics about New York City [61] [62], the average fare is 11.44 USD for a 2.8 mile trip with an 11-minutes travel time, which is 5.8 times higher than the average public transit

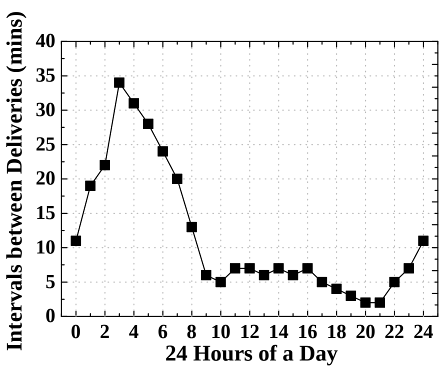


Figure 5.4: Delivery Intervals

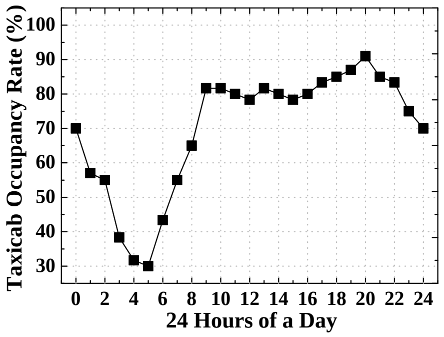


Figure 5.5: Occupancy Rate

fare (bus or subway) on average. In our statistics, the average fare of 22.9 Chinese Yuan (CNY) for taxicabs is 11 times of a bus fare of 2 CNY on average. These two datasets also provide some evidence about the low availability of taxicab services. First, in the time intervals between deliveries presented in Figure 5.4, a small interval indicates that a taxicab will pick up a new passenger right after it drops off an old passenger, i.e., low availability. In Figure 5.4, the average time interval in rush hours is small, less than 3 minutes, indicating low availability of taxicab services. Second, low availability can also be seen in the taxicab occupancy ratios in Figure 5.5, where a high occupancy ratio indicates fewer empty taxicabs. The ratios in Figure 5.5 indicate that more than 80% of taxicabs are occupied on average during the rush hour. Thus, Figures 5.4 and 5.5 indicate the low availability of taxicabs during the rush hour.

5.2.2 Opportunities for Taxicab Carpool

We show the opportunities that carpools provide to address the above inefficiencies. Specifically, we discuss three factors to show how likely carpooling services can be achieved in reality: (i) the distance between passengers' origins as well as the distance between passengers' destinations; (ii) the travel distances of shared routes between passengers; (iii) the passenger preference to the carpooling services. The benefit of carpooling servers can be further unleashed, if we have more passengers who (i) start from close origins or end at close destinations, and (ii) share the long distance common routes, and (iii) are willing to accept carpooling services.

Close Origins and Close Destinations

Based on the dataset, we show 200 consecutive trips to an airport in Figure 5.6 where most passengers came to the airport from the downtown and several hot spots.

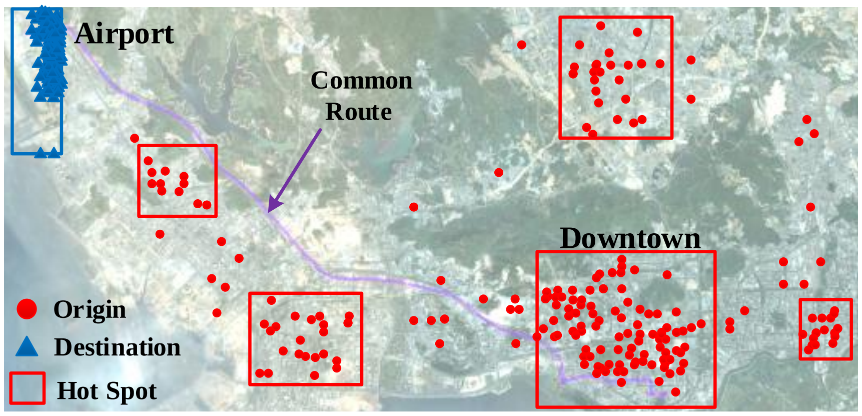


Figure 5.6: Trips to an Airport

Similarly, we show 200 consecutive trips from an airport in Figure 5.7. We observe the similar phenomenon that the most passengers came to the downtown and several hot spots from the airport.

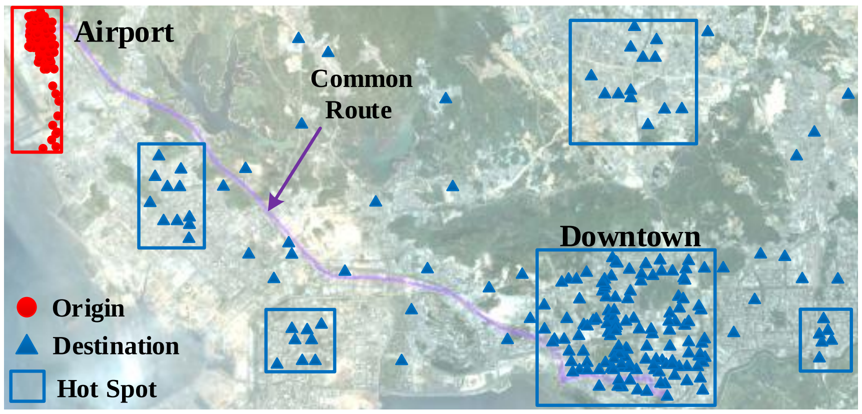


Figure 5.7: Trips from an Airport

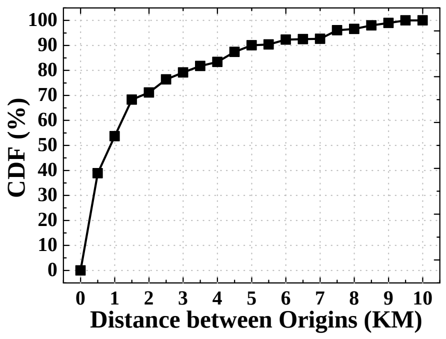


Figure 5.8: Close Origins

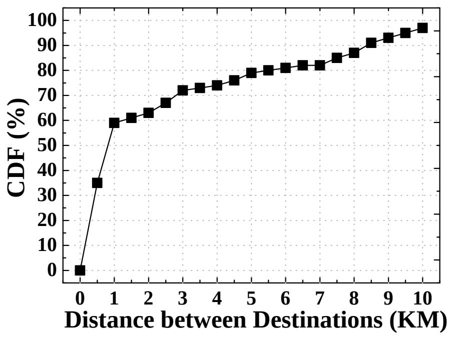


Figure 5.9: Close Destinations

In Figure 5.8, we show the CDF of distances between the origins of $1,000$ trips to the airport. Similarly, almost 50% of the trips have an origin closer than 1 KM to another origin, and almost 90% of trips have an origin closer than 5 KM to another origin. In Figure 5.9, we show the CDF of distances between destinations of $1,000$ trips from the airport. 60% of trips have a destination closer than 1 KM to another destination, and 80% of trips have a destination closer than 5 KM to another destination.

Shared Routes

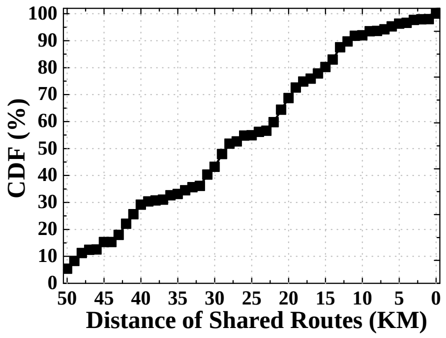


Figure 5.10: Shared Distance to Airport

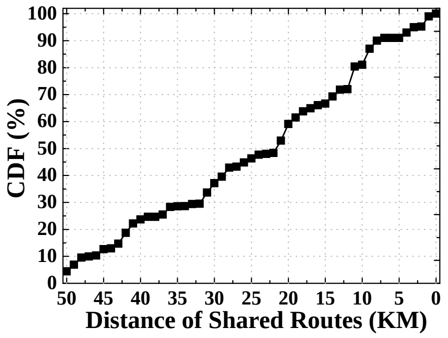


Figure 5.11: Shared Distance from Airport

Based on the dataset, Figure 5.10 shows the CDF of distances of shared routes of $1,000$ trips to the airport. More than 90% of trips share at least 7.5 KM with another trip, and more than 50% of trips share at least 20 KM with another trip. Figure 5.11 shows the CDF of distances of shared routes of $1,000$ trips from the airport. Similarly, more than 90% of trips share at least 5 KM with another trip, and more than 50% of trips share at least 20 KM with another trip.

From the above figures, we observe a good opportunity for carpools to benefit multiple parties. For passengers, a taxicab carpool can increase the availability of taxicab services in extreme weathers, peak hours or hot pickup locations, reducing the waiting time for passengers; in addition, multiple passengers in a carpool can share the fare together, reducing the fare paid by individual passengers. For taxicab drivers, a taxicab carpool can increase profits, since the aggregated carpool fare is higher than the regular service fare with the same travel distance. For operators, a taxicab carpool can provide more transportation capacity and enable more efficient gas consumption. Note that taxicab carpools do not aim to completely replace the traditional taxicab services, but serve as a key supplement for the situations where regular taxicab services are insufficient in peak hours or extreme weathers, or situations where some passengers would like to take transportation that is cheaper than traditional taxicab services yet more convenient than bus and subway.

Although taxi carpooling is not well supported at regulatory levels in many countries, according to a taxi pooling survey taken in Beijing [66], 75% of interviewees accept carpooling services; 57% of interviewees have carpooled with others at least once; 73% of interviewees accept a simple carpooling price mechanism where every passenger pays 60% of the regular service fare for the shared distance, leading to extra profits for drivers; several key concerns about carpooling pointed out by more than half of interviewees are as follows (i) prolonged travel time (64%), (ii) hard to find passengers to carpool or hard to find carpoolable taxicab (50%), and (iii) unable to print duplicated receipts for all passengers (50%). Based on the above survey, we find that most passengers are willing to accept carpools and to share the benefits of carpools with co-riders and the driver, but we still face several challenges to enable a practical carpool service in large-scale taxicab networks, which we will introduce next.

5.2.3 Challenges for Taxicab Carpool

We present three challenges and some possible solutions for implementing carpool services in the current taxicab networks.

Acquisition of Detailed Status about Taxicabs: To find the most suitable taxicab, a dispatching center should acquire detailed information about taxicabs, e.g., how many seats are left, which is difficult to obtain in current taxicab networks, where only the general taxicab status (e.g., locations, speeds, with passenger or not, etc) can be obtained by dispatching centers through real-time GPS record uploading. Thus, an onboard device should be installed in taxicabs to let dispatching centers monitor the detailed status of taxicabs and find the most suitable taxicab.

Carpool Route Calculation: After finding the most suitable taxicab, a dispatching center should calculate the optimal carpool route based on multiple received requests in a centralized way, and send the calculated route to a driver. This route should be efficient in terms of the total distance to deliver all assigned passengers. In addition, the calculation of routes should be fast enough to enable a responsive taxicab carpool service.

Fare Estimation and Calculation: With a carpool route schedule, dispatching centers should notify passengers with fare details of several carpool options for their approval. A win-win carpool fare model that estimates fares is missing in the taxicab business. Further, current fare meters can calculate only a single fare, and a more advanced fare meter that can record multiple concurrent trips is desirable for carpooling.

To address these challenges, we aim to develop a carpool system, coRide, as a hardware and software co-design with a passenger client app and an front-end onboard device called TaxiBox, along with a back-end cloud server to upgrade current taxicab networks. coRide employs multiple sensors and devices attached to TaxiBox to effectively manage taxicab networks. We provide an overview of coRide in Section 5.3.

5.3 Service Overview

In this section, we present an overview of coRide, which consists of three key parts: a cloud server, passengers clients, and onboard TaxiBox devices as shown in Figure 5.12.

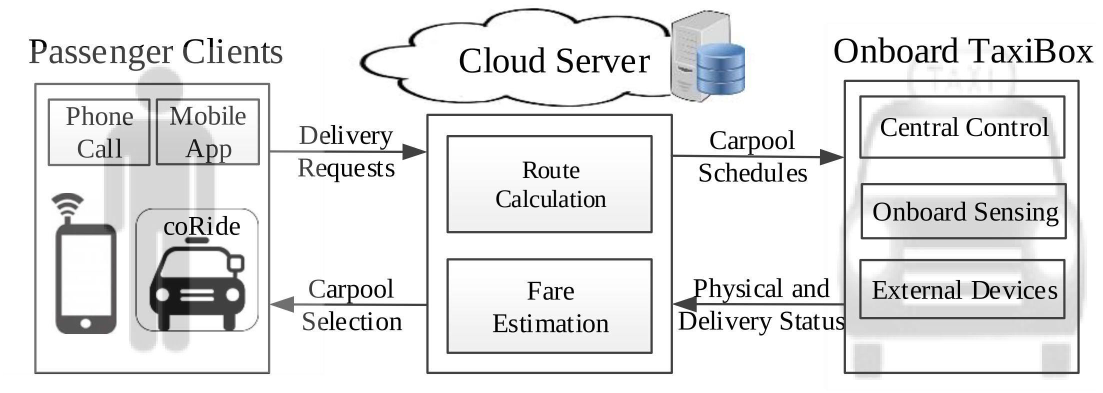


Figure 5.12: coRide Overview

Passenger Client: Passenger participation is required by our design, which can be encouraged by our win-win fare model discussed later. Assuming that passengers will be willing to participate, they will provide delivery requests to the dispatching center. The most common way to provide delivery requests is to call the dispatching center by phone to provide the number of passengers, pickup time, origin, destination, and possible delivery deadline. Further, mobile apps can be used to provide delivery requests without calling the dispatching center. Based on the delivery requests provided by passengers, the dispatching center will return a carpooling option with a reduced fare for their approval, along with a non-carpool option with a regular fare for comparison. An example of such passenger clients is given in Section 5.4.

Onboard TaxiBox: When a carpool is approved by passengers, a dispatching center will locate a suitable taxicab for the carpool based on the current status of taxicabs and then send a carpool route schedule to this taxicab's TaxiBox. The driver will respond to this carpool request by changing the status of the taxicab and then performing the carpool route schedule. These functions are performed by three key components of TaxiBox, which will be introduced in detail in Section 5.5.

Cloud Server: In this chapter, we will focus on function designs for a cloud server at dispatching centers with an emphasis on taxicab carpool services rather than regular services. In our carpool service design, a cloud server is mainly in charge of
(i) receiving delivery requests from passengers;
(ii) calculating carpool routes based on delivery requests;
(iii) estimating carpool fares for passengers to approve;
(iv) sending carpool routes to suitable taxicabs;
(v) obtaining the physical and delivery status of taxicabs.

In the rest part of the chapter, we present the detailed passenger client and TaxiBox design in Sections 5.4 and 5.5; for the dispatching cloud server, we describe three key functions, the carpool route calculation and fare model in Sections 5.6 and 5.7, respectively.

5.4 Frontend Passenger Client

We embed coRide as a carpooling service into one of our taxicab-booking app for taxicab passengers about the taxicab network in Shenzhen. In Figure 5.13, we show the app screenshots about on-duty taxicabs checking, carpooling request submitting, and phonecall for services.

For the on-duty taxicab checking function, based on the access of real-time data of taxicabs in Shenzhen, we analyze both locations and status of all taxicabs and show them with different icons based on their status on a Shenzhen city map. For the carpooling function, based on the location and time entered by users, the app sends a carpooling request to the cloud server, which returns a carpooling schedule based on the status of taxis. The current version of our app is in test based on the regulation of Shenzhen city. Although our app is specifically designed for Shenzhen, the key functions, e.g., on-duty taxi checking and carpooling request submitting, can be generalized to other cities as well.

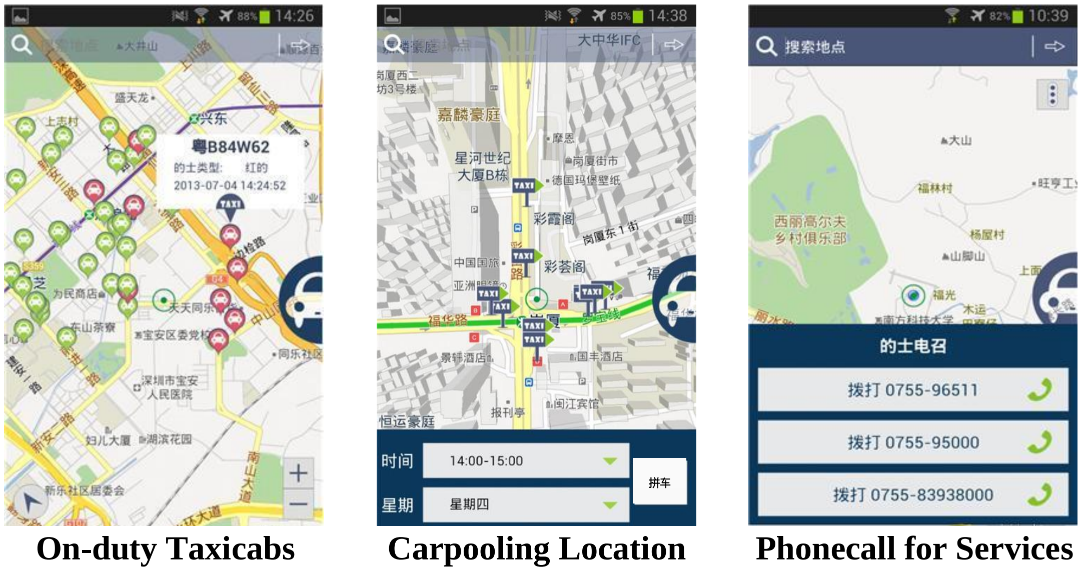


Figure 5.13: App Screenshot

5.5 Frontend Onboard TaxiBox

In this section, we present our hardware design, and then show the deployment about TaxiBox, and final propose the capability of taxicabs with TaxiBox.

5.5.1 TaxiBox Design

As shown by Figure 5.14, our onboard device, TaxiBox, consists of three main parts: central control system, onboard sensing system, and external devices.

The central control system has two key parts, the power module and the CPU module. For the power module, we employ TPS54160 from Texas Instruments, which is a 60V, 1.5A, step down SWIFT DC/DC converter with an integrated high-side MOSFET. For the CPU module, we use a 32 bit 72 MHz processor STM32F103 from ARM Cortex-M3 processors with A/D Convertors of 12-bit accuracy.

The onboard sensing system has open interfaces to multiple sensors, and the current hardware has (i) alcohol and smoke sensors, (ii) a $\pm$ 2g triaxial acceleration sensor, and (iii) a camera and a microphone. Based on the above sensors, a dispatching center is capable of monitoring the comprehensive physical status of a taxicab on streets.

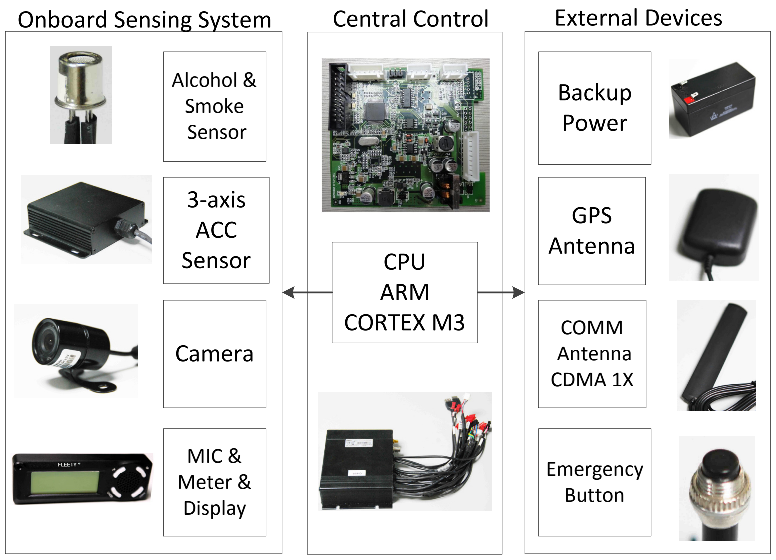


Figure 5.14: TaxiBox Hardware Design

Various external devices can be integrated into our TaxiBox. Some external devices in the current TaxiBox design include (i) a display and a speaker integrated to the display; (ii) a traditional fare meter for fare calculation and receipt printing; (iii) backup power for a situation in which the main power is not available; (iv) an emergency button; (v) a GPS module with a separate GPS antenna; and (vi) a CDMA 1X communication module with a separate antenna.

In some existing taxicab networks, the communication modules usually use GPRS (e.g., for GPS coordinates uploading) between taxicabs and dispatching centers. But in our design, taxicabs typically have a larger dataset to upload to or download from a dispatching center. Thus, a CDMA 1X, instead of GPRS, communication module is attached to TaxiBox, since CDMA employs different channels for voice and data communications, which clearly has advantages in terms of communication speed and stability, compared to GPRS that employs the same channel for data communications.

5.5.2 TaxiBox Deployment

We have deployed our TaxiBox in 98 taxicabs as shown by Figure 5.15. The alcohol and smoke sensors are installed in the ceiling of taxicabs for better sensor functions.

The camera is in front of passengers so as to take pictures from a better angle. The main part of TaxiBox is hidden above the glove box. The display is installed above the air-conditioner control panel for easier access by drivers. The 3 axis acceleration sensor is hidden under the glove box.

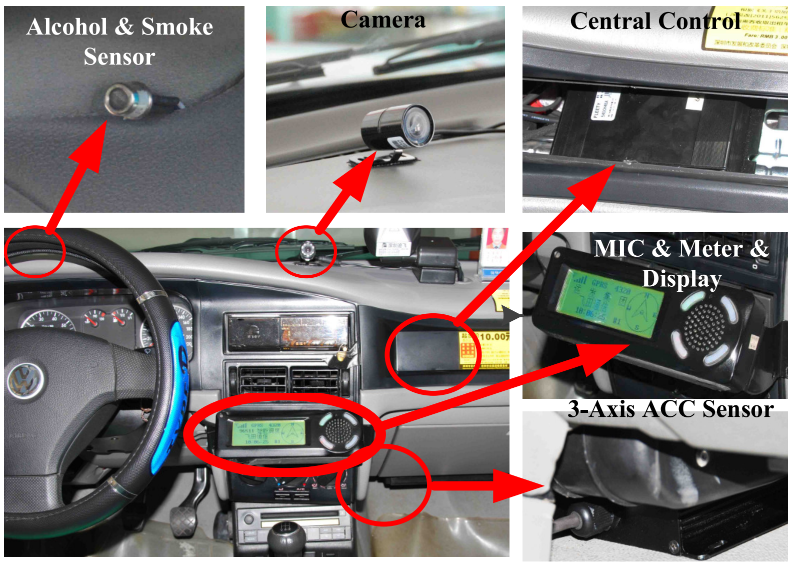


Figure 5.15: TaxiBox Deployment

5.5.3 TaxiBox Capability

In this section, based on the hardware we deployed in taxicabs, we introduce the capabilities of TaxiBox.

Taxicab Physical Status Sensing: Dispatching centers should be fully aware of the status (e.g., location, speed, *etc.*) of taxicabs to provide better carpool service. With GPS and CDMA 1X modules onboard, a taxicab can periodically upload its real-time physical status to a dispatching center. The onboard traditional fare meter and TaxiBox with a display can function together as a smart meter that can record the status of several trips, i.e., the delivery distance and fare for different passengers onboard, whereas the traditional fare meter can only record a single delivery distance. Further, a speaker is integrated into the display so dispatching centers can issue a voice schedule or voice navigation.

Taxicab Delivery Status Sensing: In addition to a taxi's physical status, a dispatching center is also interested in the real time status of its deliveries. The status of deliveries includes delivery distances, with passengers or not, fare, duration, start time, end time, pickup and dropoff location, which all can be obtained by TaxiBox and uploaded to dispatching centers.

