Data-Driven Pick-Up Location Recommendation for Ride-Hailing Services

Zhidan Liu, *Member, IEEE*, Hongquan Zhang, Guofeng Ouyang, Junyang Chen, and Kaishun Wu, *Member, IEEE*

Abstract—Ride-hailing service (RHS) has become an important transportation mode in our daily life. Although many works have been proposed to improve RHS from different aspects, only few works focus on the selections of pick-up locations, where rider and driver meet and start a trip. In this paper, we present *MPLRec*, a data-driven pick-up location recommendation system that exploits riders' specific mobility demands, e.g., destination, and historical experiences to meet riders' travel requirements. *MPLRec* generates potential pick-up locations over the road network and characterizes them with rich features that describe a location from the riders' perspective. We also build spatio-temporal indexes to organize potential pick-up locations and historical data for facilitating online recommending. When processing an online recommendation request, *MPLRec* derives candidate pick-up locations and investigates them with materialized features, which are computed from historical order and trajectory data while considering rider's mobility demands. Based on these features, a novel scoring function is used to derive the best pick-up location for each request. Moreover, we implement an RHS simulator to evaluate *MPLRec* using large-scale practical ride-hailing datasets. Extensive experiments and simulations demonstrate the effectiveness and efficiency of *MPLRec*, which can complete each request within 0.5 s and largely reduce the ride-hailing costs when compared to baseline methods.

Index Terms—Ride-hailing service, pick-up location, recommendation, mobility demand, spatio-temporal index

1 Introduction

Ride-hailing service (RHS), e.g., Didi [1] and Uber [7], has emerged as a novel on-demand transportation mode for urban citizens. Different from traditional taxi services, RHS allows a rider to easily hail a ride and timely track the vehicle's location with a smartphone, rather than standing at the roadside to wait for an available taxi. By incentivizing private vehicles to provide ride-hailing services, RHS also promotes the sharing economy and enlarges the transportation capacity of a city [38], [41]. With the great convenience and flexibility, RHS becomes increasingly popular across the world. According to a recent report, the global ride-hailing service market will reach $126.52 billions in 2025 [4].

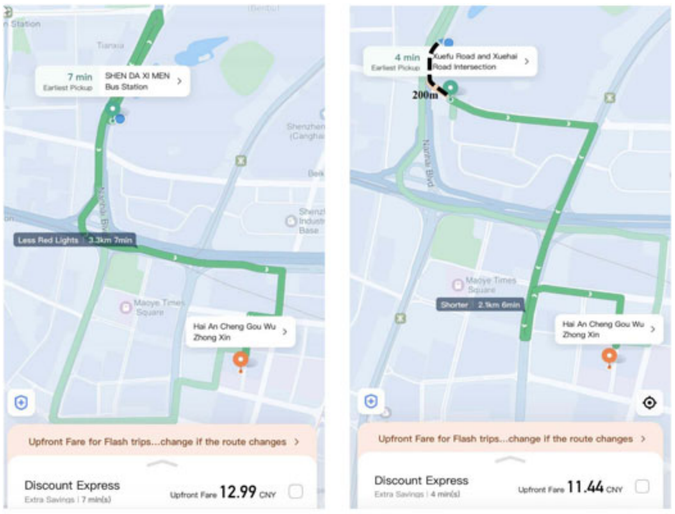
Tremendous research efforts have already been devoted to improve the efficiency of RHS systems from various aspects, e.g., order assignments to match ride-hailing orders with drivers [25], [43], vehicle dispatching to balance supply and demand across different locations [23], [41], incentive pricing model to encourage more users and drivers [8], [29], safety and privacy protection [30], [33], *etc.* while the problem of docking riders and assigned drivers at the suitable pick-up locations remains less explored. In fact, the pick-up location in RHS matters a lot. Because pick-up locations will not only determine the route planning, which can affect travel time and ride-hailing fares, but also have much influence on users' experiences, e.g., extra walking distance and waiting time.

In the reality, most riders determine pick-up locations based on their own experiences, while in an unfamiliar environment riders tend to choose the adjacent curbside as the trip origins. In the literature, there exist only a few works focusing on the selections of pick-up locations [9], [35], [51]. Most of existing works group historical pick-up locations into clusters, and then recommend popular location clusters that are the closest to riders [35], [51]. However, these works have severely overlooked the rider's destination while merely investigating walking distance or waiting time of each candidate pick-up location. As a result, such methods will recommend sub-optimal solutions that lead to detoured routes yet not satisfy riders' requirements on multiple metrics, e.g., walking distance to the recommended pick-up location, waiting time for available vehicles, overall travel time and the fare.

Some practical RHS systems, e.g., Didi [1], adopt a similar clustering based pick-up location recommendation method. Fig. 1 illustrates a real ride-hailing example, where the rider wants to take a ride to the nearby shopping mall (i.e., the orange circle). The rider sends a request to the RHS system (e.g., Didi in this example) at her current location

• *Zhidan Liu, Hongquan Zhang, Guofeng Ouyang, and Junyang Chen are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China. E-mail: {liuzhidan, junyangchen}@szu.edu.cn, {zhanghongquan2020, ouyangguofeng2021}@email.szu.edu.cn.*
• *Kaishun Wu is with The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China, and also with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China. E-mail: wu@szu.edu.cn.*

Manuscript received 23 May 2022; revised 23 August 2022; accepted 19 September 2022. Date of publication 22 September 2022; date of current version 8 January 2024.
This work was supported in part by China NSFC under Grants 62172284, 61872248, and U2001207, in part by Guangdong Basic and Applied Basic Research Foundation under Grants 2022A1515010155 and 2017A030312008, in part by Shenzhen Science and Technology Foundation under Grants ZDSYS20190902092853047 and R2020A045, in part by the Project of DEGP under Grants 2019KCXTD005 and 2021ZDZX1068, and in part by Guangdong "Pearl River Talent Recruitment Program" under Grant 2019ZT08X603.
(Corresponding author: Kaishun Wu.)
Digital Object Identifier no. 10.1109/TMC.2022.3208566


7 min Earliest Pickup
SHEN DA XI MEN Bus Station
Less Red Lights 3.3km 7min
Hai An Cheng Gou Wu Zhong Xin
Upfront Fare for Flash trips...change if the route changes
Discount Express
Upfront Fare 12.99 CNY
4 min Earliest Pickup
Xuefu Road and Xuehai Road Intersection
200m
Shorter 2.1km 6min
Hai An Cheng Gou Wu Zhong Xin
Upfront Fare for Flash trips...change if the route changes
Discount Express
Upfront Fare 11.44 CNY

(a) Default pick-up location (b) Alternative pick-up location

Fig. 1. A real ride-hailing example in our daily life, where different pick-up locations lead to distinct itinerary plans with varied costs.

(i.e., the blue circle) within our campus. Fig. 1(a) shows the default pick-up location (i.e., the green circle) at adjacent roadside, which is close to the rider, and the itinerary plan, which incurs waiting time of $7\,mins$, travel distance of $3.3\,km$, estimated travel time of $7\,mins$, with a trip fare about 12.99 CNY. From Fig. 1(a), we see a clear detour in the route. Due to the road network structure, the vehicle needs to make a U-turn at some place and then heads to the destination. If the rider is aware of this situation, she may set another pick-up location to avoid such a turn around. In fact, she can walk to another road intersection, and meets the assigned driver there, as shown in Fig. 1(b). With this new trip origin, the RHS system reschedules the itinerary plan, and we find that all trip metrics are largely reduced. According to the new itinerary plan, the rider would arrive at the destination $4\,mins$ earlier, while saving fare by 1.55 CNY. The only cost is that the rider needs to walk $200\,m$ more, with estimated walking time less than $1\,min$. Such a case is common in our daily life, where inappropriate pick-up locations cause sub-optimal itinerary plans with detoured trips and terrible user experiences. Through the analysis of real-world ride-hailing datasets, we find that about 61.21% ride-hailing orders undergo detoured routes (see more in Section 2.2).

In this paper, we present the <u>M</u>ultiobjective <u>P</u>ick-up <u>L</u>ocation <u>Rec</u>ommendation system – *MPLRec*, which considers the rider's specific mobility demands and historical experiences learned from massive ride-hailing data to recommend suitable pick-up locations for RHS systems. *MPLRec* generates potential pick-up locations over a road network, and characterizes them with rich features of traffic conditions, travel accessibility to other locations, and ride-hailing hotness, which indicates the probability of successfully hailing vehicles at a location. To optimize online recommending, *MPLRec* builds spatio-temporal indexes to organize potential pick-up locations and historical data. Therefore, *MPLRec* can efficiently investigate each candidate pick-up location using its materialized features, which are derived from historical orders and trajectories, and the rider's specific mobility demands, e.g., destination. We propose a novel scoring function, which considers ride-hailing hotness and distance-related factors, to compute the best pick-up location for each rider. Furthermore, we propose an assessment methodology, which can generate realistic ride-hailing orders and vehicle statuses and simulates the RHS operations, to comprehensively evaluate pick-up location recommendation methods.

The key contributions of this work are summarized as follows.

- To the best of our knowledge, we are the first to consider multiobjective pick-up location recommendation (MPLR) problem, which aims to recommend suitable pick-up locations for satisfying riders' multiple demands in RHS.
- To address the MPLR problem, we propose *MPLRec*, a data-driven system that can efficiently recommend pick-up locations by exploiting the rider's specific mobility demands and historical ride-hailing order/trajectory data.
- We implement an RHS simulator to simulate realistic RHS operations and evaluate *MPLRec* using real-world data.
- We have conducted extensive experiments and simulations to evaluate the performance of *MPLRec* based on the large real-world ride-hailing datasets. Results demonstrate the effectiveness and efficiency of *MPLRec*, which can complete each recommendation within 0.5 s and largely reduce various costs when compared to the baselines.

The rest of the paper is organized as follows. In Section 2, we present the motivation and problem statement. Section 3 elaborates the design of *MPLRec*, and Section 4 introduces the assessment methodology. The performance evaluation is described in Section 5. In Section 6, we review and discuss the related works. Section 7 finally concludes this paper.

2 Motivation and Problem Statement

In this section, we will first present some preliminary definitions about ride-hailing service (RHS), and then analyze the real-world ride-hailing datasets to motivate our work. Finally, we formulate the problem of multiobjective pick-up location recommendation. Table 1 summarizes the key notations used in this paper.

2.1 Preliminary

A typical RHS transaction involves three stakeholders, i.e., a rider, a driver, and the RHS system. The centralized RHS system continuously receives ride-hailing orders from riders, and assigns them to suitable drivers, who will deliver riders from the specified pick-up locations to their destinations, following the travel routes planned on a road network.

