Feeder: Supporting Last-Mile Transit with Extreme-Scale Urban Infrastructure Data

Desheng Zhang[†] zhang@cs.umn.edu
Juanjuan Zhao[‡] jj.zhao@siat.ac.cn
Fan Zhang[‡] zhangfan@siat.ac.cn
Ruobing Jiang[◇] likeice@sjtu.edu.cn
Tian He[◇†] tianhe@cs.umn.edu

[†]Department of Computer Science and Engineering, University of Minnesota, USA
[‡]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China
[◇]Shanghai Jiao Tong University, China

Abstract

In this paper, we propose a transit service Feeder 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 evaluation results show that compared to the ground truth, Feeder reduces last-mile distances by 68% and travel time by 52% on average.

Categories and Subject Descriptors

H.4 [**Information System Application**]: Miscellaneous

General Terms

Algorithms, Model, Experimentation, Application

Keywords

Last-Mile Transit, Urban Infrastructure

1 Introduction

Pubic transit contributes significantly to reduction of travel delay and gas consumption [9], *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 [4]. 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 2. This issue is known as "*the last-mile problem*", which is a key barrier to better public-transit utilization [3].

In this paper, 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. For a passenger, the wait is typically more insufferable than the ride due to short ride time in last-mile trips; the walk also requires much more effort than the ride. Therefore, Feeder has the two following objectives to enhance passenger experience on waiting 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 Walking Distance*: This objective would also 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 design challenge to infer detailed passenger last-mile transit demand (*i.e.,* exiting stations and times as well as fine-grained destinations) for Feeder optimizations

IPSN '15, April 14 - 16, 2015, Seattle, WA, USA

ACM 978-1-4503-3475-4/15/04$15.00
http://dx.doi.org/10.1145/2737095.2737121

without active contributions from passengers.

To address this challenge, we employ extreme-scale urban infrastructures to infer last-mile transit demand, 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. 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 (the most crowded city in 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 paper 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 [2].
- 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 two key components: (i) a *departure time computation* to minimize wait times based on straightforward yet efficient smartcard data processing; and (ii) a *service stop selection* to minimize last-mile walking distances based on cellphone and taxi data. 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 Tanglang 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 the paper as follows. Section 2 gives our motivation. Section 3 presents an overview. Section 4 describes the front-end devices. Sections 5, 6 and 7 depict the back-end server. Sections 8 and 9 validate Feeder with a real-world test and a large-scale evaluation. Section 10 discusses real-world issues, followed by the related work and the conclusion in Sections 11 and 12.

2 Motivation for Last-Mile Transit

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 5.

In Figure 1, we show last-mile trip lengths between a subway station XingDong in Shenzhen and inferred passenger destinations closer to XingDong than other stations. The average length is given in Figure 2. The average distance 1.4 km is longer than the distance that passengers are willing to walk [8], *i.e.*, 400 to 800 m.

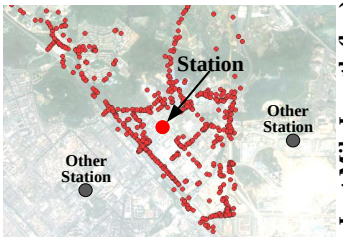


Fig 1. Last-Mile in XD

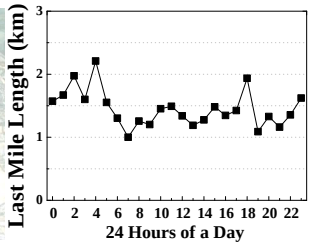


Fig 2. AVG. Length in XD

In Figure 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.