In current taxicab status design, the status of taxicab about passengers is either 0, indicating no passenger onboard, or 1, indicating passengers onboard, but exactly how many passengers cannot be obtained. But unlike regular taxicab services, for a carpool service, the number of passengers in a trip is another key status of taxicab trips, since it decides how many other passengers can be pooled into the same taxicab. But in the current taxicab network, the only way for a dispatching center to obtain the number of passengers in a taxicab is to contact working drivers, which can create a dangerous distraction for driving.

With capabilities of TaxiBox in our design, we propose the solution of counting passengers with the camera of TaxiBox. When passengers first enter a taxicab, the driver will reset the fare meter and the camera will automatically take a picture to show the current image inside the taxicab. With a face detection algorithm [67], a TaxiBox can obtain the number of passengers inside a taxicab and change the status of taxicabs from 0 to the number of passengers onboard. The reasons we process images in TaxiBox without uploading them to dispatching centers are twofold: first, uploading images costs much more communication traffic than changing a number in GPS trace records (20KB extra vs. 0 extra uploading), and the amount of data a TaxiBox can upload via CDMA is limited by a monthly payment plan (30MB per month); second, uploading images may lead to privacy violations. A picture taken from a camera inside a taxicab is shown in Figure 5.16 where with a face detection algorithm, we can count the number of passengers with TaxiBox.

Under taxicab scenarios, when the camera takes pictures, several passengers could face in different directions, so a multi-view face detection algorithm has to be considered. The face of passengers can also be hidden by other objects such as seats, so we can improve the results by taking multiple pictures. In this chapter, we focus mainly on taxicab system design, instead of image processing, so we employ a state-of-the-art face detection algorithm [67] to serve our carpool service. Its performance curve is given in

Figure 5.16: Picture inside Taxi

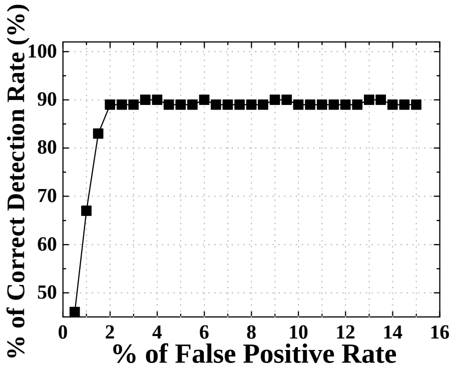


Figure 5.17: Detection Performance

Figure 5.17 with different thresholds for a false positive rate.

5.6 Backend Cloud Server

In this section, we focus on the cloud server design in term of carpool route calculation. We first propose preliminaries about our carpool work, then define a carpool route calculation problem, and finally propose its solution.

Carpools can be classified into four categories:
(i) one origin to one destination ($1 \to 1$);
(ii) one origin to many destinations ($1 \to N$);
(iii) many origins to one destination ($N \to 1$);
(iv) many origins to many destinations ($N \to N$).
For the sake of presentation, we will focus on $1 \to N$ because (i) $1 \to 1$ is a special case of $1 \to N$; (ii) $N \to 1$ can be solved with $1 \to N$ by reversing origin and destination; and (iii) $N \to N$ can be solved with a special $1 \to N$ with constraints on the order in which to visit all origins and destinations. Without loss of generality, we use $1 \to N$ model (e.g., carpool passengers from an airport) as an example for the design.

5.6.1 Preliminaries

For a carpool, a passenger will provide a *delivery request* with an origin, a destination, a start time and an optional end time (A possible end time serves as a deadline for delivery, but our model works with an unknown end time). Thus, given several requests for carpooling from the same origin as in Figure 5.18(a), we shall analyze distances between their destinations, which can be shown as a complete graph. We construct this complete graph as shown in Figure 5.18(b) by (i) treating both origin and destinations as vertices, and (ii) linking all vertices to each other with directed edges, associated edge weights with pairwise mileage costs.

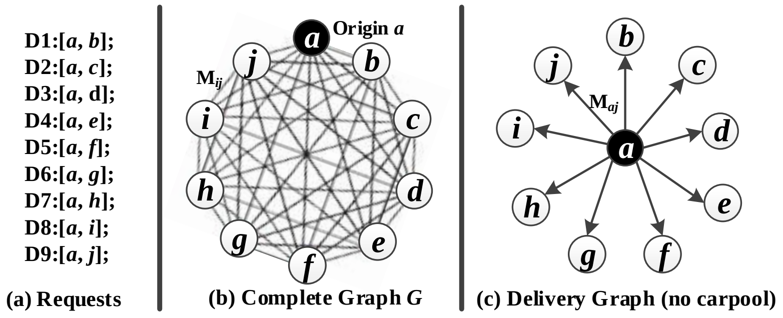


Figure 5.18: Complete Graph

Subfigures (a) and (b) in Figure 5.18 give an example of how to create a complete graph based on 9 delivery requests from the origin a. A weight on an edge (e.g., M_{ij}) indicates the real-world mileage between two locations. Given the complete graph, we can obtain a carpool route based on a *delivery graph*, which is defined as follows.

Definition 1: Delivery Graph: With a complete graph G given by delivery requests, a *delivery graph* is a subgraph of G where (i) the origin vertex can reach all destination vertices; (ii) no branches exist at any vertex but the origin vertex (i.e., spoke topology)

With this definition, a delivery graph uniquely indicates a carpool route where the total carpool mileage is equal to the sum of all its edges' weights. In Definition 1, the condition (i) is to make sure that with a carpool route, all passengers can be delivered from the origin to their destinations; the condition (ii) is to make sure that every passenger will take only one taxicab during the carpool, i.e., without relay. Subfigure (c) in Figure 5.18 gives an example of a delivery graph without carpool, i.e., all passengers are delivered by separate drivers with separate mileages, e.g., M_{aj}.

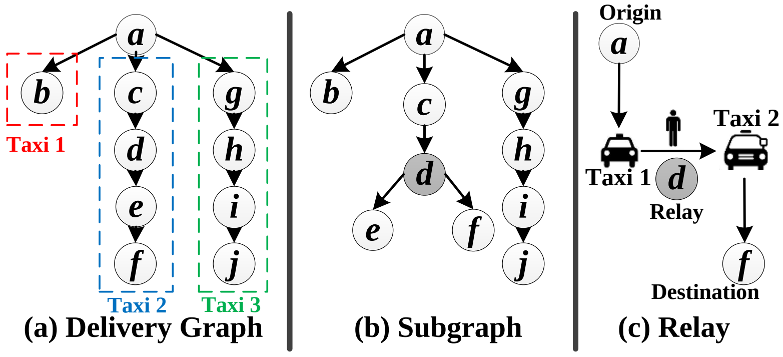


Figure 5.19: Delivery Graph with Carpool

The examples of a delivery graph DG with carpool are given in Subfigure (a) of Figure 5.19. In DG, the origin vertex a can reach all destination vertices, and no branches exist at any destination vertex. A delivery graph (e.g., DG) indicates a real-world carpool by specifying (i) a *passenger assignments* for taxicabs, and (ii) a *delivery order* for a taxicab's passenger assignment. For example, DG shows that three taxicabs fulfill passenger requests with destinations on three paths (the total weight on edges of a path indicates the real-world mileage): Taxi 1 delivers passengers to b, with a mileage M_{ab}; Taxi 2 delivers passengers to c, d, e and f, with a mileage $M_{ac}+M_{cd}+M_{de}+M_{ef}$; Taxi 3 delivers passengers to g, h, i and j, with a mileage $M_{ag}+M_{gh}+M_{hi}+M_{ij}$.

Subfigure (b) of Figure 5.19 gives an example of carpools prohibited by coRide. To carpool by this subgraph, we have to at least use two taxicabs to deliver passengers to c, d, e and f: the first taxicab delivers passengers with destination c, d, and e, but carries only the passenger with destination f from origin a to an intermediate vertex d; the second taxicab has to pick up this passenger at vertex d (a relay), and then deliver him or her to destination f as shown in Subfigure (c).

We argue that the carpool service with a relay is not practical in real world, because (i) the relayed passenger has to pay multiple times to different drivers, and (ii) the coordination between taxicabs would lead to a large layover delay. Therefore, coRide supports only the delivery graphs with a spoke topology to indicate a practical real-world carpool route schedule for drivers to deliver passengers without a relay.

5.6.2 Carpool Route Calculation Problem

Based on the delivery graph proposed in the last subsection, we propose our carpool route calculation problem: **Given a complete graph based on delivery requests, find the minimum weight delivery graph.**

The complete graph can be easily constructed based on delivery requests provided by passengers; as a subgraph, a delivery graph specifies passenger assignments and delivery orders to fulfill all delivery requests; the minimum total weight of a delivery graph indicates it fulfills all requests with a carpool spending the minimum total mileage. To perform a practical carpool, we also consider three constraints as follows.
(i) ***Taxicab Capacity*** c; it shows how many passengers can be pooled into one taxi.
(ii) ***Number of Available Taxicabs*** n; it shows how many taxicabs can be used for carpool at the origin.
(iii) ***Travel Period*** $[t_i^s, t_i^e]$; it shows the earliest pickup time t_i^s and the latest dropoff time t_i^e for a delivery request i.

Based on the above discussion, our carpool route calculation problem is related to a *multiple traveling salesmen problem* (called mTSP where multiple salesmen start from a depot to visit different cities with the minimum total distance [68]) with the special carpool constraints. An mTSP is generally solved with Integer Programming to the optimal solution. But for our large scale setting, the optimal solution results in a long running time, since it is NP-Hard. Thus, an approximation algorithm should be used to obtain a delivery graph within a reasonable time.

Another key feature of our carpool route calculation problem is that instead of booking a carpool trip a day or two in advance, some passengers may provide *online* delivery requests just tens of minutes before the starting time of their deliveries. So an *online* algorithm is also necessary. Therefore, the design agenda about our solution to the carpool route calculation problem is given as follows:
(i) we use Integer Linear Programming to formulate our carpool route calculation to obtain the optimal solution in Section 5.6.3;
(ii) for a practical (faster) solution, we propose a 2-factor approximation algorithm to obtain a sub-optimal solution in Section 5.6.4;
(iii) to consider online requests, we present our online algorithm in Section 5.6.5.

5.6.3 Optimal Algorithm

We formulate our Carpool Route Calculation with following definitions.
(1) $G = (V, A)$: a weighted complete graph where vertex a is the origin vertex where a carpool starts and $V' = V - \{a\}$ is the set for destinations, and a weight on A indicated as c_{ij} is the real-world mileage cost from vertex i to vertex j ;
(2) $x_{ij} = 1$ if edge $(i, j) \in A$ is used; $x_{ij} = 0$ otherwise;
(3) $[t_i^s, t_i^e]$: a travel period for a passenger to vertex i;
(4) n: the number of available taxicabs; c: the taxicab capacity;
(5) y_i: total number of dropped passengers before vertex i;
(6) q_i: total number of dropped passengers at vertex i;
(7) p_i: time arriving at vertex i;
(8) w_i: latest starttime of dropped passengers before vertex i;
(9) $T(i, j)$: travel time between vertex i and vertex j.

$$\min \quad \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$\text{s.t.} \quad \sum_{i\in V} x_{ij} = 1 \qquad \forall j \in V' \tag{5.1}$$

$$\sum_{j\in V} x_{ij} \in \{0,1\} \qquad \forall i \in V' \tag{5.2}$$

$$\sum_{j\in V'} x_{aj} \le n \qquad \sum_{i\in V'} x_{ia} = 0 \tag{5.3}$$

$$y_i + q_i \le c \qquad \forall i \in V' \tag{5.4}$$

$$\text{If } x_{ij} = 1 \Rightarrow y_i + q_i \le y_j \qquad \forall i, j \in V' \tag{5.5}$$

$$p_i \le t_i^e \qquad \forall i \in V' \tag{5.6}$$

$$\text{If } x_{ij} = 1 \Rightarrow p_i + T(i, j) \le p_j \qquad \forall i, j \in V' \tag{5.7}$$

$$\max\{w_i, t_i^s\} + T(a, i) \le t_i^e \qquad \forall i \in V' \tag{5.8}$$

$$\text{If } x_{ij} = 1 \Rightarrow w_i \le w_j \qquad \forall i, j \in V' \tag{5.9}$$

$$\sum_{i\notin S}\sum_{j\in S} x_{ij} \ge 1 \qquad \forall S \subseteq V' \tag{5.10}$$

$$x_{ij} \in \{0,1\} \qquad \forall i, j \in V \tag{5.11}$$

where (4.1) ensures that exactly one taxicab visits a destination; (4.2) ensures that exactly one taxicab leaves one destination for the next delivery, or the carpool is over and no delivery needs to be made; (4.3) is about the constraint on the number of available taxicabs; (4.4) and (4.5) are about the taxicab capacity constraint; (4.6), (4.7), (4.8) and (4.9) are about the travel period constraint; (4.10) is to prevent the formation of subtours not including origin vertex a. Note that though a taxicab has disjoint vertices in a delivery graph, they can share the same route in the real world when performing a carpool.

Since the traditional traveling salesmen problem is NP-Hard, as a generalized version, our problem is also NP-Hard (due to space constraint we omit the formal proof). Therefore, when the number of destinations increases, the running time to solve the above integer programming increases exponentially. Although integer linear programming is sufficient for a small number of destinations, we need to accommodate the case where the number of destinations is large with an efficient algorithm.

5.6.4 Approximation Algorithm

We first propose an approximation algorithm, and discuss impacts of three constraints.

Approximation Algorithm without Constraints

An approximation algorithm without constraints is under a scenario where all passengers' travel periods are not considered; the origin has unlimited taxicabs with unlimited capacities.

We first present some rationales. Any carpool from the same origin can be performed with two key steps: (i) we shall *assign passengers* to different empty taxicabs; (ii) we shall calculate a *delivery order* for a given passenger assignment for a particular taxicab. In the followings, we will describe how to make use of this rationale for our approximation algorithm in two steps and provide its approximation ratio.

(i) **Passenger Assignment**: Based on a complete graph G created with delivery requests, we will show how to assign passengers in Figure 5.20.

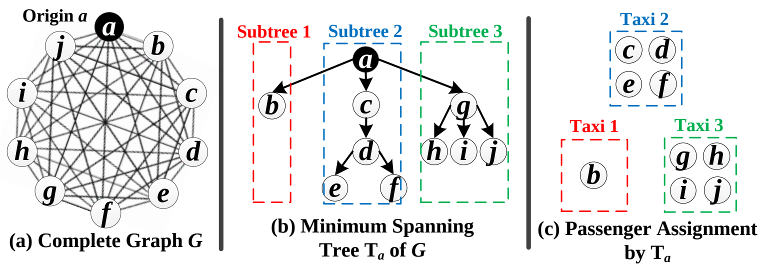


Figure 5.20: Passenger Assignment

To assign passengers to different taxicabs, we shall take into account the distances between destinations given by G in subgraph (a). The objective is to find a *minimum weight* subgraph of G to assign destination vertices to different paths (every path is used by an unique taxicab). Since the minimum spanning tree (MST) is the minimum weight subgraph of G, in this chapter we try to employ an MST to obtain a passenger assignment. Subgraph (b) gives a G's MST T_a with three subtrees rooted at origin vertex a. Based T_a, we assign the passengers, who have destination vertices in the same subtree, into the same taxicab. For example, passengers with destinations c, d, e and f

are assigned into Taxicab 2, as in Subgraph (c). Note that T_a is not a delivery graph we try to obtain, because T_a has branches at destination vertices. Thus, we have shown how to conduct passenger assignment based on a given complete graph.

(ii) **Delivery Order Calculation**: Based on the assignment in step (i), in Figure 5.21 we show how to calculate a delivery order for passengers in the same taxicab.

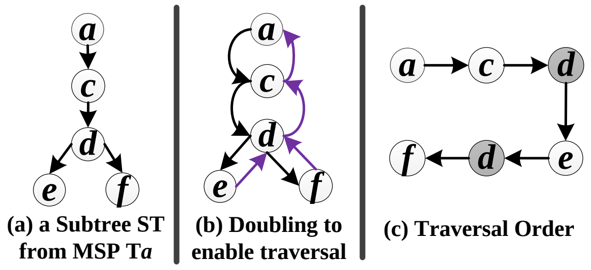
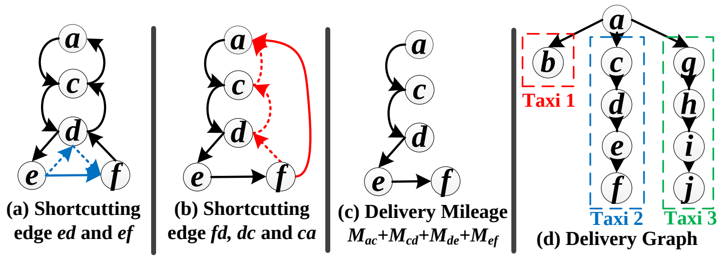

a
c
d
e
f
(a) a Subtree ST from MSP Ta
(b) Doubling to enable traversal
(c) Traversal Order

Figure 5.21: Delivery Order Calculation

As in step (i), a passenger assignment for a particular taxicab is given by a subtree rooted at the origin vertex a. Thus, we employ Subtree 2 (named ST) in the MST T_a in subfigure (a) as an example to show how to obtain a delivery order. With an observation on ST, we found that ST only gives a passenger assignment, but not a fixed delivery order since ST has a branch that requires passenger relay, which is prohibited by coRide. Thus, a new subgraph transformed from ST should be created to calculate an order without relay. In this chapter, we use a depth-first traversal from root vertex to decide a delivery order. But as we can see in subfigure (a), ST is a directed graph, and cannot be traversed based on current edges. Thus, as in subfigure (b), we double the edges in ST to create loops to enable a traversal $a \to c \to d \to e \to d \to f$. This order is not a delivery order since it involves duplicated vertices, i.e., d, thus a longer total mileage $M_{ac} + M_{cd} + M_{de} + M_{ed} + M_{df}$.

To obtain a delivery order, we use a shortcut strategy to eliminate duplicated vertices in a traversal. In Figure 5.22, we show how to shortcut some edges about duplicated vertices, thus further reducing a delivery mileage.

As in subfigure (a) and (b), we shortcut edge ed and df with a new edge ef, and then shortcut edge fd, dc and ca with another new edge fa. Note that based on triangle



(a) Shortcutting edge ed and ef
(b) Shortcutting edge fd, dc and ca
(c) Delivery Mileage $M_{ac}+M_{cd}+M_{de}+M_{ef}$
(d) Delivery Graph
Taxi 1
Taxi 2
Taxi 3

Figure 5.22: Shortcutting about Duplicated Vertices

inequality, the length of an added edge (e.g., M_{ef}) is always shorter than the sum of edges it shortcutting (e.g., $M_{ed}+M_{df}$). Further, we delete the edge fa to obtain the delivery order $a \to c \to d \to e \to f$ and the total mileage cost of 4 edges $(M_{ac}+M_{cd}+M_{de}+M_{ef})$ as in subfigure (c). Therefore, with a traversal, we have shown that how to calculate a delivery order based on a given passenger assignment. With the above two steps, we finish our approximation algorithm to obtain our delivery graph in subfigure (d).

Proof of Approximation Ratio: We have proved that our traversal algorithm has a constant performance ratio of 2, i.e., the total mileage obtained by our carpool schedule, is at most 2 times the optimal mileage we obtained by the optimal solution using integer programming. This is because (i) with shortcutting, the weight of our delivery graph $W(S)$ is smaller than a weight of a Traversal $W(T')$, i.e., $W(S) < W(T')$; (ii) a traversal is exactly two times of a MST, $W(T') = 2W(T)$; (iii) the MST is smaller than or equal to the Optimal solution since the optimal solution is a spanning tree and MST is the smallest spanning tree, $2W(T) \leq 2W(O)$. Thus, $W(S) < W(T')$ $=2W(T) \leq 2W(O)$, therefore $\frac{W(S)}{W(O)} < 2$.

During the construction of the minimum spanning tree, three constraints, i.e., Taxicab Capacity, Number of Available Taxicabs, Travel Period, have special impacts, which will be introduced in the following three subsections.

Impact of Number of Available Taxicabs n

In this section, we show how to solve a carpool problem with constraints on the number of available taxicabs n.

We can reduce the total mileage by delivering passengers separately, if they are heading in significantly different directions. In a delivery graph G, every subtree rooted at the origin is associated with a separate taxicab, which satisfies all delivery requests in this subtree. For example, in Figure 5.23, the spanning tree has three subtrees (boxed), and therefore we need three taxicabs to satisfy the deliveries.

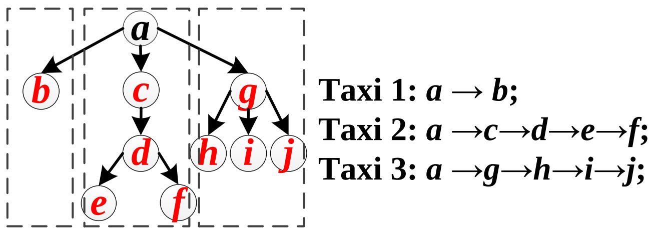


Figure 5.23: Minimum Number of Taxicabs for Deliveries

When constructing a spanning tree, we have to find one spanning tree whose number of subtrees rooted at origin is not bigger than n. Figure 5.24 shows how to impose such a constraint during a spanning tree construction.

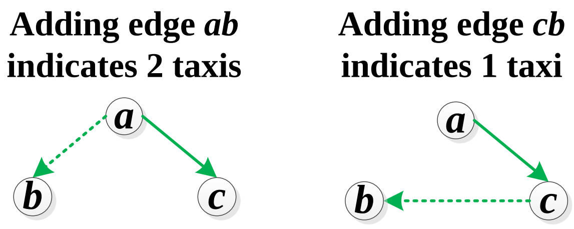

Adding edge ab indicates 2 taxis
a
b
c
Adding edge cb indicates 1 taxi
a
b
c

Figure 5.24: Constraints on Number of Available Taxicabs n

In Figure 5.24, given $n = 1$, i.e., there is only one taxicab available for origin a, suppose that after adding edge ac, currently the minimum edge that should be added to the tree is edge ab according to Prim's algorithm. But adding edge ab indicates that we need two taxicabs to fulfill the deliveries, which is against to $n = 1$. Alternatively, we can add edge cb and it will still fulfill the deliveries yet with one taxicab.

Note that if the only constraint is the number of taxicabs, there is always a spanning tree (e.g., a "path" graph where one taxicab takes all passengers) under the constraint of the number of taxicabs.

Impact of Taxicab Capacity c

In this section, we consider how to solve a carpool problem with constraints on the taxicab capacity c.

Since the taxicab capacity is limited (e.g., 4 for a sedan and 6 for a van), a delivery graph should not have infinite depth for any delivery branch. It is clear that given a fixed spanning tree, the minimum taxicab capacity is equal to the size of its biggest subtree rooted at the origin, because a taxicab has to deliver all passengers in this subtree from the origin. Figure 5.25 gives an example, where the biggest subtree has four vertex, therefore the minimum taxicab capacity is 4.

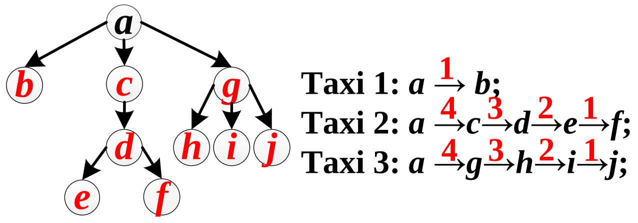

a
b
c
g
d
h
i
j
e
f
Taxi 1: $a \xrightarrow{1} b$;
Taxi 2: $a \xrightarrow{4} c \xrightarrow{3} d \xrightarrow{2} e \xrightarrow{1} f$;
Taxi 3: $a \xrightarrow{4} g \xrightarrow{3} h \xrightarrow{2} i \xrightarrow{1} j$;

Figure 5.25: Minimum Taxicab Capacity for Deliveries

When constructing a spanning tree, we have to control the sizes of subtrees to make sure the size of the largest subtree less than given capacity constraint c. Figure 5.26 shows how to consider it during the construction of a spanning tree.

