mT-Share: A Mobility-Aware Dynamic Taxi Ridesharing System

Zhidan Liu, *Member, IEEE*, Zengyang Gong, Jiangzhou Li, and Kaishun Wu, *Member, IEEE*

Abstract*—Due to the wide availability of taxis in a city and the tremendous benefits of ridesharing, taxi ridesharing becomes promising and attractive in recent years. Existing taxi ridesharing schemes simply match ride requests and taxis based on partial trip information and omit the offline passengers, who will hail a taxi at the roadside without submitting the ride requests to the system. Thus, they are still not efficient and practical. In this article, we consider the mobility-aware taxi ridesharing problem and present *mT-Share* to address these limitations. *mT-Share* fully exploits the mobility information of taxis and ride requests to achieve efficient indexing of taxis/requests and better passenger–taxi matching, while still satisfying the constraints on passengers' deadlines and taxis' capacities. Specifically, *mT-Share* makes use of both geographical information and travel directions to index taxis and ride requests and supports the shortest path-based routing and probabilistic routing to serve both online and offline ride requests. In addition, *mT-Share* proposes a novel payment model to share the ridesharing benefits among the taxi driver and passengers. Extensive evaluations using a large real-world taxi data set demonstrate the efficiency and effectiveness of *mT-Share*, which can respond each ride request in milliseconds and be with moderate detour costs and passengers' waiting time. Compared to state-of-the-art schemes, *mT-Share* can serve 42% and 62% more ride requests in peak and nonpeak hours, respectively. Furthermore, *mT-Share* can save 8.6% taxi fare for passengers and meanwhile increase 7.8% incomes for taxi drivers, when compared with the regular taxi services.

***Index Terms*—Clustering, mobility pattern, payment model, route planning, taxi ridesharing.**

I. Introduction

RIDESHARING utilizes one vehicle to serve multiple passengers, who have similar time schedules and itineraries, and thus potentially brings many benefits for an urban city, e.g., alleviating traffic congestion and reducing energy consumption [29]. Recently, taxi ridesharing becomes promising because of the wide availability of taxis in a city [24], [30], [53]. Different from the carpooling services [16], [36], which normally serve static ride requests with early planned routes, taxi ridesharing is relatively more complex. This is because both taxis and ride requests are extremely dynamic [29]. Once passengers need a ride, they submit their requests immediately with no prior planning. Even worse, some passengers prefer to hail a taxi at roadside without explicitly submitting their requests. Compared to private vehicles-based ridesharing [5], [11], [12], [36], taxis are operating all the time and thus are more flexible yet complex. These properties cause taxi ridesharing especially challenging, since ride requests have to be assigned to proper taxis timely and the schedule/route of a shared taxi should be constantly updated to guarantee the quality of service [28].

Some valuable efforts have been devoted to design taxi ridesharing schemes [18], [28]–[30], [53]. For a given ride request, these schemes usually make use of the request's origin location and geographical distribution of available taxis to determine a candidate taxi set, and then select the one, which introduces the minimum detour cost while satisfying other passengers' service requirements, to serve the request. Due to some practical limitations, however, the schemes are not sufficiently efficient yet. First, most previous schemes only utilize partial trip information, i.e., taxis' current locations and a request's origin location, to determine the candidate taxi set. As a result, they may not find the most suitable taxi to serve a request. Second, existing schemes merely consider online ride requests, while in practice, some passengers may be *offline* as they hail a taxi at roadside. Thus, such passengers are invisible to the system. Regarding users' preferences of getting taxi services, a recent taxi service research report shows that 41.68% of users prefer either online booking or offline hailing, while 13.71% of users get taxi services only in an offline manner [3]. As a matter of fact, the amount of offline requests could be quite large (i.e., 13.71%–55.39% of users), and an appropriate scheme is required to well handle such requests.

In this article, we will consider a practical taxi ridesharing problem, namely, mobility-aware taxi ridesharing (MTR). The MTR problem aims to completely exploit the mobility information from both taxis and ride requests, so as to maximize the number of served requests and minimize the overall detour cost, subject to the constraints of passengers' deadlines and taxis' capacities. However, we find that the MTR

Manuscript received June 4, 2021; revised July 26, 2021; accepted August 2, 2021. Date of publication August 5, 2021; date of current version December 23, 2021. This work was supported in part by the National Science Foundation of China (NSFC) under Grant 61802261, Grant 61872248, and Grant U2001207; in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2020A1515011502 and Grant 2017A030312008; in part by the Shenzhen Science and Technology Foundation under Grant ZDSYS20190902092853047 and Grant R2020A045; in part by the Project of DEGP under Grant 2019KCXTD005; and in part by the Guangdong "Pearl River Talent Recruitment Program" under Grant 2019ZT08X603. This article was presented in part at the IEEE 36th International Conference on Data Engineering (IEEE ICDE 2020), Dallas, TX, USA, April 20–24, 2020 [DOI: 10.1109/ICDE48307.2020.00088]. *(Corresponding author: Kaishun Wu.)*

Zhidan Liu, Jiangzhou Li, and Kaishun Wu are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: liuzhidan@szu.edu.cn; lijiangzhou2018@email.szu.edu.cn; wu@szu.edu.cn).

Zengyang Gong is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: zgongae@cse.ust.hk).

Digital Object Identifier 10.1109/JIOT.2021.3102638

problem is extremely challenging. The challenges primarily stem from two aspects, namely, the high dynamics of online requests and taxis and the uncertainty of offline requests.

To address the above problem and improve the existing schemes, we propose a novel taxi ridesharing scheme, named *mT-Share*, by fully exploiting both the known mobility information from ride requests and taxis and the hidden mobility patterns from historical data to match requests with the most suitable taxis. The key idea of *mT-Share* is that the best passenger–taxi matches should be the pairs of ride requests and taxis, which have geographically close origins and destinations and share the similar travel directions. Therefore, *mT-Share* proposes the bipartite map partitioning and mobility clustering to efficiently index ride requests and taxis from these two aspects, respectively. Thanks to the built indexes, *mT-Share* can assign the most suitable taxi to serve a request via the mobility clustering and a proper similarity measure. With the map partitions, *mT-Share* enables two routing modes and further optimizes passenger–taxi matching by improving taxi scheduling efficiency, while simultaneously satisfying the constraints on both requests' deadlines and taxi's capacity. By considering the mobility patterns of taxi orders [10], [23], we propose a novel probabilistic routing that allows shared taxis to opportunistically encounter offline requests with much higher probabilities. Furthermore, *mT-Share* proposes a payment model to share the ridesharing benefits among passengers and taxi drivers.

We summarize the contributions of this work as follows.

1) We analyze and identify the limitations of previous taxi ridesharing schemes and further consider the practical MTR problem by exploiting the mobility information to serve both online and offline ride requests.
2) We present *mT-Share* to well address the MTR problem. By incorporating the holistic mobility information of ride requests and taxis, *mT-Share* optimizes the indexing of taxis/requests and passenger–taxi matching.
3) We further present a novel payment model that is able to fairly share the ridesharing benefits among passengers and drivers, so as to encourage more riders and taxi drivers to join in the ridesharing campaign.
4) Extensive experiments have been performed to evaluate *mT-Share* using a large real-world taxi data set. The results demonstrate that *mT-Share* greatly outperforms the state-of-the-art schemes, e.g., serving 42% and 62% more ride requests in the peak and nonpeak hours, respectively. Compared with no taxi ridesharing, *mT-Share* saves 8.6% taxi fare for passengers and meanwhile increases 7.8% incomes for taxi drivers.

The remainder of this article is organized as follows. We review the related works in Section II. The MTR problem is described in Section III. We present the design of *mT-Share* in Section IV and evaluate the performances in Section V. Finally, we conclude this article in Section VI.

II. Related Work

A. Carpooling

Carpooling is also known as recurring ridesharing, which primarily deals with routine commutes, e.g., between home and workplace. As carpooling generally involves a few drivers and riders, it can be solved with linear programming to get the optimal solution [7]. GPS trajectories of users can be leveraged to discover possible carpooling opportunity [16], [37], while *coRide* [50] is proposed to design carpooling service's schedule and route. Compared to carpooling whose ride requests can be known in a prior, the ride requests of taxi ridesharing are generated instantaneously and the routes of shared taxis are constantly changing. Therefore, taxi ridesharing is more dynamic than carpooling.

Taxi ridesharing can be modeled as a variant of the famous dial-a-ride problem (DARP). In general, DARP aims to design the schedule and route for a number of riders between their origins and destinations [13]. Note that riders in DARP are assumed to specify their pick-up and drop-off locations in advance. Existing works for DARP mainly consider the static scenario [12], where ride requests are preknown.

B. Ridesharing

Recently, ridesharing has been widely studied because of its attractive benefits [29]. The dynamic ridesharing can be modeled as a combinatorial optimization problem and has been proved to be NP-hard [8]. Thus, a variety of heuristic techniques are proposed to optimize the two major stages of ridesharing, i.e., candidate taxi searching [21], [34], [36], [39] and ridesharing routing [19], [42], [57]. For example, Li *et al.* [21] took both social relations between drivers and riders and the revenue into consideration to select the top-*k* suitable vehicles for a ride request. Similarly, social preferences are also considered in the passengers–vehicle matching process [34]. Tong *et al.* [42], [45] have optimized route planning of shared mobility with a smart insertion. In particular, two recent works [44], [49] also make use of demand predictions to plan ridesharing routes, so as to serve more compatible passengers. Our work differs from them by considering both online and offline ride requests, and meanwhile optimizing passenger–taxi matching by fully exploiting the mobility information of ride requests and taxis.

Because of wide availability of taxis in an urban city [10], taxi ridesharing becomes promising and has already attracted substantial research attentions [18], [24], [28]–[30], [53]. For example, Ma *et al.* [29], [30] developed a mobile-cloud-enabled taxi-sharing system called *T-Share*, and Ma *et al.* [28] further considered the service quality of taxi ridesharing to improve *T-Share*. In particular, Hou *et al.* [18] focused on the transfer-allowed taxi ridesharing using battery-limited electric vehicles. Zhang *et al.* [53] further took the passenger's acceptance probability on taxi ridesharing into the design. Different from our work, these works do not exploit mobility information to achieve appropriate passenger–taxi matching and meanwhile have omitted the offline passengers.

Due to the increasing popularity of ridesharing, other factors of the ridesharing, e.g., the pricing models [11], [21], [41], ridesharing order dispatching [4], [20], [56], partner selection [17], destination matching [31], privacy protection [35], [43], [47], riders' satisfaction [12], and riders' attitude on ridesharing [53], have also been studied in the past years.

C. Taxi Demands and Dispatching

Taxis play an important role in urban transportation, and thus it is essential to perceive taxi demands and well reposition taxis to balance the supply–demand gap. A great number of works [40], [46], [52] have been made on taxi demand predictions by processing massive historical taxi data. For example, Geng *et al.* [15] proposed a deep learning model to predict the region-level taxi ride-hailing demands. Different from the works which predict demands in regular taxi services, we predict the offline ride requests for taxi ridesharing that is even more challenging. It is worth noting that a recent work [23] presents a ridesharing routing scheme to serve potential riders, which are predicted from the statistics of mobility data. Our scheme differs from it by considering and serving both online and offline requests.

Furthermore, many works [22], [25], [38], [52], [54], [55] have explored the taxi dispatching problem given the known taxi demands. Specifically, an order dispatching model is proposed to maximize the matching success ratio of ride requests and taxis in [52]. Lin *et al.* [22] presented a fleet management system, which explicitly coordinates idle taxis using the multiagent deep reinforcement learning (DRL) theory. Similarly, Tang *et al.* [38] also applied DRL on the order dispatching problem with a deep value network. Liu *et al.* [25] improved DRL-based taxi dispatching solutions by considering more context information. In addition, Zhao *et al.* [54] further considered the preference-aware taxi dispatching using online stable matching. These works dispatch a vacant taxi for each individual ride request, while do not consider the ridesharing among passengers.

III. Problem Statement

In this section, we present the definition, motivation, and the problem statement of MTR. We summarize the key notations used in this article in Table I.