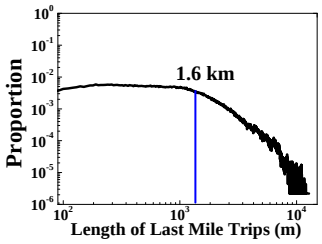


Fig 3. Last-Mile Trips

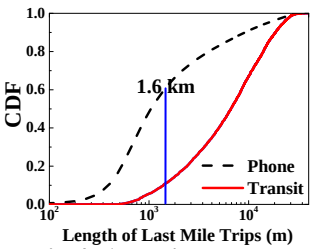


Fig 4. All Trip Lengths

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 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 taxicab trips). Since cellphone trips can be seen as proxies for all trips due to popularity of cellphones, 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.

3 Feeder Service Overview

We first present an operational scenario for a Feeder service based on Figure 5. Without the Feeder service, a passenger would (i) enter public transit at an entering station, and (ii) exit public transit at an exiting station, and (iii) walk to his/her final destination. Thus, the last-mile walking distance is from the exiting station to the destination.

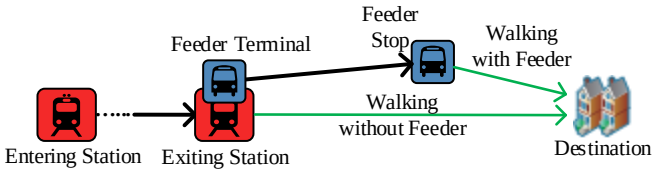


Fig 5. Feeder Operational Scenario

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, which are optimally selected by Feeder according to inferred fine-grained destinations; (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, two 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, and (ii) how to infer fine-grained destinations in order to optimize service stop locations. These challenges are solved in the following framework, which consists of three key components as in Figure 6.

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

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

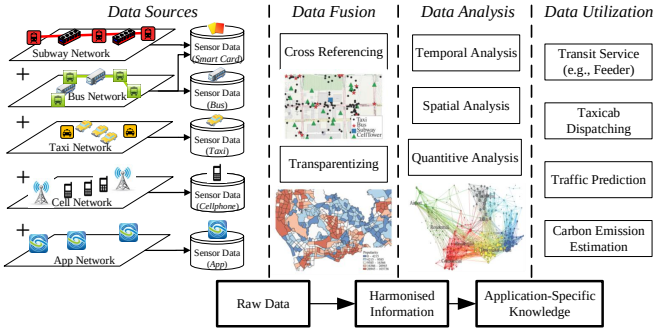


Fig 6. Feeder System Framework

Management (introduced in §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 §6): calculating effective departure times online based on mined passenger exiting stations and times to minimize passenger wait times; (iii) Stop Location Selection (introduced in §7): selecting efficient stops offline based on mined destinations to minimize last-mile walking distances.

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 §4.

4 Feeder Device Design

In our project [17], we develop a prototype for front-end data transmission to support functions in Feeder. Figure 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.

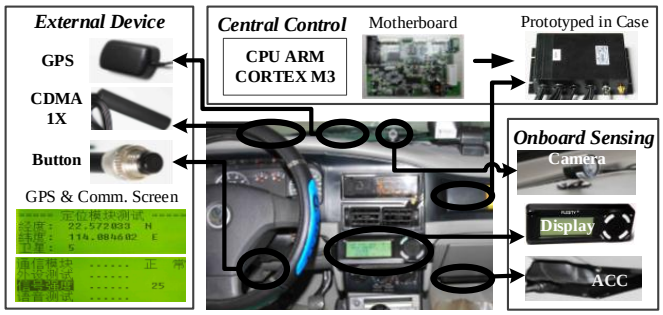


Fig 7. Feeder Device Design and Deployment

Through 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 Feeder server. The logical status, *i.e.*, numbers of onboard passengers, is also important to the Feeder service, because it affects departure

times. In this work, we envision that drivers or fare collecting devices will track the number of onboard passengers and thus change logical status to inform the Feeder 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.

5 Feeder Server: Data Management

In this section, we first present data input in Section 5.1, and then discuss our data cleaning in Section 5.2, and finally describe our data fusion in Section 5.3.

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 on average. In particular, 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.