In Figure 5.26, suppose $c = 1$ for simplicity, and suppose that after adding edge ac, currently the minimum edge should be added to the tree is edge cb according to Prim's algorithm. But adding edge cb indicates that we need taxicabs with capacity of 2 to fulfill the deliveries, which is against to $c = 1$. Alternatively, we can add edge ab and it will still fulfill the deliveries yet with capacity of 1.

Note that if the only constraint is capacity, there is always a spanning tree (e.g., a

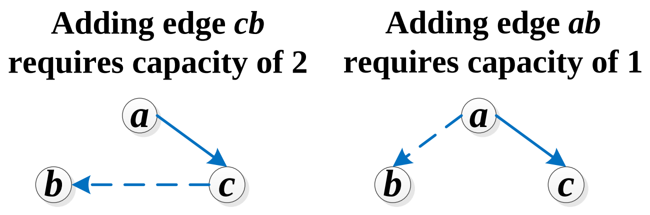

Adding edge *cb* requires capacity of 2
Adding edge *ab* requires capacity of 1
a
b
c
a
b
c

Figure 5.26: Constraints on Taxicab Capacity c

star graph where every taxicab only takes one passenger) under the taxicab capacity constraint.

Impact of Travel Period

In this subsection, we analyze the carpool problem with constraints on the travel periods of deliveries.

A travel period of a delivery i is specified by $[t_i^s, t_i^e]$, where t_i^s is the earliest time that a delivery i can start, and t_i^e is the latest time that delivery i must finish. The reason to consider travel periods is that in practice, two deliveries with non-overlapping periods cannot be carpooled together, even though they have the same origin and destination. Thus, a carpool schedule is valid only if its minimum spanning tree fits the travel periods of all deliveries on this tree, which needs to be validated when constructing the minimum spanning tree.

We first provide some rationale behind the validations. Due to a carpool, a taxicab has to leave origin vertex after the *carpool start time*, which is given by *the start time of last passenger* in this carpool. Given this carpool start time, the validation is based on the expected travel times of all deliveries. According a schedule based on the minimum spanning tree, if the carpool start time plus a travel time of a delivery i is smaller than or equal to delivery i's end time, then this spanning tree can accommodate delivery i.

To impose this constraint, for a minimum spanning tree T_a and a delivery i with travel period $[t_i^s, t_i^e]$ from origin vertex a to destination vertex i, we need to ensure that a spanning tree T_a can accommodate delivery i by satisfying:

$$\max_{k \in p} t_k^s + T(a,i) \leq t_i^e,$$

where p is a path on T_a from a to i, hence $\max_{k \in p} t_k^s$ is the start time of last passenger, and $T(a,i)$ is the travel time from a to i in p of T_a. The left-hand side is expected arrival time of delivery i by this carpool, and the right-hand side is the latest end time of delivery i, given by the passenger. Thus, if the left-hand side is not bigger than the right-hand side, it indicates that T_a can accommodate i. Figure 5.27 gives an example of how to validate whether the MST can accommodate a delivery or not.

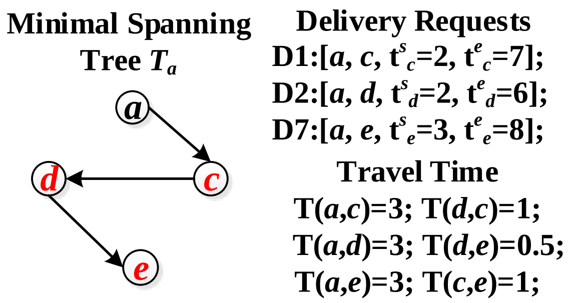

Minimal Spanning Tree T_a
a
c
d
e
Delivery Requests
D1:[a, c, t^s_c=2, t^e_c=7];
D2:[a, d, t^s_d=2, t^e_d=6];
D7:[a, e, t^s_e=3, t^e_e=8];
Travel Time
T(a,c)=3; T(d,c)=1;
T(a,d)=3; T(d,e)=0.5;
T(a,e)=3; T(c,e)=1;

Figure 5.27: Validation on Travel Period

In Figure 5.27, suppose that during the construction of a spanning tree, the next minimum edge should be added to the spanning tree according to Prim's algorithm is an edge de. Based on delivery requests and travel time in Figure 5.27, $\max_{k \in p} t_k^s = \max\{2, 2, 3\} = 3$; $T(a, e) = 3 + 1 + 0.5 = 4.5$; thus, $\max_{k \in p = \{a \to c \to d \to e\}} t_k^s + T(a, e) = 7.5 \leq t_e^e = 8$. Therefore, the edge de is a safe edge and can be added to the spanning tree.

Note that if the only constraint is the travel period, there is always a spanning tree (a star graph where every taxicab only takes one passenger) under this constraint.

Put All Constraints Together

A practical approximation algorithm shall construct a minimum spanning tree that (i) accommodates all travel periods of its deliveries, (ii) has the biggest size of subtrees not bigger than c, and (iii) has a number of subtrees not bigger than n. The details about how to impose the three constraints have been given in pervious section. We note that the order of imposing constraints should not be changed, since it is easiest to find more taxicabs to fulfill requests, relatively easier to find bigger taxicabs to fulfill requests, and harder to require passengers to change their schedules. If the conditions conflict with each other, we can always find a feasible solution by using more taxicabs. Note that with highly diverse travel periods or a small taxicab capacity, lots of taxicabs will be used to satisfy deliveries individually, which is a delivery schedule based on "fat" spanning trees with small yet many subtrees. In contrast, with a small number of available taxicabs, lots of deliveries will be pooled into one taxicab and then be fulfilled one by one, which is a delivery schedule based on "thin" spanning trees with big yet fewer subtrees.

5.6.5 Online Algorithm

Instead of providing requests a day or two earlier, some passengers may provide online requests a hour, or even several minutes, before the delivery start time. In *coRide*, we response to online requests by adding them to an existing carpool schedule. Given an online request k and a delivery graph about existing carpool schedules, our online algorithm has three key steps as shown by the example in Figure 5.28:

(i) Adding New Online Request to the Existing Delivery Graph: Based on the location of request k, we add k to the closest request f already in the delivery graph. This is the optimal solution for adding this online request. This is because if the optimal solution adds k to another request instead of f, we can always add k to f to obtain a smaller delivery graph, which is better than the optimal solution. So k must be added to f in the optimal solution.

(ii) Selecting a set of Close Requests regarding to the New Online Request: Based on the location of request k, we select a set of requests that are closer to the new online request k than any other requests already in the delivery graph. In this example, we select h and f as the close requests to k.

(iii) For Every Close Request, Adjust the Structure of the Delivery Graph for Small Weight: For every close request, we first find a route from this close request to the new online request. Then, we compare if the longest link between two requests on this route is longer than the link from the new online request to this close request.

If so, we add this link to the delivery graph, and delete the longest link. For example, in our example, for the close request h, we first find a route from h to k, which is from h to g to a to c to d to e to f to k. We select the longest link in this route, i.e., g to h, and then we compare it to the link from k to h. In our case, the distance from g to h is longer than that from k to h, so we add a new edge from k to h, and the delete the edge from g to h for a smaller total weight, as shown in the figure.

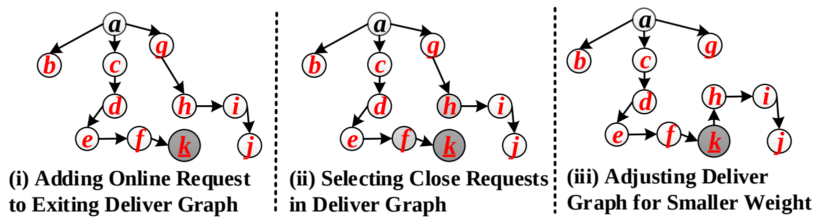

a
b
c
g
d
h
i
e
f
k
j
(i) Adding Online Request to Exiting Deliver Graph
(ii) Selecting Close Requests in Deliver Graph
(iii) Adjusting Deliver Graph for Smaller Weight

Figure 5.28: Online Algorithm

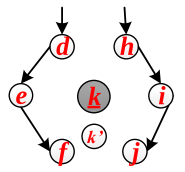


Figure 5.29: Close Request

The most important part of this online algorithm is its time complexity. Assuming we have $|C|$ close requests for an online request in Step (ii), and for every close requests, we at most go through all existing $|N|$ requests for the adjustment in Step (iii). Thus, the time complexity is $O(|C| \times |N|)$. But we can prove that $|C|$ is smaller than 7, so we have a linear time complexity $O(|N|)$. For example in Figure 5.29, for a new online request k, we at most have 6 close requests, which form a regular hexagon. If we have one more close request, e.g., k', it will make one of existing requests, i.e., f, closer to k' than k. Thus f is not a close request any more, which makes the total count of close requests smaller than or equal to 6.

5.7 Win-Win Fare Model

Generally, a taxicab fare consists of three main parts: an initial charge for every service; surcharge for luggage, waiting time, *etc*; and main charge based on traveled distance. In our model, we focus on how to consider a carpool benefit into calculations of the main charge. Such a carpool benefit shall be shared between the passengers (as a group) and the driver, as well as among the passengers themselves. The rationale behind sharing the carpool benefits with drivers is that we have to encourage drivers to participate in the non-mandatory carpool application. We believe that negotiating privately by passengers alone and sharing the benefits only between passengers will severely hurt the interests of drivers, since the total profit for all drivers will decrease significantly.

5.7.1 Carpool Benefit

A carpool benefit **B** between the total non-carpool fare and a fare paid for a carpool distance is given as follows:

$$\mathbf{B} = \sum_{i=1}^{c} \tau_i - \tau,$$

where c is the total number of passengers in this carpool; τ_i is the separate non-carpool fare for passenger i; τ is the regular fare for a distance equal to the carpool distance (not the carpool fare). Thus, the total non-carpool fare of all passengers is given by $\sum \tau_i$, and the regular fare for the carpool distance is given by τ, and their difference is a carpool benefit **B**. Given a carpool schedule, all three parameters are obtainable, and thus **B** is also obtainable.

For example, Figure 5.30 shows three passengers (with non-carpool fare $\tau_1 = 17$, $\tau_2 = 32$, $\tau_3 = 45$) carpooled together with a distance of a regular fare $\tau = 52$, leading to $\mathbf{B} = 42$. Note that $\tau = 52$ is a regular fare for a distance equal to the carpool distance, and is not the actual carpool fare all passengers will pay together under our model.

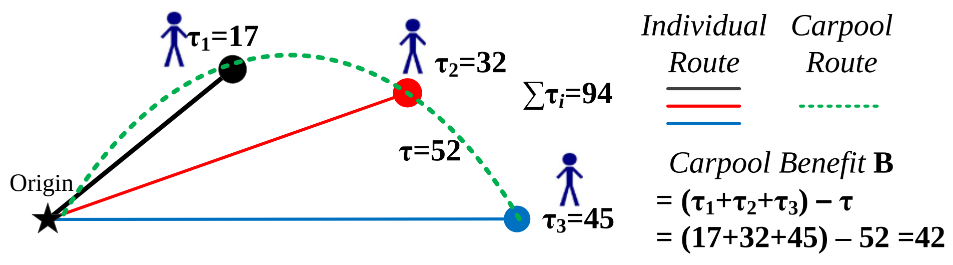


Figure 5.30: Carpool Benefit

To build a win-win fare model, we need to (i) share a carpool benefit between the driver and all passengers as a group and (ii) share the benefit within the passenger group.

5.7.2 Sharing Percentage between Driver and Passenger

We use ρ to indicate the sharing percentage of the passengers (all passengers as a group) for a given carpool benefit $\mathbf{B}$, and hence $1-\rho$ is the sharing percentage of the driver.
(i) For a carpool benefit $\mathbf{B}$, all passengers as a group pay:

$$\text{Total Fare Paid by Passengers} = \sum_{i=1}^{c} \tau_i - \rho \times \mathbf{B},$$

where $\sum \tau_i$ is the sum of regular fares by all passengers in a non-carpool situation; $\rho \times \mathbf{B}$ is the benefit to passenger group.
(ii) For a carpool benefit $\mathbf{B}$, a driver collects:

$$\text{Total Fare collected by Drivers} = \tau + (1-\rho) \times \mathbf{B},$$

where τ is the fare a driver collects for the carpool distance; $(1-\rho) \times \mathbf{B}$ is the benefit for a driver to carpool. Note it is easy to check that the total carpool fare paid by passengers equals the amount collected by the driver in our model.

In real-world scenarios, ρ can be dynamically decided based on various factors about the supply and request relationship in a taxicab network. In this chapter, we give an example to define $\rho = \frac{\text{\# of occupied taxicabs}}{\text{\# of total taxicabs}}$ in a certain area during a time window to balance the carpool incentives between the driver and the passenger. Thus, for a large ρ, i.e., more occupied taxicabs, the more benefit will be given to the passengers to encourage passengers to carpool; for a small ρ, i.e., more empty taxicabs, the more benefit will be given to the driver to discourage passengers to carpool, balancing deliveries among other empty taxicabs. In Figure 5.30, given $\rho = \frac{1}{2}$, the total carpool fare collected by drivers is $52 + \frac{1}{2} \times 42 = 73$, which is equal to the total carpool fare for all passengers, i.e., $94 - \frac{1}{2} \times 42 = 73$.

5.7.3 Sharing Percentage among Passengers

Among the total carpool benefits for all passengers, i.e., $\rho \times \mathbf{B}$, we shall decide a sharing percentage to show a carpool benefit for a particular passenger i, and thus model the carpool fare for a passenger i. It is given as follows.

$$\text{Carpool Fare Paid by a Passenger } i = \tau_i - \rho \times \mathbf{B} \times \frac{\tau_i}{\sum \tau_i},$$

where τ_i is the non-carpool fare a passenger i has to pay at a non-carpool situation; $\rho \times \mathbf{B} \times \frac{\tau_i}{\sum \tau_i}$ is the carpool benefit for a particular passenger i. In Figure 5.30, given $\rho = \frac{1}{2}$, the carpool fare paid by a passenger 3 is $45 - \frac{1}{2} \times 42 \times \frac{45}{94} \approx 34$.

Currently, we use $\frac{\tau_i}{\sum \tau_i}$ to share the carpool benefit among passengers based on their non-carpool fare. In other words, we differentiate passengers by their destinations to the common origin, not the delivery order. But the last dropped off passenger typically will have a farther destination than other passengers (since our carpool graph is based on the minimum spanning tree), so he/she will share more carpool benefit than other earlier dropped off passengers in our fare model, which implicitly compensates to the passengers with a longer traveling time. In more advanced designs, the sharing percentages among passengers can also be directly decided by the priority of services, e.g., based on the delivery order μ_i of passenger i in a carpool, the sharing percentage can be defined as $\frac{\mu_i}{\Sigma \mu_j}$.

5.7.4 Fare Model Evaluation

In this subsection, we numerically evaluate our fare model. Based on three delivery requests in Figure 5.30, Figure 5.31 shows the impact of different sharing percentages ρ on the fare that every passenger pays and the fare the driver collects. It shows when ρ increases from 0 to 1 (indicating a trend of an undersupplied taxicab services in the real world), the carpool incentive for the passenger increases from 0% fare savings to 44% fare savings, whereas the carpool incentive for the driver decreases from 80% more profit to 0% more profit. By adjusting sharing percentage ρ according to taxicab supply, our model can dynamically balance the carpool incentives for drivers and passengers.

Given requests with fixed non-carpool fares, a short carpool distance will increase the carpool benefit (the same $\sum \tau_i$, but a smaller τ), which results in a win-win situation (i.e., more profits for drivers and lower fares for passengers). Taking the passengers with $\tau_1 = 17$ and $\tau_2 = 32$ in Figure 6.22 as examples, physically, the lower bound of a fare paid for the carpool distances should be $\tau_{min} = \max\{\tau_1, \tau_2\} = 32$. In addition, passengers may not select a carpool delivery where they pay a fare together more than the sum of their regular non-carpool fares. So, logically, the upper bound for τ is $\tau_{max} = \tau_1 + \tau_2 = 49$. Figure 5.32 shows impacts of different carpool route distances (by different τ from τ_{max} to τ_{min}) on the savings percentages of passengers and profiting

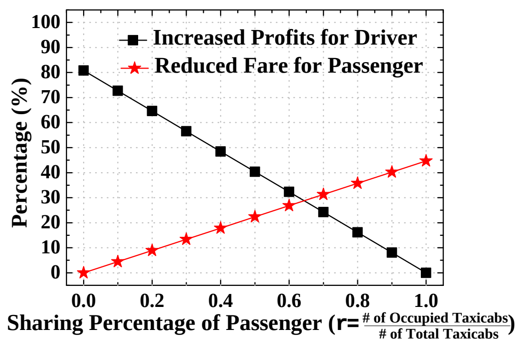


Figure 5.31: Incentive Balancing

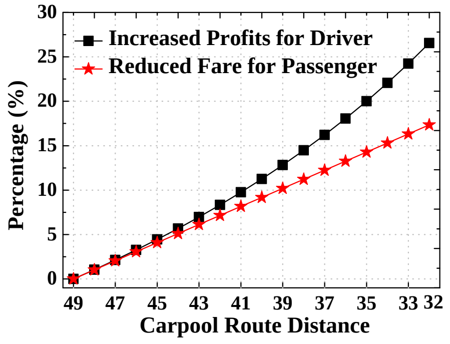


Figure 5.32: Win-Win Model

percentages of drivers. First, it shows a win-win situation as long as $\tau < \tau_{max}$. Second, the smaller τ, the higher the profit for drivers, the lower the fare for passengers.

5.8 Real-World Implementation

We have installed the customized TaxiBox in a small portion (98 taxicabs) of the taxicab network of a Chinese city Shenzhen with a population of 10 million to test the functionality of TaxiBox. We quickly learned that it takes time to install hardware in current taxicabs, and that it is much more difficult than we had anticipated. Although the taxicab operators requested that their drivers cooperate with the deployment, drivers still were not enthusiastic about installing devices to taxicabs with no immediate benefits to them. During the deployment, it was usual for drivers to not appear or to arrive late and leave early due to business or personal matters. It was also hard to persuade drivers to be more involved in system testing, e.g., logging passenger numbers for every delivery. How to provide an incentive for them to be involved in system deployment and testing is a key question we need to address.

For a large scale carpool deployment, through the operators from which we obtained datasets, the dispatching center to collect delivery requests via phone apps we introduced has been established. But the detailed regulation laws are still under progress to being passed, and hope to be completed within this year. Thus, a large scale carpool service evaluation is hard to conduct for the current situation. In this section, we describe our trial implementation of our coRide system. We rent 3 taxicabs to drive 12 volunteers from a subway station to their workplaces as in Figure 5.33.

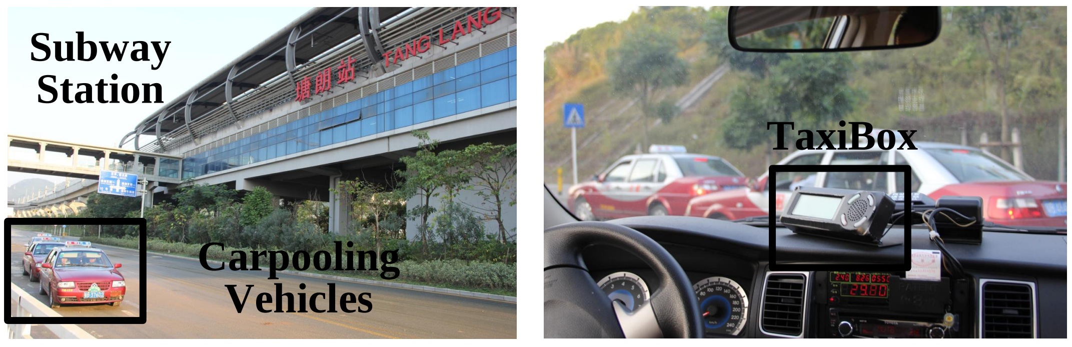


Figure 5.33: Experiment in Tanglang Station

Based on their final destinations, we formulate a request graph, and then obtain a delivery graph for them based on our approximation algorithm under several constraints, e.g., a vehicle count of 3, a vehicle capacity of 4, and a tolerated waiting time of 5 mins. The taxicabs will go back to the subway station until all volunteers are delivered. We plot the data of the 12 involved volunteers for a 31 day period evaluation in Figure 5.34. We use a metric called the percentage of reduced total mileage, which is obtained by the total mileage used to deliver all passengers with carpool, and the total mileage used to deliver all passengers without carpool. Due to different combinations of volunteers based on their starting time, the percentage of reduced mileage is different at different days even though their origins and destinations are the same. On average, we reduce mileage by 49% for all passengers.

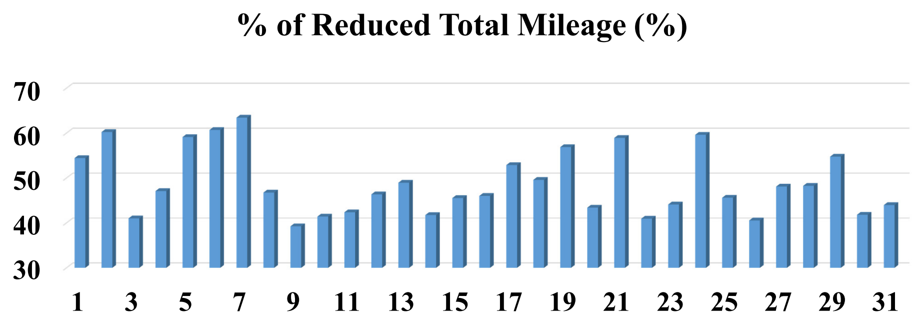
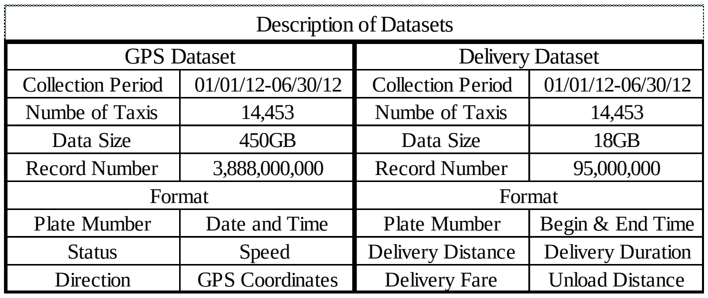


Figure 5.34: Reduced Mileage

5.9 Data-Driven Evaluation

In this section, we perform a large scale trace-driven evaluation of a real-world dataset about GPS records of 14, 453 taxicabs belonging to different taxicab companies in Shenzhen. The first dataset contains daily GPS trace data of the taxicab network, and the second dataset is about deliveries. The GPS dataset was collected by letting each taxicab upload its records 30 seconds on average to a centralized base station, and the delivery dataset was obtained by an offline method. Figure 5.35 gives details about these datasets.



Description of Datasets			
GPS Dataset		Delivery Dataset	
Collection Period	01/01/12-06/30/12	Collection Period	01/01/12-06/30/12
Numbe of Taxis	14,453	Numbe of Taxis	14,453
Data Size	450GB	Data Size	18GB
Record Number	3,888,000,000	Record Number	95,000,000
Format		Format	
Plate Mumber	Date and Time	Plate Mumber	Begin & End Time
Status	Speed	Delivery Distance	Delivery Duration
Direction	GPS Coordinates	Delivery Fare	Unload Distance

Figure 5.35: Details of Datasets

In the GPS dataset, key attributes are taxicab status and GPS coordinates, which can indicate whether a taxicab at a certain location is empty or not. In the delivery dataset, the key attributes are delivery distance, duration and fare, which can describe a taxicab delivery event. Further, the unload distance indicates the distance between the end location of the last delivery and the begin location of this delivery. By combining these two datasets, we can fully understand the daily operational situation of the entire taxicab network and conduct a valid evaluation. Due to the large size of the datasets, we mainly found two kinds of errors. (i) Location Error: GPS coordinates show that a taxicab is off the road. (ii) Missing Records: a fair amount of GPS records are missing. The errors may result from different reasons, e.g., GPS device malfunctions, software issues, *etc.* We perform a preprocessing to clean datasets to rule out taxicabs with more than 10% of missing or errant records.