TABLE I
Summary of Key Notations

Notation	Description
$\mathbf{G}(\mathbf{V}, \mathbf{E})$	The directed graph of a road network
$cost(\cdot)$	A function to calculate the travel cost
$\mathbf{r_i}$	The i-th ride request
$\bar{\mathbf{r}}_\mathbf{i}$	The i-th offline ride request
o_{r_i}	The origin of ride request $\mathbf{r_i}$
d_{r_i}	The destination of ride request $\mathbf{r_i}$
e_{r_i}	Delivery deadline of ride request $\mathbf{r_i}$
$\mathbf{t_j}$	Status of the j-th taxi
$\mathcal{S}_{t_j}$	Schedule of taxi $\mathbf{t_j}$
$\mathcal{R}_{t_j}$	Route of taxi $\mathbf{t_j}$
κ	Number of partitions for a road network
$\mathbb{P}$	A set of map partitions $\{P_z\}_{z=1}^{\kappa}$
ℓ_z	Landmark of partition P_z
$\mathbf{G}_\ell(\mathbf{V}_\ell, \mathbf{E}_\ell)$	The landmark graph
$\vec{\mathbf{v}}$	A mobility vector
C_a	A mobility cluster
θ	Travel direction difference of two mobility vectors
λ	Threshold to determine similar travel direction
γ	Searching range
$\mathbb{T}_{r_i}$	Candidate taxi set for ride request $\mathbf{r_i}$

A. Preliminary

Definition 1 (Road Network): A road network is modeled as a directed graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$, where a vertex $v \in \mathbf{V}$ denotes a geolocation (e.g., road intersection), and an edge $(u, v) \in \mathbf{E}$ represent a road segment that owns a weight $\mathrm{cost}(u, v)$ to indicate the *travel cost* from u to v.

Specifically, the travel cost is estimated as either a travel time or a travel distance. Once the travel speed of a taxi is known, they can be easily converted from one to another. Thus, we do not differentiate them throughout this article and use the travel cost consistently. Similar as previous studies [29], [42], we let traffic conditions to be stable, and thus the travel cost of each edge is constant. However, our system could easily extend to run with real-time traffic conditions if such information can be timely derived from the transportation agency or be inferred by some advanced traffic estimation methods [26], [27].

Based on a recent report [33], we assume that most riders are willing to take the taxi ridesharing services. One rider can either explicitly submit her ride request to the system via some booking App or implicitly participate in the ridesharing by hailing a taxi at roadside. The ridesharing system will select a suitable taxi to serve this request by investigating the statuses of all taxis, and update the schedule/route of the chosen taxi, subject to the service requirements.

Definition 2 (Ride Request): A ride request is denoted by $\mathbf{r_i} = <t_{r_i}, o_{r_i}, d_{r_i}, e_{r_i}>$ with a trip origin $o_{r_i} \in \mathbf{V}$ and a trip destination $d_{r_i} \in \mathbf{V}$. The ride request is released at time t_{r_i} and should be finished before time e_{r_i} by delivering passengers from origin o_{r_i} to destination d_{r_i}.

In real-world taxi ridesharing systems, we may adopt two deadlines for pick-up and drop-off, respectively [30]. However, a single deadline for delivery e_{r_i} usually suffices [42]. Given delivery deadline e_{r_i} and travel cost $\mathrm{cost}(o_{r_i}, d_{r_i})$ between o_{r_i} to d_{r_i}, the pick-up deadline can be calculated as $e_{r_i} - \mathrm{cost}(o_{r_i}, d_{r_i})$. The online request will be immediately known once $\mathbf{r_i}$ is submitted, but the offline requests could be perceived only when they are encountered by shared taxis. In particular, $\bar{\mathbf{r}}_\mathbf{i}$ is used to represent an offline request.

Definition 3 (Taxi Status): The instantaneous status of the jth taxi is denoted by $\mathbf{t_j} =< \mathrm{loc}_{t_j}, \mathcal{S}_{t_j}, \mathcal{R}_{t_j} >$, where loc_{t_j} represents taxi $\mathbf{t_j}$'s current location, and $\mathcal{S}_{t_j}$ and $\mathcal{R}_{t_j}$ are the schedule and route of taxi $\mathbf{t_j}$, respectively.

Definition 4 (Taxi Schedule): A taxi schedule $\mathcal{S}_{t_j} = \{s_1, s_2, \ldots, s_m\}$ is a sequence of events for a shared taxi, where each event corresponds to pick-up or drop-off the ridesharing passenger at some location, e.g., o_{r_i} or d_{r_i} of a ride request $\mathbf{r_i}$ and o_{r_i} should appear ahead of d_{r_i}.

Definition 5 (Taxi Route): A taxi route $\mathcal{R}_{t_j}$ is generated for a taxi schedule $\mathcal{S}_{t_j}$. It includes the travel path for any two consecutive events in schedule $\mathcal{S}_{t_j}$.

Given the ride requests to share a taxi, a valid taxi schedule is established to sequentially pick-up and deliver passengers along a ridesharing route. In previous works [11], [19], [29], [30], [42], travel path between two consecutive event locations is usually set as the shortest path. Therefore, a taxi route can be derived by concatenating a sequence of such shortest paths. When ridesharing requests are picked up or delivered by taxi $\mathbf{t_j}$, both $\mathcal{S}_{t_j}$ and $\mathcal{R}_{t_j}$ should be timely updated.

B. Motivation

The taxi ridesharing problem can be modeled as a combinatorial optimization problem and has been proved to be NP-hard [8]. Previous works [18], [28]–[30], [53] thus have proposed many heuristic techniques to match requests with suitable shared taxis. Generally, they index all taxis and requests using the grids of a road network and then process each request $\mathbf{r_i}$ through the following two major stages.

Stage 1 (Taxi Searching): Taxis within a range γ around $\mathbf{r_i}$'s origin o_{r_i} are chosen as candidate taxis for serving $\mathbf{r_i}$.

Stage 2 (Ridesharing Routing): The schedule of each candidate taxi $\mathbf{t_j}$ is investigated by inserting $\mathbf{r_i}$'s pick-up and drop-off events into $\mathcal{S}_{t_j}$, subject to passengers' delivery deadlines and taxi's capacity. The taxi, which will introduce the minimum cost (e.g., the minimum increased travel cost), is usually selected as the one to serve ride request $\mathbf{r_i}$.

In practice, the searching range γ could be enlarged [53] and the two stages can be repeated until one proper taxi is finally found [30]. However, we observe that existing works still suffer from the following two limitations.

1) *Inefficient Passenger–Taxi Matching:* Most of the existing works determine the candidate taxis with only the given ride request's origin location [28], [42], [53]. Even though [29] and [30] utilize both origin and destination to conduct a dual-side search, such location information is separately considered. Besides, rather than searching for the best taxi to serve a request, some schemes only return the valid taxi that is the first discovered [29], [30]. Relying on such partial trip information, they cannot filter out the invalid taxis at the beginning, and may also miss the best solution, which has the minimum ridesharing cost.
2) *Omitting the Offline Ride Requests:* The existing works merely consider online ride requests [18], [28]–[30], [53], while in practice, there still exist many people who prefer to hail a taxi at roadside without submitting their ride requests [23], [48]. As a concrete example, some elder citizens who are not familiar with online taxi booking and the passengers who forget to carry out mobile phones are unable to issue online ride requests. According to the statistics on users' preferences of getting taxi services in a recent taxi service research report [3], 44.61% of users only prefer online booking, while 13.71% of users merely hail taxis at the roadside. Furthermore, 41.68% of users prefer either offline hailing or online booking. These statistics indicate that the amount of potential offline ride requests could be quite large, i.e., 13.71%–55.39%. A practical taxi ridesharing system should consider and serve such offline requests. In addition, offline requests are also desirable for taxi drivers, especially in nonpeak hours where online requests are inadequate [40]. Serving these offline passengers can improve taxi utilization and meanwhile increase taxi drivers' incomes [10].

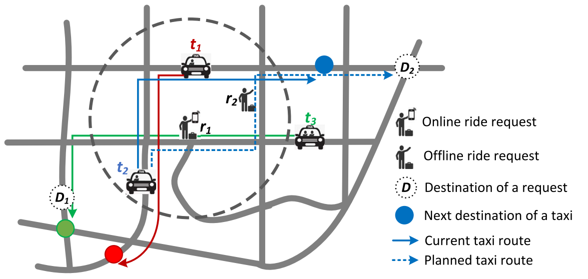


Fig. 1. Motivation example with three shared taxis (i.e., t_1, t_2, and t_3), one online request r_1, and one offline request $\bar{r_2}$. Solid lines are current taxi routes, dashed line is a planned route, and taxi routes are differentiated by colors. The gray dashed circle is the searching range for request r_1.

We explain the above arguments with an example in Fig. 1, where three shared taxis (i.e., t_1, t_2, and t_3) travel along their respective routes, and passenger r_1 submits the ride request to the system while passenger $\bar{r_2}$ will hail a taxi at roadside. Once receiving r_1's request, existing schemes will determine the candidate taxi set with a searching range around request r_1's origin. Here, t_1 and t_2 are returned for serving r_1. Then, their schedules are examined, and this procedure will involve extensive computations. We find that, however, t_2 should never be considered as it travels inversely with r_1. The examination of t_2's schedules thus introduces unnecessary computations. Although t_1 could serve r_1 with some detour, we find that t_3 actually should be the best one to serve r_1 without any detour cost. However, t_3 is even not considered as a candidate taxi for r_1. On the other hand, since passenger $\bar{r_2}$ does not explicitly submit her request, existing schemes cannot assign taxis to serve $\bar{r_2}$. If the system can perceive the existence of $\bar{r_2}$, it can serve $\bar{r_2}$ as well by slightly adjusting t_2's route.

It is necessary and possible for a taxi ridesharing scheme to consider both online and offline requests. On the one hand, there exist many passengers who still hail a taxi at the roadside rather than online booking [3], [10]. On the other hand, taxi drivers also desire to serve offline requests to improve their incomes, especially during the nonpeak hours where there are more taxi supplies than online taxi demands [48]. As a concrete example, taxi drivers in China can serve offline requests when they are not assigned with online requests. Furthermore, the mobility patterns of urban travel demands provide us an opportunity to predict the offline ride requests by exploiting the historical statistics [23], [32], [46], and thus open a new design space to improve the existing taxi ridesharing schemes.

C. Problem Definition

To improve existing works, we consider a novel MTR problem as follows.

Definition 6 (Mobility-Aware Taxi Ridesharing Problem): Given a set of online ride requests and offline ride requests to predict, and a set of taxis on road network $\mathbf{G}$, the MTR problem aims to match requests with suitable shared taxis, such that the number of served requests is maximized while the total detour cost is also minimized. The arrangements should meet the following two constraints.

1) *Capacity Constraint:* The number of passengers sharing a taxi cannot exceed the taxi's capacity at any time.
2) *Time Constraint:* Passengers of a request should be delivered to their destinations before the specific deadline.

Challenges: Different from previous works, the MTR problem considers both online requests and offline requests to improve the efficiency and practicability of taxi ridesharing. Since dynamic ridesharing is NP-hard [8], [12], [42], the MTR problem is challenging to be solved as well, mainly due to the following two challenges.

1) Both ride requests and taxis are quite dynamic, which thus requires that both taxi schedules and routes should be wisely and efficiently planned so as to guarantee the service quality of ridesharing, e.g., minimizing the detour costs.
2) Because the exact information of offline requests cannot be known in advance, it is thus difficult for shared taxis to well serve the offline ride requests. This uncertainty further makes taxi scheduling and routing to be more intricate.

IV. System Design

A. Overview

Fig. 2 illustrates the framework of *mT-Share*. In general, *mT-Share* takes historical taxi data, real-time taxi statuses and ride requests, and the road map as the input and arranges the available taxis to dynamically serve both online and offline ride requests. On the user side, passengers can either explicitly report their ride requests to *mT-Share* or hail a shared taxi at the roadside in an offline manner. On the taxi side, a shared taxi continuously uploads its status, which includes current location, available seats, etc., to the server and receives the updated schedule/route from the server.

mT-Share has three major modules, i.e., *Taxi/Request Indexing*, *Passenger–Taxi Matching*, and the *Payment Model*. Specifically, the *Taxi/Request Indexing* module exploits the mobility patterns that are discovered from historical taxi data to divide the road map into partitions and classifies taxis and ride requests into mobility clusters based on their travel directions. Both map partitions and mobility clusters are utilized to index and track the shared taxis. With these indexes, the *Passenger–Taxi Matching* module can effectively search candidate taxis and determine the best one, which introduces the minimum detour cost, to serve a given request. *mT-Share* supports both basic routing and probabilistic routing and accelerates them with the partition filtering. In particular, probabilistic routing enables a shared taxi to meet *suitable* offline requests with a much higher probability. Furthermore, *mT-Share* proposes a payment model, which determines the taxi fares by sharing the ridesharing benefits between a taxi driver and the passengers.