Definition 1 (Road Network). *A road network is denoted by a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where each vertex $v \in \mathcal{V}$ represents a geo-location (e.g., road intersection), and each edge $e \in \mathcal{E}$ represents a road segment that is associated with a weight $dist(e)$, indicating the length of road segment e.*

Definition 2 (Ride-hailing Order). *A ride-hailing order is denoted by $\mathbf{r}_i = \{i, t^o_{r_i}, \ell^o_{r_i}, \ell^p_{r_i}, \ell^d_{r_i}\}$, where i is the order ID, $t^o_{r_i}$ and $\ell^o_{r_i}$ indicate when and where the rider places the order,*

TABLE 1
Summary of the Key Notations

Notation	Description
$\mathcal{G}(\mathcal{V},\mathcal{E})$	Road network graph
$dist(\cdot)$	A function to calculate road network distance
$\mathbf{r}_i$	The ith ride-hailing order
$\ell^o_{r_i}$	Ordering location of order $\mathbf{r}_i$
$t^o_{r_i}$	Ordering time of order $\mathbf{r}_i$
$\ell^p_{r_i}$	Pick-up location of order $\mathbf{r}_i$
$t^p_{r_i}$	Pick-up time of order $\mathbf{r}_i$
$\ell^d_{r_i}$	Destination of order $\mathbf{r}_i$
$t^d_{r_i}$	Time of arriving at destination for order $\mathbf{r}_i$
$L^w_{r_i}$	Walking distance for order $\mathbf{r}_i$
$T^w_{r_i}$	Waiting time for order $\mathbf{r}_i$
$L^d_{r_i}$	Driving distance for order $\mathbf{r}_i$
$T^d_{r_i}$	Driving time for order $\mathbf{r}_i$
$\mathcal{T}_{r_i}$	Order $\mathbf{r}_i$'s vehicle trajectory
$\mathcal{F}_{r_i}$	Ride-hailing fare of order $\mathbf{r}_i$
$\mathcal{R}_{r_i}$	Travel route for order $\mathbf{r}_i$
$\Delta_j \in \mathbb{T}$	The jth time slot of $\mathbb{T} = \{\Delta_1, \Delta_2, \ldots, \Delta_{96}\}$
$s_{i,j}$	Travel speed of road segment e_i in time slot Δ_j
$p_i \in \mathbb{P}$	The ith potential pick-up location in $\mathbb{P} = \{p_1, p_2, \ldots, p_{\lvert\mathcal{E}\rvert}\}$
$h_{i,j}$	Ride-hailing hotness of pick-up location p_i in time slot Δ_j
$G_i \in \mathbb{G}$	The ith grid in $\mathbb{G} = \{G_1, G_2, \ldots, G_m\}$
β	Searching range
$f(\cdot)$	A scoring function to evaluate pick-up locations

while $\ell^p_{r_i}$ and $\ell^d_{r_i}$ are the pick-up location and destination of the order, respectively.

Typically, once a ride is needed, the rider opens an RHS APP, which can access current location $\ell^o_{r_i}$ of the rider and recommends potential pick-up locations to the rider. After determining the pick-up location $\ell^p_{r_i}$ and destination $\ell^d_{r_i}$, the rider submits an order $\mathbf{r}_i$ to the RHS system at time $t^o_{r_i}$. To serve order $\mathbf{r}_i$, the system searches nearby drivers, and then assigns this order to an appropriate driver according to some order assignment algorithms [31], [43]. Assume at time $t^a_{r_i}$, the driver accepts $\mathbf{r}_i$ at location $\ell^a_{r_i}$, and then drives to location $\ell^p_{r_i}$ to pick-up the rider at time $t^p_{r_i}$. Following a travel route planned by the RHS system, the driver will deliver the rider to destination $\ell^d_{r_i}$ at time $t^d_{r_i}$.

Based on above process, we can derive several major performance metrics for order $\mathbf{r}_i$, which are important for RHS systems:

- *Walking distance* $L^w_{r_i}$ measures the road network distance the rider walks from $\ell^o_{r_i}$ to $\ell^p_{r_i}$, i.e., $L^w_{r_i} = dist(\ell^o_{r_i}, \ell^p_{r_i})$.
- *Waiting time* $T^w_{r_i}$ indicates how long a rider will wait for the assigned driver, i.e., $T^w_{r_i} = t^p_{r_i} - t^o_{r_i}$.
- *Driving distance* $L^d_{r_i}$ is the travel distance between pick-up location and destination, i.e., $L^d_{r_i} = dist(\ell^p_{r_i}, \ell^d_{r_i})$.
- *Driving time* $T^d_{r_i}$ indicates the time a rider spends in the ride-hailing vehicle, i.e., $T^d_{r_i} = t^d_{r_i} - t^p_{r_i}$.

An RHS system will continuously log vehicle status while serving the ride-hailing orders, primarily for the purposes of fleet management and billing [23]. All status records of an order form this order's vehicle trajectory. Once the rider finally arrives at the destination, she would pay for the ride-hailing fare.

Definition 3 (Order's Vehicle Trajectory). *The vehicle trajectory $\mathcal{T}_{r_i}$ of an order $\mathbf{r}_i$ consists of a series of time-ordered records, where each record includes a timestamp, a driver ID, an order ID, and a GPS location.*

Definition 4 (Ride-hailing Fare). *The ride-hailing fare $\mathcal{F}_{r_i}$ of an order $\mathbf{r}_i$ is usually calculated based on driving distance $L^d_{r_i}$ and driving time $T^d_{r_i}$, i.e., $\mathcal{F}_{r_i} \sim (L^d_{r_i}, T^d_{r_i})$.*

Different RHS systems have different formulations to calculate the fare $\mathcal{F}_{r_i}$, while most fare formulations are directly related with $L^d_{r_i}$ and $T^d_{r_i}$. As a result, a rider usually prefers the shortest path with less driving time as the travel route that is jointly determined by pick-up location $\ell^p_{r_i}$, destination $\ell^d_{r_i}$, and traffic conditions.

2.2 Data Description, Analysis, and Motivation

To attract collective efforts on improving the performance of RHS, Didi's GAIA initiative [2] has opened some anonymized datasets. We have obtained such datasets for this study, including an order dataset and a vehicle trajectory dataset. These data were collected by Didi from downtown area of Chengdu city, China, in November 2016. Specifically, each record in the order dataset represents an order $\mathbf{r}_i$, which consists of an order ID, time $t^a_{r_i}$ and location $\ell^a_{r_i}$ of a driver while accepting this order, billing time when the rider paid for the order, and destination $\ell^d_{r_i}$. However, Didi does not open the ordering locations and time for these orders, mainly due to privacy concerns. The vehicle trajectory dataset contains trajectory records of all orders, where each trajectory record includes a driver ID, an order ID, a timestamp, and a GPS location.

We link the two datasets through order IDs. Since Didi starts logging trip details only when serving orders and stops logging once riders arrive at their destinations, we derive key information for each order $\mathbf{r}_i$ as follows: *the pick-up time $t^p_{r_i}$ and location $\ell^p_{r_i}$ are the first timestamp and GPS location, respectively, in order $\mathbf{r}_i$'s trajectory, while time $t^d_{r_i}$ of arriving at destination $\ell^d_{r_i}$ are the last timestamp and GPS location in $\mathbf{r}_i$'s trajectory, respectively.*

After filtering out erroneous data due to hardware failures and invalid orders canceled by the riders, we have in total 7065907 valid orders and their trajectories. In addition, we download road information of Chengdu city from OpenStreetMap (OSM) [5], and model the road network as a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ that consists of 214440 vertices and 466330 edges, covering an area of more than $70\,km^2$. With road network graph $\mathcal{G}$, we exploit FMM, a fast map matching algorithm [44], to map all trajectory data to the roads, and thus recover the travel route for each order.

Definition 5 (Order's Travel Route). *Order $\mathbf{r}_i$'s travel route $\mathcal{R}_{r_i}$ is a sequence of road segments, indicating the path between any two consecutive GPS locations in $\mathbf{r}_i$'s trajectory $\mathcal{T}_{r_i}$.*

Statistics About Didi's Datasets. We analyze these data and derive some statistical results about the datasets. Fig. 2(a) shows the driving distance distribution of all orders, while Fig. 2(b) shows the distribution of driving time. In general, most orders are short trips, e.g., 80% orders can be completed within $15\,mins$ with driving distances shorter than $6\,km$.

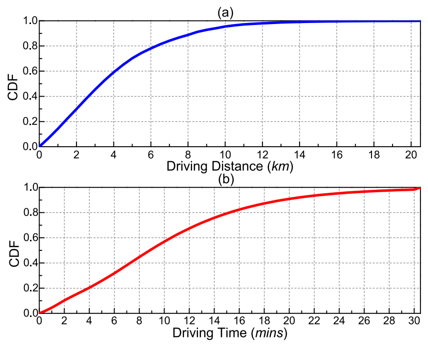


Fig. 2. CDF of driving distances and driving time for all orders.

Fig. 3(a) plots the distribution of *pick-up distances* between acceptance locations and pick-up locations. Didi indeed assigns nearby drivers to serve orders, e.g., the pick-up distances of 40% orders are only within $100\,m$ and 80% orders are within $2\,km$. On the other hand, the riders need to wait for the assigned vehicles coming. As shown in Fig. 3(b), although waiting time of 60% orders are within $10\,s$ merely, we still find that more than 30% orders incur long waiting time, e.g., more than $5\,mins$.

Motivation. As the trip origin, pick-up location is very important for RHS systems since it will affect system performance. First of all, as the meeting points for riders and assigned drivers, pick-up locations will determine the route planning. An inappropriate pick-up location may result in a detoured route that will increase both travel time and fare. Second, pick-up location will also affect user experiences. Because a rider usually places the order a bit earlier before the trip at other places, e.g., home or workplace, it thus takes time for the rider walking to the pick-up location. A far away location obviously introduces more walking distance. In addition, different locations across the city own varied probabilities to hail an available vehicle [40], [47], [50], leading to different waiting time for the riders. Lastly, the "visibility" of pick-up locations is also crucial [9], [51]. In reality, drivers often communicate with the riders multiple times before eventually finding the right meeting points, as some pick-up locations cannot be easily found.

Existing pick-up location recommendation methods primarily consider riders' current locations and recommend the most popular locations, which are the clustering results over historical pick-up locations [35], [40], [51]. These methods, however, largely overlook riders' destinations, and only derive sub-optimal solutions.

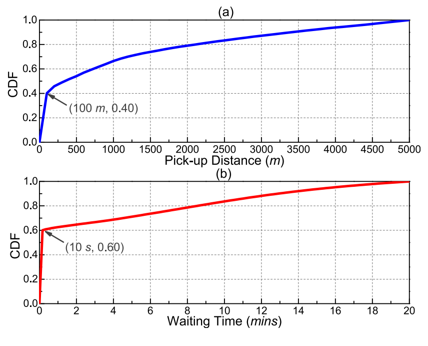


Fig. 3. CDF of pick-up distances and waiting time for all orders.

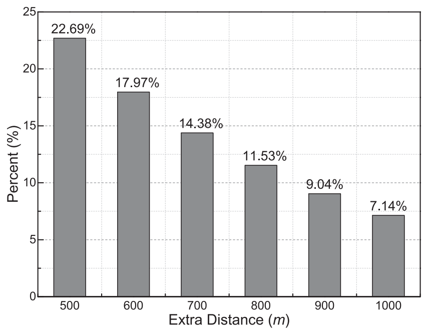


Fig. 4. Proportion of orders that have detoured routes due to inappropriate pick-up locations.