5.9.1 Evaluation Methodology

To show the effectiveness of carpool services, we compare two carpool route calculation algorithms, the ***optimal carpool*** and the approximation algorithm, indicated as ***coRide***, with the ***ground truth***, which is the original GPS traces from the dataset. To show the performance of coRide to address online requests, we also plot the performance of ***coRide online***.

The above algorithms are evaluated based on three different real-world constraints. (i) **Taxicab Capacity** c to show that how many deliveries can be pooled together in a single taxicab. (ii) **Number of Available Taxicabs** n to show that how many taxicabs can be used at an origin to fulfill all delivery requests. (iii) **Travel Period** $[t_i^s, t_i^e]$ to show the delivery start time and a tolerated end time. For travel period constraints, since we can obtain the actual travel period about every delivery in the dataset, we use a *tolerated detour time* t (minutes) plus the actual end time of a trip to show this constraint. For example, for an actual travel period $[t_i^s, t_i^e]$ in the dataset about delivery i, with a tolerated detour time t, the travel period we used to test a spanning tree is $[t_i^s, t_i^e + t]$, instead of the actual travel period $[t_i^s, t_i^e]$.

From the three perspectives of society, passengers, and drivers, we evaluate the performance of the above algorithms by several metrics. From society's perspective, with the **Percentage of Reduced Total Mileage**, we investigate how much mileage we can reduce by carpooling, given the above constraints and different time lengths between the time to provide delivery requests and time to start deliveries for online requests. We also investigate the impacts of both the hours of the day and days of the week on the percentage of reduced total mileage. From passengers' perspective, with the **Percentage of Reduced Fare** paid by passengers, we show the minimum fare they can pay, given tolerated detour times. From drivers' perspective, with the **Percentage of Increased Profit** earned by drivers, we present the maximum fare they can collect, given tolerated detour times. Further, we investigate our operating model by different carpooling locations and carpooling times. In addition, we also investigate two practical metrics, i.e., (i) the running time of the optimal algorithm to show why this optimal algorithm is not feasible in terms of running time, and (2) the increased individual mileage due to carpooling to show a possible negative effect of carpooling, i.e., increasing the travel time for passengers.

In the evaluation, for datasets about individual days of the week, we first process datasets to obtains delivery requests, and then based on the delivery requests we calculate the carpool route by different algorithms. By processing these requests on a daily basis, we show the performance when passengers provide delivery requests 24 hours earlier than the delivery start time, and for requests starting at one day and ending at the day after, we classify them into the day they start. For coRide online, we show its performance when passengers provide requests at 1, 3, 6 and 12 hours earlier than the delivery start time. The results are average outcomes of 7 days of evaluations.

5.9.2 Reduced Total Mileage

In this subsection, we evaluate coRide via the percentage of reduced total mileage at different parameters.

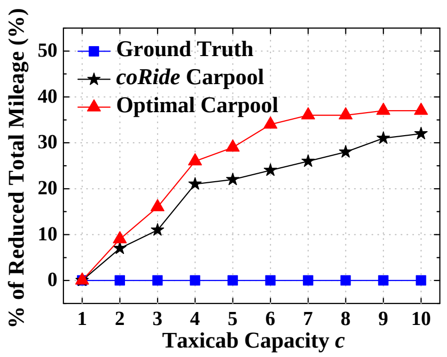


Figure 5.36: Total Mileage vs. c

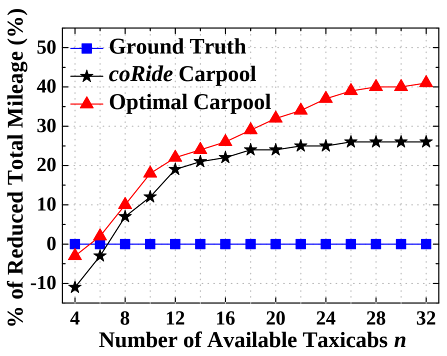


Figure 5.37: Total Mileage vs. n

Taxicab Capacity c

Figure 5.36 plots the effect of taxicab capacity c on the percentage of reduced total mileage with tolerated detour time $t = 5$ and number of available taxicabs $n = 16$. With the increase of taxicab capacity c, the percentage of reduced total mileage for coRide carpool and the optimal carpool also increases. For example, in coRide carpool, the percentage of reduced total mileage increases from 0% to 22%, when taxicab capacity c increases from 1 to 4. This is because when taxicab capacity c increases, a delivery of a taxicab can be pooled with more other deliveries, and thus it can reduce the total mileage. It implies that a carpool functions more effectively when taxicabs can carry more passengers. We observe that the optimal carpool always outperforms coRide carpool. But during the increase of taxicab capacity c from 4 to 10, the performance gain increases between the optimal carpool and coRide carpool. This indicates the optimal carpool functions better when c is larger.

Number of Available Taxicabs n

Figure 5.37 plots the effect of the different number of available taxicabs n on the percentage of reduced total mileage with tolerated detour time $t = 5$ and taxicab capacity $c = 4$. We observe that with the increase of number of available taxicabs n, the percentages of reduced total mileage in coRide carpool increase from -11% to 27%. These are some negative percentages of reduced total mileage when the number of available taxicabs n is small, and a similar situation is also shown in the performance of the optimal carpool. This is because that with fewer taxicabs at an origin, we have to pool more unrelated deliveries in this origin into the same taxicab, and drop them off one by one, and it will increase the total mileage. But when the number of available taxicabs n is larger than 8, we can reduce the total mileage by carpools. We also find that when the number of available taxicabs n is larger than 20, the performance gain between the optimal carpool and coRide increased. It may result from the fact that a spanning tree with more subtrees will not necessarily help coRide to achieve the global minimum mileage.

Travel Period $[t_i^s, t_i^e]$

Figure 5.38 plots the effect of different travel periods in terms of different tolerated detour time on the percentage of reduced total mileage with the number of available taxicabs $n = 16$ and taxicab capacity $c = 4$. In Figure 5.38, we observe that with the increase of tolerated detour time t in terms of minutes, the percentages of reduced total mileage in coRide carpool increase from 0% to 33%, while these of the optimal carpool increase from 0% to 40%, leading to a 7% performance gain. While in a carpool, with more detour time, more mileage can be reduced by pooling more deliveries together. The increase of t enables a larger travel period, making more deliveries correlated with each other in time. With the increase of tolerated detour time t, the increases of performance gain slow down between the percentages of reduced total mileage of the optimal carpool and coRide carpool.

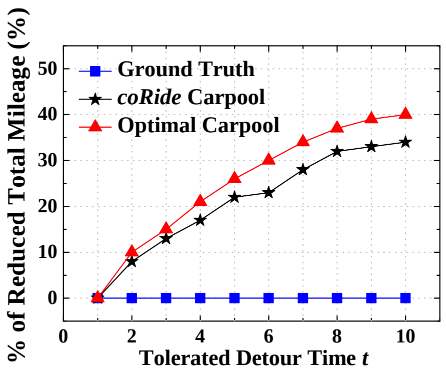


Figure 5.38: Total Mileage vs. t

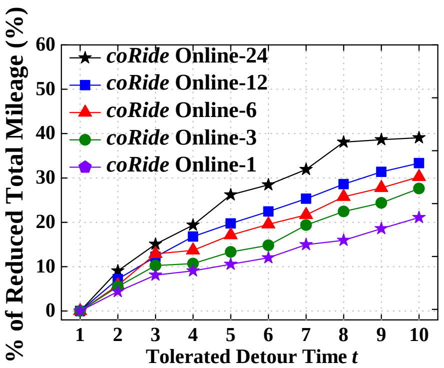


Figure 5.39: Mileage (Online)

Online Requests

In the above coRide carpool, we process requests by days, so it means we pool the delivery requests that passengers provided by 24 hours in advance (named coRide online-24). In Figure 5.39, we evaluate the performance of coRide for online requests situations where (i) the half of the passengers provides requests in advance of 24 hours, and based on them, we build carpool graphs, (ii) the other half of the passengers provides requests in advance of 1, 3, 6 and 12 hours (indicated as coRide online-1, *etc*), and we use our online algorithm to optimally add these online requests together to the existing carpool graphs every 1, 3, 6 or 12 hours, leading to new different carpool graphs. We observe that coRide online-24 outperforms all other versions, indicating the early the passengers provide requests, the better the performance. This is because with more requests to begin with, we can build a more effective spanning tree.

Hourly Windows on Weekdays and Weekends

We evaluate coRide carpool's performance via the percentage of reduced total mileage on weekdays and weekends, respectively. The other constraints are set as $t = 5$, $n = 16$ and $c = 4$. Figure 5.40 plots the average percentage of reduced total mileage in different 1 hour time windows for five weekdays. We observe that in weekday rush hours, e.g.,

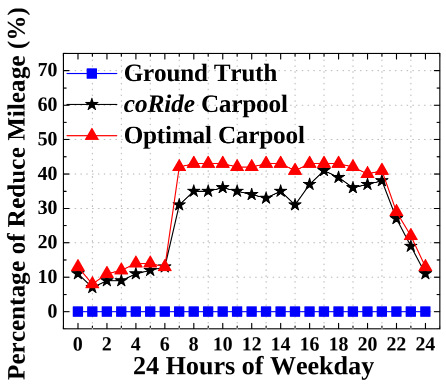


Figure 5.40: Weekday Mileage

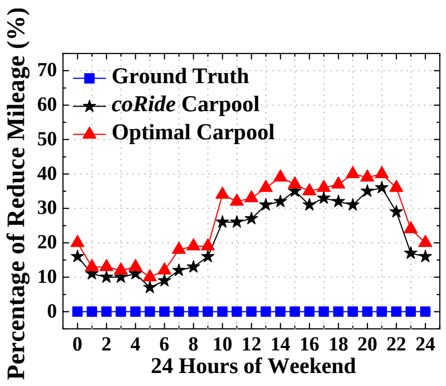


Figure 5.41: Weekend Mileage

07 : 00 − 10 : 00, the percentages of reduced total mileage for two carpool schemes, the optimal carpool and coRide carpool are both higher than 30%. In contrast, in non-rush hours, e.g., 00 : 01 − 7 : 00, the percentages of reduced total mileage for them are below 20%. Figure 5.41 shows the average percentage of reduced total mileage in a weekend. We observe that different from weekday, the high percentages of reduced total mileage in weekend are between daytime 10 : 00−21 : 00. There is no significant high percentage of reduced total mileage in certain time windows among 10 : 00 − 21 : 00 than others. But in Figure 5.40, there are higher performances in time windows 07 : 00 − 10 : 00 and 16 : 00 − 20 : 00 than others. It shows that performance of carpool on weekends is different than that on weekdays, since people would take taxicabs at different time on weekdays and weekends.

Running Time of Algorithms

Figure 5.42 shows the running time of the optimal carpool algorithm and coRide carpool algorithm at different carpool passenger numbers p at a single origin. We observe that as the passenger number p increases from 2 to 18, the running time for the coRide carpool algorithm is negligible compared to the running time for the coRide carpool algorithm. This is because that our carpool route calculation problem is NP-hard, and the optimal carpool algorithm uses Integer Programming to obtain the solution, which leads to a longer running time, and is not practical for real-world carpool route calculation with a large number of passengers.

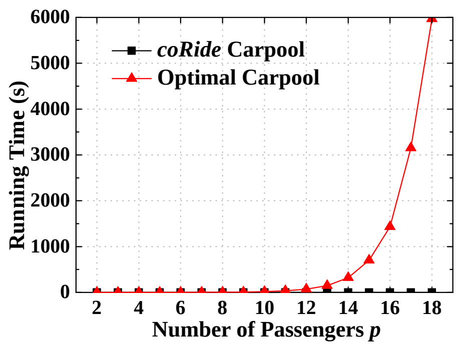


Figure 5.42: Time vs. p

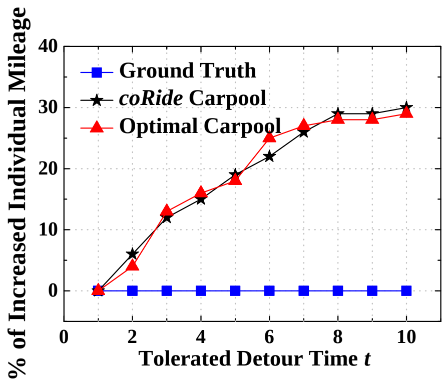


Figure 5.43: Mileage vs. t

Percentage of Increased Individual Mileage

We evaluate the performance of coRide carpool by the percentage of increased individual mileage due to carpools with different travel periods. This increased individual mileage also provides an indication of the detour time a passenger will tolerate for carpooling with others. Note that although the individual mileage increases, the fare for individual passengers is actually reduced, since more passengers will share the fare for common routes, leading to a large carpool benefit, as showed by our Fare Model in Section 5.7. Figure 5.43 plots the effect of different travel periods in terms of t on the percentage of increased individual mileage with $n = 16$ and $c = 4$. With the increase of t from 1 to 10, the percentage of increased individual mileage in coRide carpool increases from 0% to 30%, while that of the optimal carpool has a similar trend. In coRide and the optimal carpool, with more detour time, a high mileage is added to individual deliveries, since after carpool, most of the passengers will have a new yet longer route compared to the ground truth.

5.9.3 Reduced Fare

We evaluate the performance of coRide carpool in terms of maximally reducing the fare for individual passengers, based on the win-win fare model we proposed in Section 5.7. Based on the datasets, we have the ground truth for regular fares of individual passengers, and based on the carpool route, we shall have the carpool fare. We let all the passengers and the driver to evenly share the carpool benefit due to the mileage

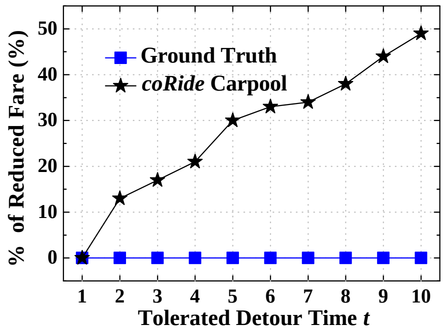


Figure 5.44: Reduced Fare

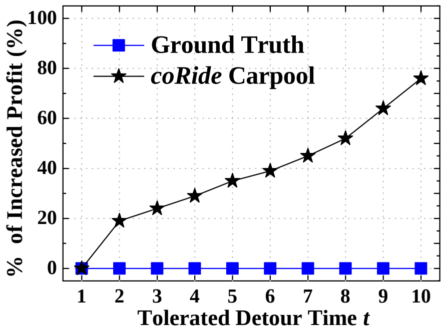


Figure 5.45: Increased Profit

reduction of a carpool route. In Figure 5.44, we observe that with the increase of tolerated detour time t, the percentages of reduced fare for individual passengers in coRide carpool increase from 0% to as much as 49%. In a carpool, with more detour time, high mileage can be shared with other passengers, thus leading to a large carpool benefit for fare reductions. It will lead to an economic incentive for passengers to carpool.

5.9.4 Increased Profit

In this subsection, we evaluate the performance of coRide carpool in terms of maximally increasing the profit for taxicab drivers based on our win-win fare model. With the method similar to that of the last subsection, we can produce an increased profit by comparing the total carpool fare collected by the taxicab driver, and the ground truth of the regular fare about the first passenger picked up in the carpool, which gives the fare the driver will collect in the case that no carpool is conducted. In Figure 5.45, we plot the effects of different travel periods in terms of t on the percentage of increased benefits. It shows that with the increase of tolerated detour time t, the percentage of increased benefits for the drivers in coRide carpool increases from 0% to as much as 76%, which leads to a considerable incentive for taxicab drivers to take carpool trips.

5.10 Conclusions

In this chapter, as one component of the application design layer for mobileCPS, we analyze, design, implement, and evaluate a prototype taxicab carpool system coRide to reduce the total mileage to deliver passengers. Our effort provides a few valuable insights and guidelines, which are hoped to be useful for realizing carpooling services commercially in near future. Specifically, (i) we found unprecedented evidence of inefficiencies of current systems, and opportunities for new systems based on our real-world datasets; (ii) we implemented a customized hardware supporting the essential functionalities for carpooling; (iii) we affirmed that complicated route functions should be implemented in a centralized cloud and near optimality can be achieved; (iv) it is important to establish incentives for all the parties involved (e.g., a win-win situation); and (v) finally our work only addresses the technical frontier, and it is even more critical to establish a right policy that would make a large scale deployment feasible.

Chapter 6

Feeder: Last-mile Transit

In this chapter, we introduce a last-mile transit application called Feeder as another component on the application design layer for mobileCPS. Feeder is also built up urban mobility modeling, but it requires more-detailed travel information at individual passenger levels in real time.

6.1 Introduction

Pubic transit contributes significantly to reduction of travel delay and gas consumption [69], e.g., in 2013, public transit reduced 865 million hours of travel delay and 450 million gallons of gas in U.S., achieving a saving of $142 billion congestion cost [70]. However, public transit (e.g., train or subway) typically stops only every mile on average to maintain a high speed, which means that most of an urban area is beyond an easy walking distance from a transit station, as shown by our large-scale empirical analysis in Section 6.2. This issue is known as "*the last-mile problem*", which is a key barrier to better public-transit utilization [71].

In this chapter, we propose a real-time transit service, called Feeder, which utilizes ridesharing-based vehicles (e.g., minibuses) to deliver passengers from their exiting transit stations to nearby dropoff locations called *service stops*, thus reducing walking distances to their destinations. Although Feeder is conceptually applicable to all public transit, we focus on the design for subway and train networks where the last-mile problem is more serious. We envision that Feeder is operated by a city transit authority with following distinctive features: different from bike systems, Feeder uses only flexible vehicles without high costs for fixed docking infrastructures or extra efforts to carry or park bikes; different from taxicabs, a passenger in Feeder pays a much lower flat fare and also travels more environmental friendly due to a large number of co-riders; different from regular bus services, Feeder is tailored for last-mile trips with a ring route starting from a high-demand station, featuring dynamic departure times and data-driven stops.

In Feeder, a passenger is mainly engaged in three phases: (i) wait for a Feeder vehicle to depart; (ii) ride the vehicle to a service stop; (iii) walk the "last-mile" to destinations. Therefore, Feeder has the three objectives to enhance passenger experience on waiting, riding and walking. (i) *Minimizing Passenger Wait Time*: This objective would be easily achieved by optimizing vehicle departure times, if passengers can provide *where* and *when* they will exit upstream transit (e.g., an exiting time in a subway station). However, in real world, passengers normally do not know future exiting times in advance. (ii) *Minimizing Passenger Riding Time*: This objective would also be easily achieved by optimizing vehicle routes based on real-time urban traffic, if we have a real-time sensor network for traffic detection at urban scale. But the traffic speed sensors, e.g., loop sensors, are only installed at major intersections in most cities. (iii) *Minimizing Passenger Walking Distance*: This objective would be achieved naturally by optimizing service stop locations, if passengers are willing to provide *fine-grained destinations* (e.g., a home address). However, passengers may be reluctant to provide such information due to extra efforts or privacy concerns. As a result, we face an essential challenge to infer detailed passenger last-mile transit demand (i.e., exiting stations, times and fine-grained destinations) for Feeder optimizations without active contributions from passengers or dedicated urban infrastructures.

To address this challenge, we employ existing extreme-scale urban infrastructures to infer last-mile transit demand and traffic speeds, transparently to passengers. In particular, we utilize various devices that generate passengers' location data (e.g., cellphones and smartcard readers) in existing infrastructures in order to infer real-time passenger *exiting times* and *station* as well as *destinations* for Feeder. Further, we use GPS-equipped vehicle networks, e.g., taxi and bus, to infer real-time traffic speeds to design optimal routes for Feeder vehicles. As a result, the key novelty of our Feeder service is that it is a completely transparent, automatic, and data-driven solution yet with neither marginal costs for deploying an *ad hoc* demand-collecting system nor extra efforts from the passenger side.

Conceptually, our core method provides a new possibility of using *heterogenous* data from *existing urban infrastructures* to improve urban efficiency, as opposed to previous monolithic and closed *ad hoc* systems. As a real-world effort, we implement this method by integrating streaming data from four infrastructures in Shenzhen, China: (i) a 10.4 million user cellular network; (ii) a 14 thousand taxicab network; (iii) a 13 thousand bus network; and (iv) an automatic fare collection system for a public transit network (i.e., subway and bus) with 16 million smartcards. We establish near real-time access to the above data sources for online analyses. Further, we store 400 million cellphone records, 32 billion GPS records, and 6 billion smartcard records for offline analyses. The key contributions of the chapter are as follows:

- We utilize various infrastructures to infer passenger last-mile demand in real time. To our knowledge, the utilized data have by far the highest standard for urban study in two aspects: (i) the most complete data including cellular, taxicab, bus and subway data for the same city, and (ii) the largest passenger coverage (i.e., 95% of 11 million permanent residents in Shenzhen). The sample data are given in [72].

- We conduct the first work to design a real-time data-driven service Feeder for the last-mile problem by a two-end solution. For the back end, we propose and implement a cloud server (called the Feeder server). It provides an online *data fusion* based on integrated heterogenous data for three key components: (i) a *departure time computation* to minimize wait times based on straightforward yet efficient smartcard data processing; (ii) a *service stop selection* to minimize last-mile walking distances based on cellphone and taxi data; (iii) an online *route calculation* with a $\frac{3}{2}$ approximation algorithm to obtain a route to connect the stops. For the front end, we customize and deploy a piece of hardware (called the Feeder device) as an onboard device to download departure times and upload status from/to the Feeder server in real time. Feeder spans the entire life cycle of data-driven application design, starting from hardware design, through data collection, cleaning, offline analysis, online processing, real-world utilization, to field evaluation.

- We implement Feeder in Shenzhen for a field study to test its real-world performance. We rent 3 cars installed with our hardware in a subway station where 12 passengers were picked up every morning from the station to their workplaces for 30 days.
- We test Feeder by a comprehensive evaluation with 4 TB Shenzhen data. The results show that Feeder reduces last-mile distances by 68% and travel times by 52% compared to the ground truth.

We organize this chapter as follows. Section 6.2 gives our motivation. Section 6.3 presents an overview. Section 6.4 describes the front-end devices. Sections 6.5, 6.6, 6.7, and 6.8 depict the back-end server. Sections 6.9 and 6.10 validate Feeder with a real-world test and a large-scale evaluation, followed by the discussion and conclusion in Sections 6.11 and 6.12.

6.2 Motivation

To justify our motivation, we explore both severity and ubiquity of last-mile trips by answering two questions: how long is a typical last-mile trip, and how frequently last-mile trips occur among all trips, based on datasets we have collected. The details of data are given in Section 6.5.

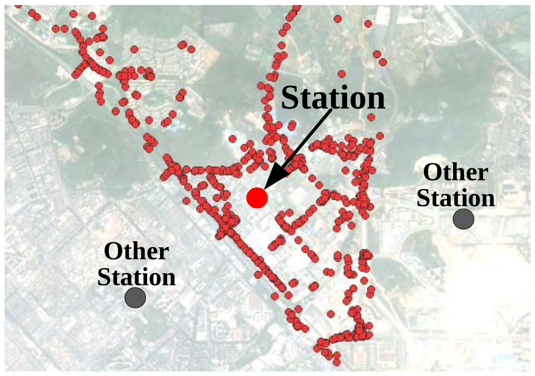


Figure 6.1: Last-Mile in XD

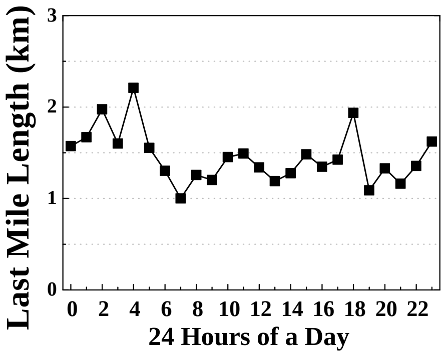


Figure 6.2: Length in XD

In Figure 6.1, we show last-mile trip lengths between a subway station XingDong in Shenzhen and inferred passenger destinations closer to it than other stations. The average length is given in Figure 6.2. The average distance 1.4 km is longer than the distance that passengers are willing to walk [73], i.e., 400 to 800 m.