B. Taxi/Request Indexing

mT-Share will index and track shared taxis and ride requests based on both geographical location and travel direction,

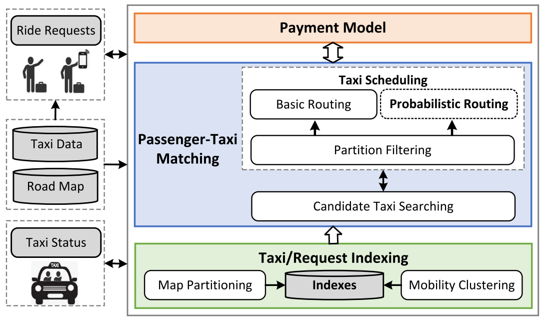


Fig. 2. Framework of *mT-Share*.

which can be achieved by performing bipartite map partitioning and mobility clustering, respectively.

1) Bipartite Map Partitioning: Rather than dividing a road network graph using geographical information [29], [30] or popular pick-up locations [28], *mT-Share* classifies the vertices of a road network graph into clusters based on their both geographical locations and transition patterns that are mined from historical taxi data. Specifically, we use the *k-means* clustering algorithm to initially classify all graph vertices into κ spatial clusters according to their geographical locations (i.e., latitude and longitude). Then, the map partitioning runs as follows.

① *Transition Probability Calculation:* For each vertex v_i and the κ spatial clusters, we calculate a transition probability vector $\vec{B}_i$ of size κ. Each item $B_{ij} \in \vec{B}_i$ ($i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, \kappa$) indicates the transition probability of ride requests that called a taxi at vertex v_i and traveled to any vertex of the jth spatial cluster. The historical taxi data can be used to calculate the transition probabilities.

② *Transition Clustering:* We regard vector $\vec{B}_i$ as vertex v_i's mobility feature, and use *k-means* clustering to group all graph vertices into k_t transition clusters based on their transition probability vectors. The vertices of a transition cluster will have the similar transition patterns. Empirically, we let $k_t < \kappa$, and set $k_t = 20$ for *mT-Share* by default.

③ *Geo-Clustering on Transition Clusters:* For each transition cluster of size n, we group its vertices into $\lfloor n\kappa/N + 1/2 \rfloor$ spatial clusters based on their locations through *k-means* clustering, where $N = |\mathbf{V}|$ denotes the number of all vertices.

Our bipartite map partitioning will repeat the above three steps until the κ spatial clusters derived in step ③ do not change. These clusters are regarded as the final *partitions* of the road network graph $\mathbf{G}$, which is represented as $\mathbb{P} = \{P_z\}_{z=1}^{\kappa}$. The vertices of a partition are both geographically close and highly similar on their transition patterns. Such properties facilitate the prediction of suitable offline ride requests by well supporting the probabilistic routing that is introduced later. In Section V-C, we conduct experiments to study the setting of κ and demonstrate the advantage of bipartite map partitioning over the traditional grid-based method on finding more offline passengers. Fig. 3(b) demonstrates the result of applying the bipartite map partitioning on the road network graph of Chengdu city, which is shown in Fig. 3(a).

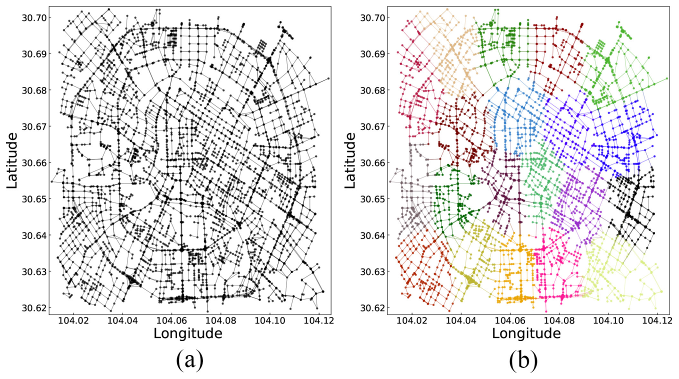


Fig. 3. (a) Road network of Chengdu city, China. (b) Result of bipartite map partitioning applied on (a), where we set $\kappa = 20$ just for a clear demonstration and the partitions are differentiated by colors.

For each spatial partition, we calculate its center point as the partition's *landmark*. With the spatial partitions and their corresponding landmarks, we construct a *landmark graph* $\mathbf{G}_\ell$ that will be exploited to speedup route planning.

Definition 7 (Landmark): The landmark of a spatial partition P_z is the vertex $\ell_z \in P_z$, which has the minimum distance to all other vertices of partition P_z.

Definition 8 (Landmark Graph): A landmark graph is represented as $\mathbf{G}_\ell(\mathbf{V}_\ell, \mathbf{E}_\ell)$, where vertices in $\mathbf{V}_\ell$ are the landmarks of all partitions and an edge between two landmarks indicates that their corresponding partitions are adjacent.

Note that the bipartite map partitioning could be periodically executed with a relatively long interval, e.g., one year, when sufficient new taxi data are accumulated to capture the latest transition patterns of passengers in an urban city. Once the map partitions are changed, the corresponding landmarks and the landmark graph should also be accordingly updated.

2) Mobility Clustering: Different from existing works that use location information to index taxis and requests only [28]–[30], [36], [42], *mT-Share* builds a *mobility vector* for each shared taxi or ride request, and further groups them through mobility clustering based on their travel directions.

Definition 9 (Mobility Vector): A mobility vector $\vec{\mathbf{v}}$ is defined as a vector pointing from an origin (lat_o, lng_o) to a destination (lat_d, lng_d), denoted by $\vec{\mathbf{v}} = (lat_o, lng_o, lat_d, lng_d)$.

For each ride request $\mathbf{r_i}$, we use its origin o_{r_i} and destination d_{r_i} to create the mobility vector $\vec{\mathbf{v}}_{r_i}$. For a taxi $\mathbf{t_j}$ that serves m passengers $\{\mathbf{r_i}\}_{i=1}^{m}$, we take its location loc_{t_j} as the origin of mobility vector $\vec{\mathbf{v}}_{t_j}$, and regard the center of all destinations of the shared passengers [i.e., $([\sum_{i=1}^{m} d_{r_i}]/m)$] as $\vec{\mathbf{v}}_{t_j}$'s destination. We do not apply mobility clustering for the empty taxis, since they have no fixed travel destinations.

We group taxis and requests into clusters according to their mobility vectors as follows. The first ride request individually forms the initial cluster, and each subsequent request will join an existing cluster or form a new cluster. For each mobility cluster C_a, we maintain a *general mobility vector* $\vec{\mathbf{v}}_{C_a}$. The origin and destination of $\vec{\mathbf{v}}_{C_a}$ are averagely calculated from the origins and destinations of all cluster members, respectively. Once a new ride request $\mathbf{r_i}$ arrives, we compare its mobility vector with each general mobility vector. If the travel direction difference between them is sufficiently small, $\mathbf{r_i}$ will be included into cluster C_a. Specifically, *cosine similarity* is adopted as the metric to measure the travel direction difference θ of two vectors, i.e.,

$$\cos(\theta) = \frac{\vec{\mathbf{v}}_{r_i} \cdot \vec{\mathbf{v}}_{C_a}}{||\vec{\mathbf{v}}_{r_i}|| \times ||\vec{\mathbf{v}}_{C_a}||}. \quad (1)$$

When $\cos(\theta) \geq \lambda$ (where λ is a predefined parameter), $\mathbf{r_i}$ is considered to travel along a similar direction with these passengers in cluster C_a and they could share a taxi. Otherwise, $\mathbf{r_i}$ is forced to form a new mobility cluster. In principle, a smaller λ (i.e., larger θ) would increase the ridesharing chances by finding more candidate taxis for a ride request. However, more candidate taxis require more examination time on the taxi schedules and thus prolong response time for each request.

mT-Share will update the mobility clusters and their corresponding general mobility vectors only when the ride requests are finished or new requests are received. The updating process introduces negligible computation overheads.

3) Index of Taxis: *mT-Share* makes use of both map partitions and mobility clusters to build the index structures, which can facilitate candidate taxi searching for ride requests later.

- *Map Partition-Based Indexing:* For each map partition P_z, *mT-Share* maintains a taxi list $P_z.L_t$ to record taxi IDs, which are now in or will arrive at map partition P_z within a time threshold T_{mp} (e.g., 1 h). According to their arrival time, these taxi IDs are sorted in an ascending order. The taxi list $P_z.L_t$ is dynamically updated.
- *Mobility Cluster-Based Indexing:* For each mobility cluster C_a, *mT-Share* also maintains a taxi list $C_a.L_t$ to contain taxi IDs, which are now serving requests and meanwhile traveling in a similar direction. The taxi list $C_a.L_t$ should be updated once mobility cluster C_a changes.

Memory Complexity: Based on our indexing structures, each taxi could be indexed by several map partitions and (at most) one mobility cluster, while each request is indexed by only one mobility cluster. Thus, the memory complexity of *mT-Share*'s indexing overhead is $\mathcal{O}((x+1)M+R)$, where M is the number of all available taxis, x is the number of map partitions a taxi could visit within time threshold T_{mp}, and R is the number of all ride requests.

C. Passenger–Taxi Matching

For each request $\mathbf{r_i}$, *mT-Share* will determine its candidate taxis using the indexing structures, and then heuristically investigate all possible taxi schedules to find the most suitable taxi that introduces the minimum ridesharing cost to serve $\mathbf{r_i}$.

1) Candidate Taxi Searching: Different from existing works, which may increase the searching range gradually and only return one valid (but may not be the best) taxi [28]–[30], [53], *mT-Share* aggressively searches the best taxi to serve each request $\mathbf{r_i}$. More specifically, the candidate taxi searching range γ for $\mathbf{r_i}$ is set as the product of a typical taxi driving speed and the waiting time Δt that is estimated as

$$\Delta t = e_{r_i} - \text{cost}\big(o_{r_i}, d_{r_i}\big) - t_{r_i} \quad (2)$$

Algorithm 1: Taxi Scheduling

```
1  Input: Ride request $\mathbf{r_i}$ and candidate taxi set $\mathbb{T}_{r_i}$;
2  Output: A taxi with updated schedule/route for $\mathbf{r_i}$;
3  foreach taxi $\mathbf{t_j} \in \mathbb{T}_{r_i}$ do
4      foreach schedule instance $\mathcal{S}'_{t_j} \leftarrow \{\mathcal{S}_{t_j}, o_{r_i}, d_{r_i}\}$ do
5          if flag then
6              $\mathcal{R}'_{t_j}$ = ProbabilisticRouting($\mathcal{S}'_{t_j}$, $\mathbf{t_j}$);
7          else
8              $\mathcal{R}'_{t_j}$ = BasicRouting($\mathcal{S}'_{t_j}$, $\mathbf{t_j}$);
9          $\omega = cost(\mathcal{R}'_{t_j}) - cost(\mathcal{R}_{t_j})$;
10 Select the taxi schedule instance with the minimum $\omega$;
```

where $e_{r_i} - \text{cost}(o_{r_i}, d_{r_i})$ and t_{r_i} are the pick-up deadline and the release time of $\mathbf{r_i}$, respectively. From the searching area that centers at $\mathbf{r_i}$'s origin o_{r_i} with radius γ, we derive a map partition set $\mathbb{S}_{r_i}$, which intersects with the searching area. For each partition $P_z \in \mathbb{S}_{r_i}$, we get its taxi list $P_z.L_t$. In addition, by comparing $\mathbf{r_i}$'s mobility vector to all existing mobility clusters, we would find a mobility cluster C_a that shares the similar travel direction with $\mathbf{r_i}$. Thus, *mT-Share* determines the candidate taxi set $\mathbb{T}_{r_i}$ for $\mathbf{r_i}$ as

$$\mathbb{T}_{r_i} = \left\{\cup_{P_z \in \mathbb{S}_{r_i}} P_z.L_t\right\} \cap C_a.L_t \tag{3}$$

mT-Share further refines set $\mathbb{T}_{r_i}$ using the following rules: 1) including empty taxis $\{\mathbf{t_j}\}$, where $\mathbf{t_j} \in P_z.L_t$ ($P_z \in \mathbb{S}_{r_i}$) and $\mathcal{S}_{t_j} = \emptyset$; 2) filtering out the taxis with no idle capacity; and 3) filtering out the taxis, which cannot reach $\mathbf{r_i}$'s locating partition P_i before $\mathbf{r_i}$'s pick-up deadline (that can be easily checked from the taxi arrival time recorded in $P_i.L_t$). These rules can remove the invalid taxis from candidate taxi set $\mathbb{T}_{r_i}$, so as to save unnecessary computation costs.