To investigate pick-up location recommendation performance of real RHS systems, i.e., Didi [1], we analyze the order data of November 18, 2016, the day having the most served orders. After filtering out short trips with driving distances less than $1\,km$, we have 191670 orders for the analysis. For each order $\mathbf{r}_i$, on the one hand we calculate its driving distance $L^d_{r_i}$ using the travel route. On the other hand, we compute the shortest path on road network graph $\mathcal{G}$ with A* algorithm [21] between the pick-up location $\ell^p_{r_i}$ and trip destination $\ell^d_{r_i}$, and denote the length of the shortest path as $L^{sp}_{r_i}$. We keep 149327 orders with $\frac{L^{sp}_{r_i}}{L^d_{r_i}} \geq 0.8$, while for the other orders we consider that drivers may intentionally choose different routes rather than the shortest ones.

For each reserved order $\mathbf{r}_i$, we try to recommend a new pick-up location. First, we search the road segments within $500\,m$ of the trip origin $\ell^p_{r_i}$,[1] and treat the midpoint of each road segment as a candidate pick-up location. Then, we calculate the shortest path between each candidate pick-up location and the trip destination $\ell^d_{r_i}$ using A* algorithm. The overall travel distance $L^{rec}_{r_i}$ is the sum of walking distance from $\ell^p_{r_i}$ to candidate pick-up location and driving distance to the destination. The candidate with the smallest overall travel distance $\hat{L}^{rec}_{r_i}$ is set as the recommended pick-up location. The difference $(L^d_{r_i} - \hat{L}^{rec}_{r_i})$ is considered as the detour distance, since drivers may detour a bit to the shortest path, due to inappropriate pick-up locations. Fig. 4 shows the proportion of orders under various detour distances. According to the statistics, we have found better pick-up locations for 61.21% orders, i.e., $(L^d_{r_i} - \hat{L}^{rec}_{r_i}) > 0$. Particularly, we find that 33878 out of 149327 orders, i.e., $\sim$ 22.69%, have detour distances larger than $500\,m$, and 7.14% orders may travel $1000\,m$ more due to the poor selections of pick-up locations.

In reality, due to unaware of road network structures and traffic conditions, most riders can only blindly take RHS's recommendations as the pick-up locations, which severely overlooks the impact of destinations on the selections of meeting points and thus leads to detoured routes. For example, riders may stand on the opposite sides of roads with respect to the destinations, and as a result the drivers have to detour to the shortest paths after picking-up the riders. Furthermore, a rider may choose an inappropriate pick-up

1. Since Didi's datasets do not contain the ordering location $\ell^o_{r_i}$ of each order $\mathbf{r}_i$, we thus try to recommend an alternative pick-up location other than the original one $\ell^p_{r_i}$ for order $\mathbf{r}_i$.

location and cause the driver run into traffic congestion, due to unaware of road traffics. Therefore, it is necessary and important to recommend suitable pick-up locations for the RHS systems, so that to provide conveniences for both riders and drivers.

2.3 Problem Statement

Recommending pick-up locations should take riders' destinations into account, and meanwhile satisfy riders' various requirements: (i) People usually prefers the nearby and conspicuous locations, which will introduce less walking distances and are easily spotted by the drivers. (ii) The hot spots with more ride-hailing vehicles passing by will be more preferable, since the riders can easily hail a vehicle there and thus avoid the long waiting time. (iii) Riders essentially want to arrive at the destinations as soon as possible, in order to save time and ride-hailing fares.

In view of above demands, we study the multiobjective pick-up location recommendation problem that is defined as follows.

Definition 6 (Multiobjective Pick-up Location Recommendation problem, MPLR). *Given a road network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, historical ride-hailing orders and vehicle trajectories, for each ride-hailing order* $\mathbf{r}$,[2] *MPLR problem aims to recommend a suitable pick-up location for* $\mathbf{r}$, *such that walking distance L_r^w, waiting time T_r^w, and ride-hailing fare $\mathcal{F}_r$ can be minimized.*

Challenges. Despite benefits of pick-up location recommendations for RHS stakeholders, it is non-trivial to address the MPLR problem, due to the following challenges.

(1) *Quantitatively evaluating pick-up locations from historical data.* Although we could collect massive ride-hailing order and vehicle trajectory data, how to exploit such data to quantitatively evaluate and recommend potential pick-up locations while considering riders' mobility demands remains unexplored. As a result, it calls for devising a novel quantitative method to comprehensively evaluate pick-up locations.

(2) *Recommending efficiency should be guaranteed.* Once a pick-up location recommendation request is received, the RHS systems should immediately recommend the best pick-up location. While it will take much time to retrieve, process, and analyze a large amount of data to finalize the recommendations. Therefore, how to optimize the computations is important and challenging.

(3) *Effective assessment mechanism for pick-up location recommendations is missing.* So far, there exist no tools or platforms to assess pick-up location recommendation methods. It is not an option to deploy such methods on real RHS systems for conducting online tests, neither. Thus, how to assess the effectiveness of pick-up location recommendations remains a challenge.

3 System Design

To address the MPLR problem, we propose a data-driven pick-up location recommendation system – *MPLRec*. In this section, we first present the system overview, and then elaborate the design details of each module.

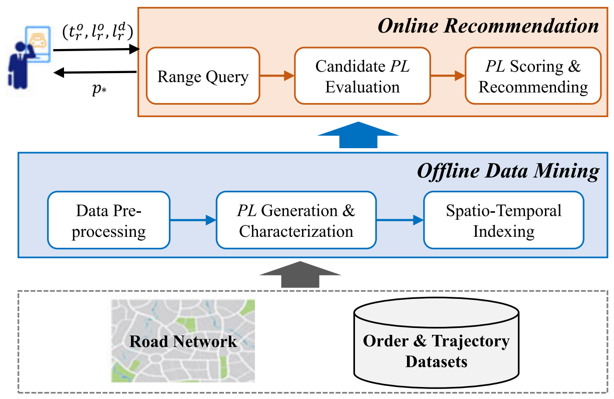


Fig. 5. The framework of *MPLRec*, where *PL* stands for *pick-up location*.

3.1 Overview

Fig. 5 illustrates the system architecture of *MPLRec*. At high-level, *MPLRec* takes the road network and historical order and vehicle trajectory data as the input to generate and characterize potential pick-up locations in an offline manner. Based on learned historical experiences, *MPLRec* searches and evaluates candidate pick-up locations for each recommendation request,[3] and recommends the one with the highest score, which is computed using a scoring function customized for the MPLR problem, for the rider.

MPLRec mainly consists of two major modules, i.e., *Offline Data Mining* and *Online Recommendation*. More specifically, the *Offline Data Mining* module aims to generate and characterize potential pick-up locations and index historical data for facilitating online recommendations. First of all, this module pre-processes all historical order and vehicle trajectory data. Then, *MPLRec* treats the midpoints of road segments as potential pick-up locations and characterizes them with features learned from historical data. In addition, *MPLRec* partitions the road network into grids using Geohash [3], which is a convenient geocoding method, and slices time of the day into slots. Therefore, the potential pick-up locations, orders and trajectories can be indexed from aspects of space and time. Considering human mobility patterns in a city, *MPLRec* further indexes historical orders via their moving directions.

The *Online Recommendation* module exploits the indexes to efficiently process online requests. Given a request $\mathbf{r}$ with ordering time t_r^o, ordering location ℓ_r^o, and destination ℓ_r^d, this module first conducts a range query around ℓ_r^o to retrieve a set of candidate pick-up locations. Then, it computes features of each candidate from historical data, including walking distance L_r^w, probability h to successfully hail a vehicle, and ride-hailing fare $\mathcal{F}_r$. With a scoring function $f(\cdot)$ that takes feature L_r^w, h, and $\mathcal{F}_r$ as the input, *MPLRec* computes a score for each candidate pick-up location, and recommends the one p_* with the highest score for request $\mathbf{r}$.

3.2 Offline Data Mining

3.2.1 Data Pre-Processing

We clean all historical data by filtering out erroneous data due to hardware failures and invalid orders canceled by the

2. We omit the subscript of orders when the context is clear.

3. We name an order, which is not submitted yet, as a *request*.

riders. For example, if the GPS locations or timestamps are invalid, we will discard these records. Then, we perform map matching on these trajectory data and meanwhile profile road speeds by exploiting map matching results.

We link the orders and their corresponding trajectory records through order IDs, and utilize the fast map matching (FMM) algorithm [44] to transform each vehicle trajectory to its actual travel route. During the map matching, noisy and erroneous GPS locations cannot match any road segment. Thus we will remove an order and its associated trajectory data if several consecutive GPS records, e.g., > 3, of this order's trajectory records are wrong.

After map matching, we can obtain the travel path between any two consecutive GPS locations of a trajectory. Note that the travel path could be on one road segment or a sequence of connected road segments. Thus, we can calculate a travel speed s, which is the ratio between road network distance of the two GPS locations and time difference between them, for the road segment(s). If a vehicle traveled multiple road segments within the time interval, we will assign travel speed s, along with the average timestamp of the two trajectory records, to all of these road segments. By scanning all trajectories and their travel routes, each road segment may be assigned with many time-stamped travel speeds.

Considering the data sparsity issue, we slice time of the day into a sequence of time slots with size of $15\,mins$. We thus have 96 time slots in total, i.e., $\mathbb{T} = \{\Delta_1, \Delta_2, \ldots, \Delta_{96}\}$. For a given road segment e_i, we classify travel speeds on e_i into time slots according to their timestamps, and calculate an average speed $s_{i,j}$ for each time slot Δ_j using speeds falling into Δ_j. In case a road segment e_i is not covered by any trajectory in time slot Δ_j, the speed $s_{i,j}$ will be temporally substituted by the average speed of this road segment in previous four time slots. Once ride-hailing vehicles have traveled on e_i later, we will update $s_{i,j}$ using the real data. Later, we will treat these travel speeds as roads' traffic condition indicators when recommending pick-up locations.

In fact, more complex time model can be adopted to represent urban traffic patterns, e.g., considering day of the week and time of the day [27], [28], while we consider time of the day for simplicity, and leave time modeling of urban traffic patterns as a future work.

3.2.2 Pick-Up Location Generation and Characterization

Previous works [9], [40], [47], [50], [51] generate pick-up locations by clustering historical pick-up locations, while the resultant clusters may be sparsely distributed and largely omit the riders' specific requirements. Considering the "visibility" requirement on pick-up locations, we propose to recommend midpoints of road segments as the meeting points for drivers and riders. On the one hand, pick-up locations along road segments can help drivers and riders easily discover each other. On the other hand, such pick-up locations are densely distributed, and we could finally seek one that will satisfy each rider's mobility demands.

For the road network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, we generate a set of pick-up locations $\mathbb{P} = \{p_1, p_2, \ldots, p_{|\mathcal{E}|}\}$, where p_i is the midpoint of road segment $e_i \in \mathcal{E}$. To facilitate the future recommendations, we characterize each pick-up location from the following aspects.

(1) *The shortest paths to other vertices in graph $\mathcal{G}$.* An important concern about each pick-up location is the driving distance to the destination, which will not only determine the travel time, but also affect the ride-hailing fare. Therefore, we pre-compute the shortest paths between the midpoints of any two road segments using A* algorithm [21], and cache these paths for future query. As drivers and riders usually prefer the shortest paths [8], while shortest path calculation is time-consuming, thus pre-computing and caching all the shortest paths can accelerate pick-up location recommendations. We cache these shortest paths in a hash map $\mathcal{M}$, where $\mathcal{M}(i, j)$ stores the path between p_i and p_j. As a result, the shortest path query takes $\mathcal{O}(1)$ time merely [37].