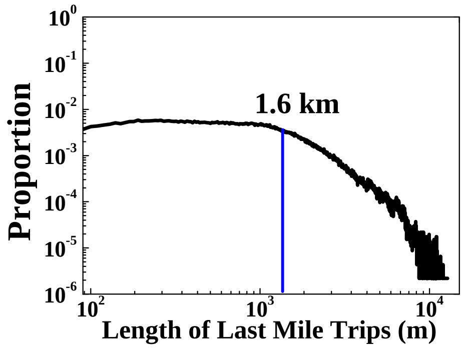


Figure 6.3: Last-Mile Trips

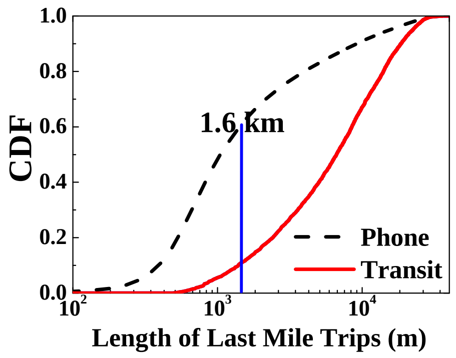


Figure 6.4: All Trip Lengths

In Figure 6.3, we plot the proportion of lengths from all inferred destinations to their closest stations, i.e., last-mile trips. In a log-log scale, a point, e.g., (1.6 km, 0.3%), indicates the last-mile trips with a length from 1.59 km to 1.6 km account for 0.3% of all last-mile trips we studied. The first part of the distribution follows an uniform distribution (i.e., the horizontal line), and the second part follows a power-law distribution (i.e., the big tail). Interestingly, the boundary is around 1.6 km. It reveals that the lengths of last-mile trips are uniformly distributed within the one-mile boundary, while outside this boundary, the longer the trip, the less frequently it occurs. Thus, we confirm the severity of the trips within the one-mile boundary.

We study the frequency of last-mile trips among all trips. Because last-mile trips are usually finished by walking, they are more likely to be captured by cellphone data, instead of transit data (including taxicab, bus, and subway). In Figure 6.4, we study CDF of lengths of trips captured by cellphone and transit data. We found that 63% of trips captured by cellphone data are shorter than 1.6 km, while only 12% of trips captured by transit data are shorter than 1.6 km (most of them are taxicabs). Since cellphone trips can be seen as proxies for all trips, we confirm the ubiquity of last-mile trips by showing that they (i.e., the trips shorter than 1.6 km) have a high frequency of 63% among all trips. We also verify that passengers normally do not use existing transit for last-mile trips since they only account for 12% of all transit trips.

6.3 Service Overview

We first present an operational scenario for Feeder based on Figure 6.5. Without the Feeder service, a passenger would (i) enter public transit at an entering station, and (ii) exit public transit at an exit station, and (iii) walk to his/her final destination. Thus, the last-mile walking distance is from the exiting station to the destination.

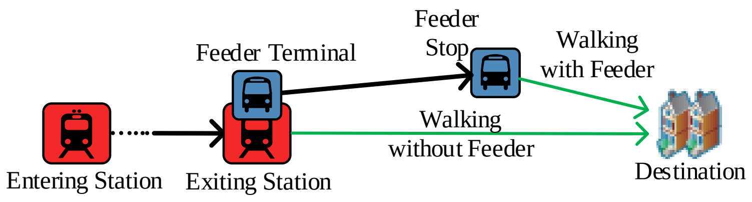


Figure 6.5: Feeder Operational Scenario

In this work, we envision that each of major transit stations has a Feeder terminal where a Feeder service is operated individually. Therefore, with Feeder, a passenger would (i) get on a Feeder vehicle at his/her exiting station (which is also a terminal of a Feeder service); (ii) wait for this Feeder vehicle to leave the terminal based on a departure time, which is optimally calculated according to inferred passenger exiting stations and times; (iii) get off this Feeder vehicle at one of service stops on the service route, which are optimally selected by Feeder according to inferred fine-grained destinations and real-time traffic info; (iv) walk to the final destination. Thus, with Feeder, the walking distance is reduced to the distance from the Feeder-service stop to the destination.

Based on the above scenario, three key design challenges for a Feeder service are (i) how to infer exiting stations and exiting times for transit passengers in order to optimize vehicle departure times, (ii) how to infer fine-grained destinations in order to optimize service stop locations, and (iii) how to infer real-time traffic info in order to optimize routes to link different stops. These challenges are solved in the following framework, which consists of three key components as in Figure 6.6.

Urban Infrastructures. They include cellular, taxicab, bus and subway networks, playing an important role in our Feeder design. We collaborate with several service

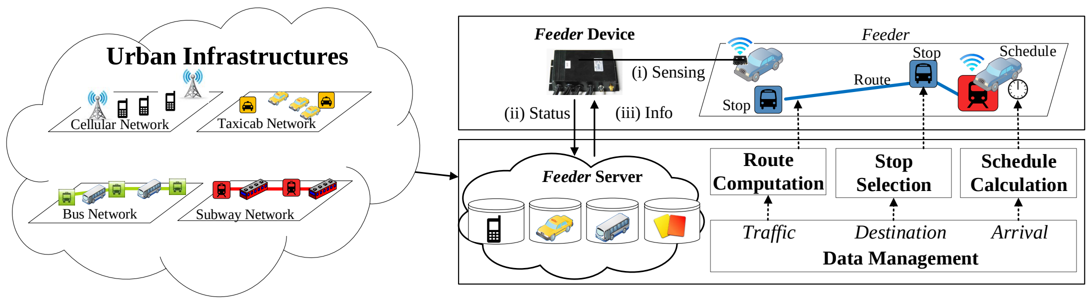


Figure 6.6: Feeder Overview

providers and government agencies to establish the real-time access from infrastructure data sources to our Feeder server. Thus, we enable a complete rendering about dynamics in last-mile transit demand for passengers in different categories, e.g., cellphone, taxicab, bus and subway users, which almost cover all residents in urban areas.

Back-end Feeder Server. A Feeder server is located at a dispatching center to receive and process real-time data from urban infrastructures. Its functions include (i) Data Management (introduced in Section 6.5): integrating heterogenous data (i.e., cellphone, taxicab, bus, and smartcard data) for real-time last-mile transit demand mining, i.e., passenger exiting stations and times as well as destinations; (ii) Departure Time Calculation (introduced in Section 6.6): calculating effective departure schedule online based on mined passenger exiting stations and times to minimize passenger wait times; (iii) Stop Location Selection (introduced in Section 6.7): selecting efficient stops offline based on mined destinations to minimize last-mile walking distances. (iv) Service Route Computation (introduced in Section 6.8): selecting efficient route online based on real-time traffic info to minimize riding times.

Front-end Feeder Device. A Feeder device is a customized device installed on a Feeder vehicle. It senses and uploads physical and logical status of each Feeder vehicle (e.g., locations and numbers of onboard passengers), as well as downloads departure times and stop locations to/from the Feeder server. These functions are performed by the three subsystems of a Feeder vehicle as introduced in Section 6.4.

6.4 Frontend Device

In our project [74], we develop a prototype for front-end data transmission to support functions in Feeder. Figure 6.7 gives a Feeder device's real-world deployment, including three subsystems: (i) an external device system with a GPS module, a CDMA 1X module, and an emergency button; (ii) a sensing system with a camera, a MIC attached to a display, and a $\pm$ 2g triaxial acceleration sensor; (iii) a central control system with a TPS54160 power module and a STM32F103 CPU module. Based on these subsystems, we discuss the capability of a Feeder device as follows.

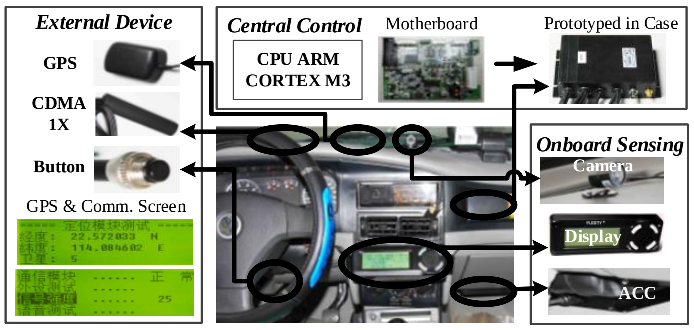


Figure 6.7: Feeder Device Design and Deployment

By Feeder devices, a Feeder server shall be fully aware of Feeder vehicles' physical status, e.g., locations. Thus, in this design, every Feeder vehicle periodically senses and uploads its physical status to the server. The logical status, i.e., numbers of onboard passengers, is also important to the Feeder service, because it affects departure times. We envision that drivers or fare collecting devices will track the number of onboard passengers and thus change logical status to inform the server.

A Feeder device shall have an efficient communication module for uploading and downloading to/from the Feeder server. In the most existing vehicular networks (e.g., Shenzhen taxicab networks), GPRS is typically used for the communication between vehicles and a dispatching center. But in our Feeder service, departure times and stops have to be sent to Feeder vehicles on time, and vehicle status is also needed to be uploaded to the Feeder server in a timely manner. Thus, we employ a CDMA 1X module utilizing separate channels, instead of GPRS, for better performance.

To summarize, the proposed Feeder device is capable of sensing detailed vehicle status and efficiently communicating with the back-end Feeder server, therefore providing a comprehensive front-end support for the Feeder service.

6.5 Data Management

In this section, we first present data input, and then discuss our data cleaning, and finally describe our data fusion.

6.5.1 Data Input

We have been collaborating with Shenzhen service providers and government agencies for access to infrastructures. Conceptually, we use four kinds of devices as *sensors* to sense real-world passenger demand in this version of reference implementation.

- **Cellphones as Sensors** are used to detect cellphone users' locations at cell-tower levels based on call detail records.
- **Taxicabs as Sensors** are used to detect taxicab passengers' locations based on taxicab status (i.e., GPS and occupancy). The locations obtained by taxicab data have a higher spatial accuracy than cellphone data and thus provide a complimentary view, since the taxicab dropoff locations are normally the locations where passengers want to get off.
- **Buses as Sensors** are used to detect bus passengers' locations by cross-referencing data of onboard smartcard readers for fare payments.
- **Smartcard Readers as Sensors** are used to detect a total of 16 million smartcards used by passengers to pay bus and subway fares. These reader sensors capture 10 million rides and 6 million passengers per day. There are two kinds of reader sensors: (i) a total of 14,270 onboard mobile reader sensors in 13 thousand buses capturing 168 thousand bus passengers per hour, and (ii) a total of 2,570 fixed reader sensors in 127 subway stations capturing 60 thousand subway passengers per hour.

We establish a secure and reliable transmission mechanism, which feeds our server the above sensor data collected by Shenzhen Transport Committee and service providers by a wired connection without impacting the original data sources. Since these data are already being collected to help service providers operate their services, our large-scale sensor data collection incurs little marginal cost. The details are given in Section 3.2.

6.5.2 Data Fusion

Our endeavor of consolidating and cleaning these data enables extremely large-scale resident sensing from different perspectives, which is unprecedented in both quantity and quality. In particular, we show the number of passengers detected by three kinds of data in 5-min slots in Figure 6.8, where we do not differentiate subway and bus passengers, since they are both detected by smartcard readers as sensors.

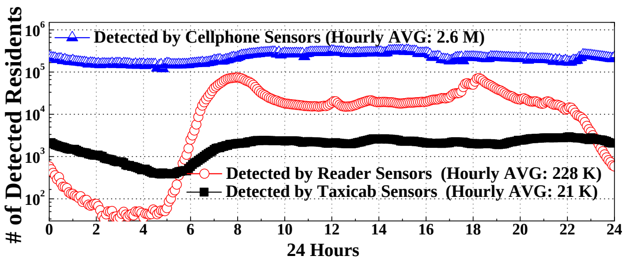


Figure 6.8: Detected Residents by Data

Though comprehensive enough, the above data are in different granularity and formats, which call for a data fusion procedure. Such a data fusion procedure aims to transparentize the heterogeneity of the above data to infer passenger demand through an integrated representation. As follows, we first discuss the heterogeneity of the utilized sensor data from the passenger coverage as well as spatial and temporal resolutions in Table 6.1.

As in Table 6.1, (i) cellphone sensors cover 95% of 11 million residents, but each sensor produces a record only when used for an activity, e.g., making a call, and the corresponding location is only given as one of 17,859 cell towers in Shenzhen; (ii) taxicab sensors cover daily taxicab passengers only accounting for 4% of all residents, but log the origins and the *real destinations* of passengers in fine GPS coordinates during 24

Table 6.1: Heterogeneous Sensor Data

Sensor Name	Resident Coverage	Temporal Resolution	Spatial Resolution
Cellphone	95%	Sparse	17,859 Towers
Taxicab	4%	Continuous	GPS Coordinates
Reader	55%	Continuous	10,448 Stations

hours of a day; (iii) reader sensors cover daily bus and subway passengers accounting for 55% of all residents, and log locations for passengers as one of 10,448 transit stations, i.e., 127 for subway and 10,321 for bus, when they use their smartcards.

Due to large scales of the heterogenous data, our fusion procedure is optimized for simplicity and speed. Thus, we utilize a unified tuple (i.e., a data record) as a generic abstraction to transparentize the heterogenous sensor data.

$$\mathbf{r} = (i, S, T),$$

where i is an ID for a cellphone, taxicab, or smartcard user; S is a location in terms of stations, cell towers, or taxicab GPS coordinates; T is an associated time based on a granularity in minutes. Note that although many residents have both cellphones and smartcards, and they may also take taxicabs, we cannot merge these three different kinds of passengers in the following Feeder server design, due to the lack of unified IDs across different datasets.

6.6 Departure Time Calculation

We first discuss why we need dynamic departure times, and show how we predict passenger-exiting stations and times for dynamic departure times, and present how we optimize departure time.

6.6.1 Motivation for Dynamic Departure Time

Our motivation for dynamic departure times is based on the key difference in passenger arrival between regular transit and Feeder. In regular transit, passengers arrive at a transit station from *various* origins; however, in Feeder, passengers arrive at a Feeder terminal mostly from *one* origin, i.e., upstream public transit, e.g., subway. As a result, passenger arrival for regular transit cannot be accurately predicted due to its various passenger origins, and thus they typically use *fixed departure times* [69]; but the passenger arrival for Feeder can be predicted by observing *current* passengers on public transit, which are known based on real-time smartcard transactions. Thus, we are inspired to predict Feeder passenger *arrival* by predicting *exiting* (in terms of stations and times) of current passengers in public transit. Such a passenger exiting prediction for public transit is used as a passenger arrival prediction for Feeder to calculate *dynamic departure times* for short wait times.

To support our motivation, based on empirical datasets, we locate an existing bus line similar to the last-mile transit with a terminal in a subway station yet with fixed departure times. In Figure 6.9, we show (i) the number of onboard passengers for its buses with *fixed departures* when leaving the terminal, and (ii) the number of passengers exiting the subway station. Without consideration of real-time fluctuation on exiting passengers, the number of passengers in buses with fixed departure times also fluctuates as in the boxes. Such fluctuates may lead to potentially longer passenger wait times, because a previous bus leaving with only few passengers may leave many passengers to the next bus, which may not have the space for all these passengers to leave together in our mid-size Feeder vehicles. Further, we simulate onboard passenger numbers about the same bus line with *dynamic departures* based on the number of exiting subway passengers. We observed that the number of onboard passengers under dynamic departure times does not fluctuate significantly. In short, it suggests that dynamic departure times can reduce wait time with well-predicted passenger demand in terms of exiting stations and times from public transit.

6.6.2 Exiting Time & Station Inference

To obtain such a real-time number of exiting passengers in a public transit station (which is also a terminal of Feeder), a trivial method is to use historical demand. But it assumes that passenger demand is stable, which is often not the case in fine-grained time periods. With real-time data, a straightforward method is to collect the demand when passengers *exit* this station for a time period. However, after such demand becomes available, it is too late to schedule departures because passengers have already been waiting during the period. We show how to predict passenger exiting times and stations as follows.

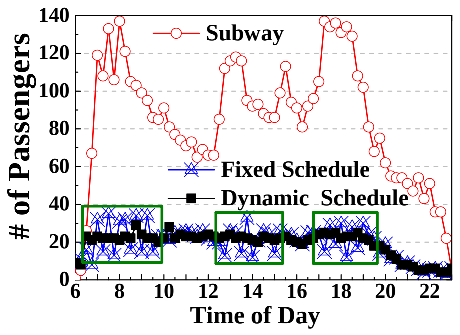


Figure 6.9: # of Passengers

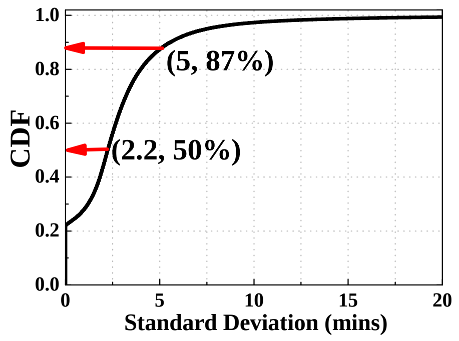


Figure 6.10: Travel Time

Exiting Times

In this work, we notice that public transit systems have relatively stable travel times between the same two stations in different periods, especially as we found in subway networks. Figure 6.10 gives the CDF of standard deviations on travel times based on our data. We found that 50% of travels have a deviation smaller than 2.2 mins, and 87% of travels have a deviation smaller than 5 mins. This nice feature allows us to use the timing information from smartcard transactions when passengers *enter*, instead of *exit*, the transit system. By predicting when passengers will exit a certain exiting station ahead of time, we have sufficient time to schedule departure times of Feeder vehicles. Our exiting time prediction using *entrance as a condition* is more accurate than the prediction based on pure historical information as shown in the evaluation.

Exiting Stations

We infer an exiting station of a passenger by inspecting the transit pattern of this passenger in the recent history under real-time contexts. This is because the majority of passengers as regular commuters exit at the same stations daily near workplaces or homes. For example, Figure 6.11 gives the CDF of distinct exiting stations for passengers in a week, and we found that 67% of all passengers only exit at two distinct stations or fewer, e.g., home and workplace. If we use more contexts (time of day), the distinct exiting stations would be even fewer. More rigorously, we show the CDF of the conditional entropy of passenger exiting stations given entering stations and times in

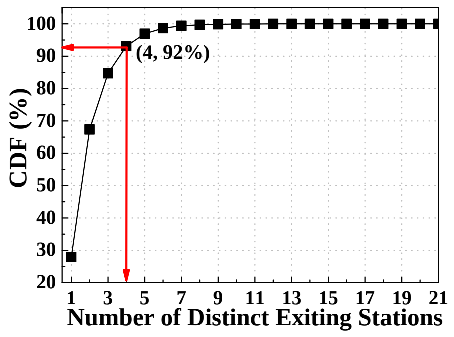


Figure 6.11: Distinct Station

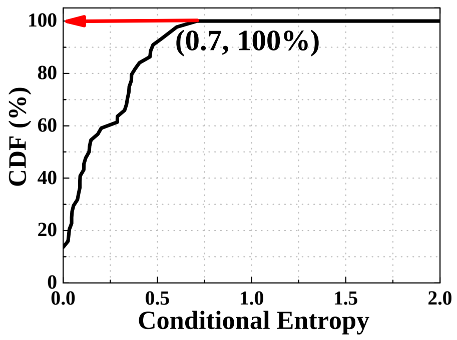


Figure 6.12: Entropy

Figure 6.12 where the conditional entropy is lower than 0.7, indicating there are only $2^{0.7}$ possible exiting stations among total 127 stations. Such a result indicates that urban transit is highly patterned by commutes, which allows us to provide accurate prediction on exiting stations, given the real-time entering contexts.

6.6.3 Departure Time Optimization

An optimization overview for a station S_j is in Figure 6.13.

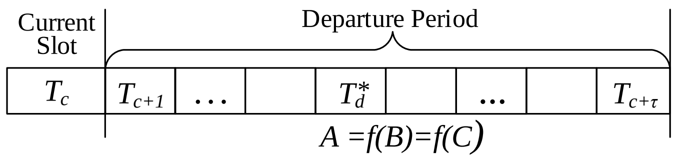


Figure 6.13: Overview of Departure Optimization

Given the current time slot is T_c and the time slot number of round-trip travel about S_j is τ, we have a departure period from the next slot T_{c+1} to the slot $T_{c+\tau}$. Among these slots, we aim to select a departure slot T_d^* with the minimum expected passenger wait time. Thus, we calculate an expected average passenger wait time (indicated as $\mathcal{A}_{T_d}$) for every possible departure slot T_d where $d \in [c+1, c+\tau]$. $\mathcal{A}$ is a function of several expected exiting-passenger numbers (indicated as $\mathcal{B}_{T_d}$ for T_d) in T_d and other slots in the departure period. Further, $\mathcal{B}_{T_d}$ is based on the aggregation on probability (indicated as $\mathcal{C}$) of passengers exiting station S_j during T_d. In the following, we use four steps to show how to obtain $\mathcal{C}$, $\mathcal{B}$, $\mathcal{A}$ and finally T_d^*. Note that we compute in a time slot unit, instead of the exact time, since it is difficult to find many transactions with the same exact times even with our large datasets. For concise notation, we match a pair of entering and exiting tuples for the same passenger to obtain an entry with the following format: (i, S^i, T^i, S_j, T_j), indicating that a passenger i entered station S^i during slot T^i and exited station S_j during slot T_j. Similarly, with $*$ as the wildcard character, we present the entry set $\{\cdot\}$ about all entries for the passenger i as $\{(i, *, *, *, *)\}$.

Step 1: For every current passenger i in the transit system, we calculate the probability $\mathcal{C}(i, S^i, T^i, S_j, T_d)$ that i who entered S^i during T^i will exit S_j during T_d as follows.

$$\mathcal{C}(i, S^i, T^i, S_j, T_d) = \frac{|\{(i, S^i, *, S_j, *)\}|}{|\{(i, S^i, *, *, *)\}|} \cdot \frac{|\{(*, S^i, T^i, S_j, T_d)\}|}{|\{(*, S^i, T^i, S_j, *)\}|},$$

where the first factor is for exiting station prediction showing that among all historical trips where i entered S^i, how many times i exited S_j; the second factor is for exiting time prediction showing that among all historical trips where *any* passenger entered S^i during T^i and exited S_j, how many times s/he exited S_j during T_d. All these subsets can be obtained by aggrergation operations on histroical data.

For example, suppose a passenger $i = 1$ entered station $S^{i=1}$ during slot $T^{i=1}$. We aim to calculate the probability that passenger 1 will exit $S_{j=0}$ during $T_{d=4}$, given the current time slot is $T_{c=3}$. Based on historical transaction entries of the passenger 1, suppose among 10 times that the passenger 1 entered S^1, s/he exited S_0 9 times. As a result, we have $|\{(1, S^1, *, *, *)\}| = 10$ and $|\{(1, S^1, *, S_0, *)\}| = 9$. Further, based on historical transaction entries of all passengers, suppose among 100 times that a passenger entered S^1 during T^1 and exited S_0, there are 80 times that a passenger exited during T_4. Thus, we have $|\{(*, S^1, T^1, S_0, *)\}| = 100$ and $|\{(*, S^1, T^1, S_0, T_4)\}| = 80$. Finally, based on the formula in Step 1, we have $\mathcal{C}(1, S^1, T^1, S_0, T_4) = \frac{|\{(1,S^1,*,S_0,*)\}|}{|\{(1,S^1,*,*,*)\}|} \cdot \frac{|\{(*,S^1,T^1,S_0,T_4)\}|}{|\{(*,S^1,T^1,S_0,*)\}|} = \frac{9}{10} \cdot \frac{80}{100} = \frac{72}{100}$.