2) Taxi Scheduling: Given set $\mathbb{T}_{r_i}$, taxi scheduling will find the most suitable taxi, which can serve ride request $\mathbf{r_i}$ while incurring the minimum detour cost. In theory, we should rearrange all events of a taxi schedule $\mathcal{S}_{t_j}$ after incorporating the pick-up event o_{r_i} and drop-off event d_{r_i} of $\mathbf{r_i}$. However, it is prohibited due to the huge computation overheads. Therefore, *mT-Share* adopts the same design choice as existing works [28]–[30], [42], which inserts o_{r_i} and d_{r_i} into $\mathcal{S}_{t_j}$ while retaining the existing schedule unchanged. The feasibility of inserting $\mathbf{r_i}$ into a schedule of taxi $\mathbf{t_j}$ is mainly decided by the time constraints of all requests that are currently served by $\mathbf{t_j}$.

Algorithm 1 presents the taxi scheduling algorithm. For each candidate taxi $\mathbf{t_j} \in \mathbb{T}_{r_i}$, we enumerate all taxi schedule instances by inserting event o_{r_i} and d_{r_i} into $\mathcal{S}_{t_j}$, where o_{r_i} is ahead of d_{r_i}. For each schedule instance $\mathcal{S}'_{t_j}$ of taxi $\mathbf{t_j}$, we will calculate a route and estimate the detour cost as

$$\text{detour cost} = \text{cost}\left(\mathcal{R}'_{t_j}\right) - \text{cost}\left(\mathcal{R}_{t_j}\right) \tag{4}$$

where $\mathcal{R}'_{t_j}$ is the updated taxi route of $\mathcal{R}_{t_j}$ once taxi $\mathbf{t_j}$ picks up request $\mathbf{r_i}$. To search the best taxi that incurs the minimum detour cost to serve $\mathbf{r_i}$, *mT-Share* needs to check all schedule instances of all candidate taxis.

Algorithm 2: Partition Filtering

```
1 Function PartitionFilter($s_z, s_{z+1}$):
2     Find partition $P_z$, $P_{z+1}$, and landmark $\ell_z$, $\ell_{z+1}$;
3     $\mathcal{P} \leftarrow \emptyset$;
4     foreach $P_i \in \mathbb{P}$ do
5         if $P_i$ satisfies the two rules then
6             $\mathcal{P} = \mathcal{P} \cup \{P_i\}$;
7     return $\mathcal{P}$;
```

Algorithm 3: Basic Routing

```
1 Function BasicRouting($\mathcal{S}$, $\mathbf{t_j}$):
2     $\mathcal{R} \leftarrow \emptyset$;
3     for $z = 1$ to $(|\mathcal{S}| - 1)$ do
4         $\mathcal{P}$ = PartitionFilter ($s_z, s_{z+1}$);
5         Build subgraph $\mathbf{G}_z$ from $\mathcal{P}$;
6         Find the shortest path $R_z$ using the Dijkstra's algorithm
            on $\mathbf{G}_z$;
7         $\mathcal{R} = \mathcal{R} \bowtie R_z$;
8     return $\mathcal{R}$;
```

To effectively serve both online and offline requests, *mT-Share* supports two routing modes. A shared taxi normally travels with the basic routing mode. When there are inadequate online requests, the taxi that has sufficient empty seats can enable the probabilistic routing mode to calculate a route to encounter suitable offline requests with a higher probability. Specifically, in Algorithm 1, if indicator *flag* is true, *mT-Share* invokes function *ProbabilisticRouting*() to compute a route for opportunistically seeking offline requests. Otherwise, function *BasicRouting*() is used to calculate the shortest path. *mT-Share* optimizes both functions with *PartitionFilter*() by pruning the searching space for fast route planning.

Because route planning usually bottlenecks the efficiency of taxi scheduling [29], [42], *mT-Share* thus optimizes both basic routing and probabilistic routing with a two-phase route planning. Given a taxi schedule instance $\mathcal{S}'_{t_j}$, *mT-Share* calculates the travel path for each consecutive event pair $(s_z, s_{z+1}) \in \mathcal{S}'_{t_j}$ through two phases, i.e., *partition filtering* that reduces the searching space of route planning and *segment-level routing* that computes the final travel path. By concatenating these travel paths of all consecutive event pairs (i.e., the operation $\bowtie$ in Algorithms 3 and 4), *mT-Share* finally derives route $\mathcal{R}'_{t_j}$ for schedule $\mathcal{S}'_{t_j}$. Next, we will detail the two phases.

Phase 1 (Partition Filtering): We execute partition filtering on landmark graph $\mathbf{G}_\ell$. For any two consecutive events $(s_z, s_{z+1}) \in \mathcal{S}'_{t_j}$, we first retrieve their locating partitions P_z and P_{z+1}, and the corresponding landmarks ℓ_z and ℓ_{z+1}. The travel cost between ℓ_z and ℓ_{z+1}, i.e., $\text{cost}(\ell_z, \ell_{z+1})$, is used to estimate the length of the shortest path between s_z and s_{z+1}. Furthermore, we generate a mobility vector $\vec{\mathbf{v}}_z$ using their two landmarks. Then, we check each map partition $P_i \in \mathbb{P}$ using the rules as follows.

- *Travel Direction Rule:* The direction difference θ between the mobility vector, which points from ℓ_z to ℓ_i, and $\vec{\mathbf{v}}_z$ is small enough, i.e., $\cos(\theta) \geq \lambda$.

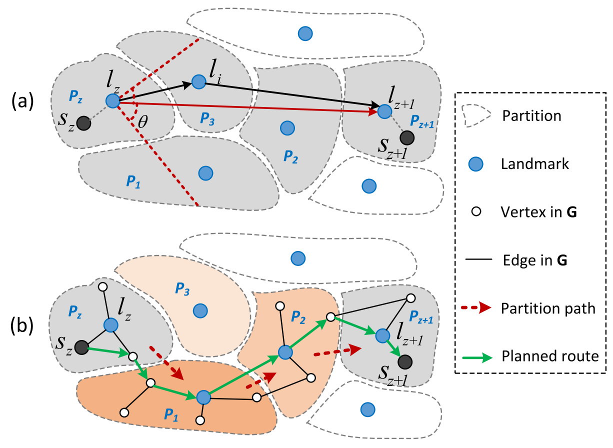


Fig. 4. (a) Illustration of partition filtering. The white partitions are filtered out, and the gray ones are retained. (b) Probabilistic routing on $\mathcal{P}'_{t_j}$, where the color of partitions indicates the probability (the darker the higher). A valid taxi route is derived from the partition path $\mathcal{H}_{t_j} = \{P_z, P_1, P_2, P_{z+1}\}$.

- *Travel Cost Rule:* The travel cost of the path, which connects ℓ_z and ℓ_{z+1} via ℓ_i, is not obviously larger than the travel cost of the shortest path between ℓ_z and ℓ_{z+1}, i.e.,

$$\text{cost}(\ell_z, \ell_i) + \text{cost}(\ell_i, \ell_{z+1}) \leq (1 + \varepsilon) \times \text{cost}(\ell_z, \ell_{z+1})$$

where ε is a predefined threshold. We conservatively set ε as 1.0 for *mT-Share*.

We retain the map partitions, which satisfy the two rules, into set $\mathcal{P}^z_{t_j}$. Algorithm 2 lists the pseudocode of partition filtering. Fig. 4(a) illustrates partition filtering applied for events (s_z, s_{z+1}), where gray partitions are retained into $\mathcal{P}^z_{t_j}$.

Phase 2 (Segment-Level Routing): Instead of planning a route on the original graph **G**, *mT-Share* builds a subgraph that consists of the vertices and edges from partitions in $\mathcal{P}^z_{t_j}$. The subgraph greatly reduces the searching space of route planning, and thus *mT-Share* will execute both basic routing and probabilistic routing on this subgraph for better efficiency. We detail the two routing modes as follows.

- *Basic Routing:* For any two consecutive events in a taxi schedule instance $\mathcal{S}'_{t_j}$, it will calculate the shortest path. In fact, the basic routing mode has been frequently adopted by existing works for ridesharing routing [28]–[30], [42]. Similarly, *mT-Share* utilizes *Dijkstra*'s algorithm [14] to compute the shortest path on the subgraph. Algorithm 3 sketches the pseudocode of the basic routing mode.
- *Probabilistic Routing:* This mode supports a shared taxi to opportunistically meet some suitable offline ride requests. We say a ride request $\mathbf{r_i}$ is *suitable* only if $\mathbf{r_i}$ travels with the similar direction as the given taxi. Instead of predicting the specified number of offline ride requests, we propose to plan a probabilistic route to maximize the probability of meeting suitable offline ride requests for a given taxi. In principle, we should calculate the probability of meeting suitable requests over each vertex in graph **G** and search for the route that can accumulate the maximum probability of meeting suitable ride requests. Such a probabilistic routing has been proved to be NP-Complete [23] and thus is computationally prohibitive.

Instead, *mT-Share* proposes a heuristic approach to calculate probabilistic routes to avoid huge computations. Specifically, for any two consecutive events (s_z, s_{z+1}) in schedule $\mathcal{S}'_{t_j}$, *mT-Share* exploits transition patterns among partitions to further refine the set $\mathcal{P}^z_{t_j}$. Then, *mT-Share* builds another much smaller subgraph from the refined partition set for probabilistic route planning. We present the pseudocode of probabilistic routing in Algorithm 4, and detail its main steps as follows.

① *Probability Calculation of Suitable Requests:* For each partition $P_i \in \mathcal{P}^z_{t_j}$ and a given candidate taxi $\mathbf{t_j}$, we determine a destination partition set $\mathcal{P}_d$ for the suitable requests. For each partition $P_a \in \mathbb{P}$, we build a mobility vector $\vec{\mathbf{v}}_a$ using the landmarks of P_i and P_a. If the travel direction difference θ between $\vec{\mathbf{v}}_a$ and taxi $\mathbf{t_j}$ is sufficiently small (i.e., $\cos(\theta) \geq \lambda$), we will retain partition P_a into set $\mathcal{P}_d$. After obtaining set $\mathcal{P}_d$, we compute a probability π_i of meeting suitable requests within P_i by summing the transition probability of each vertex in P_i to each potential destination in set $\mathcal{P}_d$. During bipartite map partitioning, we have already calculated the transition probabilities of any vertex to all partitions (see Section IV-B1). Thus, these calculation results can be reused.

② *Partition Path Planning:* We build a landmark graph $\mathbf{G}^z_\ell$ using the landmarks and edges of partitions in $\mathcal{P}^z_{t_j}$, where the landmark vertex of partition P_i is associated with probability π_i as the weight. In general, graph $\mathbf{G}^z_\ell$ would be small, we thus is able to enumerate all possible paths, each of which links the landmark of source partition and the landmark of destination partition, to find the maximum weighted path. With landmark vertices of the found path, we can retrieve their corresponding partitions to form a partition path $\mathcal{H}^z_{t_j}$, which travels from P_z to P_{z+1}. At the same time, $\mathcal{H}^z_{t_j}$ accumulates the maximum probability at the partition level.

③ *Fine-Grained Route Planning Over Partition Path:* We construct another weighted graph $\mathbf{G}_z$ using the vertices and edges of partitions in $\mathcal{H}^z_{t_j}$, where each vertex v_c is associated with a weight $(1/\psi_c)$ $(\psi_c > 0)$. Specifically, ψ_c is the accumulated transition probability from vertex v_c to the destination partition set $\mathcal{P}_d$ of v_c's locating partition. We calculate the shortest path with *Dijkstra*'s algorithm [14] on graph $\mathbf{G}_z$. The derived path should have the minimum weights and meanwhile satisfy the deadline constraints of passengers who share taxi $\mathbf{t_j}$. This path is the final taxi route R_z that has the highest probability to meet suitable offline requests between (s_z, s_{z+1}).

If no valid taxi route is found in step ③, the suboptimal partition path on $\mathcal{P}^z_{t_j}$ in step ② is returned. *mT-Share* will repeat the latter two steps until a valid taxi route (i.e., meeting requests' delivery deadlines) for event s_z and s_{z+1} is found. To avoid endless searching, we set the attempt times as 5. Otherwise, the taxi schedule instance is discarded when there exists no feasible route to link the two events. Finally, *mT-Share* concatenates all the paths of all consecutive event pairs

Algorithm 4: Probabilistic Routing

```
1  Function ProbabilisticRouting($\mathcal{S}$, $\mathbf{t_j}$):
2      $\mathcal{R} \leftarrow \emptyset$;
3      for $z = 1$ to $(|\mathcal{S}| - 1)$ do
4          $\mathcal{P}$ = PartitionFilter $(s_z, s_{z+1})$;
5          Calculate probability $\pi_i$ of meeting suitable offline
             requests for partition $P_i \in \mathcal{P}$;                ▷ step ①
6          Build weighted landmark subgraph $\mathbf{G}_\ell^z$ from $\mathcal{P}$;
7          attempt = 0;
8          while true do
9              Select the maximum weighted path from $\ell_z$ to
                 $\ell_{z+1}$ on $\mathbf{G}_\ell^z$ to form the partition path $\mathcal{H}$;
                 ▷ step ②
10             Build weighted subgraph $\mathbf{G}_z$ from $\mathcal{H}$;
11             Find the shortest path $R_z$ using the Dijkstra's
                 algorithm on $\mathbf{G}_z$;                              ▷ step ③
12             attempt = attempt + 1;
13             if $R_z$ is valid then
14                 break;
15             else if attempt > 5 then
16                 return $\emptyset$;
17         $\mathcal{R} = \mathcal{R} \bowtie R_z$;
18     return $\mathcal{R}$;
```

to derive the final taxi route. Fig. 4(b) illustrates the partition path and the final path for two consecutive events.