To enable a comprehensive offline analysis, we have stored a large amount of streaming data as in Figure 8.

Cellphone Data		Taxicab Data	
Collected From	10/1/2013	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 Number	Date and Time
Cell Tower ID	Activities	Status	GPS Coordinates
Bus Data		**Smartcard Reader Data**	
Collection Period	01/01/13-Now	Collection Period	07/01/11-Now
Number of Vehicles	13,960	Number of Cards	16,000,000
Data Size	790 GB	Data Size	600 GB
Record Number	9,855,657,663	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

Fig 8. Streaming Datasets from Urban Infrastructures

Such a big amount of data requires significant efforts for the efficient storage and management. Accordingly, 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 management, we use several MapReduce based tools, such as Pig and Hive.

5.2 Data Cleaning

Due to the extremely large size of our data, we found three main kinds of errant data. (i) Data with Logical Errors: *e.g.*, GPS coordinates show that a vehicle is far away from its previous locations. 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. (ii) Duplicated Data: *e.g.*, the smartcard datasets show two identical records for the same smartcard. Such duplicated data are detected by comparing the timestamp of every record belonging to the same data source, *e.g.,* the same smartcard. (iii) 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 consistency of incoming data for every data source, *e.g.,* a taxicab. The above errors may result from various real-world reasons, *e.g.*, hardware malfunctions, software issues, and data transmission.

To address these 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, *e.g.,* time of day and digital maps. We next store the data by dates and categories. Finally we compare the temporal consistency 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 analyses, but given the long time period, we believe our analyses are still insightful.

5.3 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 9, where we do not differentiate subway and bus passengers, since they are both detected by smartcard readers as sensors.

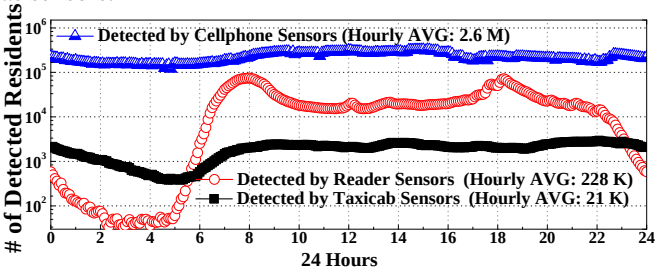


Fig 9. Detected Residents by Data

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 1.

Table 1. Heterogenous Sensors 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

As in Table 1, (i) cellphone sensors cover 95% of 11 million permanent residents, but each cellphone 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 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 Feeder Server: Departure Time Calculation

We first discuss why we need dynamic departure times in Section 6.1, and show how we predict passenger-exiting stations and times for dynamic departure times in Section 6.2, and present how we optimize departures in Section 6.3.

6.1 Motivation for Dynamic Departure Times

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* [9]; 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.

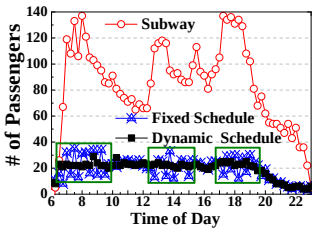


Fig 10. Passenger Number

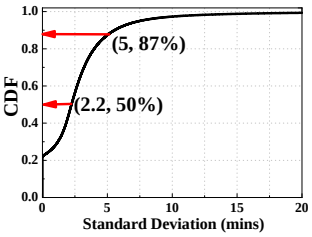


Fig 11. Travel Time

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 10, 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 subway 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. This is 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.

6.2 Exiting Times and Stations

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 of vehicles because passengers have already been waiting during the period. As follows, we show how to predict passenger exiting times and stations.

6.2.1 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 11 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.