Step 2: We aggregate probabilities for all N passengers for the expected number

$\mathcal{B}_{S_j \cdot T_d}$ of passengers who exit S_j during T_d, given entering slots and stations.

$$\mathcal{B}_{S_j \cdot T_d} = \sum_{i=1}^{N} \mathcal{C}(i, S^i, T^i, S_j, T_d).$$

In our example, suppose only one passenger $i = 1$ is in the system now, i.e., $N = 1$, we have $\mathcal{B}_{S_0 \cdot T_4} = \sum_{i=1}^{N=1} \mathcal{C}(i, S^i, T^i, S_0, T_4) = \mathcal{C}(1, S^1, T^1, S_0, T_4) = \frac{72}{100}$.

Step 3: With a length of t, we divide a potential departure period from the next slot T_{c+1} to $T_{c+\tau}$ into equal slots. If a vehicle departs from S_j right after a given time interval T_d where $d \in [c+1, c+\tau]$, we calculate the average passenger wait time $\mathcal{A}_{S_j \cdot T_d}$ for all passengers arriving during the departure period as

$$\frac{[\sum_{y=c+1}^{d} \mathcal{B}_{S_j \cdot T_y} \cdot (d-y) \cdot t] + [\sum_{z=d+1}^{c+\tau} \mathcal{B}_{S_j \cdot T_z} \cdot (\tau - (z-d)) \cdot t]}{\sum_{x=c+1}^{c+\tau} \mathcal{B}_{S_j \cdot T_x}},$$

where (i) the denominator $\sum_{x=c+1}^{c+\tau} \mathcal{B}_{S_j \cdot T_x}$ is the expected passenger number during the departure period from T_{c+1} to $T_{c+\tau}$. (ii) The first term in the numerator, i.e., $\sum_{y=c+1}^{d} \mathcal{B}_{S_j \cdot T_y} \cdot (d-y) \cdot t$, is the total wait time for the passengers who arrive before the vehicle departs (i.e., arriving from T_{c+1} to T_d) and leave with the current vehicle. The passengers arrived at T_y have an expected number of $\mathcal{B}_{S_j \cdot T_y}$ and an expected wait time $(d-y) \cdot t$. (iii) The second term in the numerator, i.e., $\sum_{z=d+1}^{c+\tau} \mathcal{B}_{S_j \cdot T_z} \cdot (\tau - (z-d)) \cdot t$, is the minimum total wait time for the passengers who arrive after the vehicle departs (i.e., arriving from T_{d+1} to $T_{c+\tau}$) and have to wait for the vehicle to come back yet with an unknown future departure time. The passengers arrived at T_z have an expected number of $\mathcal{B}_{S_j \cdot T_z}$ and the minimum expected wait time $(\tau - (z-d)) \cdot t$.

In our example, $c = 3$, $\tau = 2$, $t = 10$, $d = 4$, $j = 0$, $\mathcal{B}_{S_0 \cdot T_4} = \frac{72}{100}$, and suppose $\mathcal{B}_{S_0 \cdot T_5} = \frac{18}{100}$, so average wait time $\mathcal{A}_{S_0 \cdot T_4}$ for all passengers if the vehicle departs after T_4 is

$$\frac{[\sum_{y=4}^{4} \mathcal{B}_{S_0 \cdot T_y} \cdot (4-y) \cdot 10] + [\sum_{z=5}^{5} \mathcal{B}_{S_0 \cdot T_z} \cdot (2-(z-5)) \cdot 10]}{\sum_{x=4}^{5} \mathcal{B}_{S_0 \cdot T_x}}.$$

Thus, we have $\mathcal{A}_{S_0 \cdot T_4} = \frac{\frac{72}{100} \cdot (4-4) \cdot 10 + \frac{18}{100} \cdot (2-(5-5)) \cdot 10}{\frac{72}{100} + \frac{18}{100}} = 4$.

Step 4: We move T_d through all possible departure slots from T_{c+1} to $T_{c+\tau}$, and compare all resultant $\mathcal{A}_{S_j \cdot T_d}$, and finally select the departure time after the slot T_d^* associated with the minimum $\mathcal{A}_{S_j \cdot T_d^*}$ among all $\mathcal{A}_{S_j \cdot T_d}$.

In our example, we continue to calculate the average wait time $\mathcal{A}_{S_0 \cdot T_5}$ associated with the other possible departure slot, i.e., T_5, and then we compare $\mathcal{A}_{S_0 \cdot T_5}$ with $\mathcal{A}_{S_0 \cdot T_4}$, and finally select the smaller one to set the depart time for a minimum expected average wait time.

As an intuitive example, only one vehicle is waiting at S_j, but in our evaluation we consider a multiple vehicle situation where we select the Top n slots with the minimum average wait times for n vehicles as the departure slots. The coordination of vehicles is implicitly considered in the departure time calculation. Further, the slot length, the vehicle capacity and the data history length also have impacts on Feeder performance, which are evaluated in Section 6.10.

6.7 Stop Location Selection

We first present our motivation, and then show how to infer passengers' destination, and finally optimize stop selections.

6.7.1 Motivation for Data-Driven Stop Locations

Different from regular transit, last-mile transit aims to reduce passengers' walking distances to destinations [71]. As a result, we need a destination-driven stop selection to reduce walking distances. However, large-scale fine-grained destinations are usually unknown. We are inspired by the fact that the fine-grained destinations of cellphone and taxicab users have already been captured by cellphone and taxicab data, which have the potential to serve as proxies for destinations of all passengers.

The destinations of cellphone users are used to infer all destinations because almost every urban resident has a cellphone, e.g., in Shenzhen our cellphone records cover 95% of the permanent residents. Further, a total of 17,859 cell towers partitions the 1,991 km^2 Shenzhen area into fine-grained cells with the average coverage area of $\frac{1{,}991}{17{,}859}$km^2 $\approx 333 \times 333$m^2, which are generally within a walking distance, and thus are fine enough to serve as destinations.

The destinations of taxicab users are also good proxies for all destinations, providing a complimentary view. This is because in urban areas, the residents live in high-rise apartments in high density, so numerous residents would share the same fine-grained

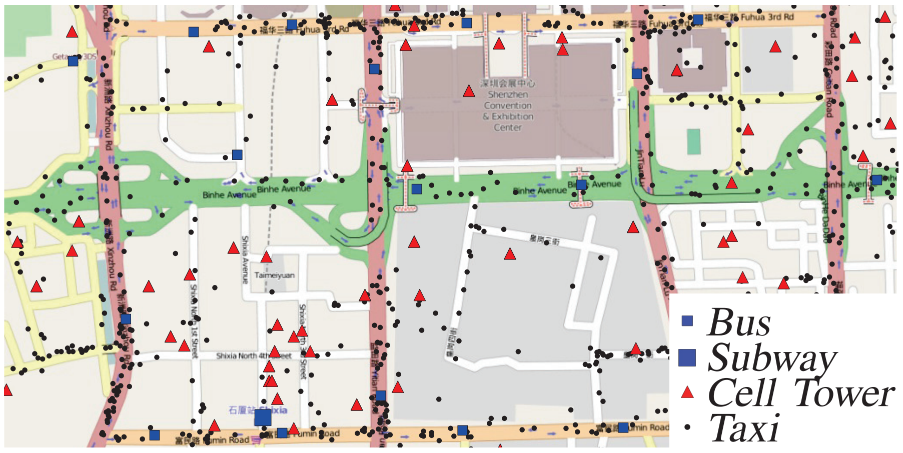


Figure 6.14: Inferred Destinations in Downtown

destinations, e.g., the front gate of a residential community. Thus, it is very common that a public transit passenger's destination is shared with a neighbor who uses taxicabs, and thus the destination of this public transit passenger is captured by taxicab data.

To support our motivation, Figure 6.14 highlights the Shenzhen downtown area with bus and subway stations, cell towers, and taxicab destinations. We found that (i) cell towers are distributed in fine granularity and more evenly than public transit stations, and (ii) taxicab destinations accumulated from one hour cover all major road segments. Note that these two modes of travel (captured by cellphones and taxicabs) have their unique advantages, which cannot be replaced by the other.

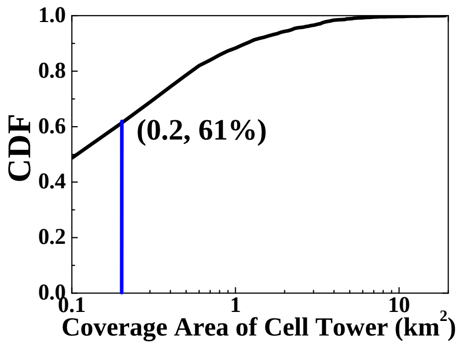


Figure 6.15: Tower Coverage

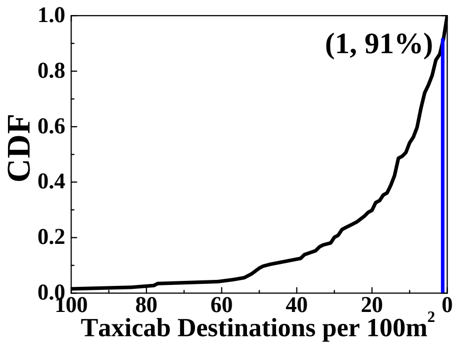


Figure 6.16: Taxicab Destinations

More rigorously, we show the CDF of coverage areas of all 17,859 cell towers in Figure 6.15 where 61% of cell towers have a coverage area smaller than 0.2 km^2. Further, we show the CDF of numbers of daily taxicab destinations per 100 m^2 among 216 Shenzhen urban regions in Figure 6.16 where 91% of regions have at least one destination per 100 m^2, which is typically within a walking distance.

6.7.2 Passenger Destination Inference

Based on the above discussion, we infer the passengers' destination set D by combining a Cellphone users' destination set D^c and a Taxicab users' destination set D^t.

To obtain D^c, we employ historical cellphone data offline for a given period (e.g., one month) to infer the two most frequently visited locations, i.e., home and workplace, for every cellphone user at cell-tower levels. This process is executed offline by finding two most frequently connected cell towers during the work time (9AM-5PM) and the non-work time (6PM-8AM) on weekdays, respectively, for every user. Based on the previous study [8], this approach has a high accuracy to infer important locations for cellphone users.

To obtain D^t, we employ taxicab data to accumulate all obtained destinations into D^t starting from the latest data, until the size of D^t is equal to the size of D^c. The reason behind this size-based accumulation is that due to lack of identifiable passenger ID in taxicab tuples, we have to accumulate all destinations in D^t for a period of time (in terms of days) to track more destinations for taxicab passengers, thus potentially more destinations shared by public transit passengers. We stop the accumulation if the size of D^t is equal to the size of D^c to avoid that D^t numerically dominates the stop selection.

6.7.3 Stop Location Optimization

We assign every destination in the destination set D to the closest public transit station based on their locations. This is because passengers usually exit public transit stations closest to their destinations. Thus, we have a subset D_j of D for a transit station S_j. As follows, we individually select stops for every public station. We first introduce Schwarz-criterion-based service stop selection, and then discuss context-aware stop updating.

Schwarz Criterion Based Section

We utilize the classic K-mean clustering on all destinations in D_j, and select the centroids of clusters as the stops for the station S_j. But one key issue is to determine K, i.e., the number of stops. The more the stops, the more delay is reduced for passengers to walk to destinations. But more stops could lead to an overfitting problem, and also incur more increased delay for onboard passengers due to frequent vehicle stopping. Thus, to balance the stop number K, we employ the Schwarz criterion [75] as follows.

$$\sum_{i=1}^{M}(l_i - c(l_i))^2 + 2\lambda K \log M,$$

where M is the total number of destinations in D_j; l_i is the GPS location of a destination; $c(l_i)$ is the nearest centroid to l_i among K centroids; λ is the regularization factor. The first term $\sum_{i=1}^{M}(l_i - c(l_i))^2$ is called the distortion term, which shows the sum of Euclidean distances of each destination to its nearest centroid. Under our Feeder context, we regard the distortion term as the average reduced delay for passengers due to the increased stops to reduce the average last-mile walking distance to their destinations. The second term $2\lambda K \log M$ is called the penalty term where K has to be regularized by M with a term $\log M$, because the penalty level of increasing K is decided by both K itself and M. This penalty term is introduced in order to avoid overfitting. In our Feeder context, we can also regard the penalty term as the average increased delay for the vehicle stopping in the increased stops.

In the above criterion, the lower the value, the better the clustering performance. However, in real-world setting, it is not practical to set too many stops for a small service area to minimize the criterion. Thus, for a station S_j with a coverage area E_j, we set the upper bound of K_j for S_j to $\frac{E_j}{100 \times 100 m^2}$, because an urban block is normally 100×100 m^2. The K_j for S_j is selected among one to its upper bound to minimize the Schwarz criterion, i.e., finding the "elbow" of the curve of this criterion against K_j.

Context-Aware Stop Updating

We explore context-based stop updating for shorter last-mile distances. This is because we found that passenger destinations are quite different under different contexts, e.g., weekdays and weekends, as shown in the evaluation. For all destinations in D_j about a station S_j, we use the day of week as a context to divide D_j into two subsets, i.e., D_j^1 to D_j^2. Each of which contains the destinations from the data for weekdays and weekends, respectively. We use each of them to update stop locations of the corresponding day. For a practical reason, we did not use other contexts (e.g., the time of day) to more frequently update stops. This is because consistently changing stops may discourage passengers to take Feeder, since they may not know where vehicles would stop in advance. The performance of this updating is tested in the evaluation.

6.8 Service Route Calculation

In this section, based on selected stops, we calculate a route $\mathcal{A}$ to connect a station S_j and all its selected stops with the minimum cost. We first introduce the speed modeling, and then present our route calculation.

6.8.1 Motivation for Traffic-based Routes

The regular transit is typically used to connect two far regions, so their routes are almost fixed due to intermediate stops [69]. In contrast, the last mile transit is used to cover a small area centered at a transit station, so it typically starts and ends at the same location with a typical ring route with few stops. As a result, it can visit all stops by several routes with different traffic speeds at different times of day, thus enabling dynamic routes to save travel time.

Based on empirical datasets, we investigate a bus line with a ring route similar to the last mile transit. Figure 6.17 gives the time of each bus took to finish the fixed route. We found some fluctuations along the time due to the traffic condition in the rush hour. In contrast, we plot the travel time of several taxicabs going through the same stops yet with dynamic routes using less time. One important reason is that different routes have different speeds at different times of day, and the experienced taxicab drivers usually select the fastest route accordingly. In short, it suggests that using traffic-based routes can reduce passenger travel time.

6.8.2 Travel Time Inference

As shown in Figure 6.17, the travel time has an online nature, i.e., the travel time is different for the same route in different times of day. To address this issue, we regard taxicabs and buses as roving sensors to continuously infer real-time traffic speeds. In particular, a tuple $\mathbf{r} = (id, l, m)$ of the taxicab or bus tuple sets indicates that a taxicab or bus passenger id was at a location l at a moment m, which is used to calculate the traffic speed on the corresponding road segment. The average time interval for those tuples is less than 30 seconds, thus enabling an accurate and continuous travel time monitoring in urban scales. Figure 6.18 shows average traffic speeds during 6PM in 496 Shenzhen regions where a warmer color indicates a slower speed. In short, we obtain the travel time based on the taxicab and bus data.

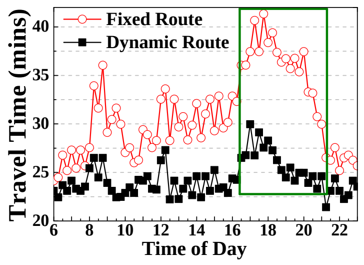


Figure 6.17: Travel Time

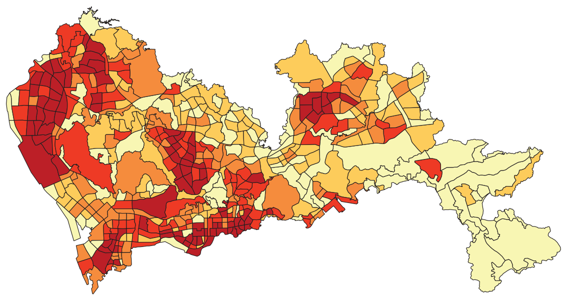


Figure 6.18: Traffic Speeds in Regions

6.8.3 Service Route Optimization

Theoretically, by regarding the station and stops as vertices, we obtain a complete graph with time-dependent weights. The weight on an edge indicates the real-time travel time (obtained by pervasive buses and taxicabs) for a particular time of day between two vertices (i.e., stops) of the edge. Thus, our route calculation problem is formulated as follows: given a complete graph including a station and all its stops, find a route to connect all stops with the minimum weight, i.e., the travel time. This problem is related to the *multiple traveling salesmen problem* (called mTSP where n salesmen start from a depot to visit different cities with the minimum weight [68]). But our problem has a relaxed constraint where we can use fewer than n vehicles to visit these stops, instead of exact n, and n is the number of available vehicles for station S_j. By an analogy to the NP-hard mTSP, our problem is also NP-hard.

To solve this NP-hard problem, we propose an approximation algorithm with a bounded performance ratio. Our algorithm produces a route to connect a station S_j to its K_j stops, given the travel time between them as the weights. Note that the travel time changes at different times of day, so the *Feeder* server recalculates the route online and sends the route to a vehicle after its arrival at the public station. Specifically, the algorithm is given in three steps.

Step 1: Connecting all stops from S_j by the minimum spanning tree T_j. We employ *the minimum spanning tree* (MST) algorithm to link the stops belonging to the public transit station S_j to obtain a MST T_j rooted at S_j as an underlying connection. Figure 6.19 gives an example to show the eight stops of S_j and the resultant MST T_j.

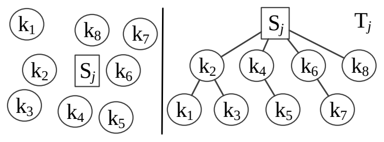


Figure 6.19: Connecting Stops by a MST

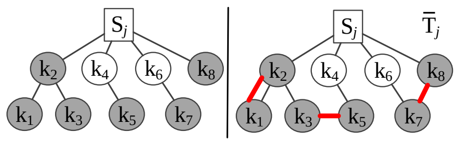


Figure 6.20: Add Perfect Matching

Step 2: Adding a special minimum perfect matching M_j to T_j to obtain an underlying structure $\bar{T_j}$. (i) We first find a vertex set V' containing all vertices with an odd degree in T_j; (ii) we construct *the minimum weighted perfect matching* M_j for all vertices in V'; (iii) we add the edges of M_j to T_j to obtain a new graph $\bar{T_j}$, as an underlying structure to calculate the final route. Note that a perfect matching M on V' is a set of pairwise non-adjacent edges (i.e., no two edges share a common vertex in V') linking all the vertices in V'; further, such a perfect matching with the minimum weight for vertices in V' can always be found in a polynomial time [76], since the number of vertices in V' is always even. In Figure 6.20, the left subfigure gives the grey vertices with an odd degree in T_j as V'; the right subfigure gives their minimum perfect matching M_j, consisting of three new edges (bold), which are added to MST T_j to obtain $\bar{T_j}$. The reason why we add M_j to T_j to obtain $\bar{T_j}$ is to enable *cycles* to calculate the route; a cycle is a vertex traversal $S_j \Rightarrow S_j$ that starts at the station S_j

and stops when it visits S_j again.

Step 3: Obtaining the final route $\mathcal{A}$ with a shortcutting based traversal on $\bar{T}_j$. (i) From the station S_j, we perform a depth-first traversal on $\bar{T}_j$; (ii) during this traversal, if we find a vertex that has already been visited before, we shortcut this vertex (except for the root S_j) to visit the next vertex directly; (iii) we obtain the resultant graph, consisting of one or more cycles about the root S_j, and each cycle is driven by *at lease one vehicle*. Figure 6.21 gives a traversal on $\bar{T}_j$ where the left figure gives the shortcutting based traversal by shortcutting k_2, i.e., deleting $k_1 \to k_2$ and $k_2 \to k_3$ yet adding $k_1 \to k_3$; the right figure gives the final route with 2 subroutes.

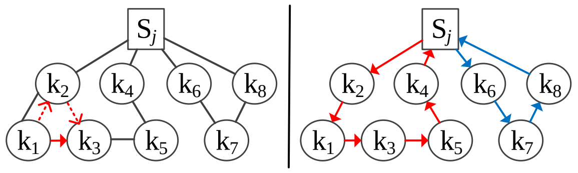


Figure 6.21: Obtaining Final Route by a Traversal

Note that we focus on the reduced travel time, so better performance can be achieved by having (i) a large vehicle capacity c and (ii) the number n_j of vehicles for S_j equal to or larger than the number of cycles about S_j. Therefore, different passengers can select vehicles for the cyclic subroute that quickly visits the stops close to their destinations without long detours. Our design accounts for the constraint on the number of vehicles n_j for S_j, e.g., if only one vehicle can be used $n_j = 1$, we merge all cycles into one big cycle by shortcutting S_j, and make a route equal to the depth-first traversal on $\bar{T}_j$, i.e., shortcutting all visited vertices, so a vehicle visits all stops and then goes back to S_j.

In the appendix, we prove that our approximation algorithm has a bounded performance ratio of $\frac{3}{2}$, i.e., the travel time of the route obtained by our algorithm is at most $\frac{3}{2}$ times of the optimal travel time. Though having the same ratio bound with the state-of-the-art solution for mTSP [68], our algorithm has a novelty in its shortcutting mechanism based on the vehicle number constraint.

6.9 Real-World Implementation

In this project, we have tried for a commercialized implementation of Feeder. The designed Feeder devices have been configured on 98 vehicles in Shenzhen, and our server has full capacities to efficiently perform Feeder server functions. However, through Shenzhen Transport Committee, we have been informed that such commercialized transit services require a government-issued permit. Alternatively, we implemented Feeder by ourselves at a subway station Tanglang in Shenzhen for a small-scale trial to show this system would function well in the real world. To enable a practical test with our 12 prearranged volunteers, we use 3 low capacity vehicles, i.e., taxicabs, as Feeder vehicles with Feeder devices to drive them to their workplaces as in Figure 6.22. But in a real-world service with more potential passengers, a Feeder vehicle shall have a high capacity, enabling more environmental friendly services.

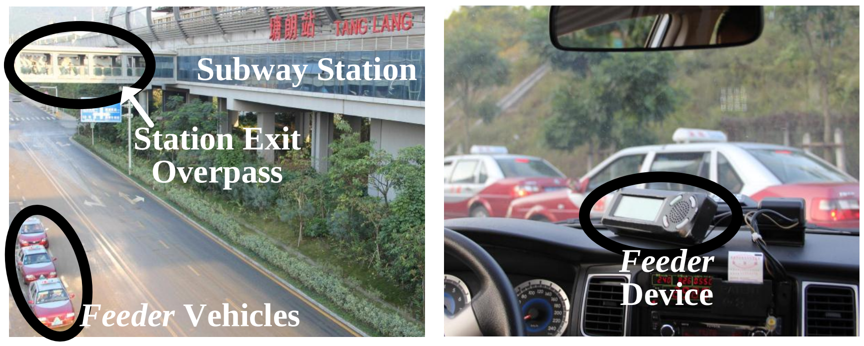


Figure 6.22: Feeder Service in Tanglang Station

6.9.1 Overview

Based on taxicab and cellphone data, we first obtain the inferred destinations that are closer to Tanglang than other stations. Next, we use these destinations to obtain eight service stops, and then find the route based on our route computation to link these stops to the Tanglang station. The stops and route are given in Figure 6.23. Further, after arriving at the final stop, the vehicles have to use the same path to go back to the station due to terrain features.