Since probabilistic routing is more computationally expensive than the basic routing, a shared taxi $\mathbf{t_j}$ may enable it only when $\mathbf{t_j}$ has sufficient empty seats and there are inadequate online requests. When taxi $\mathbf{t_j}$ encounters an offline ride request $\bar{\mathbf{r}}_\mathbf{i}$, we envision that the taxi driver can report $\bar{\mathbf{r}}_\mathbf{i}$ to the server through some App. Then, the server will investigate $\mathbf{t_j}$'s schedule and route. Taxi $\mathbf{t_j}$ will serve $\bar{\mathbf{r}}_\mathbf{i}$ only if there exists a valid schedule that guarantees the service quality for all requests (including offline request $\bar{\mathbf{r}}_\mathbf{i}$). Otherwise, the server will quickly dispatch another taxi to serve $\bar{\mathbf{r}}_\mathbf{i}$. The interaction indeed may introduce a slight delay, however, it brings potential benefits for both offline passengers and taxi drivers. Currently, *mT-Share* plans a probabilistic route to maximize the probability of meeting offline requests given the requests' delivery deadlines. How to balance the trade-off between this probability and the total detour costs will be explored in our future work.

Time Complexity: Now, we analyze the time complexity of *mT-Share*'s passenger–taxi matching, which involves the four algorithms. To process each request $\mathbf{r_i}$, *mT-Share* needs to investigate all possible schedule instances of each candidate taxi in set $\mathbb{T}_{r_i}$ and then chooses the taxi that will introduce the minimum detour cost to serve $\mathbf{r_i}$. If *mT-Share* adopts basic routing, the time complexity is $\mathcal{O}(|\mathbb{T}_{r_i}|m^3\kappa)$, where $|\mathbb{T}_{r_i}|$ is the size of candidate taxi set $\mathbb{T}_{r_i}$, m is the number of events in a taxi schedule, and κ is the total number of map partitions in $\mathbb{P}$. If *mT-Share* enables the probabilistic routing, the time complexity will be $\mathcal{O}(|\mathbb{T}_{r_i}|m^3ND|\mathcal{P}|/\kappa)$, where D is the amount of historical taxi data for calculating transition probabilities, $|\mathcal{P}|$ is the number of partitions retained by Algorithm 2, and N/κ is the average vertex number of all partitions. Similar as previous studies [12], [42], we assume the shortest path query will take $\mathcal{O}(1)$ time, because the shortest paths between any two graph vertices could be prepared and cached.

D. Payment Model

A ridesharing system should bring financial benefits for both ridesharing passengers and taxi drivers, so as to attract more participators. Some payment models have been proposed for the carpooling services [50]. Their involved requests are static and the fares are usually precalculated. Thus, they cannot work well for dynamic taxi ridesharing. There indeed exist some payment models proposed for dynamic ridesharing [5], [6], [9], [11], [23], [30], [51]. The models in [9], [11], and [23] calculate ridesharing fare for each individual passenger with a fixed discounting rate, and do not explicitly consider the benefits for drivers. Although [5], [6], [29], [30], and [51] consider both passengers and drivers, their models heavily rely on users' profiles [6] or the complex auction theory [5], [51], and thus cannot be sufficiently flexible and scalable for taxi ridesharing that needs to handle a large number of drivers and passengers. The payment model in [29] even may require passengers to pay more if they detour more distances [5], and thus is not fair. Thus, it is necessary to design a payment model that can fairly share ridesharing benefits between taxi drivers and passengers.

The benefits of taxi ridesharing mainly come from the saved travel distance of a ridesharing route when compared to no ridesharing that delivers passengers separately along different routes. Specifically, the benefit $\mathcal{B}$ for taxi ridesharing with n shared passengers is

$$\mathcal{B} = \sum_{i=1}^{n} f_{r_i}^s - \mathcal{F} \tag{5}$$

where $f_{r_i}^s$ is the regular taxi fare with no ridesharing for ride request $\mathbf{r_i}$, and $\mathcal{F}$ is the regular taxi fare for a distance equaling to the length of ridesharing route. The total fare of n passengers without ridesharing is given by $\sum_{i=1}^{n} f_{r_i}^s$, while the taxi ridesharing fare is $\mathcal{F}$. Their difference is thus the ridesharing benefit $\mathcal{B}$. *mT-Share* partitions the benefit between the taxi driver and all passengers (as a group) with a rate β. A driver will obtain benefit $(1-\beta)\times\mathcal{B}$, while all ridesharing passengers share the benefit $\beta \times \mathcal{B}$.

mT-Share splits the benefit among passengers proportionally according to their *detour rates*. Specifically, the detour rate is defined as the ratio between the detour distance and the shortest path length. For ride request $\mathbf{r_i}$ that arrives at the destination, the detour rate is

$$\sigma_i = \eta + \frac{\text{cost}(\mathcal{R}_{r_i}) - \text{cost}(\mathcal{R}_{r_i}^s)}{\text{cost}(\mathcal{R}_{r_i}^s)} \tag{6}$$

where $\mathcal{R}_{r_i}$ is the shared route $\mathbf{r_i}$ has traveled, $\mathcal{R}_{r_i}^s$ is the shortest path for request $\mathbf{r_i}$, and η is the base rate. Note that base rate η is introduced to guarantee that all passengers can gain benefits from the ridesharing even they do not take any detour, e.g., all ride requests have the same pick-up and drop-off locations. For a ride request $\mathbf{r_j}$ that has not been completed yet, the detour rate is assumed as

$$\sigma_j = \eta + \frac{\mathrm{cost}\left(\mathcal{R}_{r_j}\right) + \mathrm{cost}\left(\mathcal{R}^s_{\left(d_{r_i}, d_{r_j}\right)}\right) - \mathrm{cost}\left(\mathcal{R}^s_{r_j}\right)}{\mathrm{cost}\left(\mathcal{R}^s_{r_j}\right)} \quad (7)$$

where $\mathcal{R}_{r_j}$ is the shared route $\mathbf{r_j}$ has already traveled, and we assume the taxi will deliver $\mathbf{r_j}$ with the shortest path for the remaining trip, i.e., $\mathcal{R}^s_{(d_{r_i}, d_{r_j})}$ denotes the shortest path from $\mathbf{r_i}$'s destination to $\mathbf{r_j}$'s destination.

Therefore, for ride request $\mathbf{r_i}$ that arrives at the destination now, the taxi ridesharing benefit will be $\beta \times \mathcal{B} \times (\sigma_i / [\sum_{z=1}^{n} \sigma_z])$, and thus the taxi ridesharing fare is

$$f_{r_i} = f^s_{r_i} - \beta \times \mathcal{B} \times \frac{\sigma_i}{\sum_{z=1}^{n} \sigma_z}. \quad (8)$$

From this payment model, we see a passenger will not pay more than the regular taxi service and a taxi driver can earn more from the ridesharing. In addition, passengers with relatively large detour rates can receive more compensations. Therefore, the payment model will largely encourage more passengers and taxi drivers to participate in taxi ridesharing.

V. Performance Evaluation

A. Experimental Setup

1) Data Set: We perform extensive experiments with a large taxi data set that is publicly released by Didi's GAIA initiative [1]. The data set totally has 7 065 907 taxi transactions, which were collected within the 2nd Ring Road from Chengdu city, China, in 2016. Specifically, each transaction includes a transaction ID, a taxi ID, and a ride request, which consists of the release time, the pick-up latitude/longitude, and the drop-off latitude/longitude. We employ the data from two specific time periods to simulate two representative ridesharing scenarios.

- *Peak Scenario:* In general, taxis have to handle a large number of online requests during the peak hours. Thus, the offline requests are ignored in this scenario. The data from 8:00 A.M. to 9:00 A.M. of a typical workday that has the most hourly requests, i.e., 29 534, is used to evaluate *mT-Share* in the peak hours.
- *Nonpeak Scenario:* During nonpeak hours, most taxis have sufficient idle capacity while there exist inadequate online requests. Thus, taxi drivers can exploit probabilistic routing to seek offline passengers. Here, we assume a taxi with half of the capacity in idle will enable the probabilistic routing. The data from 10:00 A.M. to 11:00 A.M. of a typical weekend, i.e., 15 480 ride requests, is used to evaluate *mT-Share* in the nonpeak hours. Besides, we randomly pick 5000 requests out of all and make their release time and origin/destination to be invisible to the system. These requests are viewed as the offline requests.

We use the rest taxi data for bipartite map partitioning and probability calculations of meeting suitable requests. Statistics about the taxi data are shown in Fig. 5. Fig. 5(a) presents the average taxi utilization in workdays and weekends. The taxi utilization is defined as the proportion of

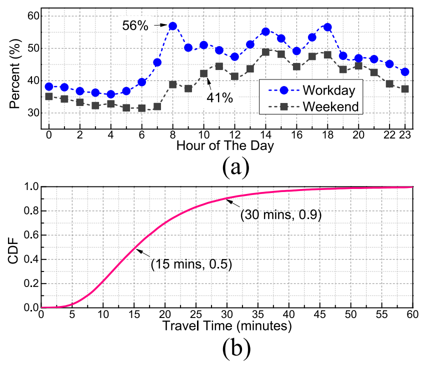


Fig. 5. Statistic of the taxi data set on (a) average taxi utilization ratio and (b) travel time distribution of taxi trips.

serving requests within each hour. From Fig. 5(a), we see that the utilization ratios of 8:00 A.M.–9:00 A.M. in workdays and 10:00 A.M.–11:00 A.M. in weekends are 56% and 41%, respectively. Fig. 5(b) further shows the travel time distribution of all taxi trips in the data set, e.g., the 90-percentile and 50-percentile trip travel time are 30 and 15 min, respectively.

We download road network data within the 2nd Ring Road of Chengdu city from OpenStreetMap [2] and model the road network as a directed graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$, which contains 214 440 vertices and 466 330 edges in total, covering an area of more than 70 km^2. The testing road network is shown in Fig. 3(a).

2) Compared Schemes: We compared *mT-Share* against the following four baseline schemes.

- *No-Sharing* operates as the regular taxi service with no ridesharing at all. It assigns a ride request to the geographically nearest idle taxi within the searching range γ.
- *T-Share* [29], [30], one of the state-of-the-art schemes, indexes all requests and taxis using grids and selects candidate taxis by conducting a dual-side search from both origin and destination of a request with the searching range γ. However, it only returns the first valid candidate rather than the best one.
- *pGreedyDP* [42], one of the state-of-the-art schemes, indexes all requests and taxis using grids like *T-Share*, and selects candidate taxis within the searching range γ around the request $\mathbf{r_i}$'s origin. To improve the taxi scheduling efficiency, it determines the event insertions of $\mathbf{r_i}$ into an existing schedule through dynamic programming.
- *mT-Sharepro* is the version of *mT-Share* with enabled probabilistic routing. Since it will introduce huge computation costs, *mT-Sharepro* is only evaluated in the *nonpeak scenario*. It is desirable for taxi drivers to serve offline requests during such periods that have inadequate online requests [10], [48].

For fair comparisons, we adjust the settings of *T-Share* and *pGreedyDP*, and enable them to serve offline requests as well. Along the taxi route provided by *T-Share* or *pGreedyDP*, if a taxi $\mathbf{t_j}$ that has sufficient empty seats happens to meet suitable offline requests $\bar{\mathbf{r_i}}$, which can be validly inserted into $\mathbf{t_j}$'s schedule, then taxi $\mathbf{t_j}$ could serve request $\bar{\mathbf{r}}_\mathbf{i}$. Similarly, they serve offline ride requests only in *nonpeak scenario* as well.

3) Performance Metrics: We evaluate all the schemes on the following performance metrics.

- *Number of served requests* is the number of ride requests, which have been timely served.
- *Response time* is the processing time for a ridesharing scheme to match a suitable taxi with the request.
- *Detour time* represents the extra travel time when compared to the travel time with no ridesharing for a request.
- *Waiting time* is measured as the time difference between the time a shared taxi picks up the passengers and the time passengers released the request.