(2) *Ride-hailing hotness.* Another crucial concern is how long a rider needs to wait for available vehicles at a location. Previous works [40], [47], [50] generally build statistical or machine learning models for a specific location to estimate the waiting time from historical data, however, such methods suffer from data sparsity. Since we may have insufficient data at some places, and as a result the derived models have poor estimation accuracy.

Instead of building the waiting time estimators, we estimate the chance of successfully hailing a vehicle at each pick-up location. To this end, we will calculate a success probability of ride-hailing, which is referred as *ride-hailing hotness* in this paper. Considering human mobility patterns, the number of available ride-hailing vehicles at a location will change across time, we thus calculate a hotness $h_{i,j}$ in each time slot Δ_j for potential pick-up location p_i. We derive $h_{i,j}$ by mining historical order and trajectory data.

Definition 7 (Ride-hailing Hotness). *The ride-hailing hotness $h_{i,j}$ of pick-up location p_i in time slot Δ_j is defined as the ratio between number $n_{i,j}$ of distinct vehicles passing by road segment e_i within Δ_j and a threshold H, i.e., $h_{i,j} = \frac{n_{i,j}}{H}$.*

Threshold H is a pre-defined system parameter, and could be set according to the RHS's requirements. For example, we can set H as the maximum number of distinct vehicles observed in a road segment within a time slot. We have calculated the ride-hailing hotness of a peak time slot for each road segment in Chengdu city by using one week of historical data, and visualize the results in Fig. 6, where we see that the hotness varies greatly across different locations. Therefore, it is necessary to take the success probability of hailing a vehicle into account for recommending the pick-up locations.

3.2.3 Spatio-Temporal Indexing

Since we aim to recommend suitable pick-up locations by leveraging massive historical data, thus it is necessary to efficiently index potential pick-up locations, order and trajectory data, which will benefit the frequent data retrieval in online recommending. To this end, we divide the road network into partitions and index potential pick-up locations using Geohash [3]. In addition, we will index all historical orders and trajectories from aspects of space and time.

Grid Based Pick-Up Location Indexing. We partition the road network $\mathcal{G}$ into grids $\mathbb{G} = \{G_1, G_2, \ldots, G_m\}$, where m is the total number of grids, with Geohash, and index pick-up locations, along with their locating road segments, using these grids.

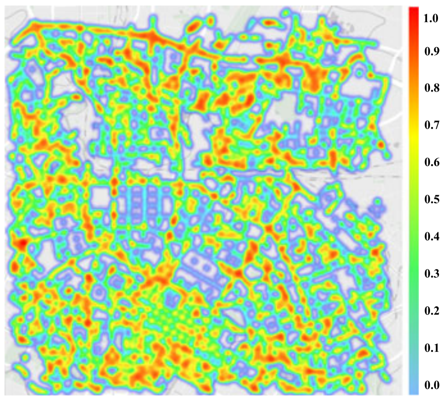


Fig. 6. The hotness distribution for Chengdu city's road network in a peak time slot, where the hotness of each road segment (i.e., pick-up location) is calculated with one week of historical data.

Geohash is a public domain geocode system that can encode a geographic location expressed by a pair of latitude and longitude into a short alphanumeric string [3]. By adopting base32 encoding, Geohash can be visualized as a division of the Earth into 32 grids, each of which is recursively divided into 32 grids until achieving the specified level [18]. It defines the grid level z as the division that results in hash codes of string length z, and each code covers a delimited area on the Earth. As a result, the child grids of a certain grid share the same prefix, i.e., the hash code of their parent. In particular, a $z = 6$ level hash code spans a grid of area about $1.22\,km \times 0.61\,km$, covering approximately $1\,km^2$ area, which is a suitable grid size for searching nearby pick-up locations. We thus choose $z = 6$ grid level to derive the grid set $\mathbb{G}$. Fig. 7 demonstrates some grids of $z = 6$ level on the Chengdu city's road network, where the nine grids share the same prefix, i.e., "*wm6n8*," and all locations within a grid, e.g., the middle one, own the same hash code, e.g., "*wm6n89*".

Given pick-up location p_i, we can derive its hash code through the Geohash function $geohash(p_i)$, and know its locating grid G_j. Therefore, we can easily index all potential pick-up locations using grids. Note that, we treat pick-up location p_i as a representative of its locating road segment e_i, thus road segments in $\mathcal{G}$ can also be indexed by grids $\mathbb{G}$ as well. Specifically, for each grid G_j, we index pick-up locations belonging to grid G_j using a list $G_j.L_p$, which is constructed according to their hash codes.

Ride-Hailing Order Indexing. We index historical orders from the aspects of space and time, which can greatly accelerate order querying later. On the one hand, we exploit time slot $\mathbb{T}$ to index orders according to their pick-up time. Specifically, for each order $\mathbf{r}$ with pick-up time t_r^p, we first assign it to time slot Δ_j where t_r^p falls into. Then for all orders generated within time slot Δ_j, we adopt B-tree structure to further index them based on their pick-up time. We denote ride-hailing orders belonging to Δ_j as $\mathcal{D}_{\Delta_j}$.

On the other hand, considering that people usually travel with varied directions within a city, we thus propose to index historical orders according to their travel directions as well. We treat true north as the zero degree of travel direction, and calculate the travel direction θ_r for each order $\mathbf{r}$ using its pick-up location ℓ_r^p and destination ℓ_r^d. We further split travel directions into interval Θ of size 5°, i.e., 72

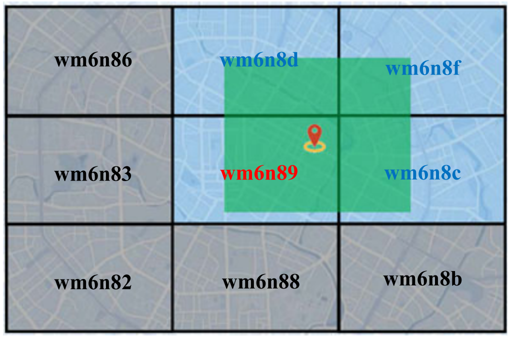


Fig. 7. Illustration of Geohash based road network partitioning, and its application for location based querying.

intervals in total, and assign all orders into direction intervals according to their travel directions. We denote ride-hailing orders belonging to direction interval Θ_j as $\mathcal{D}_{\Theta_j}$.

Trajectory Indexing. For trajectory $\mathcal{T}_r$ of order $\mathbf{r}$, we store and index its route $\mathcal{R}_r$. Due to the huge size of trajectory dataset, we directly adopt PostGIS [6], a spatial database extender for PostgreSQL database, to store all trajectories and routes. PostGIS uses the generic index structure (GIST) to index a route by expressing it with a string constituted by the ID sequence of road segments in $\mathcal{R}_r$ and a geometry. Specifically, GIST is a balanced, tree-structured access method that acts as a base template in which to implement arbitrary indexing schemes, e.g., R-tree. The geometry of a travel route is bounded with a minimum bounding rectangle [16], which is then used as the key to index this travel route. The geometry representation of orders' trajectories enables filtering-verification and can largely optimize the query efficiency.

3.3 Online Recommendation

Once *MPLRec* receives a request $\mathbf{r}$, which specifies the ordering time t_r^o, ordering location ℓ_r^o, and destination ℓ_r^d, the system will first perform a range query around ℓ_r^o to obtain a set of candidate pick-up locations, and then compute features of walking distance, hotness, and estimated fare for each candidate. Finally, *MPLRec* recommends the candidate pick-up location with the highest score calculated from a scoring function to the rider.

3.3.1 *Range Query*

Given a request $\mathbf{r}$, *MPLRec* searches for nearby pick-up locations to build a candidate set $\mathbb{C}_r$ with searching range β. According to the recent studies [40], [51], we set $\beta = 500\,m$, since people generally accept a maximum walking distance as $500\,m$. *MPLRec* exploits Geohash grids to construct $\mathbb{C}_r$ with following three steps.

① *MPLRec* computes the hash code, i.e., $geohash(\ell_r^o)$, for $\mathbf{r}$ using its ordering location ℓ_r^o, and thus obtains the grid G_r where ℓ_r^o locates. Based on the encoding rules, we can also retrieve the neighboring eight girds around G_r using grids' hash codes. The nine girds together form the candidate grid set $\mathcal{C}_r$.

② We construct a square area, which is centered at ordering location ℓ_r^o with side length as $2 \times \beta$. According to the boundary of grids and the size of square area, we can narrow down set $\mathcal{C}_r$ by only keeping the grids that intersect with the square area.

③ Finally, we retain all potential pick-up locations in the grids of $\mathcal{C}_r$ into set $\mathbb{C}_r$ as the candidates for further evaluation, i.e.,

$$\mathbb{C}_r = \{\cup_{G_j \in \mathcal{C}_r} G_j.L_p\}. \tag{1}$$

Fig. 7 illustrates above steps to narrow down the searching space for candidate pick-up locations of a request **r**, where the rider locates in the middle grid with hash code "*wm6n89*" and the potential pick-up locations in blue grids are kept in the candidate set $\mathbb{C}_r$ for further evaluation.

3.3.2 Candidate Pick-Up Location Evaluation

Given the request **r** and historical data, we compute the features, which are riders' main concerns about ride-hailing, of each candidate pick-up location, say $p_i \in \mathbb{C}_r$, from the following aspects:

(1) *Walking distance.* A walking route from ordering location ℓ_r^o to candidate pick-up location p_i can be planned using the A* algorithm [21], and thus we can calculate the walking distance $L^w_{r,p_i} = dist(\ell^o_r, p_i)$. Considering the maximum acceptable walking distance, if L^w_{r,p_i} is larger than β, we simply remove p_i out of $\mathbb{C}_r$ and continue to evaluate next candidate. Otherwise, p_i is considered as a valid candidate for further evaluation.

(2) *Ride-hailing hotness.* We determine the time slot Δ_j where ordering time t^o_r falls into, and try to compute the hotness $h_{i,j}$ of candidate pick-up location p_i within Δ_j. In particular, we propose to calculate a *customized ride-hailing hotness* for request **r**, since *Definition 7* counts all vehicles passing by road segment e_i of pick-up location p_i, resulting in a general hotness. Imagine that if all historical orders passing by e_i travel to directions that are different from **r**'s moving direction, the derived hotness seems meaningless for **r** as historical orders differ from the current case. We thus count the number $\hat{n}_{i,j}$ of distinct vehicles, which have passed by e_i within Δ_j and meanwhile share similar travel directions as **r**, e.g., direction difference within a threshold α.

We derive $\hat{n}_{i,j}$ by mining historical orders from three aspects of time, location, and moving direction. First, we calculate moving direction θ_r of request **r** using candidate pick-up location p_i and destination ℓ^d_r. Then, we obtain candidate orders O_r by exploiting time slot and moving direction based order indexes, i.e.,

$$O_r = \mathcal{D}_{\Delta_j} \cap \{\cup_{b \in \mathcal{B}} \{\mathcal{D}_{\Theta_b}\}\}, \tag{2}$$

where set $\mathcal{B}$ contains the direction intervals that are included by or intersected with moving direction range $[\theta_r - \alpha, \theta_r + \alpha]$.