We collected the data in a 30-day period about the 12 passengers who take the

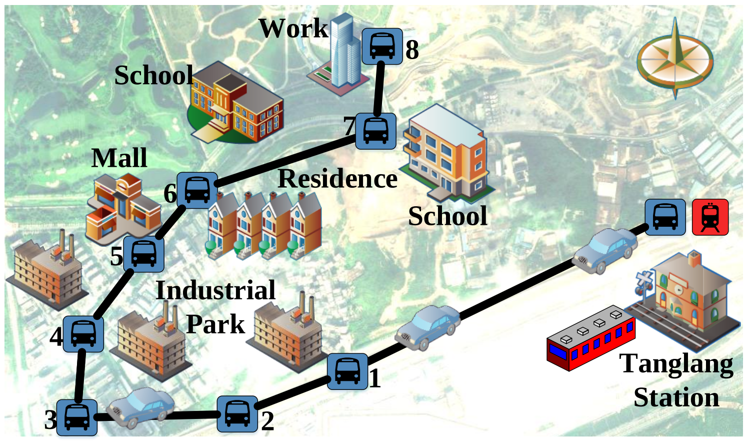


Figure 6.23: Real-World Scenario

subway to work and exit at Tanglang station every morning. After exiting the station, they were picked up individually or together based on their exiting times, and then were dropped off at their workplaces. We calculate departure times based on their smartcard data in an online fashion for vehicles to leave. The vehicles would go back to the station until all prearranged passengers were picked up and then delivered. We videotaped the service, with which arriving moments, departure moments, last-mile distance and travel time (equal to wait time plus ride time) were calculated.

6.9.2 Field Study

We use two metrics, i.e., travel time and last-mile distance, to compare Feeder with regular bus services with fixed departures. We also provide a walking time for reference. We first evaluate Feeder by the travel time, which is divided into (i) the wait time from exiting the station to leaving with vehicles; (ii) the ride time from leaving with vehicles to arriving destinations. Figure 6.24 gives the average wait and ride time among 12 passengers during 30 days, compared to using a regular bus with fixed departures. Feeder significantly reduces the travel time, compared to a 38 min bus trip. The ride time is stable around 14 mins, but the wait time is variable around 9 mins.

We evaluate the average travel time for 12 passengers in Figure 6.25. We found the wait time for some passengers is shorter than others. This is because the prediction

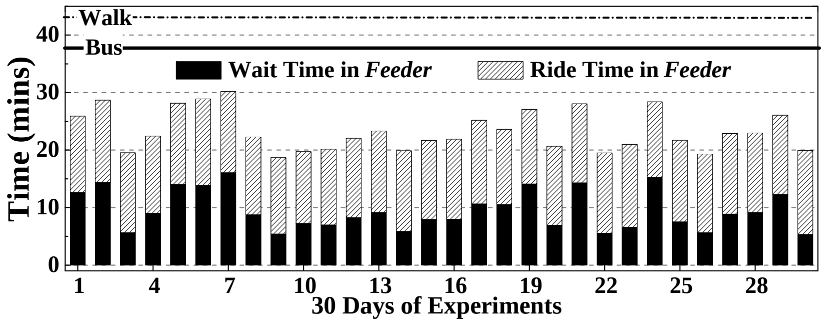


Figure 6.24: Average Travel Time in 30 days

about the passengers with highly regular patterns is accurate, which leads to effective departure times. But for the passengers with irregular patterns (e.g., they go to work from different stations), the prediction is not accurate, leading to ineffective departure times, which may increase their wait time. Feeder is better than scheduled bus because of a combined effect that the bus stop is farther than the Feeder stop to both stations and final destinations of passengers and Feeder has a better schedule.

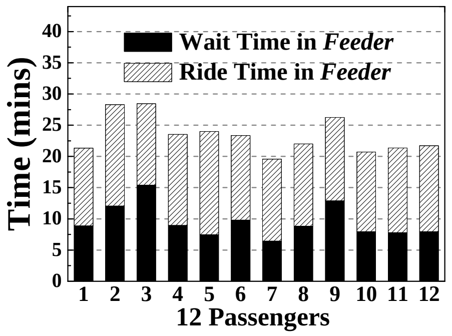


Figure 6.25: Individual Time

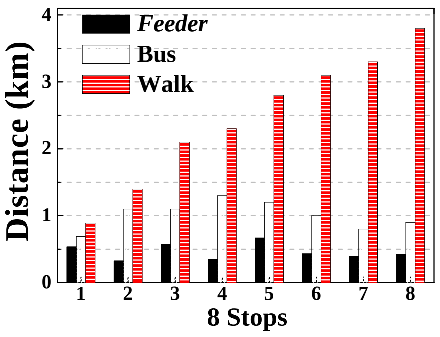


Figure 6.26: Last-Mile Distance

Finally, we evaluate Feeder by the last-mile distance. Due to the limited passengers, we utilize the taxicab and cellphone data to obtain all potential destinations along this route in one day. Then, we show if the passengers with these destinations were using Feeder to get off at the closest stops, what the average last-mile distance would be in the eight stops. We also provide a walking distance from the Tanglong station to every stop for reference. In Figure 6.26, the stops 2 and 4 are more effective since the distance for passengers who got off at these two stops is less than 300 m. For other stops, the average last-mile distance is about 500 m, still much shorter than regular buses.

6.10 Data-Driven Evaluation

With datasets introduced in Section 3.2, we perform a large-scale data-driven evaluation about 127 stations on all five of Shenzhen subway lines, though Feeder also applies to major bus stations.

6.10.1 Evaluation Methodology

For every station, we first obtain stops based on destinations of cellphone and taxicab users; then, we find the shortest route to link stops to the station; finally, we use streaming smartcard data to decide the number of exiting passengers during a given time slot to simulate a real-world scenario (with unexpected passengers), and we calculate departure times based on passenger arrival prediction with online data.

We envision that only the half of exiting subway passengers would take the Feeder service. The destinations of these passengers are randomly set to the real-world destinations of taxicab and cellphone users. We use two key metrics: Percentage of the **Reduced Last-Mile Distance** and Percentage of the **Reduced Travel Time** compared to the ground truth under different logical contexts: (i) **Time of Day**; (ii) **Day of Week**; (iii) **District Population**. In addition, we investigate several key parameters on the system performance: (i) **Departure Slot Length** t as a time unit for vehicles to leave stations (the default is 4 mins); (ii) **Historical Dataset Length** h to show the impact of historical smartcard data amounts (the default is 6 months); (iii) **Vehicle Status** in terms of the **vehicle number** n and the **vehicle capacity** c to investigate the impact of Feeder vehicles (the defaults are given later).

We compare Feeder with its three variations to show the effectiveness of Feeder design components.
(i) **Feeder+DBSCAN** utilizing DBSCAN clustering in the stop selection, which is used to show the advantage of Feeder using the Schwarz-criterion-based stop selection; (ii) **Feeder+Fixed-Schedule** utilizing the fixed departures based on vehicle numbers and the travel time without any smartcard data, which is used to show the advantage of Feeder using smartcard data for the departure computation; (iii) **Feeder+Train** utilizing real-time train arrivals as references to set the departure time, which is used to show Feeder's advantage from using individual smartcards; (iv) **Feeder+Offline**

utilizing only historical smartcard datasets to obtain departure times, which is used to show Feeder's advantage from using real-time online data and from its arrival prediction;

We evaluated Feeder extensively, but due to space limitations, we report impacts of Feeder+DBSCAN on reduced last-mile distances, and impacts of others on reduced travel time. The ground truth of last-mile distances and travel time is obtained by locations of destinations and stations, and an average walking speed of 5 km/h and an average driving speed of 35 km/h. All results are based on the average of a three-month evaluation. For scalability, we maintain transit patterns by probability distributions for every passenger exiting at a station and update them every day. Thus, in the real-time mode, the running time is negligible compared to departure periods.

6.10.2 Impacts of Logical Contexts

We test the impacts of three logical contexts as follows.

Time of Day

We evaluate impacts of the time of day during the normal public transit operating hours from 7AM to 11PM. Figure 6.27 plots the reduced last-mile distance among the evaluated subway stations in Shenzhen during 16 hours. Both of services significantly reduce the last-mile distance. But in the rush hour, Feeder outperforms Feeder+DBSCAN by 19%; whereas in the non-rush hour, Feeder has better performance with a gain of 26% over Feeder+DBSCAN. It shows Feeder's advantage by utilizing Schwarz based stop selection. Feeder has performance of a 68% last-mile distance reduction at the default time 6PM.

Figure 6.28 shows the average reduced travel time. In the non-rush hour, all services reduce the travel time by 47% on average; in the rush hour, their performance drops to 43% on average. But Feeder outperforms Fixed-Schedule shown by 11% more travel time reduction, because Feeder employs dynamic departure times based on collected data. Further, Feeder outperforms Feeder+Offline by 21% more travel time reduction, thanks to the utilization of real-time datasets. Feeder also outperforms Feeder+Train by 14%, thanks to individual smartcard based prediction. Feeder+Train cannot predict exact numbers of arriving passengers, thus leading to a suboptimal schedule. Feeder has performance of a 52% travel time reduction at the default time 6PM.

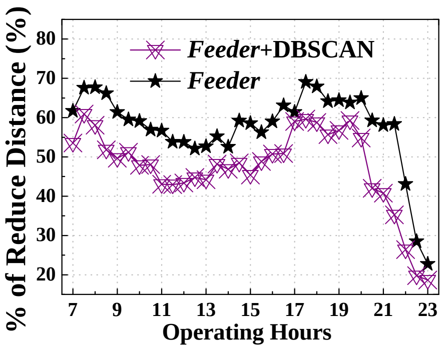


Figure 6.27: Reduced Distance

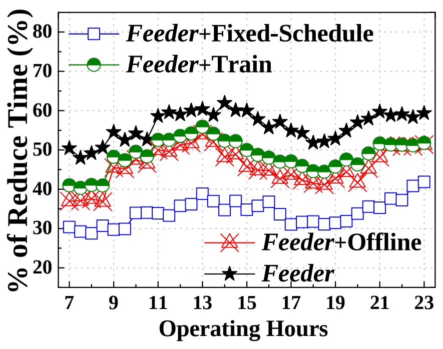


Figure 6.28: Reduced Time

Note that we show the performance of the Feeder service in terms of percentages, instead of the nominal values, because of the various travel time and last-mile distances at different subway stations. In Figures 6.29 and 6.30, we show the nominal values of the reduced travel time and the last-mile distance for the subway station CheGongMiao with the largest passenger arrival in Shenzhen. In Figure 6.29, we found that the average reduced last-mile distance fluctuates but Feeder performs better than Feeder+DBSCAN. In Figure 6.30, we observed a similar tendency as previously shown in Figure 6.28, i.e., Feeder outperforms others, and the performance is better in the non-rush hour.

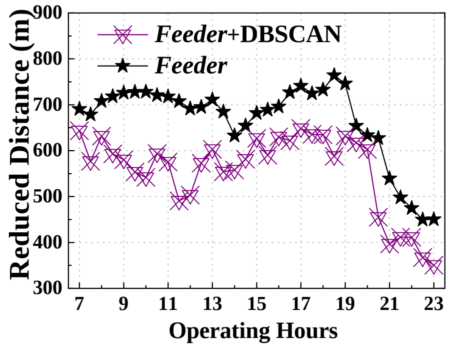


Figure 6.29: Distance at CGM

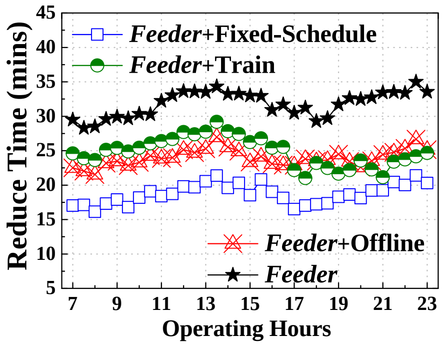


Figure 6.30: Time at CGM

Day of Week

Feeder+Weekday as well as Feeder+Weekend are used to test context-aware stop updating based on the performance of Feeder on weekdays and weekends. Figures 6.31 and 6.32 plot their reduced distance and time, respectively. In both of the figures, we found that Feeder+Weekday has higher reduced distances than Feeder+Weekend during the morning and evening rush hour. This is because the residents travel in the morning and evening rush hour on the weekday, while they travel in the regular daytime on the weekend.

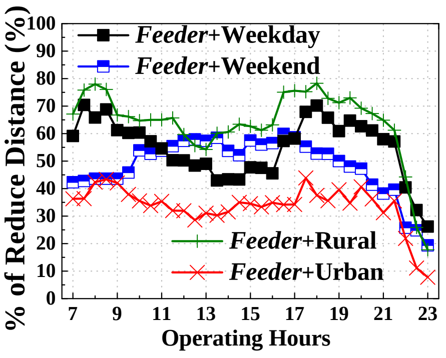


Figure 6.31: Reduced Distance

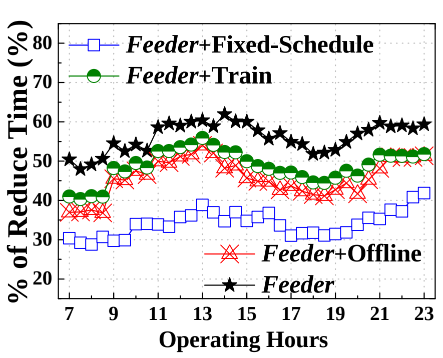


Figure 6.32: Reduced Time

District Population

Feeder+Urban gives the performance of Feeder in three urban districts (i.e., FuTian, LuoHu, and NanShan) in Shenzhen with high population levels, while Feeder+Rural gives the performance in three rural districts (i.e., Baoan, LongHua, and LongGang) with low population levels. Figures 6.31 and 6.32 plot their reduced distances and times. We found that Feeder+Rural has higher reduced distances than Feeder+Urban during all day. This is because there are fewer and sparser subway stations in the rural districts, leading to long last-mile distances.

6.10.3 Impacts of System Parameters

We test the impacts of four system parameters as follows.

Time Slot Length t

In Figure 6.33, we evaluate impacts of the slot length t, which decides the Feeder's granularity on scheduling. Note that t has no effect on Fixed-Schedule and co-design schedules with train arrivals. With the increase of t, the performance of Feeder and Feeder+Offline increases first and then decreases. This is because the prediction on exiting passengers in a smaller slot is not accurate. But when the slot becomes too long, the passenger wait times are also prolonged.

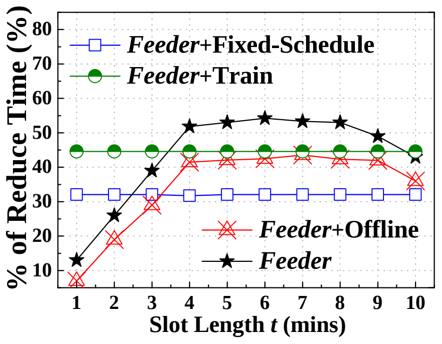


Figure 6.33: Time vs. t

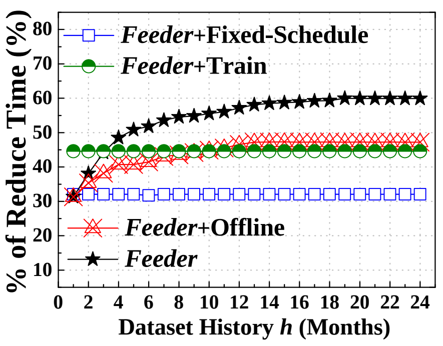


Figure 6.34: Time vs. h

Historical Dataset Length h

We investigate how much historical information is necessary for the predictions on passenger exiting stations in Figure 6.34. As expected, the longer the time, the better the performance. But a too long slot does not help much. Even with 6-month historical datasets, Feeder reduces 52% of the travel time for passengers.

Vehicle Status n & c

In Feeder, we set a different vehicle number n for each different station due to the various demand. For a station S_j, the default $n_j = \frac{N(\tau)}{c}$ where the default c is set to 20, which is the normal capacity of a MiniBus; $N(\tau)$ is the number of exiting passengers using Feeder (i.e., the half of all passengers) during the round trip time slot τ for a vehicle of a station S_j. Figure 6.35 plots the reduced time on different multiples of n.

With more vehicles, the percentage of the reduced time for Feeder increases, since the intervals between departures are reduced. The default multiple of n is 1.5.

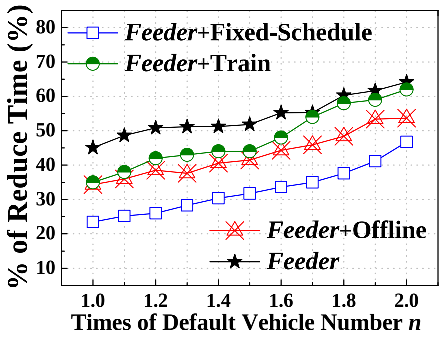


Figure 6.35: Time vs. n

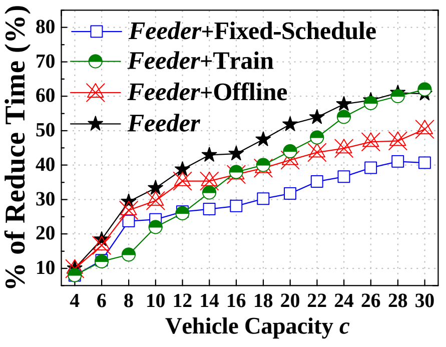


Figure 6.36: Time vs. c

We investigate the impact of the vehicle capacity c on Feeder in Figure 6.36. With the increase of c, the reduced time for Feeder increases. This is because a vehicle with a large capacity carries more passengers, and thus reduced the wait time. It implies that Feeder functions more effectively when vehicles can carry more passengers. The performance of Feeder+Train is depended on capacity since it cannot predict passenger numbers of each train, and a larger vehicle can reduce uncertain of passenger arrivals.

6.10.4 Evaluation Summary

We have the following observations based on the results. (i) The performance of Feeder is depended on the time of the day as shown by Figures 6.27, 6.28, 6.29 and 6.30. The day of week and district population also have significant impacts on Feeder as in Figures 6.31 and 6.32. Among these three real-world contexts, the district population has the largest affects on the performance, and then the day of week, and finally the time of day. (ii) The slot length has significant impacts on Feeder's performance, and generally as in Figure 6.33, the longer the slots, the more accurate the prediction about exiting passenger numbers, yet the longer the wait time. But when the slot length is set between 4 to 8 mins, the difference in performance is not obvious. (iii) How much historical data to be used by Feeder significantly affects the performance of Feeder as in Figure 6.34. Normally the longer the history, the better the performance. But the effect becomes less obvious when the history is longer than 6 months. (iv) The Feeder vehicle status, i.e., vehicle number and vehicle capacity, has big impacts on the passenger travel time as in Figures 6.35 and 6.36. It seems Feeder is more sensitive to the vehicle capacity than the vehicle number, which motivate us to use few big vehicles, instead of more small vehicles, in real-world large-scale implementation. (v) The three design components of Feeder, i.e., stop selection, route computation, and departure time computation are more effective than DBSCAN-based stop selection, fixed departure times and routes, as shown by the fact that Feeder outperforms others.

6.11 Discussion

Passenger Involvement. Feeder is described as an automatic and transparent service for passengers who do not have to provide any additional information, *e.g.*, arriving time at public transit stations or real destinations such as home and work addresses. But unfortunately the majority of passengers is not willing to provide detailed travel demand due to several reasons such as manual efforts and privacy. Sampling a subset of passengers who are willing to provide requests would introduce a bias against other passengers.

First-Mile Travel and Other Types of Travel. In this work, we focus on the last-mile problem only, and do not aim to address generic travels or the first-mile travel where passengers travel from origins to transit stations. It has a different setting where the time of a passenger starting the travel from an origin cannot be accurately predicted without active passenger involvement such as smartphone apps. A dedicated first-mile service based smartphone apps may also be used to address the last-mile problem if passengers would like to participate by providing detailed demand.

Privacy Protections. We took three steps to protect passenger privacy. (i) Anonymization: all data are anonymized by providers and all identifiable IDs in data are replaced with serial identifiers. (ii) Minimal Exposure: we only store and process the data that are useful for our Feeder service, and drop other information for the minimal exposure. (iii) Aggregation: our Feeder service uses the aggregated results and is not focused on individual residents.

Real-world Deployment Issues. We focus on technical aspects of Feeder, and here we discuss some real-world issues. (i) Focusing on data utilization, we envision that a passenger would pay a flat fare for short last-mile transit in Feeder. But more sophisticated fare models can be designed based on unique public transit fare structures in targeted cities. (ii) A portion of passengers (*e.g.*, visitors) may pay cash to purchase temporary cards, so we have no historical data about these passengers. But our method still applies because we can infer their exiting stations and times by general travel trends given entering stations and times. (iii) In a city where exiting a station does not require using smartcards, we can still infer an exiting station of a passenger by exploring his/her next entering station, assuming most passengers take round trips. (iv) If passengers use their smartcards in the Feeder service, Feeder design would be easier because we would know their real destinations. But we still need Feeder to predict passenger-exiting times in subway networks to schedule vehicle departures. (v) The main deployment cost for Feeder is the service vehicle, which we envision would be carpooling-based passenger vehicles such as passenger vans or minibuses, instead of regular taxis. Based on this carpooling feature, Feeder would significantly reduce passenger fare comparing to the taxicabs. In Feeder, the most of calculation is performed at the server side because we have to use real-time data consolidated in the server for prediction. If the real-time smartcard data can be accessed by frontend onboard devices, the calculation can also be dispatched to the frontend.

6.12 Conclusions

In this chapter, as one component of the application design layer for mobileCPS, we analyze, design, implement, and evaluate a service Feeder to tackle the last-mile problem with extreme-scale urban sensing infrastructures, reducing 68% of last-mile distances and 52% of travel time on average. Our technical endeavors provide a few valuable insights, which are hoped to be useful for commercially implementing Feeder-like data-driven services in the near future. Specifically, (i) we found unprecedented evidence of the last-mile problem, and design guidelines based on large-scale infrastructure datasets; (ii) we customized an onboard device supporting the essential functionalities (e.g., communication and sensing) for real-time on-demand services; (iii) we combined several yet independent datasets to design a data-driven service and affirmed that complicated functions (e.g., stop location and departure time optimizations) should be designed based on real-world data.

Chapter 7

Future Work

In this chapter, I present my future work along with some concrete directions.

7.1 Overview

My future work is to address the fundamental challenge in CPS with mobile urban systems. In short, it is to balance real-time mobility demand and supply based on heterogeneous models from imperfect data. (i) Due to loose-control sensing in heterogeneous systems, the physical data we have are far from perfect. They are noisy, sparse, implicit, untimely, and inconsistent. (ii) Based on these imperfect data from different systems, the models we have are heterogeneous in terms of scale, timeliness, and granularity, and completeness, especially when we consider the multi-source data driven models. (iii) Based on these models, we need some application-specific designs to improve the urban mobility efficiency, which typically requires domain knowledge to dynamically balance mobility demand and supply in real-world setting.

To address those challenges, my future work essentially has two parts. The theoretical part is to design and test novel computer science techniques focused on data life cycles, including low-quality heterogeneous data cleaning, data fusion, model integration, model predictive control along with privacy and security issues in urban systems. The application part is to implement several novel multi-source data-driven applications.

7.2 Specific Directions

I use three concrete directions to introduce my future work in three directions.

7.2.1 Imperfect-data Inference

The first direction is about imperfect-data inference. Currently, we are working with our collaborators in Shenzhen to improve the Shenzhen bus arrival prediction service. The bus system in Shenzhen has more than 14,000 buses, and 10,000 stations with more than a 10 million ridership. The goal of this project is to accurately infer bus arrival time by real-time bus GPS. The preliminary version of the service has been deployed, and the user request account is given in Figure 7.1.

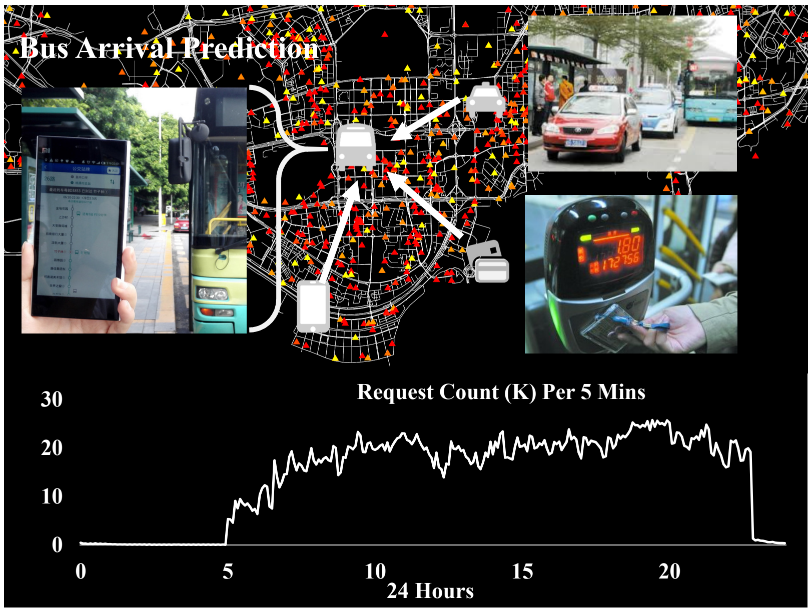


Figure 7.1: Bus Arrival Prediction

But the key challenge we have for this service is that the bus GPS data is missing all the time. To address this challenge, we propose a solution to integrate multi-source data from nearby taxis, smartcards and user app data to increase the accuracy of this service. The technical approach is to use context-aware tensor decomposition to recover missing bus GPS data by minimizing recovering errors.