4) Implementation: We implemented *mT-Share* and all compared schemes in Python. Similar as previous works [11], [12], [42], we premap each ride request $\mathbf{r_i}$'s origin and destination to the closest vertex in road network graph $\mathbf{G}$, respectively. We set the delivery deadline e_{r_i} of request $\mathbf{r_i}$ with the flexible factor ρ. This factor indicates the extra travel cost that can be tolerable by passengers when compared to the shortest path [12]. Given the release time t_{r_i} and travel cost $\text{cost}(o_{r_i}, d_{r_i})$ of request $\mathbf{r_i}$, the delivery deadline of $\mathbf{r_i}$ is thus expressed as

$$e_{r_i} = t_{r_i} + \text{cost}\big(o_{r_i}, d_{r_i}\big) \times \rho. \quad (9)$$

We set the initial location of each taxi as a random vertex from graph $\mathbf{G}$. When a taxi $\mathbf{t_j}$ is delivering passengers, it should abide by the established taxi schedule $\mathcal{S}_{t_j}$ and taxi route $\mathcal{R}_{t_j}$. As with previous studies [11], [23], [55], we assume that all taxis drive with a constant speed as 15 km/h for simplicity. By default, we fix the searching range $\gamma = 2.50$ km that is equivalent to a waiting time of $\Delta t = 10$ min. In addition, we fix $\beta = 0.80$ and $\eta = 0.01$ for the payment model.

We conduct the experimental evaluations on a server with Intel Core i7-6700 CPU@3.40 GHz and 16-GB memory. Each experimental setting will be repeated ten times and we only report the average results. To speedup the route planning, we precalculate the shortest paths for any two vertices in graph $\mathbf{G}$ and the travel cost for any two landmarks in graph $\mathbf{G}_\ell$, and cache these results in the memory [19], [55] for quick retrieves by all schemes. We summarize the parameter settings in Table II, where the default values are marked in bold.

TABLE II
SETTING OF MAJOR PARAMETERS

Parameter	Setting
Number of taxis	500, 1000, 1500, **2000**, 2500, 3000
Number of partitions κ	50, 100, **150**, 200, 250
Taxi capacity	2, **3**, 4, 5, 6
Flexible factor ρ	1.1, 1.2, **1.3**, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0
Threshold λ	$0.867, \mathbf{0.707}, 0.500, 0.259$, (*i.e.*, max $\theta = 30°, \mathbf{45°}, 60°, 75°$)
Taxi searching range γ	$1.25\,km$, $\mathbf{2.50\,km}$, $3.75\,km$, $5.00\,km$

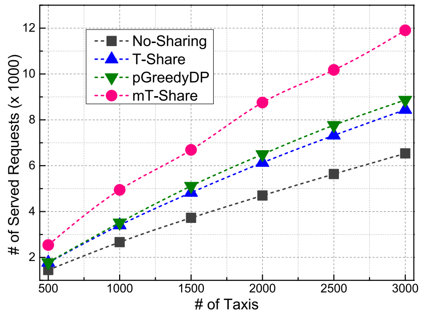


Fig. 6. Served requests in *peak scenario*.

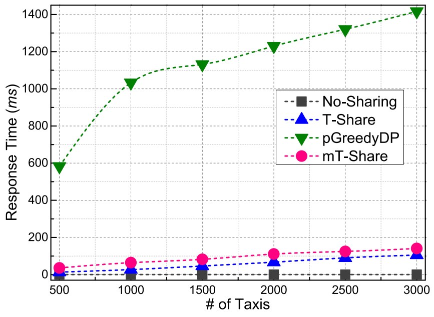


Fig. 7. Response time in *peak scenario*.

B. Performance Comparison

We compared *mT-Share* with baseline schemes in both *peak scenario* and *nonpeak scenario* by varying the number of taxis from 500 to 3000, with a step of 500, similar as [29] and [42].

1) Comparisons in the Peak Scenario: Fig. 6 shows that the four schemes can serve more requests when there are more available taxis. With *No-Sharing*, a taxi can accomplish about two ride requests on average within one peak hour. Compared to *No-Sharing*, taxi ridesharing can indeed serve much more requests. From Fig. 6, we find that *pGreedyDP* outperforms *T-Share* because it has optimized the ridesharing routing and thus is able to find better passenger–taxi matches. Among all the schemes, *mT-Share* serves the most requests since *mT-Share* has optimized both candidate taxi searching and passenger–taxi matching by efficiently exploiting the mobility information. Taking the case of 3000 taxis as an example, *No-Sharing*, *T-Share*, *pGreedyDP*, and *mT-Share* have served 6534, 8441, 8868, and 11 906 ride requests, respectively. Compared to *T-Share* and *pGreedyDP*, i.e., the state-of-the-art works, *mT-Share* can serve 42% and 36% more ride requests, respectively.

When there are more available taxis, the response time of all schemes increases, as shown in Fig. 7. *No-Sharing* can respond a request within 1 ms. *mT-Share* will take a bit more time to process a request than *T-Share*, and *pGreedyDP* has the largest response time. It is possibly because *pGreedyDP* spends much time to determine the low bound detour cost for each candidate taxi. In general, *mT-Share* responds a ride request within 35–140 ms and outperforms *pGreedyDP* by 4–10 times on the metric of response time.

To better understand the results in Figs. 6 and 7, we present the average numbers of candidate taxis for a request for different schemes in Table III. With the same searching range γ, *No-Sharing* has the smallest candidate taxi set since it only considers vacant taxis. *T-Share* has much fewer candidate taxis than *pGreedyDP* and *mT-Share* because its

TABLE III
AVERAGE NUMBER OF CANDIDATE TAXIS IN *Peak Scenario*

# of Taxis	*No-Sharing*	*T-Share*	*pGreedyDP*	*mT-Share*
500	4	5	91	86
1000	6	14	179	170
1500	9	32	266	220
2000	12	53	352	327
2500	15	79	439	375
3000	17	110	527	487

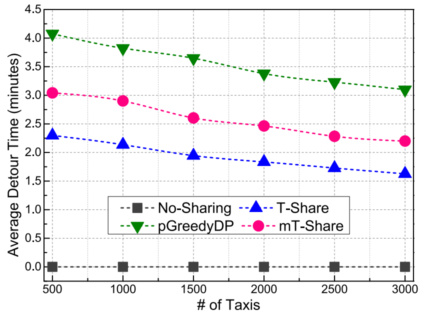


Fig. 8. Detour time in *peak scenario*.

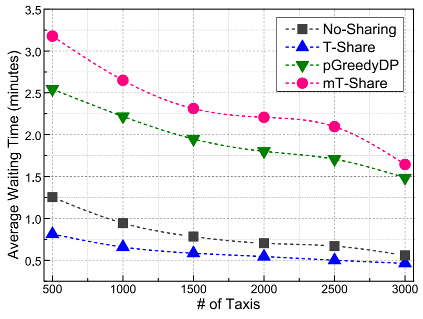


Fig. 9. Waiting time in *peak scenario*.

dual-side search mistakenly removes many possible taxis [42]. *pGreedyDP* has the most candidate taxis among the four schemes. According to the candidate taxi searching strategy in Section IV-C, *mT-Share* will aggressively consider all possible candidate taxis in the partitions that intersect with the searching range, and filter out invalid candidates by comparing the travel directions of a taxi and the request. As a result, *mT-Share* can initially remove many invalid candidate taxis while preserving possible ones. These are the reasons why *mT-Share* can respond each request quickly while serving the most requests.

Fig. 8 reports the detour time. *No-Sharing* introduces no detour, while the ridesharing schemes have detour time of 1–4 min. More taxis potentially allow all ridesharing schemes to search a more suitable taxi for a request, which thus reduces the detour time. In general, *T-Share* has the minimum detour time, while *mT-Share* is quite close to *T-Share* and holds the second place. However, *pGreedyDP* nearly doubles the detour time of *T-Share*. Compared to *pGreedyDP*, *mT-Share* improves the average detour time by 31%–40%.

We study the waiting time of all schemes and report the results in Fig. 9. In general, more taxis potentially allow each scheme to find a nearby taxi to serve each request, and thus

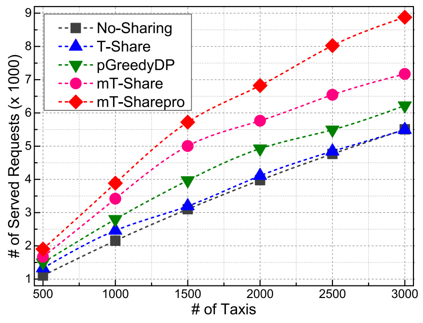


Fig. 10. Served requests in *nonpeak scenario*.

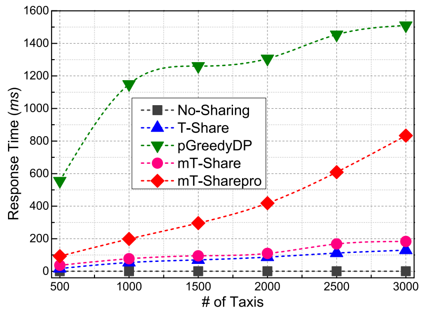


Fig. 11. Response time in *nonpeak scenario*.

the waiting time can be reduced. With no ridesharing, *No-Sharing* has the minimum taxi supplies among all schemes given the same number of taxis. Thus, it has a relatively larger waiting time around 1 min. *T-Share* has the smallest waiting time since it usually returns the nearest vacant taxi. Both *mT-Share* and *pGreedyDP* try to maximize the number of served requests and minimize the total detour costs, they thus have large waiting time. *mT-Share* has a bit longer waiting time than *pGreedyDP*, while the gap is quite small, i.e., < 0.5 min.

2) Comparisons in the Nonpeak Scenario: The number of served requests in the *nonpeak scenario* is shown in Fig. 10. By comparing with Fig. 6, we observe that the advantage of ridesharing over *No-Sharing* is diminishing. This is possible because there are much fewer requests in the nonpeak hours. We even see that *T-Share* has similar performances as *No-Sharing* on the number of served requests in some settings. *mT-Share* and *mT-Sharepro* still serve much more requests than *T-Share* and *pGreedyDP*. The probabilistic routing indeed helps *mT-Sharepro* to serve more requests, e.g., improving *mT-Share* by 13%–24%. Compared to *T-Share* and *pGreedyDP*, i.e., state-of-the-art schemes, *mT-Sharepro* can serve 62% and 58% more requests, respectively.

Fig. 11 shows the response time of all schemes in the *nonpeak scenario*. Comparing to their results in Fig. 7, we find *No-Sharing*, *T-Share*, *pGreedyDP*, and *mT-Share* have quite similar performances both scenarios. As probabilistic routing involves huge computation costs for finding the route with the highest probability of meeting suitable requests, the response time of *mT-Sharepro* is thus much greater than *mT-Share*, i.e., 2.5–4.5 times slower. The performance gap between *mT-Sharepro* and *pGreedyDP* becomes smaller when

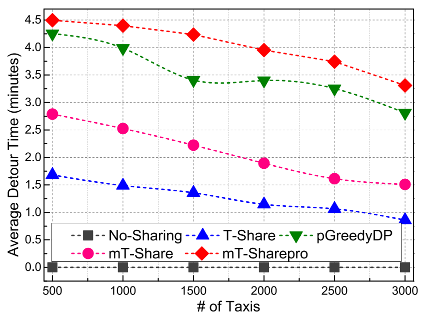


Fig. 12. Detour time in *nonpeak scenario*.

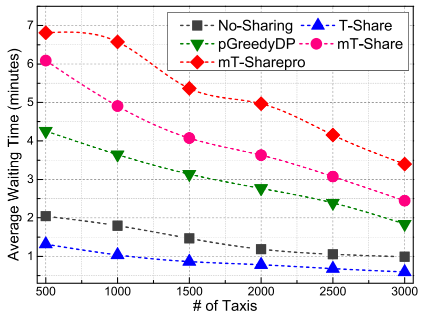


Fig. 13. Waiting time in *nonpeak scenario*.

there are more taxis. However, *mT-Sharepro* still responds a ride request much faster than *pGreedyDP* with 81%–497% performance gain. Figs. 7 and 11 also demonstrate that our scheme has a good scalability and can respond each request quickly.

Since the route planning algorithms of *No-Sharing*, *T-Share*, *pGreedyDP*, and *mT-Share* are the same in both scenarios, their detour time results in the *nonpeak scenario* as shown in Fig. 12 are quite similar with the results in the *peak scenario* in Fig. 8. Because probabilistic routing may return some long taxi routes to encounter offline passengers by chance, *mT-Sharepro* has the largest detour time among all schemes. Even so, the detour time difference between *mT-Sharepro* and *pGreedyDP* is still small, i.e., $\leq$ 0.5 min. It implies that *mT-Sharepro* greatly improves *pGreedyDP* at a minor cost.