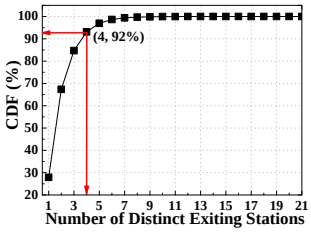


Fig 12. Distinct Stations

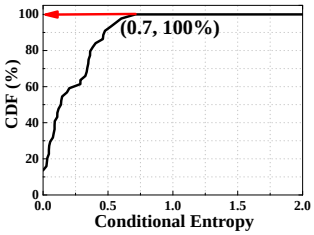


Fig 13. Conditional Entropy

6.2.2 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 12 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 13 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.

6.3 Departure Time Optimization

An optimization overview for a station S_j is in Figure 14.

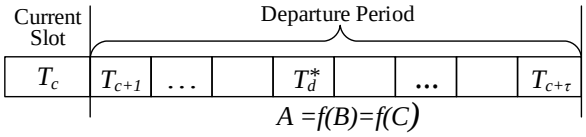


Fig 14. Overview of Departure Optimization

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 in the system 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 further 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 9.

7 Feeder Server: Stop Location Selection

We first present our motivation in Section 7.1, and then show how to infer passengers' destination in Section 7.2, and finally optimize stop selections in Section 7.3.

7.1 Using Cellphone and Taxicab Data

Different from regular transit, last-mile transit aims to reduce passengers' walking distances to destinations [3]. As a result, we need a destination-driven stop selection to reduce walking distances. However, large-scale fine-grained destinations are usually unknown. In this work, 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 urban 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 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 15 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.

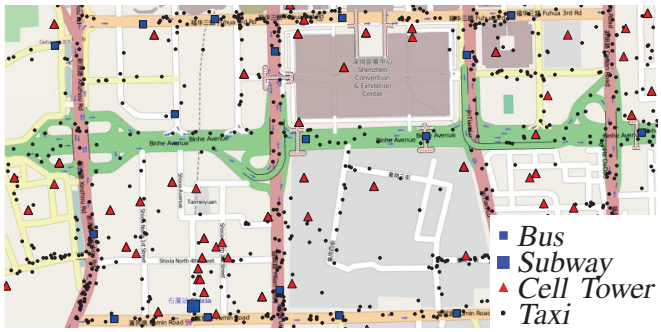
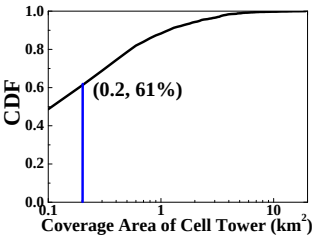


Fig 15. Inferred Destinations in Shenzhen Downtown

More rigorously, we show the CDF of coverage areas of all 17,859 cell towers in Figure 16 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 17 where 91% of regions have at least one destination per 100 m^2, which is typically within a walking distance.



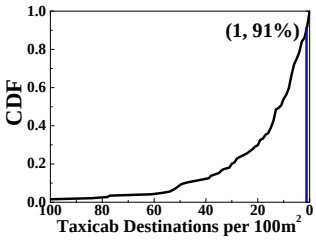


Fig 16. Cell Tower Coverage **Fig 17. Taxicab Destination**

7.2 Destinations for Public Transit Passengers

Based on the above discussion, we infer the transit 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 [11], this approach has a high accuracy to infer important locations for cellphone users.

To obtain D^t, we employ offline 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.

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.

7.3.1 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 [13] 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, due to real-world considerations, 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 $\frac{E_j}{100\times 100m^2}$, 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.

7.3.2 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.

8 Feeder 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 18. But in a real-world service with more potential passengers, a Feeder vehicle shall have a high capacity, which enables more environmental friendly services.

Fig 18. Feeder Service in Tanglang Station

8.1 Implementation 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 shortest route to link these stops to the Tanglang station, which is similar to the classic traveling salesman problem. We developed a $\frac{3}{2}$ approximation algorithm based on traversals on the minimum spanning tree to obtain the route linking all stops, but due to space limitation, we focus on data-driven components. The stops and route are given in Figure 19. 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.