Second, candidate order set O_r may still include many irrelevant orders, we thus make use of the geometry of trajectory index to further filter out orders. We construct one circle, which is centered at p_i with radius $\beta = 500\,m$, and leverage the *ST_intersects* function of PostGIS [6] to retain the orders, whose routes' geometries intersect with the circle. These retained orders may pass through the nearby area of p_i.

Finally, we scan the route string of each retained order and only keep these orders in O_r, whose route strings contain the ID i of rode segment e_i. After above operations, $\hat{n}_{i,j}$ is set as the number of orders in O_r, i.e., $\hat{n}_{i,j} = |O_r|$. We thus calculate the customized ride-hailing hotness as $h_{i,j} = \frac{\hat{n}_{i,j}}{H}$. Although there are numerous historical orders to be examined, we find that spatio-temporal index based filtering-verification is quite efficient, which can be completed around $160\,ms$ (See Section 5.2).

(3) *Estimated ride-hailing fare.* As the fare $\mathcal{F}_{r,p_i}$ is highly correlated with the driving distance L^d_{r,p_i} and driving time T^d_{r,p_i}, which starts the trip at candidate pick-up location p_i. Thus, we will estimate L^d_{r,p_i} and t^d_{r,p_i}, and then estimate the fare $\mathcal{F}_{r,p_i}$.

- *Driving distance.* We first map destination ℓ^d_r to the closest road segment e_{d_r}, and thus derive its midpoint p_{d_r}. Then, we retrieve the shortest path $\mathcal{R}_{i,d_r} = \mathcal{M}(i, d_r)$, which links the midpoints of road segment e_i and e_{d_r}, from the cache. Therefore, we calculate the driving distance L^d_{r,p_i} for request **r** as $L^d_{r,p_i} = dist(\mathcal{R}_{i,d_r})$.
- *Driving time.* Given the trip origin p_i, destination ℓ^d_r, and the travel route $\mathcal{R}_{i,d_r}$, we aim to estimate the travel time by considering general road traffic conditions. Although there exist some estimated travel arrival (ETA) methods [39], [45], which exploit deep learning models to mine massive historical travel data for inferring ETA between any two locations, we instead prefer a simple yet effective method to avoid the cumbersome model training and predictions. Given the route $\mathcal{R}_{i,d_r}$ and road traffic conditions that are dynamically updated from historical trajectory data, we can estimate the travel time of route $\mathcal{R}_{i,d_r}$ as

$$T^d_{r,p_i} = \sum_{e_z \in \mathcal{R}_{i,d_r}} \frac{dist(e_z)}{s_{z,j}}, \tag{3}$$

where j is the index of time slot ordering time t^o_r falls into, and $dist(e_z)$ returns the length of road segment e_z or the part covered by route $\mathcal{R}_{i,d_r}$. For simplicity, we use road travel speeds of time slot Δ_j for driving time estimation.

In our implementation, we directly adopt Didi's fare formulation [1] to estimate the fare $\mathcal{F}_{r,p_i}$[4] for request **r** if the rider chooses p_i as the pick-up location. Specifically, Didi favors more about the driving distance than the driving time, and calculates ride-hailing fares with the following equation:

$$\mathcal{F}_{r,p_i} = L^d_{r,p_i} \times 1.10 + T^d_{r,p_i} \times 0.20. \tag{4}$$

3.3.3 Pick-Up Location Scoring and Recommending

Once we have calculated the features for each candidate pick-up location, we need a mechanism to comprehensively evaluate all candidates and recommend the best one to the rider. Considering the relationship among walking distance L^w_{r,p_i}, ride-hailing hotness $h_{i,j}$, and fare $\mathcal{F}_{r,p_i}$, we define the scoring function $f(\cdot)$ to investigate each candidate pick-up location p_i as follows:

$$f(p_i) = \varphi \cdot h_{i,j} + (1-\varphi) \cdot \frac{10}{L^w_{r,p_i} \times 1.25 + \mathcal{F}_{r,p_i}}, \tag{5}$$

4. We simplify the formulation by omitting the flag-down price and these time-dependent pricing factors for fair and direct comparisons later.

where φ is a system parameter and satisfies $0 \leq \varphi \leq 1$. Since people usually prefer to walk less, we thus assign a larger weight (i.e., 1.25) for walking distance L^w_{r,p_i} than driving distance L^d_{r,p_i} (i.e., 1.10 in Eq. (4)). Specifically, the first term in Eq. (5) measures the probability of successfully hailing a vehicle at location p_i, while the second term in Eq. (5) transforms the costs of time and travel distances into ride-hailing fare. Therefore, we introduce φ to balance the impacts of hotness and distance-related factors.

We calculate a score for each candidate pick-up location with Eq. (5), and return $p_* = \arg\max_{p_i \in \mathbb{C}_r} f(p_i)$, as the recommended pick-up location for request $\mathbf{r}$.

4 Assessment Methodology

To validate and assess *MPLRec* under practical settings, we design and implement an RHS simulator, which can generate realistic ride-hailing orders by mining real-world order and trajectory data and manage a fleet of vehicles to serve these orders.

4.1 Data Generation

For an RHS system, ride-hailing orders and vehicle statuses are the two most important input data sources. To simulate practical RHS transactions, the simulator is designed to learn the distribution of real-world orders on the road network over time of the day and the mobility patterns of ride-hailing vehicles. For a simulation, we will select a specific date and leverage the order and trajectory data of that date to generate orders and initialize vehicles statuses.

Generating Ride-Hailing Orders. We take orders generated at road segment $e_i \in \mathcal{E}$ within time slot $\Delta_j \in \mathbb{T}$ as an example to illustrate how we generate realistic ride-hailing orders by mining historical data. Similar as previous works [32], [35], [47], [50], we assume the arrivals of orders on road segment e_i approximately follow a Poisson distribution (with parameter λ^i_j) during time slot Δ_j. We count the number N^i_j of orders that originated from e_i within Δ_j in the order dataset, and compute the parameter λ^i_j as

$$\lambda^i_j = \frac{N^i_j}{|\Delta_j|}, \tag{6}$$

where $|\Delta_j|$ indicates the size of time slots.

For each generated ride-haling order $\mathbf{r}$, we enrich its details as follows. The Poisson distribution assumes $\mathbf{r}$ is generated at p_i, i.e., the midpoint of road segment e_i, with a random timestamp within Δ_j, which is the ordering time t^o_r of $\mathbf{r}$. Other than directly adopting p_i, we generate the ordering location ℓ^o_r as a random point on a road segment within a circle area, which is centered at p_i with radius $\beta = 500\,m$. To generate the destination for $\mathbf{r}$, we mine historical data to derive transition patterns of trips originated on road segment e_i. Specifically, we construct a vector $\mathbf{B}^j_i$, whose element b^j_{ik} $(k = 1, 2, \ldots, |\mathcal{E}|)$ indicates the transition probability of riders, who had taken a ride-hailing vehicle on road segment e_i and traveled to road segment e_k. Probability b^j_{ik} is calculated as

$$b^j_{ik} = \frac{N^{ik}_j}{N^i_j}, \tag{7}$$

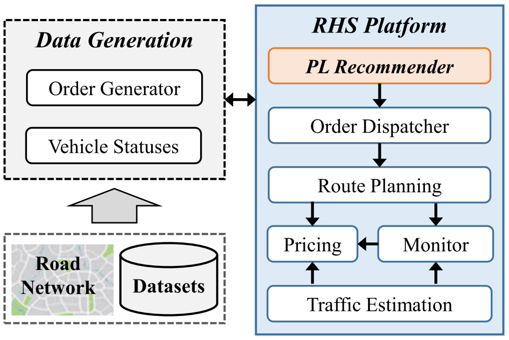


Fig. 8. Framework of the RHS simulator for investigating pick-up location recommendation methods, where *PL* stands for pick-up location.

where N^{ik}_j is the number of historical orders that originated from e_i and destined to e_k during Δ_j. Thus, we generate destination ℓ^d_r for order $\mathbf{r}$ based on the transition probability vector $\mathbf{B}^j_i$, and set the midpoint p_k of selected road segment e_k as the destination ℓ^d_r.

Note that historical order dataset only contains orders that have been successfully served, while those unsatisfied orders are not recorded. To consider this situation, we introduce parameter η for the simulator, which will generate η times the number of orders observed in the dataset. For example, the simulator will generate $\eta \times N^i_j$ orders for road segment e_i within time slot Δ_j.

Updating Vehicle Statuses. To simulate the practical scenario, we initialize ride-hailing vehicles by referring to the trajectory data of a specific experiment date. Specifically, we select a date and a particular time t_0 of that date, and treat vehicles appearing in the trajectory records within a time range $[t_0 - 5,\ t_0 + 5]$ $(mins)$ as the testing vehicles. The last location of each vehicle appearing in the trajectory records is set as its initial location. In addition, we assume all testing vehicles are vacant at the beginning of a simulation. Each vehicle is managed by the RHS's centralized platform, and will serve an assigned order following the scheduled route, or cruise to a nearby road segment e_i, which is within $1\,km$ and has a high probability of meeting potential riders, e.g., with a large number N^i_j of observed orders within time slot Δ_j. The cruising strategy simulates real-world taxi drivers, who will drive to hot spots for seeking future passengers [12], [34], [47], [48].

4.2 Workflow of Simulator

Fig. 8 presents the framework of RHS simulator, which takes road network and historical order/trajectory datasets as the input to generate synthetic data and simulates an RHS system's operations. The RHS simulator consists of two modules, i.e., *Data Generation* and *RHS Platform*. Given an experiment date and start time t_0, the *Data Generation* module initializes the status of testing vehicles, and continuously generates realistic orders, as explained in above subsection. On the other hand, the *RHS Platform* module recommends suitable pick-up location for each request, and serves this request once it is submitted by the rider. Next, we explain the simulation workflow for each generated request $\mathbf{r}$ as follows:

① Once a ride is needed, a rider will open the RHS App, which can access the rider's current location ℓ^o_r, and inputs the destination ℓ^d_r. After knowing ℓ^o_r and ℓ^d_r, the *PL Recommender* component of RHS platform tries to recommend a suitable pick-up location by executing some pick-up location recommendation methods, e.g., our *MPLRec* or any other alternative methods like recommending nearby popular pick-up locations [9], [35], [51]. The recommended pick-up location then is displayed on the RHS App, while the rider confirms this location ℓ_r^p as the trip origin and submits an order $\mathbf{r} = \{i, t_r^o, \ell_r^o, \ell_r^p, \ell_r^d\}$ to the RHS platform, which assigns this order with an ID i.

② After receiving order **r**, the *Order Dispatcher* component will query nearby vacant vehicles, and assigns this order to the nearest available vehicle (advanced order assignment algorithms can be adopted to maximize certain utility [43]). For each order-vehicle match, the *Route Planning* component will first plan a shortest route for the assigned vehicle to the pick-up location ℓ_r^p, and then calculates the shortest path $\mathcal{R}_r$ to order **r**'s destination ℓ_r^d once the rider gets in the vehicle. These shortest paths can be pre-computed and cached to improve the simulation efficiency.

③ Similar as real-world RHS systems, the *Monitor* component continuously updates and records vehicle statuses with an interval of $10\,s$. These vehicle statuses, e.g., instant location, are mainly influenced by road traffic conditions. For simplicity, we use road travel speeds calculated from trajectory records of the experiment date to simulate the real-time road traffic conditions. Therefore, the *Traffic Estimation* component stores the time-slot indexed travel speeds for all road segments in $\mathcal{G}$, and the simulator can easily get a travel speed $s_{k,j}$ for a road segment e_k and time slot Δ_j current simulation time falls into. As a result, the *Monitor* component can update the instant locations of a vehicle according to its travel route and road traffic conditions.