We present the waiting time of all schemes in Fig. 13. We observe an obvious decrease in the trend of waiting time when the number of available taxis is increased. Compared to the results in Fig. 9, the waiting times of the five schemes in the *nonpeak scenario* become larger. This is because we have fewer requests in the nonpeak hours, and a taxi usually needs to travel more distances to pick up the passengers. Because of the probabilistic routing, *mT-Sharepro* has the largest waiting time, e.g., 2 min longer than *pGreedyDP*.

3) Memory Overhead: The ridesharing schemes usually index both taxis and ride requests to accelerate the passenger–taxi matching, which consumes system memory. We thus compared their memory overheads with 3000 taxis in the *peak scenario*, which indicates the upper bound of memory costs. It is worth noting that *mT-Share* and *mT-Sharepro* have the same memory costs. Table IV presents the statistic results. Compared to *T-Share* and *pGreedyDP*, *mT-Share*

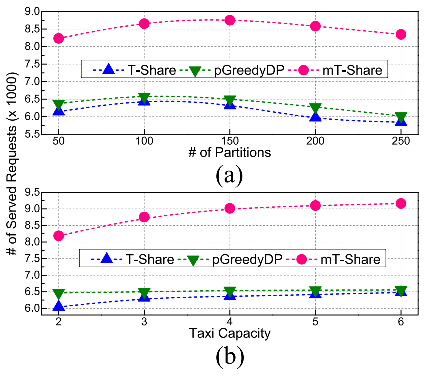


Fig. 14. Impact of the (a) number of partitions κ and (b) taxi capacity in *peak scenario*.

TABLE IV
STATISTICS OF MEMORY OVERHEAD FOR RIDESHARING SCHEMES

Metric	*T-Share*	*pGreedyDP*	*mT-Share*
Index Size (*Bytes*)	34404	34617	48018
Overall Memory (*MB*)	381	314	442

builds indexes with both map partitions and mobility clusters, and thus *mT-Share* has about 39.5% and 38.7% larger indexes, respectively. As a result, *mT-Share* consumes 16.0% and 40.7% more memories than *T-Share* and *pGreedyDP*, respectively. Fortunately, such memory overheads are negligible since current servers are equipped with sufficient memory.

C. Detailed Evaluation

Next, we will conduct experiments to explore the impacts of important parameters and alternative designs for *mT-Share*.

1) Impact of Partition Number κ: In this experiment, we vary the partition number κ while keeping other settings in *peak scenario*. As shown in Fig. 14(a), for the three schemes, the number of served requests increases at the beginning and then decreases beyond $\kappa = 150$. The κ values, either too small or oversize, will lead to fewer candidate taxis, and thus influence ridesharing performances. When κ is varied from 50 to 150, the average size of candidate taxi sets increases by 17% and meanwhile the number of served requests increases from 8234 to 8753. In contrary, more partitions lead to a shrinking candidate taxi set, and the served ride requests are reduced.

2) Impact of Capacity: Except varying the taxi capacity, we conduct this experiment using default settings as well in the *peak scenario*. When the taxi capacity is enlarged, the same number of available taxis will have much more supplies and thus they could serve more ride requests. As shown in Fig. 14(b), the larger taxi capacity brings more served ride requests. Specifically, compared to capacity $= 2$, *mT-Share* serves 12% more requests when capacity $= 6$.

3) Impact of Map Partitioning Strategies: Rather than using grids on the road network graph as previous works [29], [30], [39], [42], *mT-Share* instead adopts a bipartite map partitioning strategy, which considers both geographical information and mobility patterns. We compare the performances of *mT-Share* with different partitioning strategies in both scenarios with the default settings, and present the results

TABLE V
COMPARISONS OF DIFFERENT MAP PARTITION STRATEGIES

Scenario	Metric	Grid	Bipartite
peak scenario	# of served requests	8200	8753
	Detour time ($minutes$)	2.46	2.39
non-peak scenario	# of served requests	6317	6687
	Detour time ($minutes$)	2.02	1.88

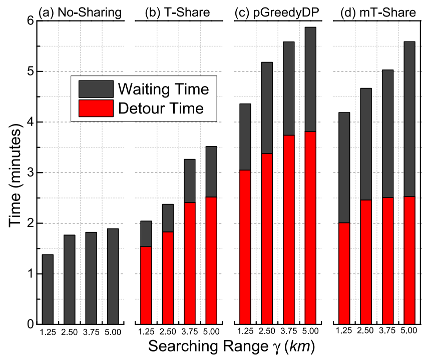


Fig. 15. Impact of searching range γ on detour time and waiting time in *peak scenario*.

in Table V. We see bipartite map partitioning indeed improves the performances in both scenarios. Specifically, bipartite map partitioning improves the number of served requests by 6% at least, while reducing the detour time by 3%–7%. The results prove that mobility-aware map partitioning can benefit ridesharing.

4) Impact of Searching Range γ: All schemes determine a candidate taxi set for each request with a searching range γ, and we study its impacts on the detour time and waiting time in the *peak scenario*. Fig. 15 shows that a larger searching range γ usually leads to more both detour time and waiting time. *No-Sharing* has no detour at all. Typically, ridesharing schemes will find more candidate taxis in a larger searching range, and the selected taxi may be farther from the requests and meanwhile has more detour cost. The sum of detour time and waiting time can implicitly represent the service quality of a taxi ridesharing scheme, where a larger value indicates that the passengers need to spend more extra time for the taxi trip. Fig. 15 reports that *T-Share* wins the best service quality and *mT-Share* has better service quality than *pGreedyDP*.

5) Impact of Routing Schemes: We study whether probabilistic routing benefits for finding offline requests. In this experiment, we combine basic routing or probabilistic routing with *T-Share*, *pGreedyDP*, or *mT-Share*, and then run each combinatorial scheme in the *nonpeak scenario* with default settings. Fig. 16 illustrates the compositions of served requests for different combinations. From Fig. 16(a), although the basic routing-based schemes may encounter a few offline passengers by chance, probabilistic routing indeed enlarges the probability by serving more offline requests, as shown in Fig. 16(b). By comparing Fig. 16(a) and (b), we find that probabilistic routing brings 89%, 46%, and 34% more offline requests for *T-Share*, *pGreedyDP*, and *mT-Share*, respectively. By exploiting the mobility patterns for effective route planning, overall they serve 26%, 17%, and 14% more requests, respectively.

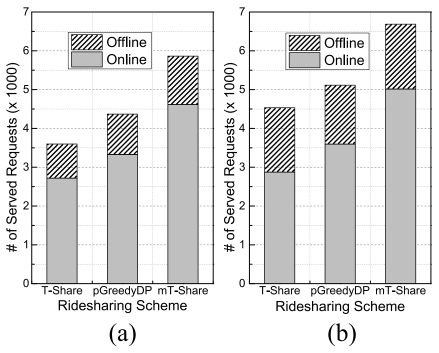


Fig. 16. Impact of (a) basic routing and (b) probabilistic routing in *nonpeak scenario*.

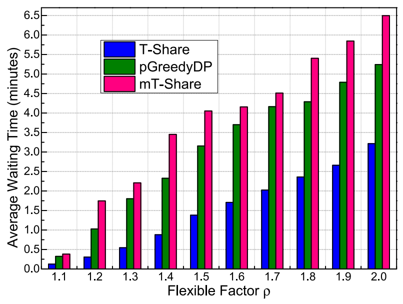


Fig. 17. Impact of ρ on the average waiting time of passengers.

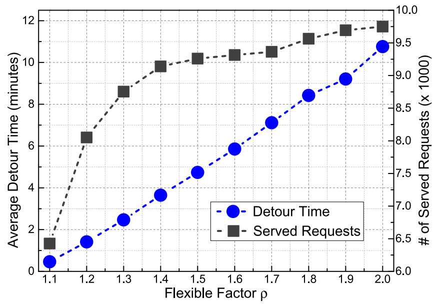


Fig. 18. Impact of ρ on average detour time and number of served requests.

6) Impact of Flexible Factor ρ: We perform experiments to explore the impact of the flexible factor ρ in the *peak scenario* with default settings. First, we study the impact of ρ on the passengers' waiting time and plot the results in Fig. 17. As ρ does not influence *No-Sharing*, we thus omit it. A larger ρ essentially indicates that passengers can tolerant more detour time, and thus much farther taxi may be selected to serve them, resulting in longer waiting time. Generally, *T-Share* has the shortest waiting time, and *mT-Share* has relatively longer waiting time. The performance gap between *pGreedyDP* and *mT-Share* is quite small, i.e., < 1.2 min.

Fig. 18 shows that when we increase ρ, the detour time also increases. The shared taxis can serve more requests with a larger ρ, but the number of served requests slightly increases beyond $\rho = 1.3$. It implies that more detour time brings about negligible benefit when we choose even larger ρ. For example, the number of served requests and detour time are 8753 and 2.5 min, respectively, when $\rho = 1.3$. The numbers increase to 9140 and 3.6 min, respectively, when $\rho = 1.4$. It means that

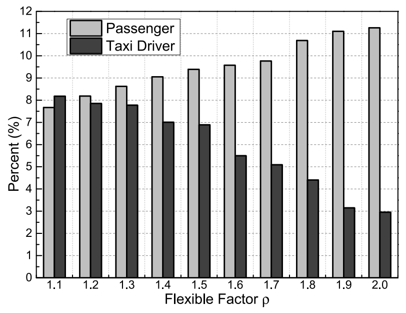


Fig. 19. Impact of ρ on the fare reduction of passengers and the profit increase of drivers.

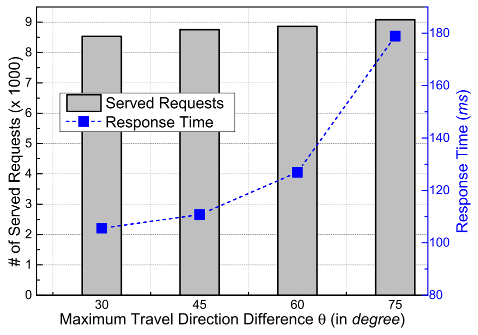


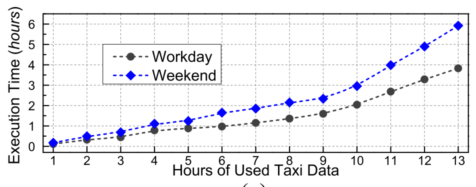
Fig. 20. Impact of maximum travel direction difference θ.

4% improvement on served requests comes at the expense of 48% increase of detour time.

We study the monetary benefits of ridesharing for both passengers and taxi drivers based on our payment model. Fig. 19 shows that taxi ridesharing indeed has economic advantages. Specifically, a larger flexible factor ρ will save more fares for ridesharing passengers while the profit for taxi drivers is decreasing. This is because larger ρ cannot lead to a remarkable increase of served requests (as reported in Fig. 18) while the travel distance grows greatly. As a result, the benefit is reduced for drivers. Specifically, the passengers can save 8.6% taxi fare while the drivers can obtain 7.8% more incomes when $\rho = 1.3$, which is a good setting for both sides.

7) Impact of Threshold λ*:* We study the impact of parameter λ in mobility clustering by varying the maximum travel direction difference θ from 30° to 75°. The experiment is performed in the *peak scenario* with default settings. Fig. 20 shows that when we increase θ (i.e., decreasing λ), the number of served requests increases slightly while the response time increases greatly. This is because a smaller λ brings more candidate taxis for each request and thus enhances the ridesharing chances. However, it also introduces huge computation costs to examine the possible taxi schedules. Therefore, we adopt $\lambda = 0.707$ (i.e., $\theta = 45°$) to balance the number of served requests and the response time for a better system performance.

8) Impact of Used Data Amounts: To investigate the scalability of our system, we make use of the taxi data during 7:00 A.M.–20:00 P.M. of one typical workday and one typical weekend for experiments. We run *mT-Share* and *mT-Sharepro* for workday data and weekend data, respectively. For each



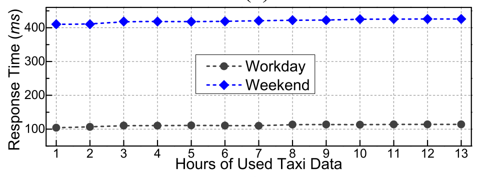


Fig. 21. Impact of the used data amounts on (a) execution time and (b) response time.

setting of experiments on weekend data, we assume one third of all requests as the offline. We gradually increase the hours of used taxi data and present the results in Fig. 21. As shown in Fig. 21(a), the total execution time raises linearly, when we increase the data amounts for both workday and weekend. For all the 13 h of taxi data, *mT-Share* can complete all calculations within 4 h for workday data. Although probabilistic routing takes time, *mT-Sharepro* can still finish within 6 h for weekend data. Fig. 21(b) shows that the response time is quite stable in both workday and weekend. Specifically, the average response time for workday and weekend are 110 and 420 ms, respectively. These results prove that our scheme is computationally efficient and can well scale to large amounts of data for a large city.