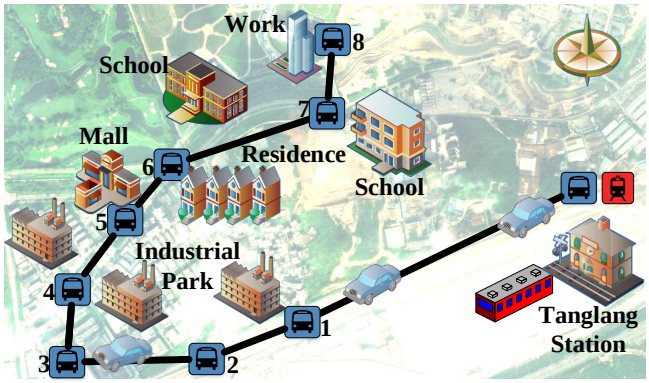


Fig 19. Real World Scenario

We collected the data in a 30-day period about the 12 passengers who take the 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 by three phones, with which arriving moments, departure moments, last-mile distance and travel time (equal to wait time plus ride time) were calculated.

8.2 Implementation Evaluation

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 20 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.

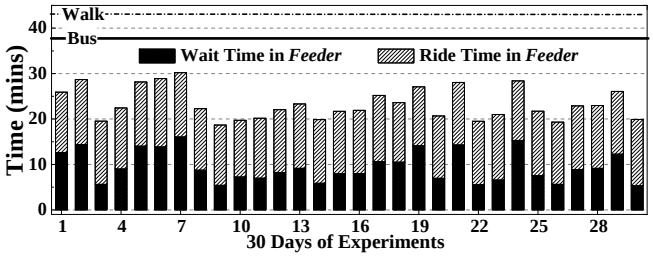


Fig 20. Average Travel Time in 30 days

We evaluate the average travel time for 12 passengers in Figure 21. We found the wait time for some passengers is shorter than others. This is because the prediction 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 always better than scheduled bus because of a combined effect that the bus stop is farther than the Feeder stop to both subway stations and final destinations of passengers as well as Feeder has a better schedule.

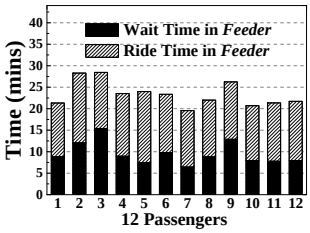


Fig 21. Individual Time

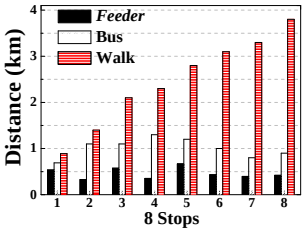


Fig 22. Last-Mile Distance

Finally, we evaluate Feeder by the last-mile distance. Due to the limited passengers in our trial, 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 22, 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.

9 Feeder Data Driven Evaluation

With datasets introduced in Figure 8, 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. 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+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;

(iv) **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.

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.

9.1 Impacts of Logical Contexts

We test the impacts of three logical contexts as follows.

9.1.1 Time of Day

We evaluate impacts of the time of day during the normal public transit operating hours from 7AM to 11PM. Figure 23 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.

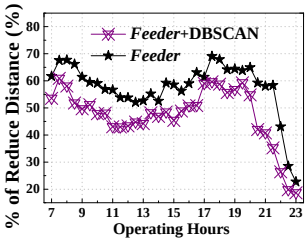


Fig 23. Reduced Distance

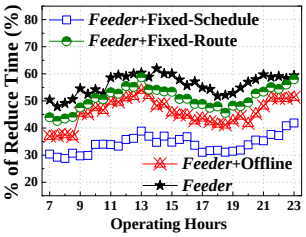


Fig 24. Reduced Time

Figure 24 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.

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 25 and 26, 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 25, we found that the average reduced last-mile distance fluctuates but Feeder performs better than Feeder+DBSCAN. In Figure 26, we observed a similar tendency as previously shown in Figure 24, *i.e.*, Feeder outperforms others, and the performance is better in the non-rush hour.