④ Once the vehicle arrives at order **r**'s destination ℓ_r^d, the *Pricing* model will calculate the ride-hailing fare $\mathcal{F}_r$ for **r**. In general, the fare $\mathcal{F}_r$ is computed based on the driving distance and time. Although some complicated pricing models [8], [29] can be used, we adopt Didi's pricing model, i.e., Eq. (4), for calculating orders' fares in the RHS simulator. In addition, once the order **r** is completed, the *Monitor* component will supplement other details to the order **r**'s record, e.g., the time t_r^d arriving at destination ℓ_r^d. These order details can be used for computing some performance metrics, e.g., driving distance and driving time.

5 Performance Evaluation

In this section, we evaluate the performance of *MPLRec* by conducting data-driven experiments and simulator based simulations.

5.1 Experimental Setup

We make use of the Didi's datasets, which have been introduced in Section 2.2, and Chengdu city's road network, downloaded from OpenStreetMap [5], for the performance evaluations. We keep the data of last week for testing, while the remaining data are used for calculating ride-hailing hotness of potential pick-up locations and profiling road segments' speeds.

Baselines. We compare *MPLRec* with the following methods:

- *GT* is the real data, which serves as the benchmark of a practical RHS systems' strategy for recommending pick-up locations. We can derive performance metrics from the order details and trajectory data in the datasets.
- *Closest* will recommend the closest pick-up location, i.e., midpoint of the road segment closest to the rider's current location. In general, when people need a ride at unfamiliar places, they usually choose the road segment that is the closest to themselves as the location to wait for taxis.
- *Prob* calculates a probability of successfully hailing a vehicle, like our ride-hailing hotness, for each potential pick-up location by mining historical trajectory data. Given a rider's ordering location, this method queries nearby pick-up locations, and recommends the one with the highest probability [40], [50]. Typically, this method allows a rider to get a vehicle quickly, and thus reduces waiting time.
- *Cluster* classifies historical pick-up locations, which are within a searching range of the rider's ordering location, into clusters by exploiting some clustering algorithms, e.g., *K-means* [17], and recommends the one that satisfies some conditions [35], [51], e.g., closest to the rider's location.
- *Shortest* calculates the overall travel distance, including both walking distance and driving distance, for each candidate pick-up location, and recommends the one with the smallest overall travel distance. Based on the road network graph $\mathcal{G}$, this method utilizes the A* algorithm [21] to plan a route that traverses the ordering location, candidate pick-up location and the destination.

Performance Metrics. For data-driven experiments, we adopt the metrics of *driving distance*, *driving time*, *walking distance*, *hotness*, *ride-hailing fare*, and *average score* calculated using Eq. (5) to evaluate all methods. For simulator-based simulations, we utilize two additional metrics of *waiting time* and *order completion rate* to compare different methods. Specifically, *waiting time* is the time interval between ordering time t_r^o and pick-up time t_r^p for each order **r**. The *order completion rate* is the ratio between the number of served orders and the number of totally received orders. In general, RHS systems try to serve all ride-hailing orders, while some orders may not be served due to insufficient vehicles.

Implementation. We implement *MPLRec* and the other baseline methods in Java. We set the searching range β as $500\,m$ for all methods. In addition, we set the number of clusters as $K = 9$ for the *Cluster* method. We configure *MPLRec* as follows: the size of time slot is $15\,mins$; road network $\mathcal{G}$ of Chengdu city is partitioned into 130 grids using Geohash; the similar direction threshold $\alpha = 30^\circ$. To calculate the relative hotness for a request, we set threshold H as the maximum number of vehicles passing by all candidate road segments. By default, we set $\varphi = 0.70$ to balance the impacts of hotness and distance-related factors in Eq. (5). For the RHS simulator, we generate $\eta = 1.50$ times of the number of observed orders for each road segment.

The evaluation experiments are all conducted on a server with CPU of Intel(R) Core(TM) i7-10700 K 3.80 GHz and

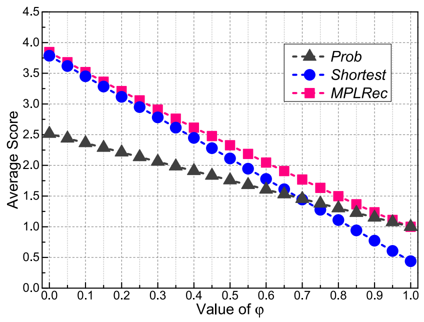


Fig. 9. Impact of parameter φ on recommendation scores of *Prob*, *Shortest*, and *MPLRec*.

memory of 16 GB. We report the average of 5 runs as the experiment results.

5.2 Data-Driven Experiments

Impact of Parameter φ. First, we explore the impact of parameter φ on the scoring function $f(\cdot)$, which balances the relationship between hotness and distance-related factors. In principle, a larger φ favors candidate pick-up locations with higher relative hotness, i.e., working similarly as the *Prob* method, while a smaller φ tends to recommend candidates with less driving distance and time, i.e., degrading into the *Shortest* method. We thus use Eq. (5) to evaluate the pick-up location recommendations of *Prob*, *Shortest*, and our *MPLRec*, and present the average scores of different methods under various φ settings, as shown in Fig. 9. In general, the average scores of the three methods decrease when we increase φ. When $\varphi = 0.70$, *Prob* and *Shortest* achieve the similar score, which implies a good trade-off between hotness and distance-related factors to select the suitable pick-up locations. Therefore, we set $\varphi = 0.70$ for the following experiments and simulations.

Comparisons on Different Metrics. We summarize the experiment results of various methods on different metrics into two day types, i.e., *workday* and *weekend*, and compare them in Table 2. In general, each method derives relatively larger driving distances L^d and driving time T^d on workday than weekend, possibly due to busier urban traffic on workdays. Since ride-hailing fare $\mathcal{F}$ is mainly determined by driving distance and driving time, the fares of orders on workdays are also a bit more than orders of weekend.

From Table 2, we find that *Closest* always recommends these pick-up locations with the fewest walking distance L^w, while *Prob* performs the best on the metric of hotness h. In addition, *Shortest* can recommend the locations with the fewest driving distances and driving time, and as a result, it can save more money for the riders. In contrary, *Cluster* achieves the moderate performance on these metrics. Although our *MPLRec* does not perform outstandingly on any individual metric, it can achieve the most comprehensive and balanced performance on these metrics, by wining the most times of second best results. For example, *Shortest* indeed recommends some pick-up locations with the fewest driving distances and time, while those recommended locations are less popular, as few ride-hailing vehicles have passed by them in the historical trajectories. Fig. 10 further compares the average scores of pick-up locations recommended by different methods. We see that *MPLRec* has the highest scores on both workdays and weekend. It also implies that our scoring function $f(\cdot)$ in Eq. (5) can investigate the comprehensive quality of candidate pick-up locations.

TABLE 2
Comparisons on Various Performance Metrics (Including *walking distance* L^w, *driving distance* L^d, *driving time* T^d, *hotness* h, and *ride-hailing fare* $\mathcal{F}$) Among Different Methods

Day Type	Method	L^w (m)	L^d (m)	T^d (s)	h	$\mathcal{F}$
Workday	*GT*	/	3854.69	890.94	0.48	7.21
	Closest	**83.64**	3678.49	706.88	0.44	6.40
	Prob	335.45	3730.49	705.82	**1.00**	6.46
	Cluster	359.33	3714.54	770.21	0.56	6.65
	Shortest	237.24	**3353.46**	**660.88**	0.39	**5.89**
	MPLRec	325.05	3631.82	687.77	0.96	6.29
Weekend	*GT*	/	3646.54	863.97	0.56	6.89
	Closest	**111.02**	3544.76	674.27	0.53	6.15
	Prob	333.32	3602.64	674.57	**1.00**	6.21
	Cluster	358.25	3536.86	788.17	0.57	6.52
	Shortest	238.97	**3256.43**	**633.78**	0.50	**5.69**
	MPLRec	319.07	3485.89	654.76	0.95	6.02

The best result for each metric is marked in bold, while the second is marked with underline.

We compare the performance of each method with the ground-truth trajectory, i.e., *GT*, and calculate the proportion of orders that have better performance than *GT* on each individual metric, e.g., fewer driving distance/time or higher hotness/score than *GT*. These statistics in Fig. 11 are in accordance with the results in Table 2, e.g., *Shortest* outperforms *GT* on distance-related factors while *Prob* beats *GT* on the hotness. Different from these methods, *MPLRec* has a well-balanced performance on these metrics.

Recommendation Efficiency. To recommend suitable pick-up locations, *MPLRec* involves three key operations: (1) *range query* that searches candidate pick-up locations within a range β; (2) *trajectory query* that retrieves historical trajectories passing by each candidate pick-up location to calculate its relative hotness; (3) *route planning* that computes a walking route and a driving route for each candidate pick-up location. Noting that the driving route between any two road segments has been pre-computed and cached, while the walking route needs to be calculated just-in-time. We summarize the average computation time for above operations in Table 3. It shows that these key operations can be efficiently handled by *MPLRec* within only a few hundreds

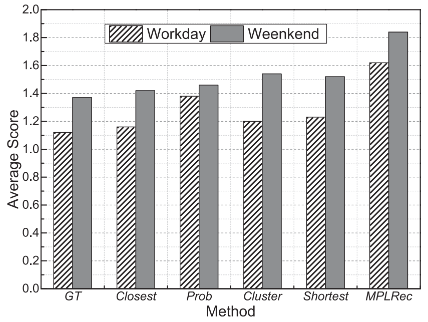


Fig. 10. Comparisons on the recommendation scores derived by different methods.

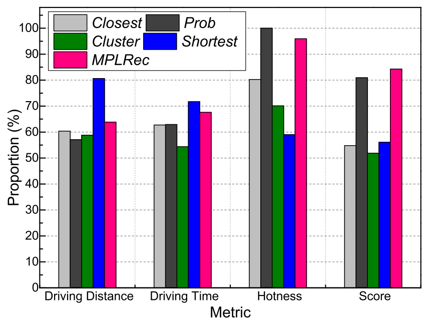


Fig. 11. Proportion of orders that outperform *GT* on each individual metric.

TABLE 3
Computation Time (Unit: ms) for Each Key Operation in *MPLRec*'s Pick-Up Location Recommendations

Day Type	*Range Query*	*Trajectory Query*	*Route Planning*	Total
Workday	83.53	164.81	210.50	**458.84**
Weekend	92.02	160.80	129.43	**382.25**

of milliseconds, thanks to the spatio-temporal indexes that accelerate the retrieval and processing of potential pick-up locations, historical orders and trajectories. On average, *MPLRec* can process each request with only $458.84\,ms$ and $382.25\,ms$ on workdays and weekend, respectively.

Case Study. Fig. 12 shows a concrete example, where *MPLRec* recommends a better pick-up location than the original one and thus significantly reduces the driving distance and time. Compared to actual order route (i.e., the black line), the rider only needs to walk a distance of $207\,m$ to the recommended location for hailing the vehicle, and can save driving distance and time by $1573.82\,m$ ($\sim 28.17\%$) and $484.74\,s$ ($\sim 74.62\%$), respectively.