VI. Conclusion

This article presents a novel dynamic taxi ridesharing scheme—*mT-Share*, which fully exploits the mobility information of taxis and ride requests. To improve the existing works, *mT-Share* utilizes both map partitions and mobility clusters to index taxis and ride requests and has optimized passenger–taxi matching with efficient routing. In particular, *mT-Share* supports the shared taxis to well serve both online and offline ride requests. Based on a large real-world taxi data set, our experimental evaluations show that *mT-Share* can respond each request within milliseconds and greatly outperforms the state-of-the-art works, e.g., serving 42% and 62% more ride requests in peak and nonpeak hours, respectively.

References

[1] *Gaia Initiative*. Accessed: Jun. 1, 2021. [Online]. Available: https://outreach.didichuxing.com/research/opendata

[2] *Openstreetmap*. Accessed: Jun. 1, 2021. [Online]. Available: http://www.openstreetmap.org/

[3] *Taxi Service Research Report*. Accessed: Jun. 1, 2021. [Online]. Available: http://www.transformcn.com/Topics/2018-08/02/b7944fb3-1b99-4840-89d7-eecaaec67bea.pdf

[4] A. O. Al-Abbasi, A. Ghosh, and V. Aggarwal, "Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 12, pp. 4714–4727, Dec. 2019.

[5] M. Asghari, D. Deng, C. Shahabi, U. Demiryurek, and Y. Li, "Price-aware real-time ride-sharing at scale: An auction-based approach," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst. (GIS)*, 2016, pp. 1–10.

[6] M. Asghari and C. Shahabi, "An on-line truthful and individually rational pricing mechanism for ride-sharing," in *Proc. 25th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst. (GIS)*, 2017, pp. 1–10.

[7] R. Baldacci, V. Maniezzo, and A. Mingozzi, "An exact method for the car pooling problem based on Lagrangean column generation," *Oper. Res.*, vol. 52, no. 3, pp. 422–439, 2004.

[8] X. Bei and S. Zhang, "Algorithms for trip-vehicle assignment in ride-sharing," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3–9.

[9] B. Cao, L. Alarabi, M. F. Mokbel, and A. Basalamah, "SHAREK: A scalable dynamic ride sharing system," in *Proc. 16th IEEE Int. Conf. Mobile Data Manag. (MDM)*, 2015, pp. 4–13.

[10] P. S. Castro, D. Zhang, C. Chen, S. Li, and G. Pan, "From taxi GPS traces to social and community dynamics: A survey," *ACM Comput. Surveys*, vol. 46, no. 2, p. 17, 2013.

[11] L. Chen, Q. Zhong, X. Xiao, Y. Gao, P. Jin, and C. S. Jensen, "Price-and-time-aware dynamic ridesharing," in *Proc. IEEE 34th Int. Conf. Data Eng. (ICDE)*, 2018, pp. 1061–1072.

[12] P. Cheng, H. Xin, and L. Chen, "Utility-aware ridesharing on road networks," in *Proc. ACM Int. Conf. Manag. Data (SIGMOD)*, 2017, pp. 1197–1210.

[13] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem (DARP): Variants, modeling issues and algorithms," *Quart. J. Belgian French Italian Oper. Res. Soc.*, vol. 1, no. 2, pp. 89–101, 2003.

[14] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[15] X. Geng *et al.*, "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 3656–3663.

[16] W. He, K. Hwang, and D. Li, "Intelligent carpool routing for urban ridesharing by mining GPS trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2286–2296, Oct. 2014.

[17] Y. He, J. Ni, X. Wang, B. Niu, F. Li, and X. Shen, "Privacy-preserving partner selection for ride-sharing services," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 5994–6005, Jul. 2018.

[18] Y. Hou, W. Zhong, L. Su, K. Hulme, A. W. Sadek, and C. Qiao, "TASeT: Improving the efficiency of electric taxis with transfer-allowed rideshare," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9518–9528, Dec. 2016.

[19] Y. Huang, F. Bastani, R. Jin, and X. S. Wang, "Large scale real-time ridesharing with service guarantee on road networks," *Proc. VLDB Endowm.*, vol. 7, no. 14, pp. 2017–2028, 2014.

[20] M. Li *et al.*, "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *Proc. World Wide Web Conf. (WWW)*, 2019, pp. 983–994.

[21] Y. Li *et al.*, "Top-*k* vehicle matching in social ridesharing: A price-aware approach," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 3, pp. 1251–1263, Mar. 2021.

[22] 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. Disc. Data Min. (KDD)*, 2018, pp. 1774–1783.

[23] Q. Lin, L. Dengt, J. Sun, and M. Chen, "Optimal demand-aware ride-sharing routing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2018, pp. 2699–2707.

[24] Z. Liu, Z. Gong, J. Li, and K. Wu, "Mobility-aware dynamic taxi ridesharing," in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, 2020, pp. 961–972. [Online]. Available: https://ieeexplore.ieee.org/xpl/conhome/9093725/proceeding

[25] Z. Liu, J. Li, and K. Wu, "Context-aware taxi dispatching at city-scale using deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, early access, Nov. 3, 2020, doi: 10.1109/TITS.2020.3030252.

[26] 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.

[27] 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.

[28] Q. Ma, Z. Cao, K. Liu, and X. Miao, "QA-Share: Toward an efficient QoS-aware dispatching approach for urban taxi-sharing," *ACM Trans. Sens. Netw.*, vol. 16, no. 2, pp. 1–21, 2020.

[29] S. Ma, Y. Zheng, and O. Wolfson, "T-Share: A large-scale dynamic taxi ridesharing service," in *Proc. IEEE 29th Int. Conf. Data Eng. (ICDE)*, 2013, pp. 410–421.

[30] 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.

[31] V. M. de Lira, R. Perego, C. Renso, S. Rinzivillo, and V. C. Times, "Boosting ride sharing with alternative destinations," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2290–2300, Jul. 2018.

[32] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Predicting taxi–passenger demand using streaming data," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1393–1402, Sep. 2013.

[33] Online New York Post news. (2016). *Apps See Surge in Riders Willing to Get Comfy With Strangers*. [Online]. Available: https://nypost.com/2016/08/13/apps-see-surge-in-riders-willing-to-get-comfy-with-strangers/

[34] S. Saisubramanian, C. Basich, S. Zilberstein, and C. V. Goldman, "Satisfying social preferences in ridesharing services," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, 2019, pp. 3720–3725.

[35] A. B. Sherif, K. Rabieh, M. M. E. A. Mahmoud, and X. Liang, "Privacy-preserving ride sharing scheme for autonomous vehicles in big data era," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 611–618, Apr. 2017.

[36] N. Ta, G. Li, T. Zhao, J. Feng, H. Ma, and Z. Gong, "An efficient ride-sharing framework for maximizing shared route," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 2, pp. 219–233, Feb. 2018.

[37] L. Tang, Z. Duan, Y. Zhu, J. Ma, and Z. Liu, "Recommendation for ridesharing groups through destination prediction on trajectory data," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 1320–1333, Feb. 2021.

[38] X. Tang *et al.*, "A deep value-network based approach for multi-driver order dispatching," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2019, pp. 1780–1790.

[39] R. S. Thangaraj, K. Mukherjee, G. Raravi, A. Metrewar, N. Annamaneni, and K. Chattopadhyay, "Xhare-a-Ride: A search optimized dynamic ride sharing system with approximation guarantee," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, 2017, pp. 1117–1128.

[40] Y. Tong *et al.*, "The simpler the better: A unified approach to predicting original taxi demands based on large-scale online platforms," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2017, pp. 1653–1662.

[41] Y. Tong, L. Wang, Z. Zhou, L. Chen, B. Du, and J. Ye, "Dynamic pricing in spatial crowdsourcing: A matching-based approach," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2018, pp. 773–788.

[42] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, J. Ye, and K. Xu, "A unified approach to route planning for shared mobility," *Proc. VLDB Endowm*, vol. 11, no. 11, pp. 1633–1646, 2018.

[43] D. Wang and X. Zhang, "Secure ride-sharing services based on a consortium blockchain," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2976–2991, Feb. 2021.

[44] J. Wang *et al.*, "Demand-aware route planning for shared mobility services," *Proc. VLDB Endowm.*, vol. 13, no. 7, pp. 979–991, 2020.

[45] Y. Xu, Y. Tong, Y. Shi, Q. Tao, K. Xu, and W. Li, "An efficient insertion operator in dynamic ridesharing services," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, 2019, pp. 1022–1033.

[46] H. Yao *et al.*, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2588–2595.

[47] H. Yu, H. Zhang, X. Yu, X. Du, and M. Guizani, "PGRide: Privacy-preserving group ridesharing matching in online ride hailing services," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5722–5735, Apr. 2021.

[48] 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.

[49] C. F. Yuen, A. P. Singh, S. Goyal, S. Ranu, and A. Bagchi, "Beyond shortest paths: Route recommendations for ride-sharing," in *Proc. World Wide Web Conf. (WWW)*, 2019, pp. 2258–2269.

[50] D. Zhang *et al.*, "Carpooling service for large-scale taxicab networks," *ACM Trans. Sens. Netw.*, vol. 12, no. 3, p. 18, 2016.

[51] J. Zhang, D. Wen, and S. Zeng, "A discounted trade reduction mechanism for dynamic ridesharing pricing," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 6, pp. 1586–1595, Jun. 2016.

[52] L. Zhang *et al.*, "A taxi order dispatch model based on combinatorial optimization," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2017, pp. 2151–2159.

[53] W. E. Zhang, A. Shemshadi, Q. Z. Sheng, Y. L. Qin, X. Xu, and J. Yang, "A user-oriented taxi ridesharing system with large-scale urban GPS sensor data," *IEEE Trans. Big Data*, vol. 7, no. 2, pp. 327–340, Jun. 2021.

[54] B. Zhao, P. Xu, Y. Shi, Y. Tong, Z. Zhou, and Y. Zeng, "Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 2245–2252.

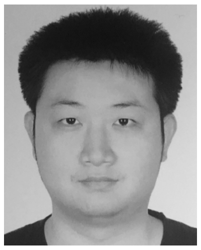
[55] L. Zheng, L. Chen, and J. Ye, "Order dispatch in price-aware ridesharing," *Proc. VLDB Endowm.*, vol. 11, no. 8, pp. 853–865, 2018.

[56] L. Zheng, P. Cheng, and L. Chen, "Auction-based order dispatch and pricing in ridesharing," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, 2019, pp. 1034–1045.

[57] M. Zhu, X.-Y. Liu, and X. Wang, "An online ride-sharing path-planning strategy for public vehicle systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 2, pp. 616–627, Feb. 2019.

Zhidan Liu (Member, IEEE) received the Ph.D. degree in computer science and technology from Zhejiang University, Hangzhou, China, in 2014.

He worked as a Research Fellow with Nanyang Technological University, Singapore. In 2017, he joined Shenzhen University, Shenzhen, China, as an Assistant Professor. His research interests include distributed sensing and mobile computing, big data analytics, Internet of Things, and urban computing.



Zengyang Gong received the B.S. degree in computer science and technology from Wuhan University of Science and Technology, Wuhan, China, in 2017, and the master's degree in computer science and technology from Shenzhen University, Shenzhen, China, in 2020. He is currently pursuing the Ph.D. degree with Hong Kong University of Science and Technology, Hong Kong.

His research interests are in the areas of big data, including traffic modeling, pervasive computing, and urban computing.

Jiangzhou Li received the B.S. degree in software engineering from Qingdao University, Qingdao, China, in 2018. He is currently pursuing the master's degree with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China, under the supervision of Dr. Z. Liu.

His research interests are in the areas of big data, including data analysis, urban computing, and reinforcement learning.

Kaishun Wu (Member, IEEE) received the Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2011.

He worked as a Research Assistant Professor with HKUST. In 2013, he joined Shenzhen University, Shenzhen, China, as a Distinguish Professor. He has coauthored two books and published over 100 high-quality research papers in international leading journals and primer conferences, such as IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, ACM MobiCom, and IEEE INFOCOM. He is the inventor of six U.S. and over 90 Chinese pending patents.

Dr. Wu received the 2012 Hong Kong Young Scientist Award, the 2014 Hong Kong ICT Awards: Best Innovation, and the 2014 IEEE ComSoc Asia–Pacific Outstanding Young Researcher Award. He is an IET Fellow.