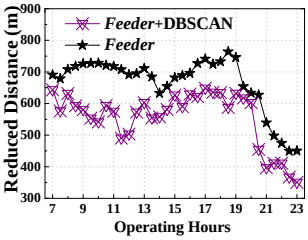


Fig 25. Distance at CGM

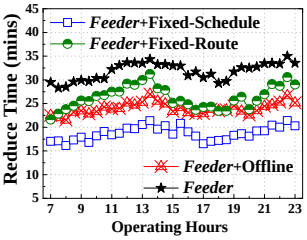


Fig 26. Time at CGM

9.1.2 *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 27 and 28 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.

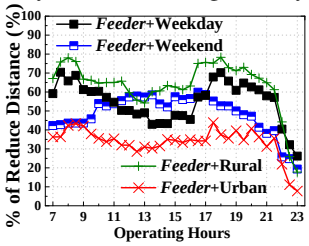


Fig 27. Reduced Distance

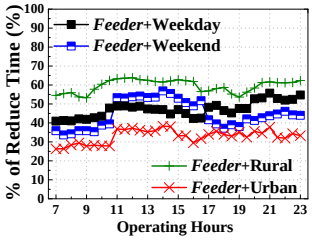


Fig 28. Reduced Time

9.1.3 *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 27 and 28 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.

9.2 Impacts of System Parameters

We test the impacts of four system parameters as follows.

9.2.1 *Time Slot Length t*

In Figure 29, 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.

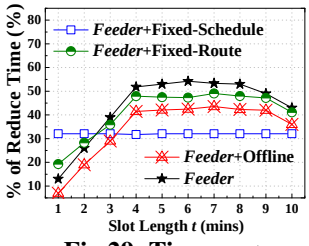


Fig 29. Time vs. t

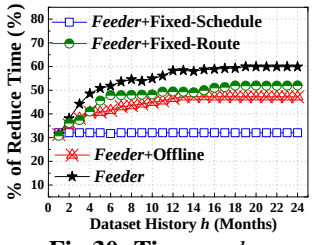


Fig 30. Time vs. h

9.2.2 *Historical Dataset Length h*

We investigate how much historical information is necessary for the predictions on passenger exiting stations in Figure 30. 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.

9.2.3 *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 31 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.

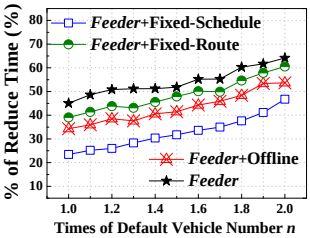


Fig 31. Time vs. n

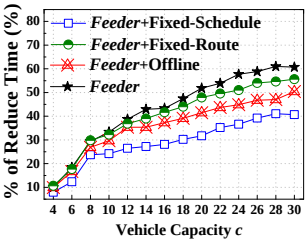


Fig 32. Time vs. c

We investigate the impact of the vehicle capacity c on Feeder in Figure 32. 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.

10 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 travel 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. 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.

Implementation in Different Cities. A transit service typically has different performance in different cities due to the geographic and demographic features, *e.g.*, travel patterns, urban density, and data accessibility. It is therefore extremely important to evaluate the proposed Feeder 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 a potential evaluation.

11 Related Work

In addition to walking, bikes, taxicabs, and personal vehicles discussed in the introduction, taxicab ridersharing and minibus services are two major efforts for the last-mile problem. Some cities, *e.g.*, New York City [15], Beijing [12] and Shenzhen [18], 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 [1], use minibuses to deliver passengers closer to their destinations, but they have fixed routes and schedules. The key difference between Feeder and ridersharing is that Feeder learns passenger demand automatically, while ridersharing assumes demand is given in advance. Feeder is also different from the above services in terms of low infrastructure costs, flexible network coverages, and real-time supports from the Feeder server with online data from urban infrastructures.