5.3 Simulations

Since human dynamics are not consistent across time and demonstrate different mobility patterns [28], [41]. We thus chose two periods of November 27 (*Sunday*) and November 28 (*Monday*), 2016, as experiment dates and time to evaluate different methods under various mobility patterns. Specifically, we consider workday and weekend, and for each day type we further select time periods to represent the *non-peak hours* (e.g., 5:30AM-7:30AM) and *peak hours* (e.g., 7:30AM-9:30AM). Therefore, we will consider varied scenarios of four types, i.e., *workday/non-peak hours* (*W-NP*), *workday/peak hours* (*W-P*), *weekend/non-peak hours* (*NW-NP*), and *weekend/peak hours* (*NW-P*). For each scenario, we utilize real order and trajectory data of corresponding time periods to generate synthetic data and initialize vehicle statuses for the simulations.

Effectiveness of Order Generator. Fig. 13 has visualized the distributions of real orders and generated orders on the peak hours of a typical workday, where we see that generated orders share a similar heat distribution as the real orders on the vast majority of locations. It implies that our simulator can accurately learn riders' mobility patterns from historical order and trajectory data and generate realistic orders for effective simulations.

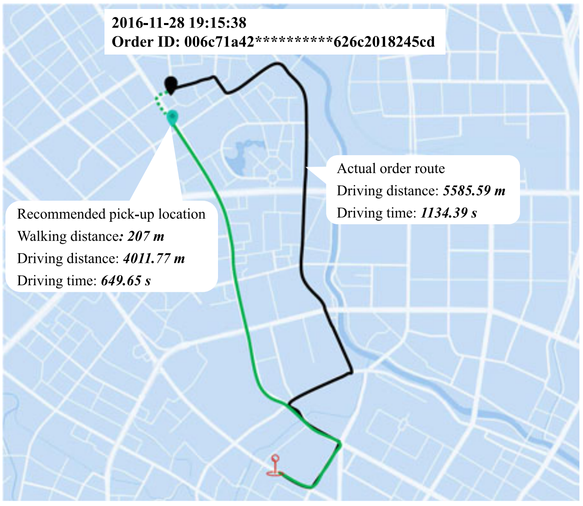

2016-11-28 19:15:38
Order ID: 006c71a42**********626c2018245cd
Recommended pick-up location
Walking distance: 207 m
Driving distance: 4011.77 m
Driving time: 649.65 s
Actual order route
Driving distance: 5585.59 m
Driving time: 1134.39 s

Fig. 12. A concrete example of pick-up location recommendation that performs better than the actual order route on driving distance and time.

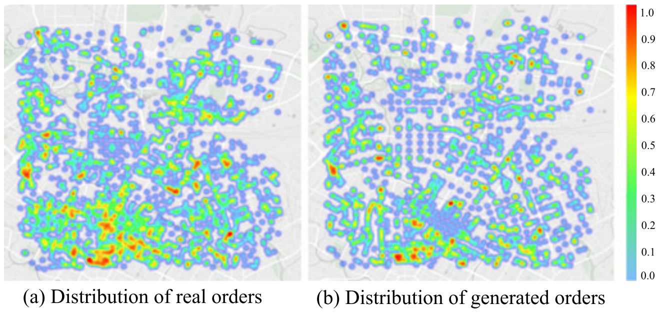

1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
(a) Distribution of real orders
(b) Distribution of generated orders

Fig. 13. Comparison on the distributions of real orders and generated orders on the peak hours of a typical workday.

Performance Comparisons. We compare the performances of different methods using the RHS simulator and summarize the average results in Table 4. Similar to the results in Table 2, the five methods have quite similar performances and relative rankings on the metrics of *walking distance*, *driving distance*, *driving time*, and *hotness* on both workday and weekend. Different from Table 2, we consider the metric of *waiting time* in the simulations, and we find that *Cluster* performs quite well with relatively fewer waiting time. *MPLRec* performs well on this metric as well, with close results as the best one. Table 4 demonstrates that *MPLRec* achieves the most comprehensive performance with pretty good results across all metrics, and thus has the highest *score* among the five methods.

We further compare *order completion rates* of different methods under the four scenarios and show the results in Fig. 14. In general, we find that there are more ride-hailing vehicles on peak hours than non-peak hours across all simulations. Except *NW-NP* scenario that has only about 180 vehicles, we usually have around 2500 vehicles for the other scenarios. We thus find the order completion rates are quite low for all methods in the *NW-NP* scenario. This is because there are insufficient vehicles for serving all orders. In contrary, all methods can achieve almost 100% order completion rates in the *W-NP* scenario. Among the five methods, *Shortest* can achieve the highest order completion rates in all scenarios. Such a greedy method could finish each order

TABLE 4
Simulation Result Comparisons on Various Performance Metrics (Including *walking distance* L^w, *waiting time* T^w, *driving distance* L^d, *driving time* T^d, *hotness* h, and *score*) Among Different Methods

Method	Workday						Weekend					
	L^w (m)	T^w (s)	L^d (m)	T^d (s)	h	*score*	L^w (m)	T^w (s)	L^d (m)	T^d (s)	h	*score*
Closest	**93.91**	325.61	3991.91	773.37	0.32	0.78	**112.58**	353.61	4082.38	791.56	0.42	0.87
Prob	331.69	247.14	3982.74	769.41	**1.00**	$\underline{1.25}$	332.05	291.27	4053.55	787.66	**1.00**	$\underline{1.25}$
Cluster	424.81	**221.97**	3989.96	853.50	0.52	0.83	405.85	$\underline{268.77}$	4029.54	922.61	0.87	1.13
Shortest	$\underline{234.38}$	264.24	**3683.10**	**727.08**	0.39	0.90	$\underline{247.30}$	**266.67**	**3752.37**	**745.23**	0.46	0.96
MPLRec	325.69	$\underline{246.93}$	$\underline{3928.55}$	$\underline{758.81}$	$\underline{0.98}$	**1.31**	322.15	278.60	$\underline{3984.85}$	$\underline{774.93}$	$\underline{0.98}$	**1.33**

The best result for each metric is marked in bold, while the second is marked with underline.

with the fewest driving time (as shown in Table 4) and thus can serve more orders given the same number of vehicles. *MPLRec* has the moderate order completion rates, which are actually quite close to *Shortest*'s results. The results in Table 4 and Fig. 14 demonstrate that our method can recommend suitable pick-up locations for the riders, while still assuring the efficiency of RHS systems.

6 Related Work

Recommendations in Transportation. Numerous research efforts have been made to provide recommendations for drivers or riders to improve transportation services in urban cities. These works can be classified into the following two categories.

(1) *Recommendations for drivers.* In the past decades, various methods have been proposed to help drivers enhance their services. By analyzing massive taxi GPS trajectory data, traditional methods either recommend hot spots for drivers to find potential passengers [10], [36], [40], [46], [47], or plan the cruising routes for drivers to maximize the probability of "hunting" future passengers [11], [12], [13], [34], [48]. As an example of the former works, *TaxiRec* [40] segments a road network into road clusters and proposes a ranking-based extreme learning machine model to recommend the candidate road clusters for taxi drivers to seek passengers. As a representative of the latter works, Guo et al. [13] recommend the right road segment to drivers at every intersection by employing a force-directed approach that considers rich features extracted from multi-source urban data. In addition, Guo et al. [14] also explore the relationship between driver revenue and factors relevant to seeking strategies, which could help drivers improve their revenues in the ride-on-demand services. Recent works [19], [20] tend to improve traditional methods by leveraging deep learning models to learn passenger-hunting experiences from historical data.

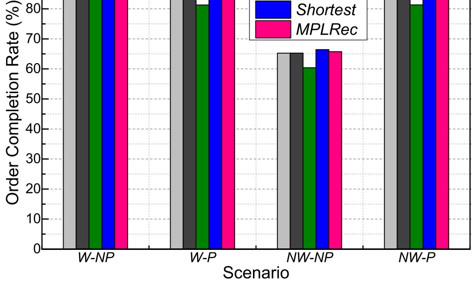


Fig. 14. Comparisons on *order completion rates* of different methods.

With the growing popularity of ride-hailing service (RHS) that can be aware of both vehicle statuses and online orders, increasing research interests [23], [26], [41] turn to proactively dispatching vehicles to certain regions, where vehicle supplies are insufficient for travel demands. Compared to conventional ad-hoc recommendations for each individual driver, the vehicle dispatching methods can balance supply-demand across different areas and maximize the efficiency of RHS systems [23].

Different from above works, we aim to recommend suitable pick-up locations for riders in the RHS scenario. While it is worthy noting that our pick-up location recommendations can also benefit drivers and RHS systems for improving the docking efficiency.

(2) *Recommendations for riders.* Research efforts of the other category are devoted to improve users' travel experiences in transportation systems. In addition to locating potential passengers, taxi trajectory data can also implicitly help riders know where to find available taxis [42], [47], [50]. For example, Zheng et al. present a waiting time prediction method that recommends some places for passengers to quickly find vacant taxis [50]. In addition, Liu et al. implement *Hydra* [24], which provides personalized and context-aware multimodal transportation routes according to riders' preference and situational context. Besides, there exist not a few works [22], [49] that aim to recommend next destination for users given their personalized travel patterns.

In the literature, however, only a few works [9], [35], [51] focus on recommending pick-up locations for riders in the RHS scenario. Specifically, Chen et al. recommend a set of locations as the alternatives of user's common choice to defend location inference attacks [9]. The other works primarily cluster historical pick-up locations within a searching range and recommend the clusters according to estimated waiting time [35] or certain traffic factors (e.g., lane setting of the roads and identification difficulty) [51]. The technique details of [35], however, are quite unclear. Our work differs from these works by considering the multiobjective pick-up location recommendation (MPLR) problem, which aims to recommend suitable pick-up locations by satisfying riders' multiple mobility demands. Furthermore, we propose an assessment methodology for the MPLR problem by generating realistic ride-hailing orders and vehicle statuses, which will inspire and benefit future studies on the pick-up location recommendations for RHS.

Ride-Hailing Service. RHS has already become an important transportation mode in our daily life, and various research works have been conducted to improve the performance of RHS systems from aspects of ride-hailing demand prediction [38], order-vehicle matching [25], [43], vehicle dispatching [23], [41], pricing model [8], [15], [29], safety and user privacy [30], [33], *etc*. In this paper, we consider a different problem, i.e., MPLR problem, which can improve RHS's efficiency and is compatible with those works.

7 Conclusion

In this paper, we study the multiobjective pick-up location recommendation problem and present *MPLRec*. Different from previous works, *MPLRec* explicitly takes rider's specific mobility demands, e.g., destination, into account, and exploits the experiences mined from historical order and trajectory data to recommend suitable pick-up locations, which can satisfy the riders' various requirements in the RHS scenario. Furthermore, we design and implement an RHS simulator to generate realistic ride-hailing orders and vehicle statuses for evaluating pick-up location recommendation methods. Extensive experiments and simulations have demonstrated the effectiveness and efficiency of *MPLRec*.