7.2.2 Heterogeneous-Model Integration

The second direction is heterogeneous model integration. We are working with our collaborators in Shenzhen to model real-time energy consumption as in Figure 7.2.

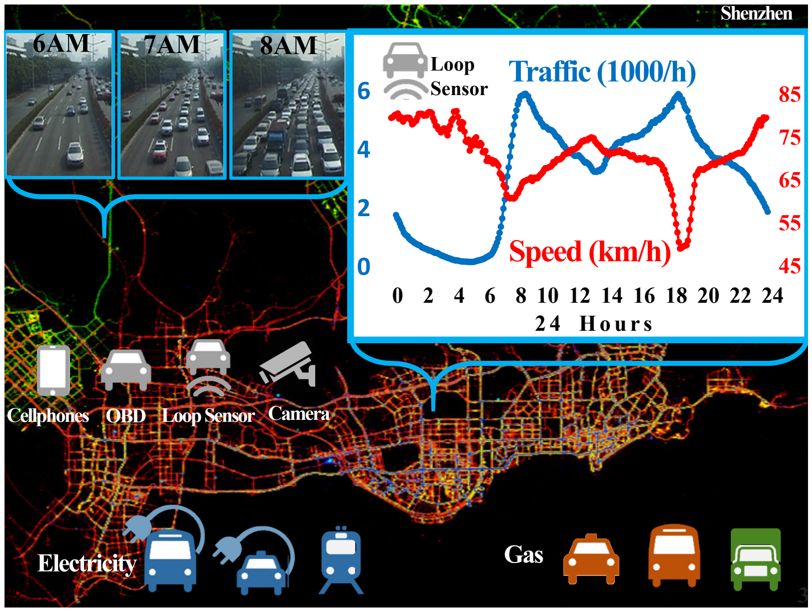


Figure 7.2: Transportation Energy Modeling

Our goal is to infer the real-time energy consumption at road segment level. The challenge is the fine spatiotemporal granularity from multiple heterogeneous data sources. Based on our datasets in Shenzhen, we can model electricity and gas consumption of commercial vehicles, but the key challenge is about private vehicles since they do not upload their data to the cloud server. We aim to use existing urban infrastructures, e.g., cellphone, OBD devices, cameras and loop sensors to infer the traffic volume and speed, and thus to infer the energy consumption as shown in the figure. The technical approach is to preform heterogeneous model integration based on semi-supervised multi-view learning to minimize disagreement between heterogeneous models.

7.2.3 Mobility Demand and Supply Rebalancing

Finally, the last direction is about mobility demand and supply rebalancing in urban systems. We are working with our collaborators in Department of Transportation in Washington D.C. to design an efficient bike rebalancing algorithm for the D.C. Bike System as Figure 7.3.

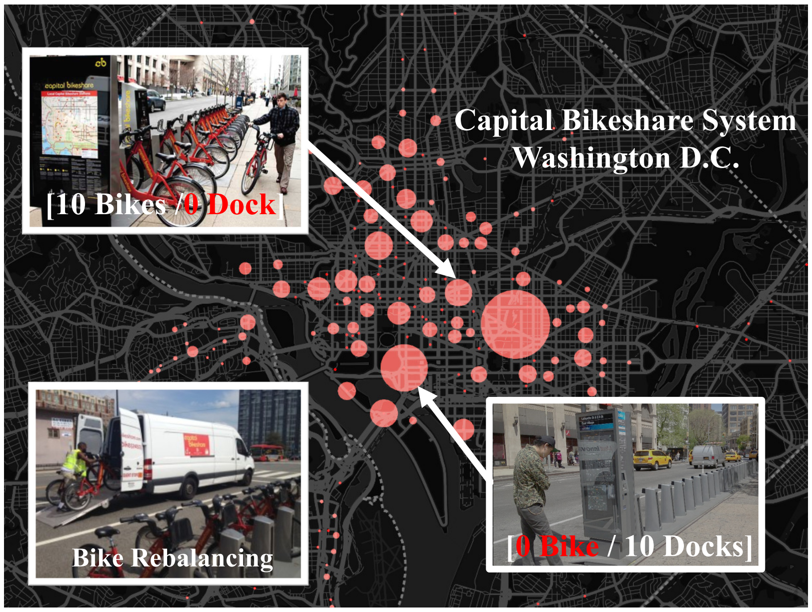


Figure 7.3: D.C. Bike Rebalancing

It gives the station map for the DC bike system. Every dot is a station, and the bigger the station, the higher the demand. Due to high demand, we often find some stations without any bikes (where passengers cannot rent bikes), and some nearby stations with any docks (where passengers cannot return bikes). To address this issue, the system operators need to use trucks to move bikes between stations to ensure sufficient bike supply. But the key challenge we have is the dynamics passenger demand in the D.C. bike system. We aim to propose a uncertain demand and supply model based on real-time data with correlated contexts, e.g., weather, train schedule, other events. The technical approach is to preform robust model predictive control based on uncertain demand models to balance bike supply among different stations.

Chapter 8

Conclusions

In this dissertation, we propose a three-layer cyber-physical system called mobileCPS for the human mobility modeling and their urban applications. Specifically, for the real-time data feed layer, we found that integrating multi-source data from mobile urban systems enables high spatiotemporal granularity and coverage, making it suitable for many urban phenomenon modeling and application design. But such large-scale multi-source data have many errors, which have to be fixed before their potentials can be fully released. For the mobility abstraction layer, we found that the human mobility modeling based on single-view data introduces biases, which can be addressed by using multi-view learning methods. Every view itself is incomplete but they are often complementary to each other, and thus it is essential to model the completeness degree of a view before inferring the mobility. For application design layer, we targeted at two specific applications, i.e., carpooling service coRide and last-mile transit service Feeder. Both coRide and Feeder are driven by real-time multi-source urban data and improve the urban mobility efficiency. Finally, the key insight from this dissertation is that mobile urban infrastructures, e.g., cellphone, taxi, bus and subway systems, are amazingly complex distributed systems, which interact with each other explicitly or implicitly in real time at urban scale. The multi-source mobility data generated by these mobile systems can be used to infer the real-time urban phenomena, e.g., urban mobility patterns, which can be used to design and evaluate future mobility services, e.g., carpooling and last-mile transit, to address the most pressing concerns related to sustainability in our urbanization.

References

[1] Desheng Zhang, Juanjuan Zhao, Fan Zhang, and Tian He. UrbanCPS: a Cyber-Physical System Based on Multi-source Data with Model Integration. In *the ACM/IEEE 6th International Conference on Cyber-Physical Systems (ICCPS'15).*

[2] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: Concepts, methodologies, and applications. *ACM Trans. Intell. Syst. Technol.*, 5(3):38:1–38:55, September 2014.

[3] Yu Zheng. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.*, 6(3):29:1–29:41, May 2015.

[4] Filippo Simini, Marta C. Gonzlez, Amos Maritan, and Albert-Lszl Barabsi. A universal model for mobility and migration patterns. Nature, 2014.

[5] Javed Aslam, Sejoon Lim, Xinghao Pan, and Daniela Rus. City-scale traffic estimation from a roving sensor network. In *Proceedings of 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12.

[6] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: User movement in location-based social networks. KDD '11.

[7] Desheng Zhang, Jun Huang, Ye Li, Fan Zhang, Chengzhong Xu, and Tian He. Exploring human mobility with multi-source data at extremely large metropolitan scales. MobiCom '14, pages 201–212, 2014.

[8] Sibren Isaacman, Richard Becker, Ramón Cáceres, Margaret Martonosi, James Rowland, Alexander Varshavsky, and Walter Willinger. Human mobility modeling at metropolitan scales. MobiSys '12.

[9] Sibren Isaacman, Richard A. Becker, Ramón Cáceres, Stephen G. Kobourov, Margaret Martonosi, James Rowland, and Alexander Varshavsky. Ranges of human mobility in los angeles and new york. In *PerCom Workshops*, 2011.

[10] Kateřina Dufková, Jean-Yves Le Boudec, Lukáš Kencl, and Milan Bjelica. Predicting user-cell association in cellular networks. MELT'09.

[11] Sourav Bhattacharya, Santi Phithakkitnukoon, Petteri Nurmi, Arto Klami, Marco Veloso, and Carlos Bento. Gaussian process-based predictive modeling for bus ridership. UbiComp '13.

[12] Neal Lathia and Licia Capra. How smart is your smartcard?: Measuring travel behaviours, perceptions, and incentives. UbiComp '11.

[13] Raghu Ganti, Mudhakar Srivatsa, Anand Ranganathan, and Jiawei Han. Inferring human mobility patterns from taxicab traces. UbiComp '13.

[14] Fosca Giannotti, Mirco Nanni, Dino Pedreschi, Fabio Pinelli, Chiara Renso, Salvatore Rinzivillo, and Roberto Trasarti. Unveiling the complexity of human mobility by querying and mining massive trajectory data. *The VLDB Journal*.

[15] Lars Backstrom, Eric Sun, and Cameron Marlow. Find me if you can: Improving geographical prediction with social and spatial proximity. In *WWW '10*.

[16] New York Times. Limited share-a-cab test to begin soon. *http://www.nytimes.com/2010/02/22/nyregion/22ataxis.html*.

[17] Supershuttle service. http://www.supershuttle.com/.

[18] 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 *SenSys*, page 9, 2013.

[19] Shuo Ma, Yu Zheng, and Ouri Wolfson. T-share: A large-scale dynamic taxi ridesharing service. In *ICDE 2013*. IEEE, April 2013. The Best Paper Runner-Up Award.

[20] Shuo Ma and Ouri Wolfson. Analysis and evaluation of the slugging form of ridesharing. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 64–73, New York, NY, USA, 2013. ACM.

[21] Johanna Meurer, Martin Stein, David Randall, Markus Rohde, and Volker Wulf. Social dependency and mobile autonomy: Supporting older adults' mobility with ridesharing ict. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 1923–1932, New York, NY, USA, 2014. ACM.

[22] Uber. *Announcing UberPool*, 2015. Available at `http://newsroom.uber.com/announcing-uberpool/`.

[23] Ronald Fagin and John H. Williams. A fair carpool scheduling algorithm. *IBM J. Res. Dev.*

[24] Ganesh Ananthanarayanan, Maya Haridasan, Iqbal Mohomed, Doug Terry, and Chandramohan A. Thekkath. Startrack: a framework for enabling track-based applications. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, MobiSys '09.

[25] Jiannong Cao Guihai Chen Wei Lou Nianbo Liu, Ming Liu. When transportation meets communication: V2p over vanets. In *2010 International Conference on Distributed Computing Systems*, ICDCS '10.

[26] Moni Naor. On fairness in the carpool problem. *J. Algorithms.*

[27] Don Coppersmith, Tomasz Nowicki, Giuseppe Paleologo, Charles Tresser, and Chai Wah Wu. The optimality of the online greedy algorithm in carpool and chairman assignment problems. *ACM Trans. Algorithms.*

[28] Rajesh Krishna Balan, Khoa Xuan Nguyen, and Lingxiao Jiang. Real-time trip information service for a large taxi fleet. In *Proceedings of the international conference on Mobile systems, applications, and services*, MobiSys '11.

[29] Wei Wu, Wee Siong Ng, Shonali Krishnaswamy, and Abhijat Sinha. To taxi or not to taxi? - enabling personalised and real-time transportation decisions for mobile users. In *Proceedings of the 2012 IEEE 13th International Conference on Mobile Data Management*, MDM '12.

[30] Yong Ge, Chuanren Liu, Hui Xiong, and Jian Chen. A taxi business intelligence system. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11.

[31] 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*, KDD '10, 2010.

[32] 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*, UbiComp '11, 2011.

[33] Yan Huang and Jason W. Powell. Detecting regions of disequilibrium in taxi services under uncertainty. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, 2012.

[34] Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xie Xing. Discovering spatio-temporal causal interactions in traffic data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, 2011.

[35] Daqing Zhang, Nan Li, Zhi-Hua Zhou, Chao Chen, Lin Sun, and Shijian Li. i-bat: detecting anomalous taxi trajectories from gps traces. In *13th conference on ubiquitous computing*, UbiComp '11.

[36] Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. Constructing popular routes from uncertain trajectories. In *Proceedings of the international conference on Knowledge discovery and data mining*, KDD '12.

[37] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *Proceedings of the international conference on Knowledge discovery and data mining*, KDD '11.

[38] Wangsheng Zhang, Shijian Li, and Gang Pan. Mining the semantics of origin-destination flows using taxi traces. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, 2012.

[39] Jing Yuan, Yu Zheng, and Xing Xie. Discovering regions of different functions in a city using human mobility and pois. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, 2012.

[40] Yu Zheng, Yanchi Liu, Jing Yuan, and Xing Xie. Urban computing with taxicabs. UbiComp '11.

[41] James Biagioni and Jakob Eriksson. Map inference in the face of noise and disparity. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12.

[42] Xuemei Liu, James Biagioni, Jakob Eriksson, Yin Wang, George Forman, and Yanmin Zhu. Mining large-scale, sparse gps traces for map inference: comparison of approaches. In *Proceedings of the international conference on Knowledge discovery and data mining*, KDD '12.

[43] 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. SenSys '13, 2013.

[44] MiniBus in Hong Kong. *http://www.minibus.hk/*.

[45] Yu Zheng, Yukun Chen, Quannan Li, Xing Xie, and Wei-Ying Ma. Understanding transportation modes based on gps data for web applications. *ACM Trans. Web*, 4(1), January 2010.

[46] Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. Learning transportation mode from raw gps data for geographic applications on the web. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, 2008.

[47] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on gps data. In *Proceedings of the 10th international conference on Ubiquitous computing*, UbiComp '08, 2008.

[48] Anandatirtha Nandugudi, Taeyeon Ki, Carl Nuessle, and Geoffrey Challen. Pocketparker: Pocketsourcing parking lot availability. UBICOMP '14, 2014.

[49] James Biagioni, Tomas Gerlich, Timothy Merrifield, and Jakob Eriksson. Easytracker: Automatic transit tracking, mapping, and arrival time prediction using smartphones. SenSys '11, 2011.

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

[51] Rijurekha Sen and Rajesh Krishna Balan. Challenges and opportunities in taxi fleet anomaly detection. SENSEMINE'13, 2013.

[52] Shan Jiang, Gaston A. Fiore, Yingxiang Yang, Joseph Ferreira, Jr., Emilio Frazzoli, and Marta C. González. A review of urban computing for mobile phone traces: Current methods, challenges and opportunities. UrbComp '13, 2013.

[53] Lijun Sun, Der-Horng Lee, Alex Erath, and Xianfeng Huang. Using smart card data to extract passenger's spatio-temporal density and train's trajectory of mrt system. UrbComp '12.

[54] Yu Zheng and Xing Xie. Learning travel recommendations from user-generated gps traces. *ACM Trans. Intell. Syst. Technol.*

[55] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. Nature, 2008.

[56] T. G. Kolda and B. W Bader. Tensor decompositions and applications. In *SIAM Review 51*, 2009.

[57] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: N-dimensional tensor factorization for context-aware

collaborative filtering. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 79–86, New York, NY, USA, 2010. ACM.

[58] Dimitri P. Bertsekas. Non-linear programming. In *Athena Scientific*, 1999.

[59] Transport Committee of Shenzhen. Statistical data for transportation in shenzhen. http://www.sz.gov.cn/jtj/tjsj/zxtjxx/, 2014.

[60] Taxi user surveys. 2013. San Francisco Municipal Transportation Agency.

[61] Schaller Consulting. The new york city taxicab fact book. *http://www.schallerconsult.com/taxi/taxifb.pdf.*

[62] NYC Taxi and Limousine Commission. Taxi of tomorrow survey. 2011.

[63] nationmaster. Energy statistics. *http://www.nationmaster.com/ graph/ene_oil_con-energy-oil-consumption.*

[64] Wikipedia. Emissions trading or cap and trade. *http://en.wikipedia.org/wiki/Emissionstrading.*

[65] 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*, GIS '10.

[66] Data100 Company. *Taixcab Carpooling Survey*, 2012. http://wenku.baidu.com/view/2f0fea1f964bcf84b9d57b4c.html.

[67] Lie Gu, Stan Z Li, and Hong-Jiang Zhang. Learning probabilistic distribution model for multi-view face detection. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[68] Paul Oberlin, Sivakumar Rathinam, and Swaroop Darbha. A transformation for a multiple depot, multiple traveling salesman problem. In *Proceedings of the conference on American Control Conference*, ACC'09.

[69] Brian Ferris, Kari Watkins, and Alan Borning. Onebusaway: Results from providing real-time arrival information for public transit. CHI '10, 2010.

[70] American Public Transportation Association. In *http://www.apta.com/mediacenter/ptbenefits/Pages/default.aspx*, 2010.

[71] The Last Mile Problem. *http://en.wikipedia.org/wiki/Lastmile(transport)*.

[72] Desheng Zhang, Juanjuan Zhao, Fan Zhang, Ruobing Jiang, and Tian He. Feeder: Supporting Last-Mile Transit with Extreme-Scale Infrastructure Data. In *the 14th ACM Conference on Information Processing in Sensor Networks (IPSN '15)*.

[73] Christopher MacKechnie. Walking distance to transit. *http://publictransport.about.com*.

[74] Desheng Zhang, Ye Li, Fan Zhang, Mingming Lu, Yunhuai Liu, and Tian He. coRide: Carpool Service with a Win-Win Fare Model for Taxicab Networks. In *the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys'13)*.

[75] A. Moore. K-means and hierarchical clustering. In *http://www. autonlab.org/tutorials/kmens11.pdf*, 2001.

[76] WILLIAM COOK and ANDR E ROHE. Computing minimum-weight perfect matchings. In *INFORMS Journal on Computing*, 1999.

Appendix A

Published Work

In addition to this dissertation, my research results are also given by the following published work.

A.1 Urban-Data Analytics

- Desheng Zhang, Juanjuan Zhao, Fan Zhang, and Tian He. coMobile: Real-time Human Mobility Modeling at Urban Scale by Multi-View Learning In the 23rd ACM Conference on Advances in Geographic Information Systems (SIGSPATIAL'15). Acceptance Rate: 38/216=18%.

- Desheng Zhang, Ruobing Jiang, Shuai Wang, Yanmin Zhu, Bo Yang, Tian He, and Jian Cao. Everyone Counts: Fine-Grained Digital Media Advertising in Urban Metro Systems. In the IEEE International Conference on Big Data (BigData'15). Acceptance Rate: 62/363=17%.

- Desheng Zhang, Tian He, Shan Lin, Sirajum Munir, and John A. Stankovic. Dmodel: Online Taxicab Passenger Demand Model from Large Roving Sensor Networks. In the 3rd IEEE International Congress on Big Data (BigData'14). Acceptance Rate: 38/200=19%.

- Desheng Zhang, Tian He, Yunhuai Liu, and John A. Stankovic. CallCab: a Unified Recommendation System for Carpooling and Regular Taxicab Services.

In the IEEE International Conference on Big Data (BigData'13). Acceptance Rate: 45/259=17.4%.

A.2 Cyber-Physical Systems

- Desheng Zhang, Juanjuan Zhao, Fan Zhang, Ruobing Jiang and Tian He. Feeder: Supporting Last-Mile Transit with Extreme-Scale Infrastructure Data. In 14th ACM Conference on Information Processing in Sensor Networks (IPSN'15). Acceptance Rate: 27/110=24.3%.
- Desheng Zhang, Juanjuan Zhao, Fan Zhang, and Tian He. UrbanCPS: a Cyber-Physical System Based on Multi-Source Data with Model Integration. In the ACM/IEEE 6th International Conference on Cyber-Physical Systems (ICCPS'15). Acceptance Rate: 25/91=27.4%.
- F. Miao, S. Lin, S. Munir, J. A. Stankovic, H. Huang, Desheng Zhang, T. He, and G. J. Pappas. Taxi Dispatch with Real-Time Data in Metropolitan Areas - a Receding Horizon Control Approach. In the ACM/IEEE 6th International Conference on Cyber-Physical Systems (ICCPS'15). Acceptance Rate: 25/91=27.4%.
- Desheng Zhang, Jun Huang, Ye Li, Fan Zhang, Chengzhong Xu, and Tian He. Exploring Human Mobility with Multi-Source Data at Extremely Large Metropolitan Scales. In the 20th ACM International Conference on Mobile Computing and Networking (MobiCom'14). Acceptance Rate: 36/220=16.3%.
- Desheng Zhang, and Tian He. Collaborative Sensing and Control in Large-Scale Transportation Systems. Invited Paper. In the International Conference on Collaboration Technologies and Systems (CTS'14).
- Desheng Zhang, Ye Li, Fan Zhang, Ming Lu, Yunhuai Liu, and Tian He. coRide: Carpool Service with a Win-Win Fare Model for Taxicab Networks. In the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys'13). Acceptance Rate: 21/123=17.1%.
- Desheng Zhang, and Tian He. pCruise: Reducing Cruising Miles for Taxicab Networks. In the 33rd IEEE Real-time Systems Symposium (RTSS'12). Acceptance

Rate: 35/157=22.2%.

A.3 Wireless Networking

- Desheng Zhang, Tian He, Yunhuai Liu, Yu Gu, Fan Ye, Raghu K. Ganti, and Hui Lei. Acc: Generic On-Demand Accelerations for Neighbor Discovery. In the 10th ACM Conference on Embedded Networked Sensor Systems (SenSys'12). Acceptance Rate: 23/123=18.6%.

- Desheng Zhang, Tian He, Fan Ye, Raghu K. Ganti, and Hui Lei. EQS: Neighbor Discovery and Rendezvous Maintenance with Extended Quorum System. In the 32nd International Conference on Distributed Computing Systems (ICDCS'12). Acceptance Rate: 71/515=13.7%.

- Desheng Zhang, Jinbao Li, and Longjiang Guo. MCR: a Dynamic and Optimal Duty Cycle Based MAC Protocol for Wireless Sensor Networks. In the 4th Conference for Wireless Sensor Networks (CWSN'10). Best Paper Award, 1/200+.

- Jinbao Li, and Desheng Zhang. M&M: A Multi-Channel MAC Protocol with Multiple Channel Reservation for Sensor Networks. In the Conference on Cyber-Enabled Computing (CyberC'10). Best Paper Runner-up Award, 1/297.

A.4 Journal Publication

- Desheng Zhang, Tian He, Yunhuai Liu, Yu Gu, Fan Ye, Raghu K. Ganti, and Hui Lei. Generic Neighbor Discovery in Mobile Applications. In ACM Transactions on Sensor Networks (TOSN), 2015.

- Desheng Zhang, Tian He, Lin Shan, Sirajum Munir, and John A. Stankovic. Online Cruising Mile Reduction for Large-Scale Taxicab Networks. In the IEEE Transactions on Parallel and Distributed Systems (TPDS). Early Access Article, Oct. 2014.

- Desheng Zhang, Tian He, Yunhuai Liu, and John A. Stankovic. A Unified Recommendation System for Carpooling and Regular Taxicab Services. In Special

Issue on Big Data of the IEEE Transactions on Emerging Topics in Computing (TETC). Volume 2, No. 3, Sept. 2014.