Another type of related work to Feeder is urban data-driven applications. The increasing availability of GPS has encouraged a surge of research for urban data-driven applications. Many novel applications are proposed to assist urban residents or city officials, *e.g.*, finding parking spots for drivers [14], inferring real-world maps based on GPS data [6], predicting bus arrival times [7], enabling passengers to query taxicab availability to make informed transit choices [5], predicting passenger demand for taxicab drivers [10], modeling the urban transit [19], and navigating new drivers based on GPS traces of experienced drivers [16]. 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.

12 Conclusion

In this work, 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.

13 Acknowledgements

The authors thank our shepherds and Dr. Ling Yin in SIAT for the data support. This work was supported in part by US NSF Grant CNS-0845994, CNS-1239226, NSFC Grant U1401258, and China 973 Program 2015CB352400.

14 References

[1] MiniBus in Hong Kong. *http://www.minibus.hk/*.
[2] Sample data. http://cloud.siat.ac.cn/Feeder.html.
[3] The Last Mile Problem. *http://en.wikipedia.org/wiki/Lastmile(transport)*.
[4] AMERICAN PUBLIC TRANSPORTATION ASSOCIATION. In *http://www.apta.com/mediacenter/ptbenefits/Pages/default.aspx*.
[5] BALAN, R. K., NGUYEN, K. X., AND JIANG, L. Real-time trip information service for a large taxi fleet. MobiSys '11.
[6] BIAGIONI, J., AND ERIKSSON, J. Map inference in the face of noise and disparity. SIGSPATIAL '12.
[7] BIAGIONI, J., GERLICH, T., MERRIFIELD, T., AND ERIKSSON, J. Easytracker: Automatic transit tracking, mapping, and arrival time prediction using smartphones. SenSys '11.
[8] DITTMAR, H., AND OHLAND, G. The new transit town: Best practices in transitoriented development. *Island Press*.
[9] FERRIS, B., WATKINS, K., AND BORNING, A. Onebusaway: Results from providing real-time arrival information for public transit. CHI '10.
[10] GE, Y., XIONG, H., TUZHILIN, A., XIAO, K., AND GRUTESER, M. An energy-efficient mobile recommender system. KDD '10.
[11] ISAACMAN, S., BECKER, R., CÁCERES, R., MARTONOSI, M., ROWLAND, J., VARSHAVSKY, A., AND WILLINGER, W. Human mobility modeling at metropolitan scales. MobiSys '12.
[12] MA, S., ZHENG, Y., AND WOLFSON, O. T-share: A large-scale dynamic taxi ridesharing service. In *ICDE 2013*.
[13] MOORE, A. K-means and hierarchical clustering. In *http://www. autonlab.org/tutorials/kmens11.pdf* (2001).
[14] NANDUGUDI, A., KI, T., NUESSLE, C., AND CHALLEN, G. Pocketparker: Pocketsourcing parking lot availability. UBICOMP '14.
[15] NEW YORK TIMES. Limited share-a-cab test to begin soon. *http://www.nytimes.com/2010/02/22/nyregion/22ataxis.html*.
[16] YUAN, J., ZHENG, Y., XIE, X., AND SUN, G. Driving with knowledge from the physical world. KDD '11.
[17] ZHANG, D., LI, Y., ZHANG, F., LU, M., LIU, Y., AND HE, T. coRide: Carpool Service with a Win-Win Fare Model for Taxicab Networks. In *the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys'13)*.
[18] ZHANG, D., LI, Y., ZHANG, F., LU, M., LIU, Y., AND HE, T. coride: Carpool service with a win-win fare model for large-scale taxicab networks. SenSys '13.
[19] ZHENG, Y., CHEN, Y., LI, Q., XIE, X., AND MA, W.-Y. Understanding transportation modes based on gps data for web applications. *ACM Trans. Web 4*, 1 (Jan. 2010).