References

[1] Didi. Accessed: Aug. 2022. [Online]. Available: https://www.didiglobal.com/

[2] GAIA Didi initiative. Accessed: Aug. 2022. [Online]. Available: https://outreach.didichuxing.com/research/opendata/

[3] Geohash. Accessed: Aug. 2022. [Online]. Available: http://geohash.org/site/tips.html

[4] Market-Global Revenue. Accessed: Aug. 2022. [Online]. Available: https://www.alliedmarketresearch.com/ride-hailing-service-market

[5] OpenStreetMap. Accessed: Aug. 2022. [Online]. Available: https://www.openstreetmap.org/

[6] PostGIS. Accessed: Aug. 2022. [Online]. Available: https://postgis.net/

[7] Uber. Accessed: Aug. 2022. [Online]. Available: https://www.uber.com/

[8] L. Chen, Y. Gao, Z. Liu, X. Xiao, C. S. Jensen, and Y. Zhu, "PTrider: A price-and-time-aware ridesharing system," *Proc. VLDB Endowment*, vol. 11, no. 12, pp. 1938–1941, 2018.

[9] Y. Chen, M. Li, S. Zheng, C. Lal, and M. Conti, "Where to meet a driver privately: Recommending pick-up locations for ride-hailing services," in *Proc. Int. Workshop Secur. Trust Manage.*, 2021, pp. 43–61.

[10] M. G. Demissie, L. Kattan, S. Phithakkitnukoon, G. H. de Almeida Correia, M. Veloso, and C. Bento, "Modeling location choice of taxi drivers for passenger pickup using GPS data," *IEEE Intell. Transp. Syst. Mag.*, vol. 13, no. 1, pp. 70–90, Spring 2021.

[11] Y. Ding, S. Liu, J. Pu, and L. M. Ni, "HUNTS: A trajectory recommendation system for effective and efficient hunting of taxi passengers," in *Proc. IEEE 14th Int. Conf. Mobile Data Manage.*, 2013, pp. 107–116.

[12] H. Dong, X. Zhang, Y. Dong, C. Chen, and F. Rao, "Recommend a profitable cruising route for taxi drivers," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst.*, 2014, pp. 2003–2008.

[13] S. Guo et al., "A force-directed approach to seeking route recommendation in ride-on-demand service using multi-source urban data," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1909–1926, Jun. 2022.

[14] S. Guo et al., "ROD-revenue: Seeking strategies analysis and revenue prediction in ride-on-demand service using multi-source urban data," *IEEE Trans. Mobile Comput.*, vol. 19, no. 9, pp. 2202–2220, Sep. 2020.

[15] S. Guo et al., "A simple but quantifiable approach to dynamic price prediction in ride-on-demand services leveraging multi-source urban data," *Proc. ACM Interactive, Mobile, Wearable Ubiquitous Technol.*, vol. 2, no. 3, pp. 1–24, 2018.

[16] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1984, pp. 47–57.

[17] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *J. Roy. Statist. Society. Ser. C. (Appl. Statist.)*, vol. 28, no. 1, pp. 100–108, 1979.

[18] Q. Huang, J. Du, G. Yan, Y. Yang, and Q. Wei, "Privacy-preserving spatio-temporal keyword search for outsourced location-based services," *IEEE Trans. Serv. Comput.*, early access, Jun. 11, 2021, doi: 10.1109/TSC.2021.3088131.

[19] Z. Huang, G. Shan, J. Cheng, and J. Sun, "TRec: An efficient recommendation system for hunting passengers with deep neural networks," *Neural Comput. Appl.*, vol. 31, no. 1, pp. 209–222, 2019.

[20] Z. Huang, J. Tang, G. Shan, J. Ni, Y. Chen, and C. Wang, "An efficient passenger-hunting recommendation framework with multitask deep learning," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7713–7721, Oct. 2019.

[21] R. E. Korf, "Depth-first iterative-deepening: An optimal admissible tree search," *Artif. Intell.*, vol. 27, no. 1, pp. 97–109, 1985.

[22] N. Lim et al., "Origin-aware next destination recommendation with personalized preference attention," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2021, pp. 382–390.

[23] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1774–1783.

[24] H. Liu, Y. Tong, J. Han, P. Zhang, X. Lu, and H. Xiong, "Incorporating multi-source urban data for personalized and context-aware multi-modal transportation recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 2, pp. 723–735, Feb. 2022.

[25] Z. Liu, Z. Gong, J. Li, and K. Wu, "Mobility-aware dynamic taxi ridesharing," in *Proc. IEEE 36th Int. Conf. Data Eng.*, 2020, pp. 961–972.

[26] Z. Liu, J. Li, and K. Wu, "Context-aware taxi dispatching at city-scale using deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 1996–2009, Mar. 2022.

[27] Z. Liu, Z. Li, M. Li, W. Xing, and D. Lu, "Mining road network correlation for traffic estimation via compressive sensing," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 1880–1893, Jul. 2016.

[28] Z. Liu, P. Zhou, Z. Li, and M. Li, "Think like a graph: Real-time traffic estimation at city-scale," *IEEE Trans. Mobile Comput.*, vol. 18, no. 10, pp. 2446–2459, Oct. 2019.

[29] Y. Lu, Y. Qi, S. Qi, Y. Li, H. Song, and Y. Liu, "Say no to price discrimination: Decentralized and automated incentives for price auditing in ride-hailing services," *IEEE Trans. Mobile Comput.*, vol. 2, no. 2, pp. 663–680, Feb. 2022.

[30] Y. Lu et al., "Safety warning! Decentralised and automated incentives for disqualified drivers auditing in ride-hailing services," *IEEE Trans. Mobile Comput.*, early access, Aug. 27, 2021, doi: 10.1109/TMC.2021.3108012.

[31] Y. Luo, X. Jia, S. Fu, and M. Xu, "pRide: Privacy-preserving ride matching over road networks for online ride-hailing service," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 7, pp. 1791–1802, Jul. 2019.

[32] S. Ma, Y. Zheng, and O. Wolfson, "Real-time city-scale taxi ridesharing," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 7, pp. 1782–1795, Jul. 2015.

[33] A. Pham, I. Dacosta, G. Endignoux, J. R. T. Pastoriza, K. Huguenin, and J.-P. Hubaux, "ORide: A privacy-preserving yet accountable ride-hailing service," in *Proc. 26th USENIX Conf. Secur. Symp.*, 2017, pp. 1235–1252.

[34] B. Qu, W. Yang, G. Cui, and X. Wang, "Profitable taxi travel route recommendation based on big taxi trajectory data," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 2, pp. 653–668, Feb. 2020.

[35] L. Song et al., "TaxiHailer: A situation-specific taxi pick-up points recommendation system (demo)," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2014, pp. 523–526.

[36] H. Tang, M. Kerber, Q. Huang, and L. Guibas, "Locating lucrative passengers for taxicab drivers," in *Proc. 21st ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst.*, 2013, pp. 504–507.

[37] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, J. Ye, and K. Xu, "A unified approach to route planning for shared mobility," *Proc. VLDB Endowment*, vol. 11, no. 11, 2018, Art. no. 1633.

[38] D. Wang, W. Cao, J. Li, and J. Ye, "DeepSD: Supply-demand prediction for online car-hailing services using deep neural networks," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, 2017, pp. 243–254.

[39] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? Estimating travel time based on deep neural networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2500–2507.

[40] R. Wang et al., "TaxiRec: Recommending road clusters to taxi drivers using ranking-based extreme learning machines," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 3, pp. 585–598, Mar. 2018.

[41] X. Xie, F. Zhang, and D. Zhang, "PrivateHunt: Multi-source data-driven dispatching in for-hire vehicle systems," *Proc. ACM Interactive, Mobile, Wearable Ubiquitous Technol.*, vol. 2, no. 1, pp. 1–26, 2018.

[42] X. Xu, J. Zhou, Y. Liu, Z. Xu, and X. Zhao, "Taxi-RS: Taxi-hunting recommendation system based on taxi GPS data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 1716–1727, Aug. 2015.

[43] Z. Xu et al., "Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 905–913.

[44] C. Yang and G. Gidofalvi, "Fast map matching, an algorithm integrating hidden Markov model with precomputation," *Int. J. Geographical Inf. Sci.*, vol. 32, no. 3, pp. 547–570, 2018.

[45] H. Yuan, G. Li, Z. Bao, and L. Feng, "Effective travel time estimation: When historical trajectories over road networks matter," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2020, pp. 2135–2149.

[46] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun, "Where to find my next passenger," in *Proc. 13th Int. Conf. Ubiquitous Comput.*, 2011, pp. 109–118.

[47] N. J. Yuan, Y. Zheng, L. Zhang, and X. Xie, "T-finder: A recommender system for finding passengers and vacant taxis," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2390–2403, Oct. 2013.

[48] D. Zhang and T. He, "pCruise: Reducing cruising miles for taxicab networks," in *Proc. IEEE 33rd Real-Time Syst. Symp.*, 2012, pp. 85–94.

[49] K. Zhao et al., "Discovering subsequence patterns for next POI recommendation," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2020, pp. 3216–3222.

[50] X. Zheng, X. Liang, and K. Xu, "Where to wait for a taxi?," in *Proc. ACM SIGKDD Int. Workshop Urban Comput.*, 2012, pp. 149–156.

[51] W. Zhu, J. Lu, Y. Li, and Y. Yang, "A pick-up points recommendation system for ridesourcing service," *Sustainability*, vol. 11, no. 4, 2019, Art. no. 1097.

Zhidan Liu (Member, IEEE) received the PhD degree in computer science and technology from Zhejiang University, Hangzhou, China, in 2014. After that, he worked as a research fellow in Nanyang Technological University, Singapore. He is currently an associate professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include mobile computing, Big Data analytics, Internet of Things, and urban computing. He is a member of ACM, and CCF.

Hongquan Zhang received the BS degree in IoT engineering from Hunan Agricultural University, Changsha, China, in 2020. He is currently working toward the second-year master's degree with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China, under the supervision of Dr. Zhidan Liu. His research interests are in the areas of trajectory data analysis and urban computing.

Guofeng Ouyang received the BS degree in electronical information science and technology from Zhaoqing University, Zhaoqing, China, in 2021. He is currently working toward the first-year master's degree with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China, under the supervision of Dr. Zhidan Liu. His research interests are in the areas of trajectory data analysis and urban computing.

Junyang Chen received the PhD degree in computer and information science from the University of Macau, Macau, China, in 2020. He is currently an assistant professor with the College of Computer Science and Software Engineering, Shenzhen University, China. His research interests include graph neural networks, text mining, and recommender systems.

Kaishun Wu (Member, IEEE) received the PhD degree in computer science and engineering from The Hong Kong University of Science and Technology (HKUST), Hong Kong, China, in 2011. After that, he worked as a research assistant professor with HKUST. In 2013, he joined Shenzhen University as a distinguish professor. Currently, he is a professor of the DSA & IoT Thrust Area under the Information Hub with the Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China. He has co-authored 2 books and published more than 100 high quality research papers in international leading journals and primer conferences, like *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Parallel and Distributed Systems*, ACM MobiCom, IEEE INFOCOM. He is the inventor of 6 US and more than 90 Chinese pending patents. He received 2012 Hong Kong Young Scientist Award, 2014 Hong Kong ICT awards: Best Innovation, and 2014 IEEE ComSoc Asia-Pacific Outstanding Young Researcher Award. He is an IET fellow.